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 tamanio 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);
```

```
          mensaje+=new String(a,"UTF-8").replace("*","")+"\n";
          /* Limpia la pantalla de todo el texto para actualizar todo el chat */
          System.out.print("\033[H\033[2J");
          System.out.flush();
          System.out.println("Bienvenido: "+nombre);
          System.out.println(mensaje);
          System.out.print("Escribe un mensaje: ");
        }
      }catch(Exception e){
        System.err.println("Exception: "+e.getClass()+" with "+e);
      }
    }
  }

  public static void main(String[] args) throws Exception{
    if(args.length<1){
      System.err.println("Error usage\njava Chat <username>");
      System.exit(1);
    }
    Worker w = new Worker();
    w.start();
    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.
    */
    System.out.println("Bienvenido: "+nombre);

    String msg="";
    int i;
    System.out.print("Escribe el mensaje: ");
    while(true){
      msg=b.readLine();
      msg=nombre+" escribe: "+msg;
      /* Usa el tamanio maximo de MTU de la interfaz de red de linux */
      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);
      }
    }
  }
}
```

```
# Archivo Makefile
# @author Adrian Gonzalez Pardo
JVC=javac
SRCC=$(wildcard *.java)
OBJS=$(SRCC:.java=.class)
all: ${OBJS}

%.class: %.java
   ${JVC} $<

.PHONY: clean

clean:
   rm *.class
```

3. Ejecución en red

Para este caso se realizo 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 uniran 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 envio los datos con el siguiente script:

```
#!/usr/bin/env bash

# @author Adrian Gonzalez Pardo

# Envia los archivos java y el makefile a los equipos que estan
#   relacionados via ssh
scp *.java Makefile d3v@192.168.100.3:~/chat-multicast
scp *.java Makefile pi@192.168.100.103:~/chat-multicast
scp *.java Makefile pi@192.168.100.195:~/chat-multicast


# Ejecuta la instruccion solo para ser ejecutado mas tarde
#   es decir compila el codigo fuente de acuerdo a la version
#   de jdk que tenga instaladas
#   (puede que no tenga compatibilidades)
ssh d3v@192.168.100.3 "cd chat-multicast && make"
ssh pi@192.168.100.103 "cd chat-multicast && make"
ssh pi@192.168.100.195 "cd chat-multicast && make"
```

4. Capturas y descripción del programa

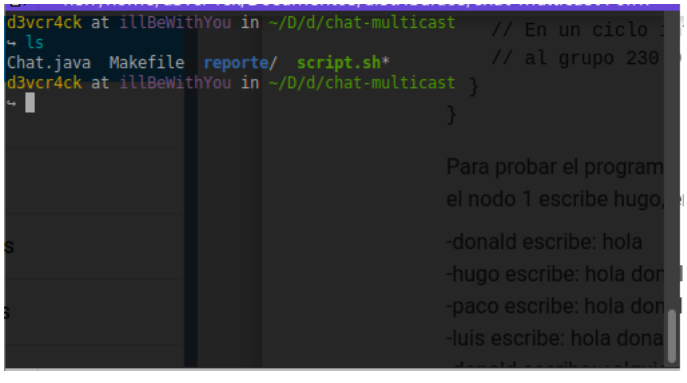

Figura 1: Antes de la compilación con Makefile

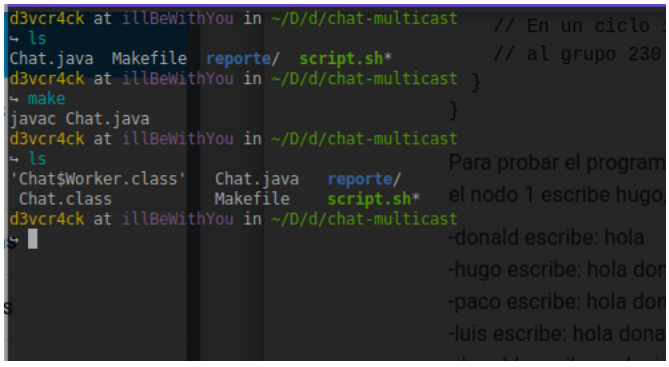

Figura 2: Compilación

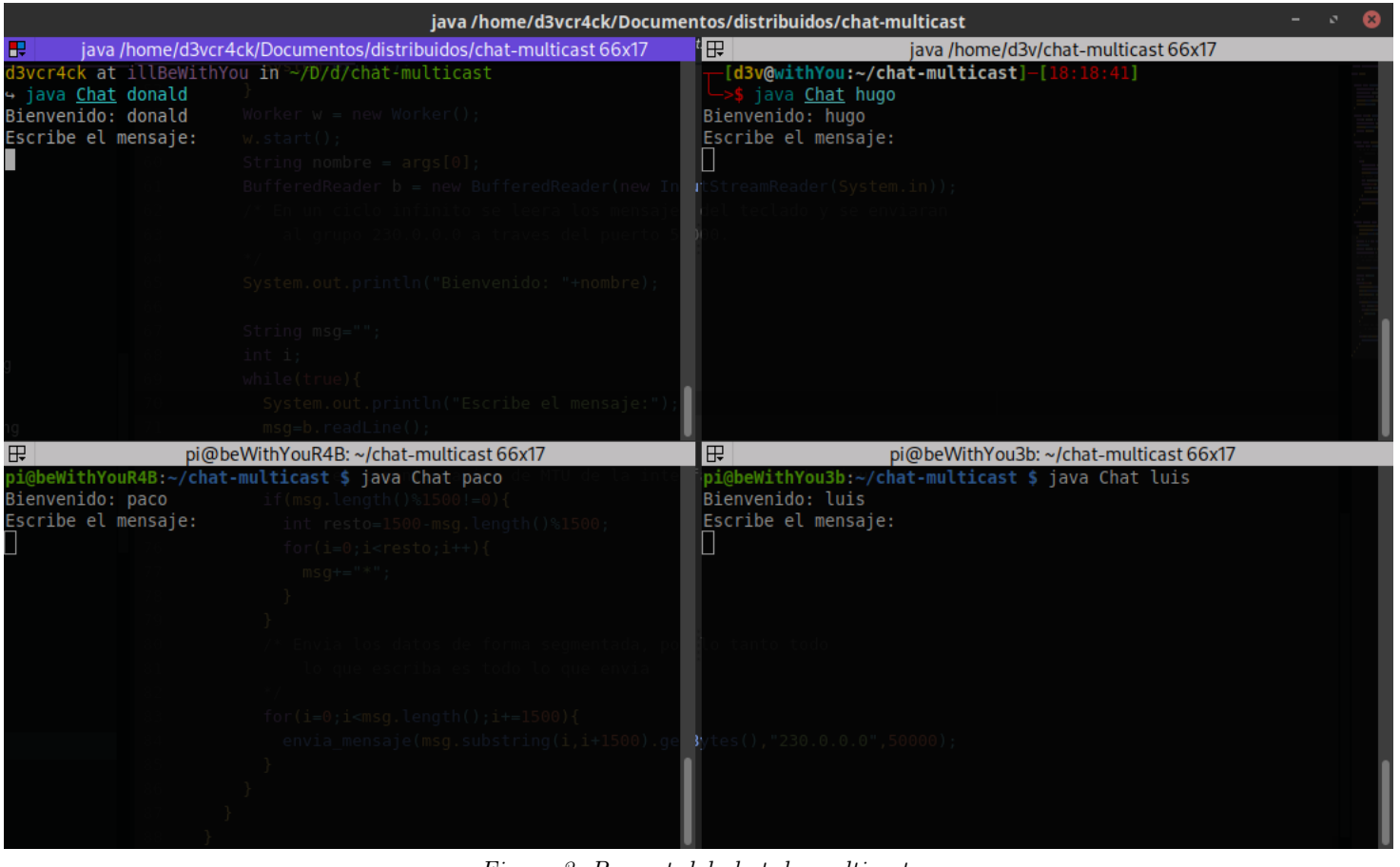

Figura 3: Prompt del chat de multicast.

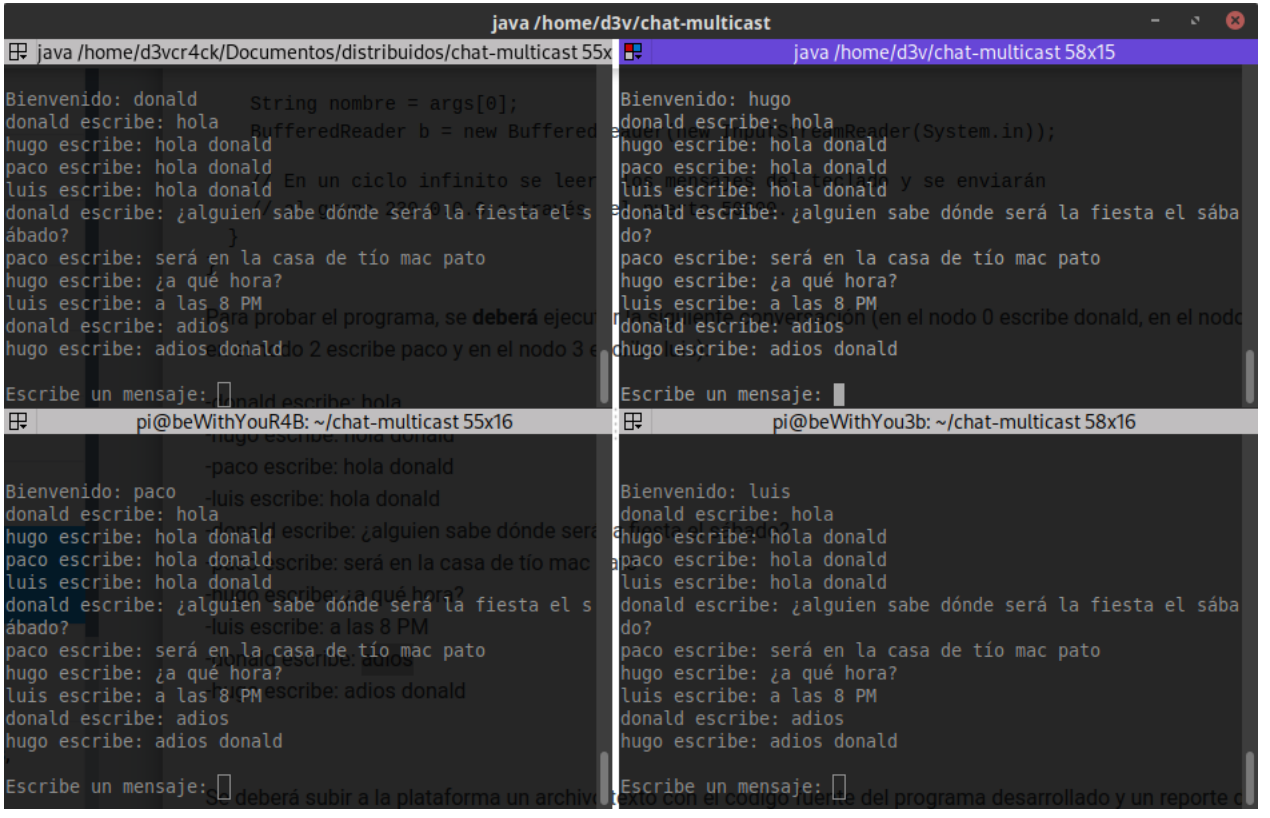

Figura 4: Ejecución del chat donde llegan los nuevos mensajes.

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.