



Instituto Politécnico Nacional Escuela Superior de Cómputo

Desarrollo de Sistemas Distribuidos

Actividad: Chat Multicast

Curso impartido por el profesor: Pineda Guerrero Carlos

Grupo: 4CV1

21/01

Alumno: Adrian González Pardo



Ultima fecha modificado: 10 de noviembre de 2020

1. Desarrollo

Para el desarrollo necesario de esta practica es necesario realizar el uso de paquetes nativos de java los cuales son el uso Sockets, Buffer de lectura de datos, Entrada de datos por teclado, Hilos, y algunas otras cosas más que vienen en el código fuente. por ello se programo un chat de multicast el cual debido a que no conocemos el tamaño de datos de entrada, haremos uso de una restricción que viene en nuestra tarjeta de red en Linux, el cual es un maximo de datos de 1500 por ello en caso de que sobren datos el mismo programa enviara los datos de forma serializada y un poco instantanea.

2. Código fuente:

```
1 import java.net.DatagramPacket;
2 import java.io.IOException;
3 import java.net.DatagramPacket;
4 import java.net.DatagramSocket;
5 import java.net.InetAddress;
6 import java.nio.ByteBuffer;
7 import java.net.InetAddress;
8 import java.lang.Thread;
9 import java.io.BufferedReader;
10 import java.io.InputStreamReader;
11 import java.net.MulticastSocket;
12 import java.net.InetAddress;
13
14 /*
15  * @author Adrian Gonzalez Pardo
16  */
17
18 class Chat{
19
20     public static String nombre;
21     static void envia_mensaje(byte[] buffer,String ip,int puerto) throws
        IOException{
22         DatagramSocket socket = new DatagramSocket();
23         InetAddress grupo = InetAddress.getByName(ip);
24         DatagramPacket paquete = new DatagramPacket(buffer,buffer.length,grupo,
            puerto);
25         socket.send(paquete);
26         socket.close();
27     }
28
29     static byte[] recibe_mensaje(MulticastSocket socket,int longitud) throws
        IOException{
30         byte[] buffer = new byte[longitud];
31         DatagramPacket paquete = new DatagramPacket(buffer,buffer.length);
32         socket.receive(paquete);
33         return buffer;
34     }
35
36     static class Worker extends Thread{
37         public void run(){
38             /* En un ciclo infinito se recibiran los mensajes enviados al grupo
39              230.0.0.0 a traves del puerto 50000 y se desplegaran en la pantalla.
40             */
41             try{
42                 InetAddress grupo = InetAddress.getByName("230.0.0.0");
43                 MulticastSocket socket = new MulticastSocket(50000);
44                 socket.joinGroup(grupo);
45                 /* Maximo tamaño de datagrama por la interfaz de red */
46                 int max_size=1500;
47                 String mensaje="";
48                 while(true){
49                     byte[] a=recibe_mensaje(socket,max_size);
```

```

50         mensaje+=new String(a,"UTF-8").replace("*","")+"\n";
51         /* Limpia la pantalla de todo el texto para actualizar todo el chat */
52         System.out.print("\033[H\033[2J");
53         System.out.flush();
54         System.out.println("Bienvenido: "+nombre);
55         System.out.println(mensaje);
56         System.out.print("Escribe un mensaje: ");
57     }
58 }catch(Exception e){
59     System.err.println("Exception: "+e.getClass()+" with "+e);
60 }
61 }
62 }
63
64 public static void main(String[] args) throws Exception{
65     if(args.length<1){
66         System.err.println("Error usage\njava Chat <username>");
67         System.exit(1);
68     }
69     Worker w = new Worker();
70     w.start();
71     nombre = args[0];
72     BufferedReader b = new BufferedReader(new InputStreamReader(System.in));
73     /* En un ciclo infinito se leera los mensajes del teclado y se enviara
74        al grupo 230.0.0.0 a traves del puerto 50000.
75     */
76     System.out.println("Bienvenido: "+nombre);
77
78     String msg="";
79     int i;
80     System.out.print("Escribe el mensaje: ");
81     while(true){
82         msg=b.readLine();
83         msg=nombre+" escribe: "+msg;
84         /* Usa el tamaño máximo de MTU de la interfaz de red de linux */
85         if(msg.length()%1500!=0){
86             int resto=1500-msg.length()%1500;
87             for(i=0;i<resto;i++){
88                 msg+="*";
89             }
90         }
91         /* Envía los datos de forma segmentada, por lo tanto todo
92            lo que escriba es todo lo que envía
93         */
94         for(i=0;i<msg.length();i+=1500){
95             envia_mensaje(msg.substring(i,i+1500).getBytes(),"230.0.0.0",50000);
96         }
97     }
98 }
99 }

```

```

1 # Archivo Makefile
2 # @author Adrian Gonzalez Pardo
3 JVC=javac
4 SRCC=$(wildcard *.java)
5 OBJJS=$(SRCC:.java=.class)
6 all: ${OBJJS}
7
8 %.class: %.java
9     ${JVC} $<
10
11 .PHONY: clean
12
13 clean:
14     rm *.class

```

3. Ejecución en red

Para este caso se realizó una implementación a nivel red Local con al menos 4 equipos, se puede hacer lo siguiente (destacando que ya hay un emparejamiento de llaves ssh) y ellos mismos se unirán al grupo de Multicast

A nivel local se conoce la lista de los siguientes equipos:

- Lenovo IP: 192.168.100.69
- Acer IP: 192.168.100.3
- Raspberry Pi 3B IP: 192.168.100.194
- Raspberry Pi 4B IP: 192.168.100.103

Para el cual se envió los datos con el siguiente script:

```
1 #!/usr/bin/env bash
2
3 # @author Adrian Gonzalez Pardo
4
5 # Envía los archivos java y el makefile a los equipos que están
6 # relacionados via ssh
7 scp *.java Makefile d3v@192.168.100.3:~/chat-multicast
8 scp *.java Makefile pi@192.168.100.103:~/chat-multicast
9 scp *.java Makefile pi@192.168.100.195:~/chat-multicast
10
11
12 # Ejecuta la instrucción solo para ser ejecutado más tarde
13 # es decir compila el código fuente de acuerdo a la versión
14 # de jdk que tenga instaladas
15 # (puede que no tenga compatibilidades)
16 ssh d3v@192.168.100.3 "cd chat-multicast && make"
17 ssh pi@192.168.100.103 "cd chat-multicast && make"
18 ssh pi@192.168.100.195 "cd chat-multicast && make"
```

4. Capturas y descripción del programa

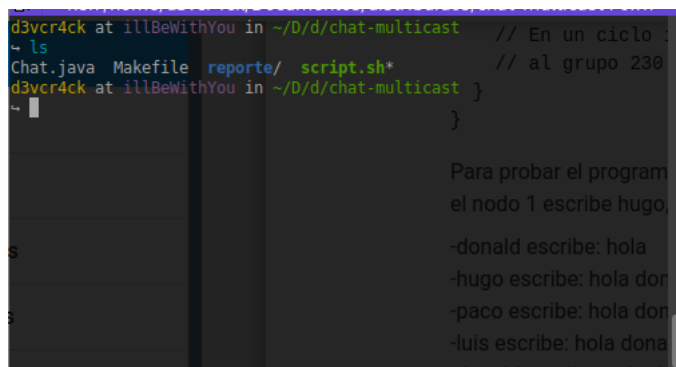


Figura 1: Antes de la compilación con Makefile

```

d3vcr4ck at illBeWithYou in ~/D/d/chat-multicast // En un ciclo.
└─ ls
Chat.java Makefile reporte/ script.sh* // al grupo 238
d3vcr4ck at illBeWithYou in ~/D/d/chat-multicast }
└─ make
javac Chat.java }
d3vcr4ck at illBeWithYou in ~/D/d/chat-multicast
└─ ls
'Chat$Worker.class' Chat.java reporte/ Para probar el program
Chat.class Makefile script.sh* el nodo 1 escribe hugo
d3vcr4ck at illBeWithYou in ~/D/d/chat-multicast
└─
-donald escribe: hola
-hugo escribe: hola don
-paco escribe: hola don
-luis escribe: hola dona

```

Figura 2: Compilación

```
java /home/d3vcr4ck/Documentos/distribuidos/chat-multicast
d3vcr4ck at illBeWithYou in ~/D/d/chat-multicast
$ java Chat donald
Bienvenido: donald
Escribe el mensaje:
}
Worker w = new Worker();
w.start();
String nombre = args[0];
BufferedReader b = new BufferedReader(new InputStreamReader(System.in));
// En un ciclo infinito se leera los mensajes del teclado y se enviaran
// al grupo 230.0.0.0 a traves del puerto 50000.
while(true){
    System.out.println("Bienvenido: "+nombre);

    String msg="";
    int i;
    while(true){
        System.out.println("Escribe el mensaje:");
        msg=b.readLine();
        if(msg.length()%1500!=0){
            int resto=1500-msg.length()%1500;
            for(i=0;i<resto;i++){
                msg+=" ";
            }
        }
        // Envia los datos de forma segmentada, por lo tanto todo
        // lo que escriba es todo lo que envia
        for(i=0;i<msg.length();i+=1500){
            envia_mensaje(msg.substring(i,i+1500).getBytes(),"230.0.0.0",50000);
        }
    }
}

pi@beWithYouR4B: ~/chat-multicast 66x17
pi@beWithYouR4B:~/chat-multicast $ java Chat paco
Bienvenido: paco
Escribe el mensaje:
}

pi@beWithYou3b: ~/chat-multicast 66x17
pi@beWithYou3b:~/chat-multicast $ java Chat luis
Bienvenido: luis
Escribe el mensaje:
}
```

Figura 3: Prompt del chat de multicast.

```
java /home/d3v/chat-multicast 55x16
Bienvenido: donald
donald escribe: hola
hugo escribe: hola donald
paco escribe: hola donald
luis escribe: hola donald
donald escribe: ¿alguien sabe dónde será la fiesta el sábado?
paco escribe: será en la casa de tío mac pato
hugo escribe: ¿a qué hora?
luis escribe: a las 8 PM
donald escribe: adiós
hugo escribe: adiós donald
Escribe un mensaje:

java /home/d3v/chat-multicast 58x15
Bienvenido: hugo
donald escribe: hola
hugo escribe: hola donald
paco escribe: hola donald
luis escribe: hola donald
donald escribe: ¿alguien sabe dónde será la fiesta el sábado?
paco escribe: será en la casa de tío mac pato
hugo escribe: ¿a qué hora?
luis escribe: a las 8 PM
donald escribe: adiós
hugo escribe: adiós donald
Escribe un mensaje:

pi@beWithYouR4B: ~/chat-multicast 55x16
Bienvenido: paco
donald escribe: hola
hugo escribe: hola donald
paco escribe: hola donald
luis escribe: hola donald
donald escribe: ¿alguien sabe dónde será la fiesta el sábado?
paco escribe: será en la casa de tío mac pato
hugo escribe: ¿a qué hora?
luis escribe: a las 8 PM
paco escribe: adiós donald
donald escribe: adiós
hugo escribe: adiós donald
Escribe un mensaje:

pi@beWithYou3b: ~/chat-multicast 58x16
Bienvenido: luis
donald escribe: hola
hugo escribe: hola donald
paco escribe: hola donald
luis escribe: hola donald
donald escribe: ¿alguien sabe dónde será la fiesta el sábado?
paco escribe: será en la casa de tío mac pato
hugo escribe: ¿a qué hora?
luis escribe: a las 8 PM
donald escribe: adiós
hugo escribe: adiós donald
Escribe un mensaje:
```

Figura 4: Ejecución del chat donde llegan los nuevos mensajes.

5. Conclusiones

El realizar este tipo de programas cuya ejecución se puede realizar de forma universal o en cualquier equipo de cómputo, es gracias a que existe un estandar de red y gracias a la parte de Java de ser un lenguaje multiplataforma el cual nos permite compilar en multiples arquitecturas con diferentes versiones del mismo jdk, por ello es necesario reconocer que igual existen otros lenguajes los cuales nos permiten realizar las mismas tareas.