**Instituto Politécnico Nacional.**

**Escuela Superior De Cómputo.**





**Materia:**

**Desarrollo de Sistemas Distribuidos.**

**Tema:**

**Chat Multicast.**

**(Tarea 05)**

**Profesor:**

**Carlos Pineda Guerrero.**

**Alumno:**

**Mario Alberto Miranda Sandoval.**

**Grupo:**

**4CM5**

**Objetivo.**

Desarrollar un programa en Java que implemente un chat utilizando comunicación multicast mediante datagramas.

**Desarrollo.**

Para desarrollar esta práctica es necesario implementar dos funciones vistas en la clase, las cuales son envia\_mensaje y recibe\_mensaje, además que se debió completar el código base facilitado por el profesor.

Después de añadir las funciones vistas en clase, procedemos a completar la clase Worker, donde primero declaramos un objeto de tipo MulticastSocket y ese objeto se inicializará con el sokcet multicast que pasemos al Worker al instanciar la clase.

Posteriormente en el método run, en un ciclo infinito, colocamos un arreglo de bytes que devolverá la función recibe\_mensaje, la cantidad de bytes que se mandaran a la función para crear el paquete será de 100 bytes, esto viene siendo equivalente a 100 caracteres por mensaje en la línea de código, por último, mostramos el mensaje en pantalla.

1. **static** **class** Worker **extends** Thread {
2. MulticastSocket socket;
4. **public** Worker(MulticastSocket socket) {
5. **this**.socket = socket;
6. }
8. **public** **void** run() {
9. **while**(**true**) {
10. **try** {
11. **byte**[] a = recibe\_mensaje(socket, 100);
12. System.out.println(**new** String(a, "UTF-8"));
13. } **catch**(IOException e) { e.printStackTrace(); }
14. }
15. }
16. }

Ahora en el main, primero validamos que se este mandando el nombre de usuario en la consola, una vez validado esto pasamos a crear el socket multicast y unirnos al grupo, con el socket ya unido al grupo lo mandamos a la clase Worker cuando se crea la instancia de esta, luego como ya es costumbre con el método start comenzamos la ejecución del hilo.

1. **public** **static** **void** main(String[] args) **throws** Exception {
2. **if**(args.length != 1) {
3. System.err.println("Se debera pasar por consola el nombre de usuario");
4. System.exit(1);
5. }
7. InetAddress grupo = InetAddress.getByName("230.0.0.0");
8. MulticastSocket socket = **new** MulticastSocket(50000);
9. socket.joinGroup(grupo);
11. Worker w = **new** Worker(socket);
12. w.start();

Por último, recuperamos el nombre de usuario de los argumentos, abrimos la entrada de datos por teclado y en un ciclo infinito nos mantenemos a la escucha de la entrada por teclado, una vez dada la entrada por teclado pasamos a usar una secuencia de control del terminal en Linux (posiblemente no funcione en otro sistema operativo), donde lo que se hace es regresar la línea anterior y colocarla al inicio de la terminal, de este modo se “borra” el mensaje escrito desde el teclado, luego le damos el formato deseado concatenando todo en un String, por último enviamos el mensaje con la función envia\_mensaje.

1. String nombre = args[0];
2. BufferedReader b = **new** BufferedReader(**new** InputStreamReader(System.in));
4. **while**(**true**) {
5. String ínea = b.readLine();
6. System.out.print(“\33[1ª\33[2K”);
7. String formato = “-“ + nombre + “ escribe: “ + ínea.trim();
8. envia\_mensaje(formato.getBytes(), “230.0.0.0”, 50000);
9. }

**Ejecución.**

Captura de pantalla de computadora

Descripción generada automáticamenteCaptura de pantalla de computadora

Descripción generada automáticamenteCaptura de pantalla de computadora

Descripción generada automáticamenteCaptura de pantalla de computadora

Descripción generada automáticamente

Como se usa el mismo programa solo lo compilaremos una vez, el warning sale debido a que uso el JDK 15 y el método joinGroup ya se encuentra depreciado.

Ahora podemos ver la ejecución de cada usuario en el chat, se puede apreciar como la conversacion va de manera fluida.

Captura de pantalla de computadora

Descripción generada automáticamenteCaptura de pantalla de computadora

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamenteCaptura de pantalla de computadora

Descripción generada automáticamente

Por último, el uso de la secuencia de control se ve de manera más general en este ejemplo, donde se ve que el usuario manda **hola** y sobre la misma línea donde escribió se coloca **-donald escribe: hola**.

Captura de pantalla de computadora

Descripción generada automáticamenteCaptura de pantalla de computadora

Descripción generada automáticamente

Si la secuencia de control no se hubiera ejecutado el resultado en consola se vería del siguiente modo:

**hola**

**-donald escribe: hola**

Lo cual al momento de usar el chat haría que fuera poco estética la vista.

**Conclusiones.**

Se puede observar y comparar la distribución de multicast es demasiado eficiente en ciertos tipos de comunicación, este mismo ejercicio comparado con sockets unicast requeriría de manera forzosa un servidor y este servidor debería tener una manera de almacenar los mensajes enviados y después distribuirlos respetando el orden, además de tener especial cuidado con la concurrencia de los mensajes, todo esto se ve resuelto con la comunicación multicast.