INSTITUTO POLITÉCNICO NACIONAL

Escuela Superior de Cómputo

Aplicaciones para comunicaciones en red

Práctica 2. Buscaminas

Alumnos:

- Godínez Montero Esmeralda
- Martinez Martinez Fernando

Grupo: 3CM17

Profesor: Axel Ernesto Moreno Cervantes

INSTITUTO POLITÉCNICO NACIONAL

ESCOM

Contenido

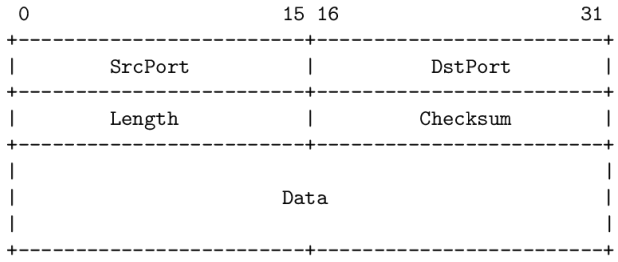
Introducción teórica

Protocolo UDP

El protocolo UDP (Protocolo de Datagrama de Usuario) proporciona puertos que se emplean para distinguir entre múltiples programas que se estén ejecutando en una misma máquina (demultiplexación). Por lo tanto, además de los datos enviados, cada mensaje UDP tiene una cabecera que contiene tanto el número de puerto de destino como el número de puerto de origen, haciendo posible que el protocolo UDP en el destinatario envíe el mensaje al destino correcto y al destinatario enviar una contestación

Además, utiliza un sistema no orientado a la conexión, es decir, no asegura que los mensajes han llegado correctamente al destinatario, no reordena los mensajes recibidos y no proporciona control de flujo de información. Así, los mensajes UDP pueden perderse, duplicarse o llegar fuera de orden y, además, los paquetes pueden llegar más deprisa de lo que el receptor puede procesarlos.

Cada mensaje UDP, denominado datagrama de usuario, consta de dos partes fundamentales: la cabecera y el área de datos. La cabecera se divide en cuatro campos de 16 bits que especifican el puerto desde el cual se envía el mensaje, el puerto al cual está destinado el mensaje, la longitud del datagrama de usuario, y un checksum para comprobar que el datagrama llega a su destino sin haber sufrido ninguna alteración.



Estructura de un paquete UDP

El campo Puerto de Origen es opcional, cuando se utiliza, especifica el puerto al que se debería enviar la contestación, si no se utiliza, debe ser 0. Por su parte, el campo Longitud contiene el número de octetos del datagrama UDP incluyendo la cabecera y el campo de datos de usuario, por lo tanto, el valor mínimo de este campo es 8, la longitud de la cabecera (es por lo tanto posible enviar datagramas en los cuales la longitud del campo de datos sea 0).

UDP en Java

En las aplicaciones en red es muy común el paradigma cliente-servidor. El servidor es el que espera las conexiones del cliente (en un lugar claramente definido) y el cliente es el que lanza las peticiones a la maquina donde se está ejecutando el servidor. A través de las clases del paquete java.net, los programas Java pueden utilizar TCP o UDP para comunicarse a través de Internet.

Los Datagram Socket son más eficientes que TCP, pero no está garantizada la fiabilidad: los datos se envían y reciben en paquetes, cuya entrega no está garantizada, en este caso, se tienen los siguientes métodos:

- **Constructor public DatagramSocket (int port):** Se encarga de construir un socket para datagramas especificando el número de puerto.
- **public void receive (DatagramPacket p):** Recibe un DatagramPacket del socket, y llena el búfer con los datos que recibe.
- **public void send (DatagramPacket p):** Envía un DatagramPacket a través del socket.
- **public int getLocalPort():** Retorna el número de puerto en el host local al que está conectado el socket.

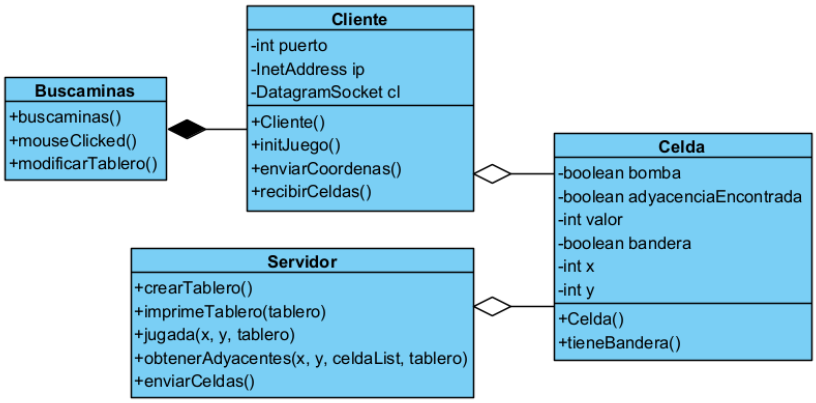
Por su parte, un DatagramPacket contiene la información relevante. Cuando se desea recibir un datagrama, éste deberá almacenarse bien en un búfer o en un array de bytes. Y cuando se prepara un datagrama para ser enviado, el DatagramPacket no sólo debe tener la información, sino que además debe tener la dirección IP y el puerto de destino, que puede coincidir con un puerto TCP.

- **Constructor public DatagramPacket(byte ibuf[], int ilength):** Implementa un DatagramPacket para la recepción de paquetes de longitud length, siendo el valor de este parámetro menor o igual que ibuf.length.
- **public byte[] getData():** Retorna los datos a recibir o a enviar.
- **public InetAddress getAddress ():** Retorna la dirección IP del host al cual se le envía el datagrama o del que el datagrama se recibió.
- **public int getLength()**: Retorna la longitud de los datos a enviar o a recibir.
- **public int getPort():** Retorna el número de puerto de la máquina remota a la que se le va a enviar el datagrama o del que se recibió.

Desarrollo

En esta práctica, se programó un buscaminas cliente-servidor que permite elegir una dirección IP y un numero de puerto de servidor, luego aparece un tablero de buscaminas en el cual el cliente puede comenzar su partida.

Se consideró que en el servidor solo se pudiera conectar un cliente a la vez. A continuación de presenta el diseño del juego programado en Java.



Cliente

El cliente ejecuta el archivo Buscaminas.java donde se encuentran los siguientes métodos:

buscaminas()

```
public void buscaminas(){
        //Se inicializa la ventana
        setBounds(20, 20, 700, 700);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        //Se incializa el panel superior y sus componentes
        JPanel panelSup = new JPanel();
```

```
        banderaIconoLabel = new JLabel();
        ImageIcon icono = new ImageIcon("img/bandera.png");
        banderaIconoLabel.setIcon(new
ImageIcon(icono.getImage().getScaledInstance(38,38,Image.SCALE_SMOOTH)));
        panelSup.add(banderaIconoLabel);
        panelSup.setBackground(Color.WHITE);

        banderasLabel = new JLabel("40");
        banderasLabel.setBounds(0,0,20,20);
        panelSup.add(banderasLabel);
        this.add(panelSup,"North");

        botones = new JButton [16][16];
        JPanel tablero = new JPanel(new GridLayout(16,16));
        Color colores[] = new Color[2];
        colores[0] = new Color(170, 215, 81);
        colores[1] = new Color(162, 209, 73);
        for(int i=0;i<16;i++)
            for(int j=0;j<16;j++){
                //  Crear boton
                botones [i][j] = new JButton();
                botones[i][j].setOpaque(true);
                botones[i][j].setBackground(colores[(j+i)%2]);
                botones[i][j].setBorderPainted(false);
                botones[i][j].setSize(11, 11);
                botones[i][j].setName(i+"-"+j);
                //Colocar en el panel
                tablero.add(botones[i][j]);
                //  Action Listener
                botones[i][j].addMouseListener(this);
            }
        tablero.setBackground(Color.WHITE);
        this.add(tablero, "Center");
        setResizable(false);
        setVisible(true);
    }
```

El objetivo de este método es mostrar el tablero del buscaminas; en este método se especifican todas las variables para su diseño. Se agregaron dos paneles, uno superior y otro inferior. En el panel superior se agregó la imagen de una bandera junto a una etiqueta que se utiliza para el contador de las banderas del juego.

En el panel inferior se agregó un grid de 16x16, en donde se colocaron botones de dos colores distintos. A ellos también se les agregó la interfaz AddMouseListener.

mouseClicked()

```
   public void mouseClicked(java.awt.event.MouseEvent e) {
        String coordenadas = ((JButton)e.getSource()).getName();
        if(SwingUtilities.isRightMouseButton(e)){//Se detecta click derecho
            int x = Integer.parseInt(coordenadas.split("-")[0]);
            int y = Integer.parseInt(coordenadas.split("-")[1]);
            if(botones[x][y].getIcon() == null){// Se marca bandera
                bandera--;
                banderasLabel.setText(Integer.toString(bandera));
                ImageIcon icono = new ImageIcon("img/bandera.png");
                botones[x][y].setIcon(new
ImageIcon(icono.getImage().getScaledInstance(20,20,Image.SCALE_SMOOTH)));
            }else{//Se desmarca una bandera
                bandera++;
                banderasLabel.setText(Integer.toString(bandera));
                botones[x][y].setIcon(null);
            }
        }else{//Se detecta click izquierdo
            if(SwingUtilities.isLeftMouseButton(e)){//Se muestra valor
                int x = Integer.parseInt(coordenadas.split("-")[0]);
                int y = Integer.parseInt(coordenadas.split("-")[1]);
                if(botones[x][y].getIcon() == null){
                    cliente.enviarCoordenadas(x,y);
                    ArrayList<Celda> celdas = cliente.recibirCeldas(); //Se
obtienen las celdas afectadas por el movimiento
                    modificarTablero(celdas); //Se hacen los cambios en el
tablero
                }
            }
        }
    }
```

En este método se verifica si el botón del mouse presionado fue derecho o izquierdo. Si fue izquierdo, el cliente envía al servidor las coordenadas del botón presionado y recibe un arreglo de celdas que fueron afectadas por la jugada. Luego se modifica el tablero con el método modificarTablero().

Por otra parte, si se presiona el botón derecho del mouse, se verifica que se tenga o no bandera para marcarla o desmarcarla.

modificarTablero()

```
public void modificarTablero(ArrayList<Celda> celdas){
        String imagen;
        for(int i=0; i<celdas.size(); i++){
            if(celdas.get(i).bomba){//Se acaba el juego
                ImageIcon icono = new ImageIcon("img/bomba.png");
                botones[celdas.get(i).x][celdas.get(i).y].setIcon(new
ImageIcon(icono.getImage().getScaledInstance(20,20,Image.SCALE_SMOOTH)));
                System.out.println("Bombaa!!");
                JOptionPane.showConfirmDialog(null, "Has perdido", "Booom!",
JOptionPane.DEFAULT_OPTION);
                cliente.enviarCoordenadas(-1, -1);
                System.exit(0);
            }else{//Se despliegan las celdas (Cargar imagenes)
                if(celdas.get(i).valor == 0){
                    if(botones[celdas.get(i).x][celdas.get(i).y].getBackgrou
nd() != tierrita){
                        botones[celdas.get(i).x][celdas.get(i).y].setBackgro
und(tierrita);
                        ganador++;
                    }
                }else{
                    if(botones[celdas.get(i).x][celdas.get(i).y].getIcon()
== null){
                        imagen = Integer.toString(celdas.get(i).valor);
                        ImageIcon icono = new
ImageIcon("img/"+imagen+".png");
                        botones[celdas.get(i).x][celdas.get(i).y].setIcon(ne
w ImageIcon(icono.getImage().getScaledInstance(20,20,Image.SCALE_SMOOTH)));
                        botones[celdas.get(i).x][celdas.get(i).y].setBackgro
und(new Color(255,255,255));
                        ganador++;
                    }
                }
                System.out.println(ganador);
                if(ganador == 216){
                    JOptionPane.showMessageDialog(null, "Has ganado",
"Yei!", JOptionPane.DEFAULT_OPTION);
                    cliente.enviarCoordenadas(-1, -1);
                    System.exit(0);
                }
            }
        }
    }
```

El método modificarTablero actualiza las celdas del grid que fueron afectadas en la jugada. Esta función recibe como parámetro el arreglo de celdas enviado por el servidor en el método mouseClicked(). Este arreglo de celdas se verifica, si se ha enviado de una bomba entonces el juego acaba y se cierra la aplicación, si no, simplemente se muestran los valores de las celdas enviada para desplegarse.

Servidor

El servidor ejecuta el archivo Servidor.java donde se encuentran los siguientes métodos:

crearTablero()

```
public static Celda[][] crearTablero(){
        //Se crean las celdas
        Celda[][] tablero = new Celda[16][16];
        for(int i=0; i<16; i++){
            for(int j=0; j<16; j++){
                tablero[i][j] = new Celda();
                tablero[i][j].x = i;
                tablero[i][j].y = j;
            }
        }
        int x,y;
        //Se colocan las bombas
        for(int i=0; i<40; i++){
            x = (int)(Math.random()*16);
            y = (int)(Math.random()*16);
            if(!tablero[x][y].bomba){
                tablero[x][y].bomba = true;
                //Se calculan los valores adyacentes
                for(int j=x-1; j<=x+1; j++){
                    for(int k=y-1; k<=y+1; k++){
                        if(!(j<0 || j>15 || k<0 || k>15)){
                            if(!tablero[j][k].bomba){
                                tablero[j][k].valor += 1;
                            }
                        }
                    }
```

```
                }
            }else{
                i--;
            }
        }

        return tablero;
    }
```

El método crearTablero() crea un tablero que inicializa las coordenadas de cada celda y coloca bombas aleatoriamente, también calcula los valores adyacentes de cada bomba.

imprimeTablero()

```
public static void imprimeTablero(Celda[][] tablero){
        System.out.println("Imprime tablero");
        for(int i=0; i<16;i++){
            for(int j=0; j<16;j++){
                if(tablero[i][j].bomba) System.out.print(" B ");
                else System.out.print(" " + tablero[i][j].valor + " ");
            }
            System.out.println("");
        }
    }
```

Este método imprime en consola los valores de un tablero.

jugada()

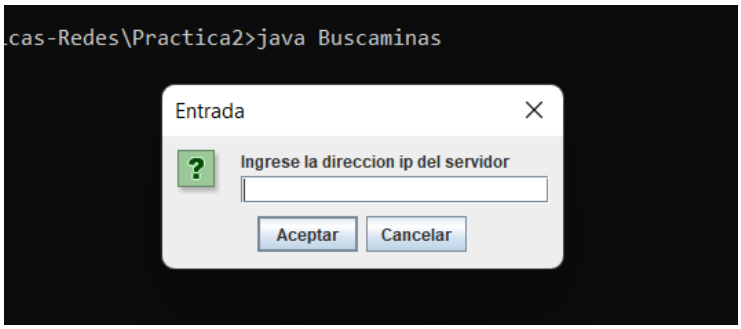
```
public static ArrayList<Celda> jugada(int x, int y, Celda[][] tablero){
        //Objetivo: Enviar celdas que se revelan
        ArrayList<Celda> celdas = new ArrayList<Celda>();
        if(tablero[x][y].bomba){//Si es bomba se envÃa para que pierda
            celdas.add(tablero[x][y]);
        }else{//Si la celda es un numero se valida la jugada
            if(tablero[x][y].valor > 0){//Si es numero se muestra
                celdas.add(tablero[x][y]);
            }else{ //Si es cero se obtienen sus adyacentes
                obtenerAdyacentes(x,y,celdas, tablero);
            }
```

```
        }
        return celdas;
    }
```

El método jugada() crea una lista de celdas para enviar al cliente, recibe los valores (‘x’ y ‘y’) de la coordenada del botón que presionó el cliente, también recibe el tablero. Si la coordenada recibida es una bomba, reenvía la bomba, si es un valor mayor a 0, entonces solo agrega esa celda a la lista, pero si se ha presionado una celda con valor de 0, entonces se llama a la función recursivamente.

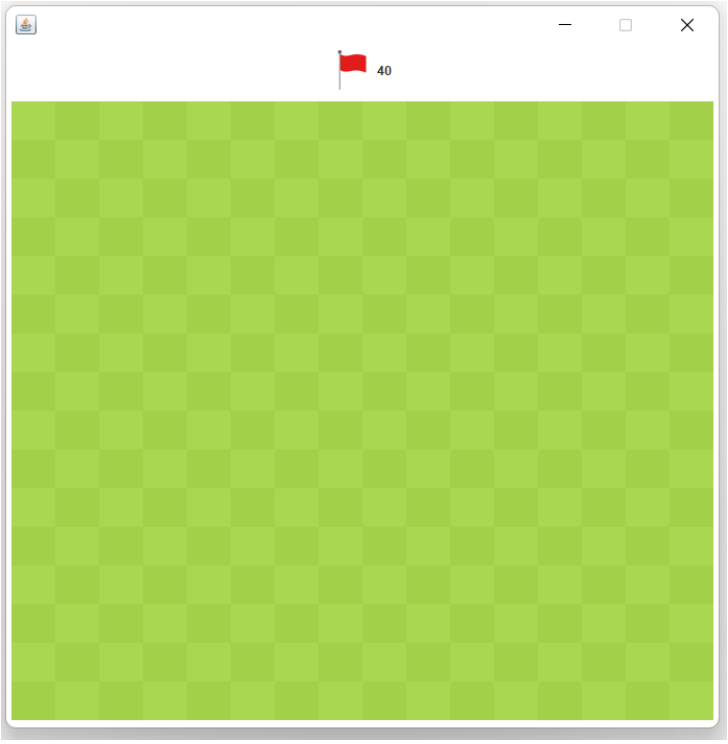
Ejecución

A continuación, se muestra la ejecución del programa, lo primero que se muestra es la ventana para ingresar la ip y el puerto asociado al servidor, es por ello que antes el servidor debe estar listo para recibir datagramas.



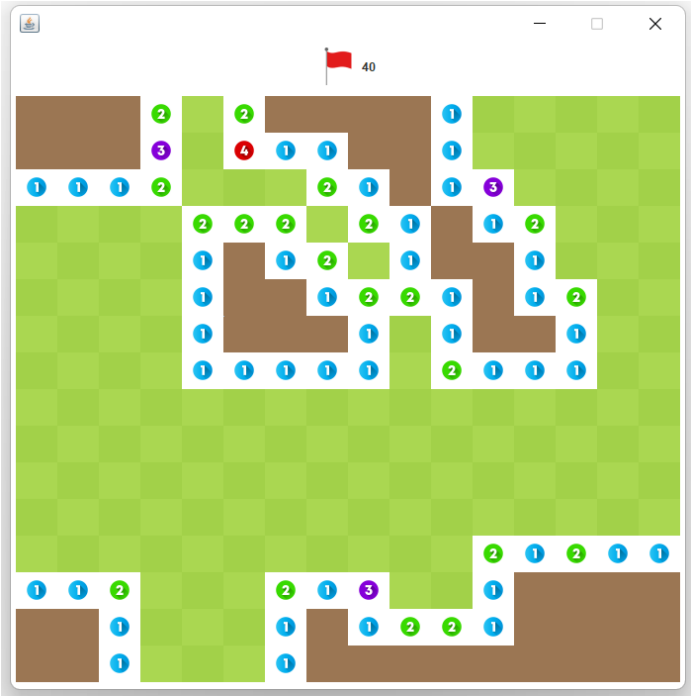
Ventana para ingreso de ip

Posteriormente, se ejecuta la interfaz gráfica del cliente, en el que el cliente podrá seleccionar la celda dando click izquierdo o marcando con una bandera mediante el click derecho.



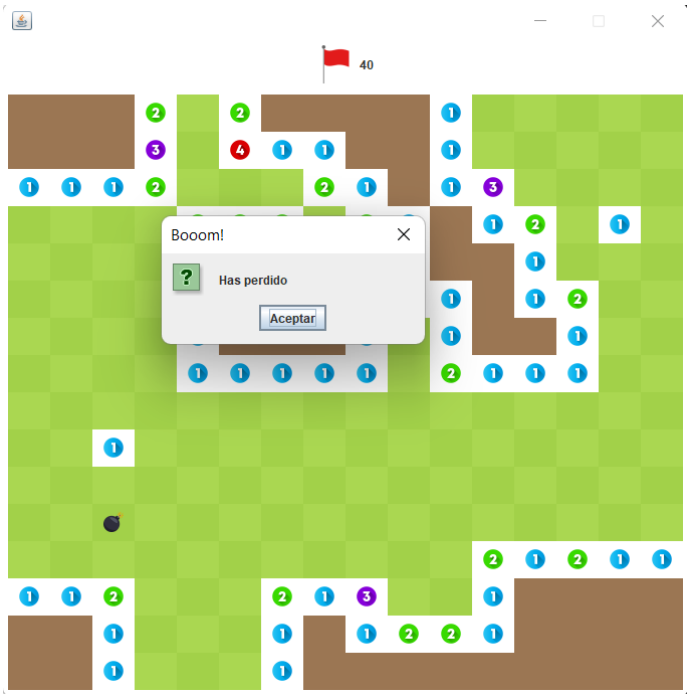
Tablero inicial del juego

De esta manera el cliente puede seguir el desarrollo del juego, tal como se muestra a continuación.



Progreso del juego

Finalmente, el juego termina cuando el cliente selecciona una celda que contiene una mina, por lo que se le muestra una ventana con este aviso.



Fin del juego

Conclusión

Los sockets permiten la comunicación bidireccional entre dos entidades, una cliente y otra servidor, sin embargo, la manera en que los datos se envían varían dependiendo del protocolo utilizado. Para el envío de mensajes mediante el protocolo de transporte UDP, se utilizan sockets de datagramas.

Para ello, en java se usó la clase DatagramSocket para crear un nuevo socket y DatagramPacket para crear paquetes para enviar. En esta práctica la interacción entre cliente y servidor ocurría al seleccionar cada casilla del tablero del buscaminas, en ese momento el cliente enviaba las coordenadas de la casilla y el servidor respondía con una lista de celdas que se procesaban para seguir con la siguiente jugada o terminar la partida.

Referencias

- García Gutiérrez E. (s.f). “Protocolos de interconexión de redes”. (17/03/2022). Recuperado de: https://ocw.unican.es/pluginfile.php/1357/course/section/1682/Tema%203.pdf

- Perez Manuel. (s.f). “Protocolo de transporte: UDP”. (17/03/2022). Recuperado de: http://umh2266.edu.umh.es/wp-content/uploads/sites/197/2013/04/T2_UDP.pdf
- /Díaz Gómez Fernando. (s.f). “Sockets en Java”. (17/03/2022). Recuperado de: https://www.infor.uva.es/~fdiaz/sd/doc/java.net.pdf