

# Multicast | Bully

## Programación Distribuida

Alumnos:

Alan Job de la Luz Hernández

Aviel Aldama Díaz

Asesor:

M.C. Mariano Larios Gómez

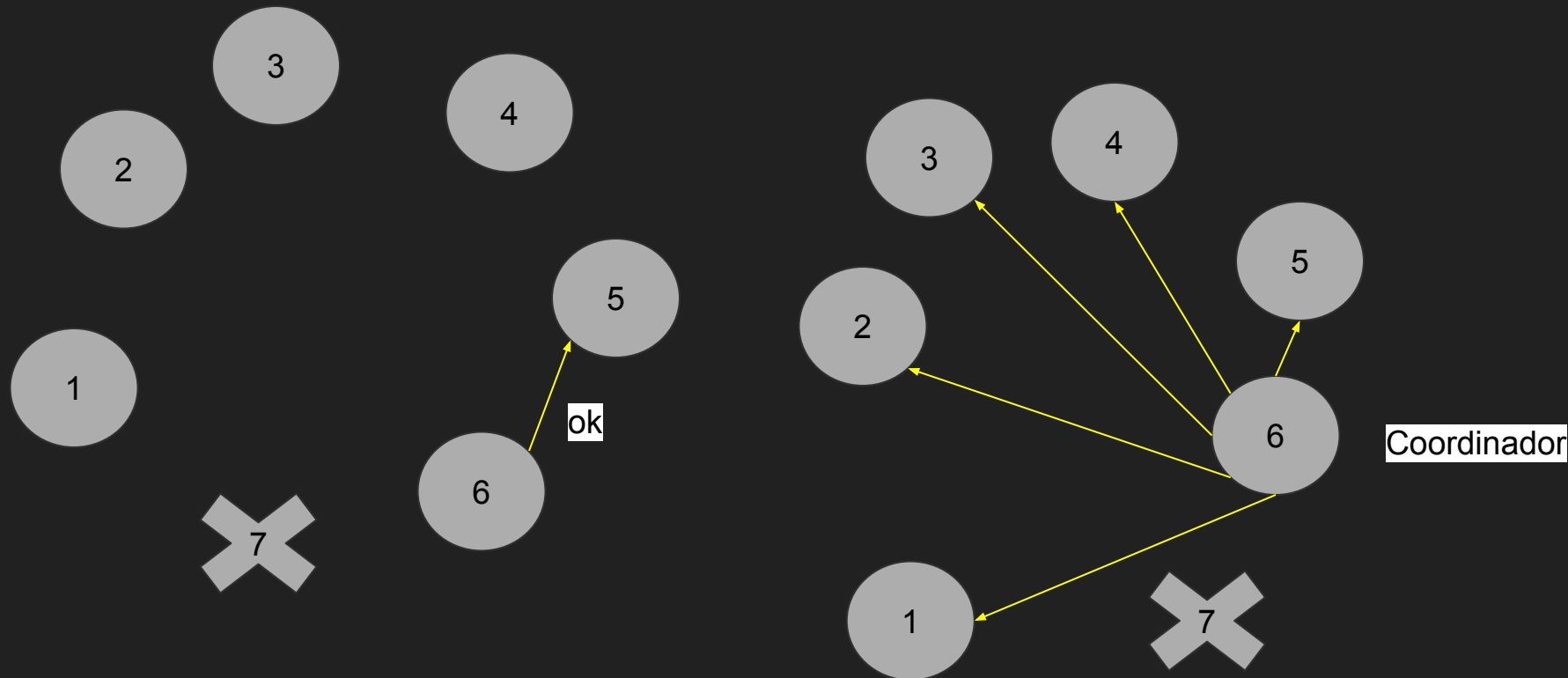
# Bully-Monitoreo

- En una red de varios ordenadores conectados entre sí, donde a cada uno denominaremos Peer.
- Un Peer de la red tiene la función de ser coordinador enviando un mensaje cada cierto tiempo al resto de Peers que no son coordinadores.
- Si un Peer no es Coordinador y detecta que el Coordinador actual fallo inicia un proceso para seleccionar a un nuevo Coordinador

# Bully-Elección

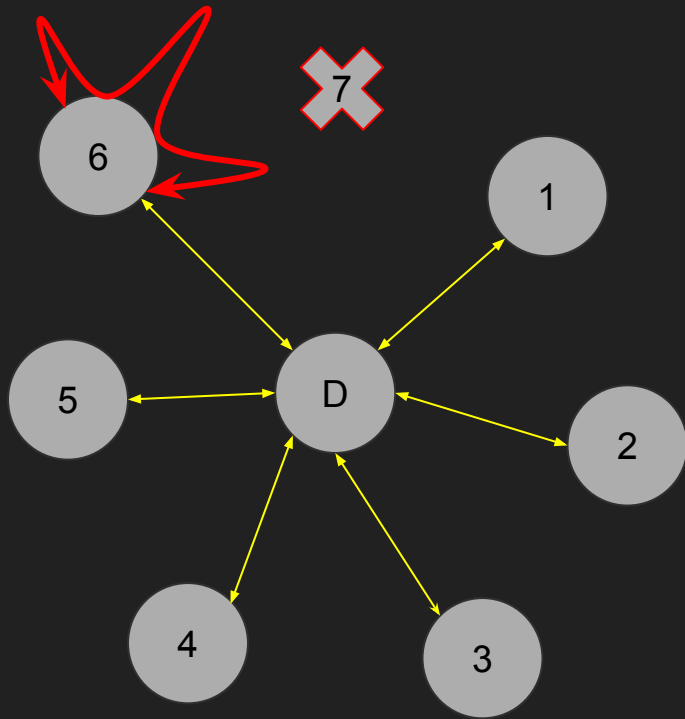
- Un Peer emite un mensaje para iniciar el proceso de elección esperando como respuesta un “ok” confirmando .
- 
- Si el Peer no recibe algún mensaje de confirmación por un tiempo determinado entonces se declara como Coordinador e inicia el proceso de enviar los mensajes cada cierto tiempo notificando el resultado de la elección.
- 
- Si el Peer recibe un mensaje de elección de un Peer con un id mayor este se queda a la espera del nuevo coordinador.
- 
- Si recibe un mensaje de elección de un Peer con un id menor, envía un mensaje de respuesta “ok” e inicia de nuevo el proceso de elección

# Ejemplo

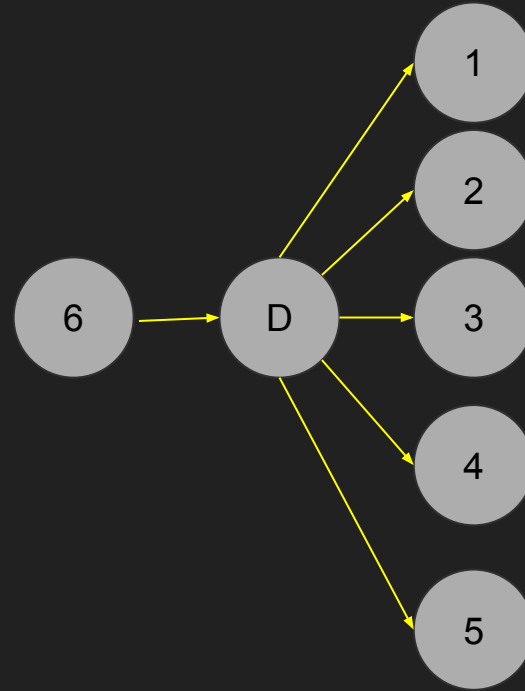


# Bully-Multicast

- Método para la difusión de mensajes a múltiples destinos, se reservan las redes de tipo D para la multidifusión en el rango 224.0.0.0 a 239.255.255.255
- Peer's monitorean al coordinador, en caso de fallo se inicia la elección.
- Los Peer's leen el mensaje de elección con el id que detectó la caída.
- Si leen un mensaje de elección que proviene de un peer con id mayor se quedan a la espera del nuevo coordinador.
- si lee un mensaje de elección con su mismo id este incrementa un contador, para simular tiempo de espera.
- cuando el id recibe más de 2 mensajes con su mismo id, simulando un tiempo de espera de 3 segundos, se declara Coordinador.



## Ejemplo



D - Redes Tipo D en el rango 224.0.0.0 a 239.255.255.255 Multicast

# Demostración NetBeans 8.1

Java jdk 1.8

# Tipos de mensaje-Coordinador

- Mensaje Coordinador, notifica que el coordinador está activo.

```
/**
 * Se envia el mensaje para informar a todos los peer que es el coordinador actual
 * @param id identificador del peer que se declara coordinador.
 */
private void msg_Coordinador(int id){
    byte buffer []= ("Coordinador "+id).getBytes();
    DatagramPacket paquete = new DatagramPacket(buffer,buffer.length,host,port);
    try {
        socket.send(paquete);
    } catch (IOException ex) {
        Logger.getLogger(Peer.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

- Se envía la cadena de caracteres “Coordinador” + id del mismo



# Tipos de mensaje-Elección

- Se envía una cadena de caracteres “Elección” + id del Peer

```
/**
 * Se envia el mensaje para el proceso de eleccion a todos los peer informando
 * que es un posible candidato a coordinador
 * @param id identificador del peer que pretende ser coordinador
 */
private void msg_Eleccion(int id){
    byte buffer[] = ("Eleccion "+id).getBytes();
    DatagramPacket paquete = new DatagramPacket(buffer,buffer.length,host,port);
    try {
        socket.send(paquete);
    } catch (IOException ex) {
        Logger.getLogger(Peer.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

## Envío/Coordinador

- Se utiliza el método para enviar el mensaje coordinador a los peer que no son coordinadores.

```
if(Coordinador){  
    //System.out.println("Soy el coordinador "+id);  
    msg_Coordinador(id);  
    mensaje+="\nSoy el coordinador "+id+"\n";  
    if(ar1!=null)  
        ar1.setText(mensaje);  
}
```

# Recepción/No Coordinador

- Recibimos los mensajes del grupo multicast al que estamos anexados
- El conjunto de bytes recibidos serán convertidos a una cadena que dividiremos en su valor entero y el contexto del mensaje que analizaremos mensaje Coordinador/Elección

```
if(!Coordinador){
    try {

        byte buffer[] = new byte[20];
        paquete = new DatagramPacket(buffer, buffer.length);
        socket.setSoTimeout(3000);
        socket.receive(paquete);

        pack = to_Split_Datagram(paquete.getData());

        String msg = String.valueOf(pack.get(0));
        int id_rec = Integer.parseInt(String.valueOf(pack.get(1)));
    }
}
```

# Coordinador Activo

- Si el tipo de mensaje es la cadena “Coordinador” simplemente mostramos el id del Peer que es coordinador y el id del peer que lo recibe.

```
if(msg.equalsIgnoreCase("Coordinador")){  
    Elector_lock=false;  
    //System.out.println("El coordinador actual es: "+id_rec+" soy: "+id);  
    mensaje+=("El coordinador actual es: "+id_rec+" soy: "+id+"\n");  
    if(arl!=null)  
        arl.setText(mensaje);  
}
```

# Caída del Coordinador

- Si en un tiempo de espera determinado no se recibe mensaje del Coordinador, Capturamos la excepción y si el peer aun es candidato al proceso de elección

```
    } catch (IOException ex) {  
  
        if(!Elector_lock){  
  
            //System.out.println("Se envia mensaje eleccion: "+id);  
            msg_Eleccion(id);  
            mensaje+="Se envia mensaje eleccion: "+id+"\n";  
            if(arl!=null)  
                arl.setText(mensaje);  
        }  
    }
```

- Puede no ser candidato por estar a la espera del Coordinador mientras ya fue iniciado el proceso de elección por otro peer y este recibió un id mayor

# Proceso de Elección / Bully

- Si el tipo de mensaje es la cadena "Eleccion"
- Cuando el id que recibimos resulta ser mayor al del peer actual bloqueamos el proceso de elección, a la espera de un nuevo coordinador.

```
if(msg.equalsIgnoreCase("Eleccion")){  
    if(count>2){  
        Coordinador=true;  
        //System.out.println("Count: "+count+" en: "+id+"Coordinador: "+Coordinador);  
        mensaje+=("Count: "+count+" en: "+id+"Coordinador: "+Coordinador);  
        if(ar1!=null)  
            ar1.setText(mensaje);  
    }  
    if(id<id_rec){  
        Elector_lock=true;  
    }  
    if(id==id_rec){  
        count++;  
    }  
}
```

- Si a la escucha del multicast el id recibido es igual al actual incrementamos un contador simulando el tiempo de espera 1s por cada mensaje al ser mayor a 2 el peer actual se declara como el nuevo coordinador.

# Fase Experimental

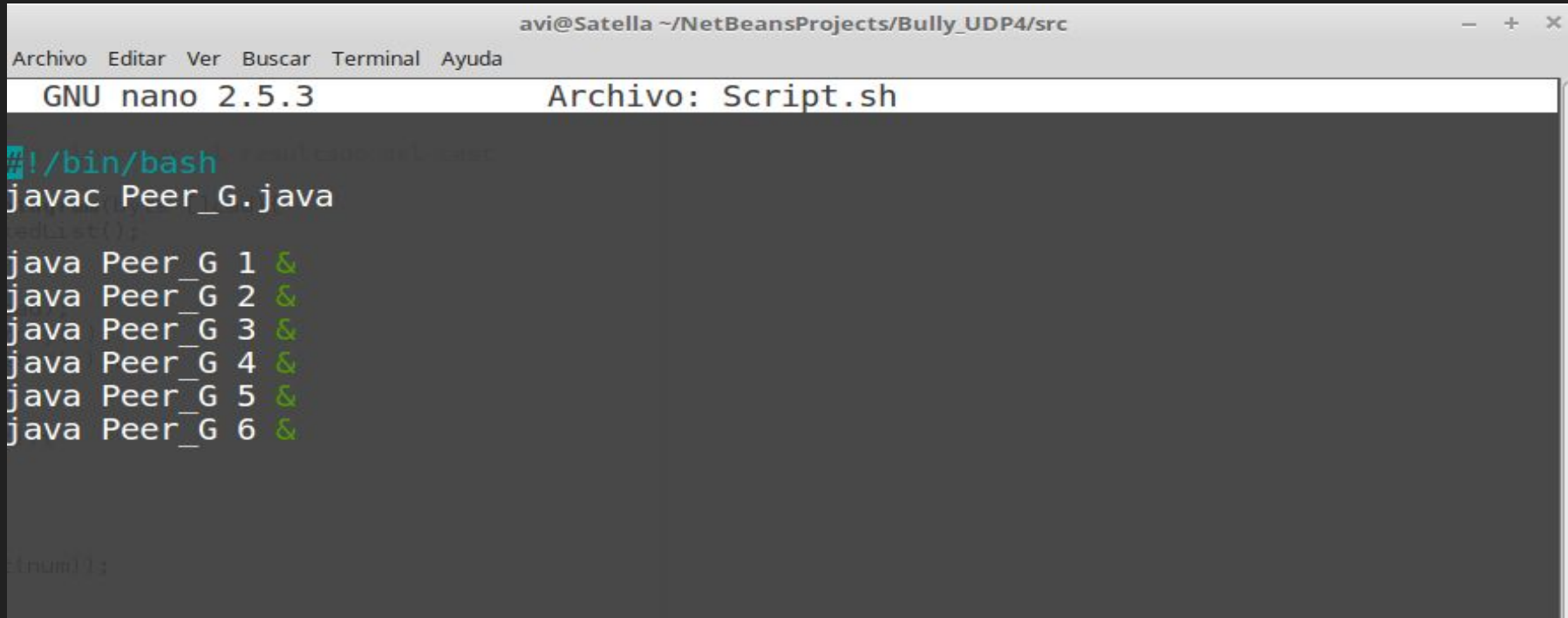
## GNU/Linux(Linux Mint Sarah)

# Preparando El Entorno Experimental

- Creamos un archivo en el entorno GNU/Linux mediante el intérprete de comandos bash .
- El Script estará encargado de compilar las clases .java y ejecutar el archivo Peer\_G agregando un argumento del tipo entero para el id del peer.
- Consecutivamente será acompañado de un símbolo “&” ampersand, en entornos Linux este símbolo se usa para que un proceso, al momento de ser lanzado se vuelvan daemons y sean independientes del proceso padre que los crea, de esta forma podemos cerrar cada peer una vez ejecutado para simular la caída del coordinador.
- Deshabilitar Firewall y Conectarse a una red con un Router para el direccionamiento Multicast.



# Script - Bash



The screenshot shows a NetBeans IDE window with a terminal pane. The title bar indicates the user is 'avi@Satella' and the current project is '~/NetBeansProjects/Bully\_UDP4/src'. The terminal window is titled 'GNU nano 2.5.3' and 'Archivo: Script.sh'. The script content is as follows:

```
#!/bin/bash
javac Peer_G.java
addList();
java Peer_G 1 &
java Peer_G 2 &
java Peer_G 3 &
java Peer_G 4 &
java Peer_G 5 &
java Peer_G 6 &
```

The terminal shows the script being executed, with the prompt 'avi@Satella' visible at the bottom left.

# Clases \*.java

- Script.sh: Script para lanzar el entorno experimental de los Peer.
- Peer.java: Clase en la que se encuentra la implementacion del algoritmo.
- Peer\_G.java: Clase generada por NetBeans 8.1 como controlador para el entorno grafico.
- Peer\_G.form: archivo XML encargado de la vista sobre el entorno grafico

# Clases

avi@Satella ~/NetBeansProjects/Bully\_UDP4/src

Archivo Editar Ver Buscar Terminal Ayuda

```
avi@Satella src $ ls
bully.sh  Peer_G.form  Peer_G.java  Peer.java  Script.sh
avi@Satella src $
```

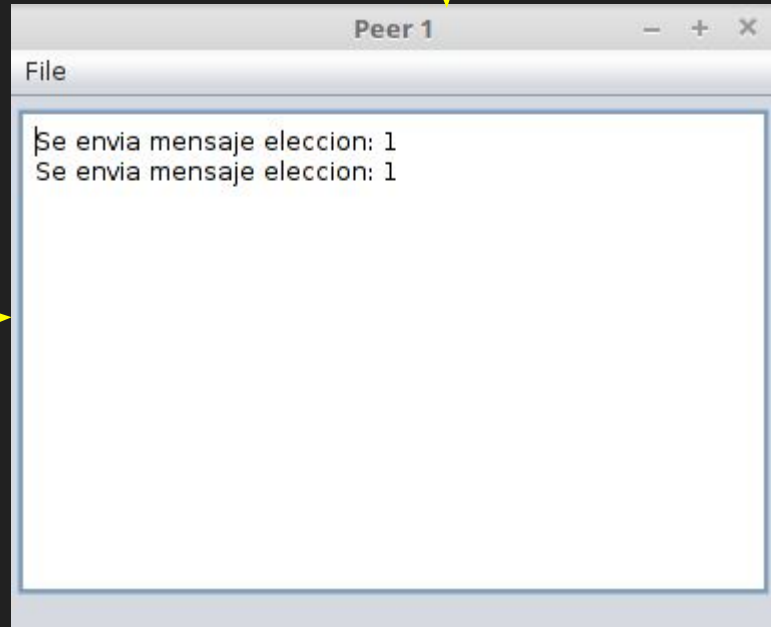
```
Formacion del datagrama
Resultado de bytes
Mediante un hilo para almacenar el resultado del cast
```

```
list to Split Datagram(byte [][]cad){
    lista= new LinkedList();
    a = 0;
    for (int i=0; i<cad.length; i++){
        na=new String(cad[i]);
        cadena.toCharArray();
        if (Character.isAlphabetic(c[i])){
            lista.add(na);
        }
    }
}
```

# Interfaz Gráfica

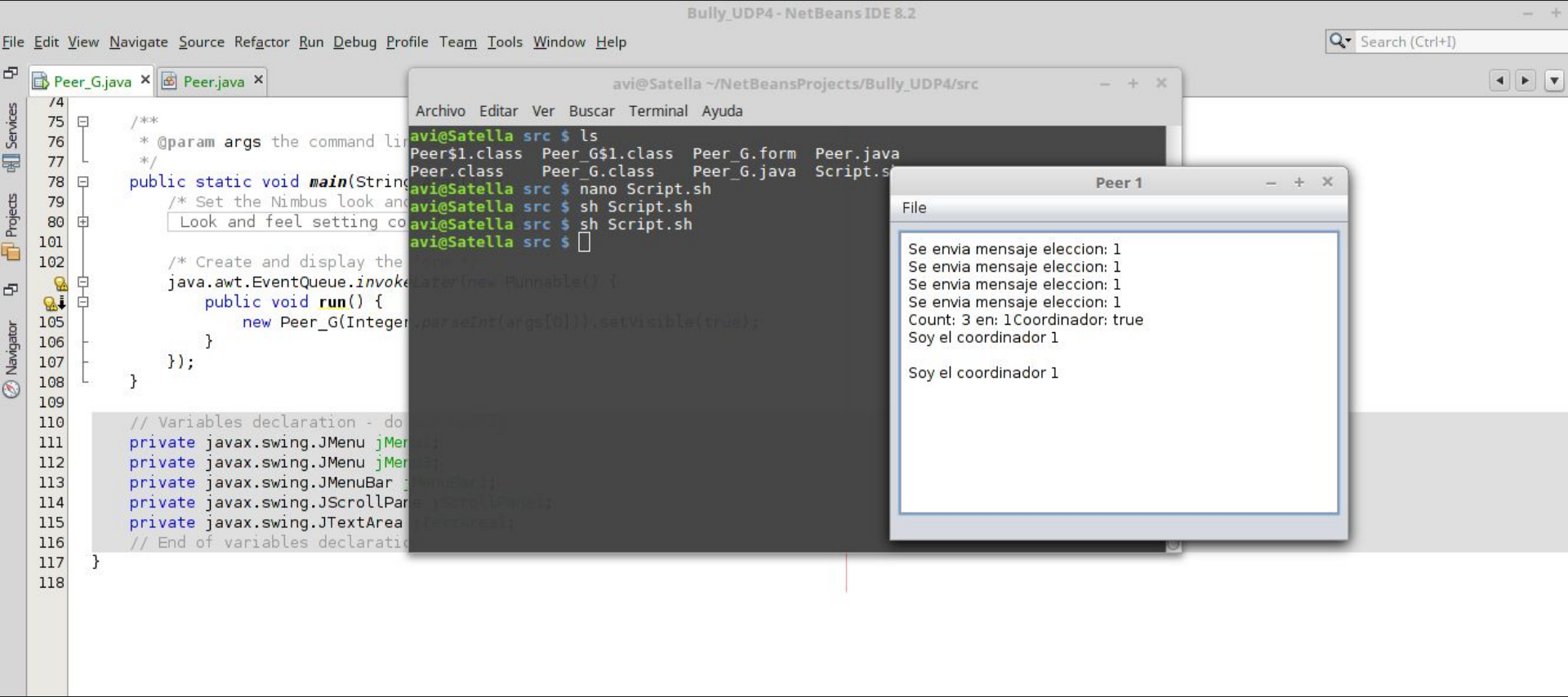
ID del Peer

Terminar Proceso



Estado actual  
Monitoreo Coordinador  
Proceso de Elección

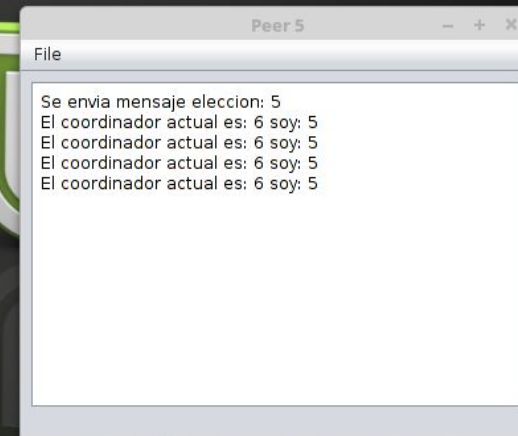
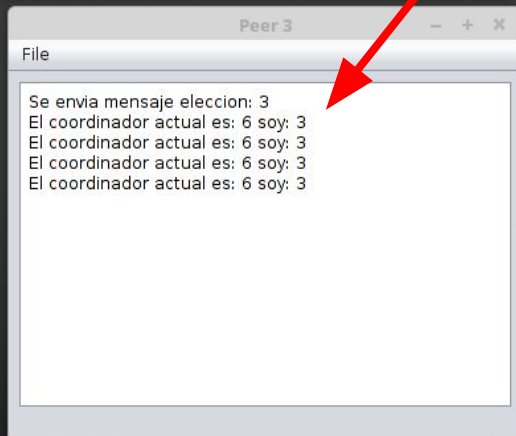
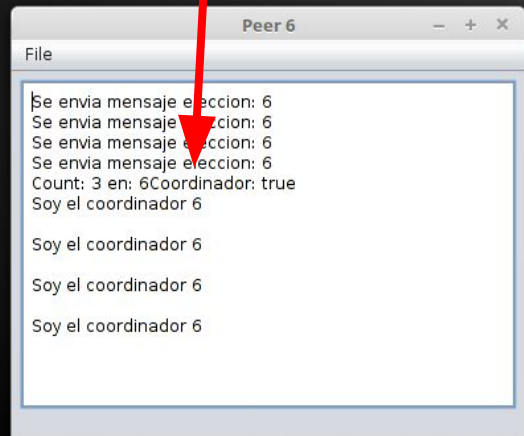
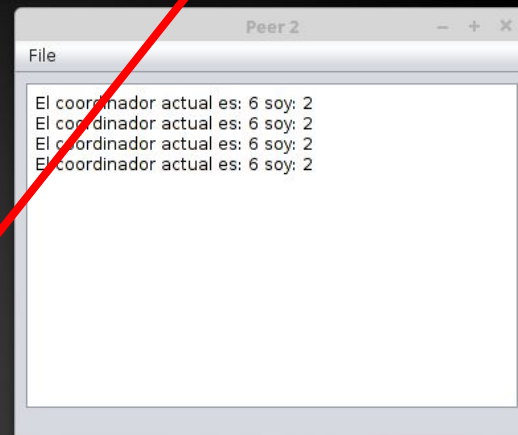
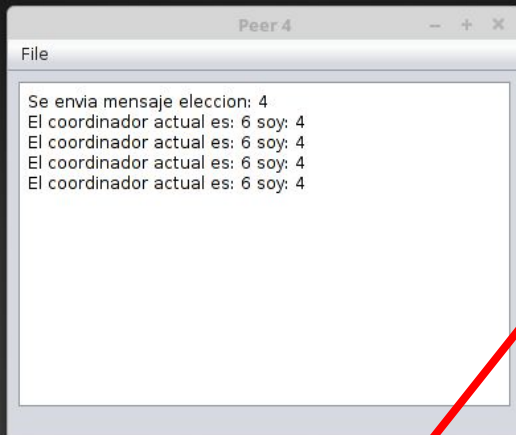
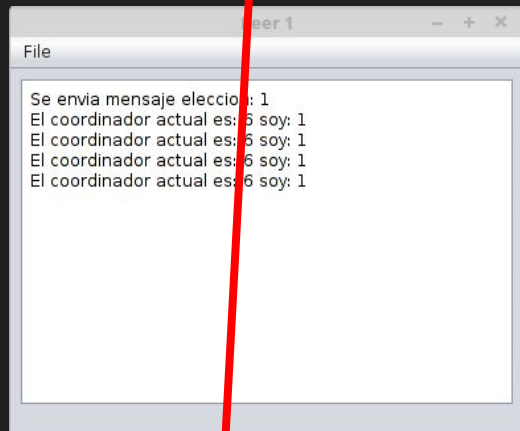
# Caso Trivial



Coordinador inicial

# Prueba 6 Peers

Monitoreo al Coordinador



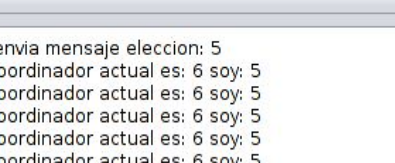
# Simulando Fallo de Peer 6

[illegible]A screenshot of a terminal window titled "Peer 4". The window has standard OS controls (minimize, maximize, close) in the top right corner. Below the title bar is a menu bar with the word "File". The main area of the terminal displays a series of messages:

Se envia mensaje eleccion: 4  
El coordinador actual es: 6 soy: 4  
El coordinador actual es: 6 soy: 4  
El coordinador actual es: 6 soy: 4  
El coordinador actual es: 6 soy: 4  
El coordinador actual es: 6 soy: 4  
El coordinador actual es: 6 soy: 4  
El coordinador actual es: 6 soy: 4  
El coordinador actual es: 6 soy: 4  
El coordinador actual es: 6 soy: 4  
El coordinador actual es: 6 soy: 4  
El coordinador actual es: 6 soy: 4  
El coordinador actual es: 6 soy: 4  
El coordinador actual es: 6 soy: 4  
Se envia mensaje eleccion: 4

A large red arrow points upwards towards the bottom of the message list.A screenshot of a terminal window titled "Peer 2". The window has standard OS controls (minimize, maximize, close) in the top right corner. A menu bar at the top left shows the word "File". The main area of the terminal contains ten identical lines of text:

El coordinador actual es: 6 soy: 2

The text is displayed in a monospaced font, typical of code editors or terminals.[illegible]

Peer 5

File

Se envia mensaje eleccion: 5

El coordinador actual es: 6 soy: 5

El coordinador actual es: 6 soy: 5

El coordinador actual es: 6 soy: 5

El coordinador actual es: 6 soy: 5

El coordinador actual es: 6 soy: 5

El coordinador actual es: 6 soy: 5

El coordinador actual es: 6 soy: 5

El coordinador actual es: 6 soy: 5

El coordinador actual es: 6 soy: 5

El coordinador actual es: 6 soy: 5

El coordinador actual es: 6 soy: 5

Se envia mensaje eleccion: 5

Se envia mensaje eleccion: 5

Peer 6 Fallo

## Inicio del proceso de Elección

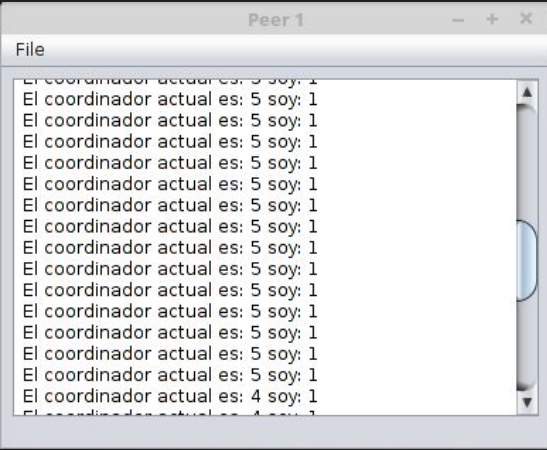


de 2016-11-15 01-26-

## Nuevo Coordinador Seleccionado



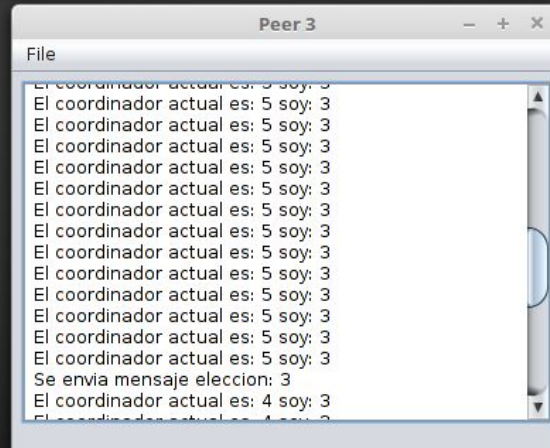
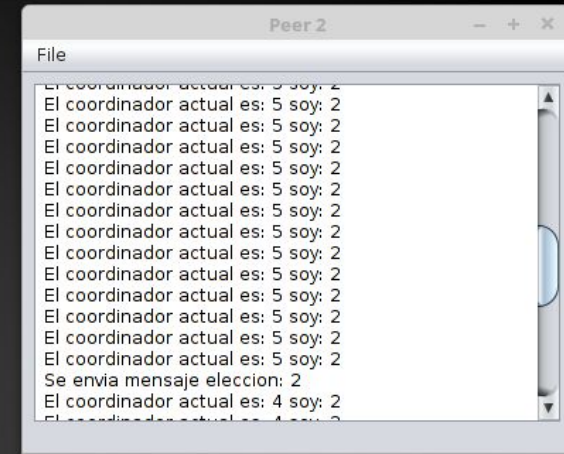
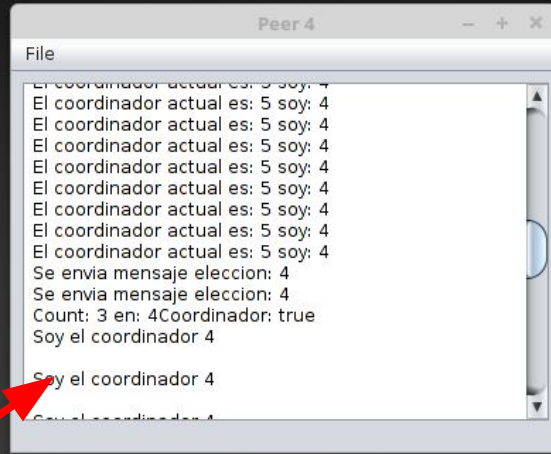
## Simulando una segunda caída



de 2016-11-15 01-26-41.png

Se selecciona un nuevo Coordinador

nuevo



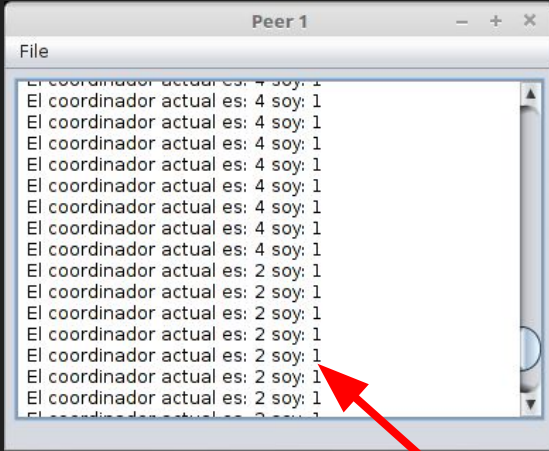
# Una caída no Significativa



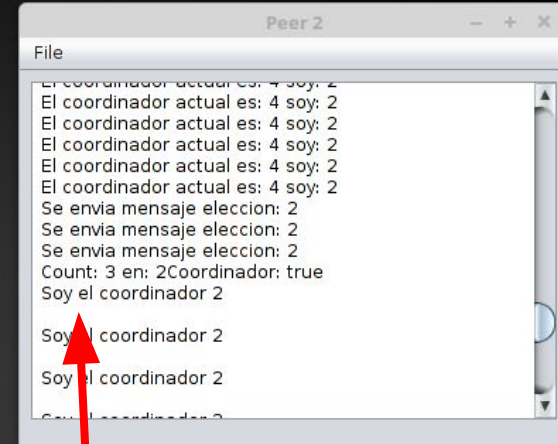
No se inicia ningún proceso de Elección ya que el Coordinador continua activo, el Peer 3 ahora ausente no influye en nuestro entorno.



# Finalizando la Fase Experimental



```
File
El coordinador actual es: 4 soy: 1
El coordinador actual es: 4 soy: 1
El coordinador actual es: 4 soy: 1
El coordinador actual es: 4 soy: 1
El coordinador actual es: 4 soy: 1
El coordinador actual es: 4 soy: 1
El coordinador actual es: 4 soy: 1
El coordinador actual es: 4 soy: 1
El coordinador actual es: 4 soy: 1
El coordinador actual es: 2 soy: 1
El coordinador actual es: 2 soy: 1
El coordinador actual es: 2 soy: 1
El coordinador actual es: 2 soy: 1
El coordinador actual es: 2 soy: 1
El coordinador actual es: 2 soy: 1
El coordinador actual es: 2 soy: 1
El coordinador actual es: 2 soy: 1
El coordinador actual es: 2 soy: 1
El coordinador actual es: 2 soy: 1
```



```
File
El coordinador actual es: 4 soy: 2
El coordinador actual es: 4 soy: 2
El coordinador actual es: 4 soy: 2
El coordinador actual es: 4 soy: 2
El coordinador actual es: 4 soy: 2
El coordinador actual es: 4 soy: 2
El coordinador actual es: 4 soy: 2
El coordinador actual es: 4 soy: 2
El coordinador actual es: 4 soy: 2
Se envia mensaje eleccion: 2
Se envia mensaje eleccion: 2
Count: 3 en: 2Coordinador: true
Soy el coordinador 2
Soy el coordinador 2
Soy el coordinador 2
```

Se analiza el comportamiento del algoritmo en esta simulación con 6 Peer hasta el caso previo al trivial, donde se continúan respetando los principios de funcionamiento

# Conclusiones

- El algoritmo resulta ser eficaz.
- La cantidad de mensajes usados para el proceso de elección puede ser hasta  $O(n^2)$  o en el mejor caso donde el Peer con el id inmediato más bajo al coordinador detecta el fallo  $O(n)$ .
- Si un peer con id anteriormente coordinador se recupera, este no reclama los derechos de ser coordinador, pero si es participe en un siguiente proceso de elección.

# Referencias Bibliográficas

1. George Coulouris, Jean Dollimore, Tim Kindberg, Gordon Blair. (2012). DISTRIBUTED SYSTEMS. United States of America: Addison-Wesley.
2. J. Lopez. (2003). Leyendo y escribiendo en un UDPSocket. 7-11-2016, de udec Sitio web:  
<http://www.inf.udec.cl/~jlopez/DSWR/LABORATORIOS/javaudpsocket.html>