Universidad Autónoma Metropolitana
Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licienciatura en Ingeniería en Computación

Sistema para la visualización gráfica de la topología de una red de computadoras

Proyecto que presenta:

Jorge Alberto Ozuna Cruz

para obtener el titulo de:

Ingeniero en Computación

Asesor:

M. en C. Oscar Alvarado Nava

México, D.F. **Abril de 2014**

Resumen

El objetivo de este proyecto, fue desarrollar una aplicación que permitiera descubrir y dibujar la topología de una red de área local. Para lograrlo se utilizó el protocolo SNMP y el lenguaje de programación Java.

El protocolo SNMP es soportado por la gran mayoría de los dispositivos utilizados actualmente en las redes de computadoras, la principal dificultad con este método de descubrimiento y dibujado, es que este protocolo debe ser instalado y configurado en cada dispositivo existente dentro de la red, de lo contrario no podrá ser detectado y por lo tanto no se mostrara en la topología resultante.

Una vez configurado el protocolo de manera correcta, basta proveer a la aplicación con las direcciónes IP que se desea dibujar y este desplegará la estructura e información de la red.

Agradecimientos

- A la División de Ciencias Básicas e Ingeniería de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco.

- Al Departamento de Sistemas y al Departamento Electrónica.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Planteamiento del problema

Una red es un conjunto de computadoras separadas fisicamente pero interconectadas[1] mediante dispositivos como switches y routers, con el propósito de intercambiar y compartir información, recursos o servicios.

Actualmente las redes de comunicación han tenido un rápido desarrollo y se han convertido en el núcleo de casi todos los sistemas productivos de la sociedad, especialmente en aquellos que demandan de una gran eficiencia en el almacenamiento, transmisión y acceso seguro de la información, empresas como bancos, aerolíneas, hospitales, universidades y estancias de gobierno transfieren grandes cantidades de información a diario. Para satisfacer esta demanda son necesarios una gran cantidad de dispositivos de diferentes tipos, con funciones diferentes y deben tener la capacidad de comunicarse entre sí para poder realizar todas las tareas necesarias.

En la actualidad cualquier tipo de organización cuenta con un gran numero de dispositivos en funcionamiento, el crecimiento de la organización conlleva al crecimiento de la red, ocasionando que no siempre se cuente con una documentación apropiada acerca de la manera en que esta se encuentra construida, provocando que el detectar problemas y vulnerabilidades dentro de la red sea una tarea complicada y tome mucho tiempo pudiendo provocar pérdidas económicas a la empresa.

La topología de una red refleja las características básicas de esta, por lo que conocerla puede ayudar al administrador a realizar su trabajo de manera más conveniente, pues el conocer la estructura le permitirá detectar problemas de manera rápida y sencilla, esto convierte a la topología de redes en una de las bases más importantes dentro de la administración. En la figura 1.1 se muestran dos ejemplos de redes de diferente complejidad.

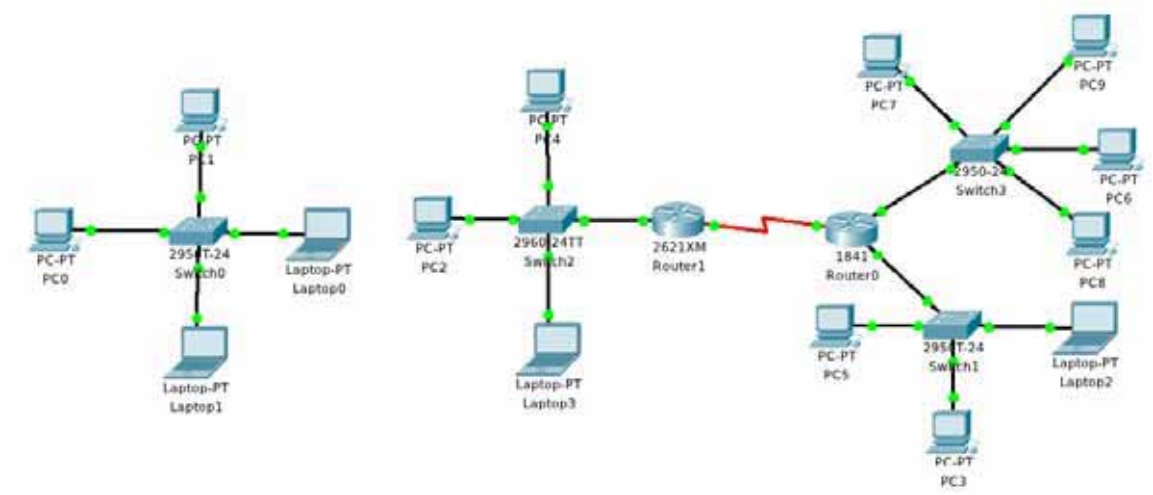

Figura 1.1: Ejemplos de redes.

En la figura de la izquierda, se puede observar una pequeña red compuesta por cuatro computadoras, interconectadas mediante un Switch. En la figura de la derecha hay tres redes distintas, conectadas por medio de dos routers.

1.2. Marco Teorico

Las computadoras fueron creadas con el propósito de realizar cálculos de una manera mas rápida y precisa de lo que los humanos somos capaces, en un principio eran dispositivos que ocupaban grandes volumenes y eran muy costosas, las empresas y otras instituciones como universidades sólo contaban con una o dos computadoras y cada una de ellas era independiente.

Con el rápido avance tecnológico, las computadoras se hicieron mucho mas pequeñas y poderosas, en cuanto a capacidad de procesamiento y almacenamiento, y fue posible la producción de millones de estas. También creció la necesidad de obtener, procesar y enviar información.

Las primeras redes de computadoras fueron creadas con el propósito de compartir el uso de otro dispositivo, por ejemplo, poder utilizar una misma impresora desde múltiples computadoras, reduciendo costos para la empresa. Posteriormente fue posible compartir los recursos[1] de las computadoras, o que una de ellas prestara un servicio [2] a otras.

Este desarrollo fue continuo hasta el punto en que hoy lo conocemos, las redes de computadoras son la base de distintos tipos de organizaciones cuyas redes abarcan miles de kilómetros y sin embargo en mili-segundos una computadora es capaz de comunicarse con otra que se encuentra fisicamente muy separada.

La demanda de comunicación conllevo a la creación de dispositivos especializados en permitir la comunicación entre diferentes computadoras, todos estos dispositivos, como un switch[3] o un router[4], simplemente son sistemas de computo dentro de la red, la diferencia es que cuentan con un hardware[5] y un sistema operativo[6] distintos a los de las computadoras tradicionales.

Para mantener en funcionamiento estos dispositivos y por tanto a una red, se necesitan especialistas que conozcan el hardware y software de los distintos dispositivos que la componen, se encarguen de monitorearlos, configurarlos y mantenerlos para asegurar su correcto funcionamiento, así como administrar servicios, cuentas de usuarios, etc. Estas personas son los administradores de red.

---

[1] Espacio de almacenamiento, capacidad de procesamiento, archivos, aplicaciones, etc.

[2] Programa que ejecuta una tarea en beneficio de otro proceso llamado cliente.

[3] Un conmutador o switch es un dispositivo digital lógico de interconexión de equipos que opera en la capa de enlace de datos del modelo OSI.

[4] También conocido enrutador o encaminador de paquetes, es un dispositivo que proporciona conectividad a nivel de red o nivel tres en el modelo OSI

[5] Componentes electrónicos y mecánicos de un dispositivo.

[6] Programa encargado de gestionar el hardware de un dispositivo.

La topología de una red, es la forma en que se encuentran conectados los distintos dispositivos que la componen, es decir, la forma en que se encuentra diseñada física o lógicamente. Por esto conocer la topología permitirá al administrador proporcionar una mejor configuración y administración a la red, lo cual podría verse reflejado en un mejor funcionamiento de esta[2].

Junto con la creación de dispositivos y sistemas operativos especializados en garantizar la comunicación, se crearon también modelos y protocolos de comunicación con el fin de estandarizar la tecnología y evitar que distintos fabricantes crearan los propios lo cual a futuro ocasionaría una incompatibilidad entre los distintos protocolos y por lo tanto no seria posible la comunicación.

Un modelo es la representación de pequeños procesos que pertenecen a un proceso mayor, su función es permitir el análisis de la interacción entre estos procesos, con el fin de lograr una relación que les permita cumplir su función dentro del proceso mayor. El modelo teórico más conocido dentro de las redes es el modelo OSI (*Open System Interconnection*), fue un primer paso hacia la estandarización internacional de los protocolos utilizados en varias capas[1], este modelo de red descriptivo divide en siete capas, el proceso de transmisión de la información entre equipos informáticos, cada capa se encarga de ejecutar una determinada parte del proceso global para permitir la interconexion y comunicación de los dispositivos.

Un protocolo de comunicación es un conjunto de reglas, normas o procedimientos cuyo fin es estandarizar la secuencia de mensajes que ocurren durante la comunicación entre dispositivos de la red. Uno de los múltiples protocolos es el SNMP (Simple Network Management Protocol) se encuentra en la capa de aplicación[7] y permite el intercambio de información de administración entre dispositivos de red, permitiendo al administrador supervisar el funcionamiento de la red, detectar y solucionar problemas dentro de ella.[3]

1.3. Objetivo general

Diseñar e implementar un sistema que permita descubrir y mostrar la topología de la red que se está administrando, basado en el protocolo SNMP.

1.4. Objetivos específicos

- Programar el módulo encargado de obtener la información necesaria para descubrir la topología de red.
- Implementar el algoritmo encargado de descubrir la topología de la red.

[7]Capa que proporciona la interfaz y servicios para las herramientas que ve el usuario.

- Diseñar e implementar el módulo encargado de dibujar la topología.

Capítulo 2

Desarrollo

2.1. Dispositivos

Un dispositivo de red se encarga de cumplir determinadas tareas con el fin de permitir la comunicación entre computadoras. Dentro de las redes de computadoras los dispositivos más comunes son los switches y los routers cada uno cumple con funciones distintas y la comunicación entre estos permite la realización de diversas tareas y el intercambio de grandes cantidades de información entre computadoras.

Dentro del proyecto, cada dispositivo de la red sera representado mediante un objeto, definido por la clase mostrada en la Figura 2.1.

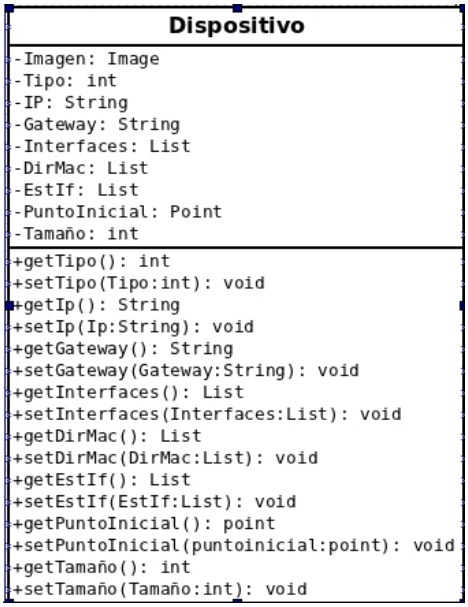

Figura 2.1: Clases de dispositivos.

2.1.1. Computadoras

La computadora es el dispositivo más común dentro de una red y con el que el usuario tiene contacto, actualmente son indispensables para realizar muchas tareas de la vida diaria. La definición exacta del término computadora es: "Máquina electrónica capaz de recibir, procesar y devolver resultados en torno a determinados datos y que para realizar esta tarea cuenta con un medio de entrada y uno de salida".

Una computadora será representada por el icono mostrado en la Figura 2.2.

Figura 2.2: Representación de una Computadora.

2.1.2. Switch

Un switch es un dispositivo para la interconexión de computadoras, trabaja en la capa de enlace de enlace de datos[1] del modelo OSI. La principal función de un switch es interconectar varios segmentos de red permitiendo que se comporten como una sola red y por tanto permitiendo la comunicación entre todos los dispositivos.

Esta función se logra gracias a que los switches tienen la capacidad de "aprender" y almacenar la dirección MAC (*Media Access Control*) de los dispositivos a los que puede llegar desde cada uno de sus puertos. Adicionalmente funcionan como un filtro dentro de la red, permitiendo mejorar el rendimiento y seguridad de esta.[1]

Los switches serán representados por el icono mostrado en la Figura 2.3.

Figura 2.3: Representación de un Switch.

[1]Capa encargada de lograr una comunicación confiable y eficiente entre dos dispositivos.

2.1.3. Routers

El Router es un dispositivo que trabaja en la capa de red,[2] este dispositivo de encarga de proporcionar comunicación entre diferentes subredes, encaminando los paquetes de datos de una red a otra por medio de la ruta mas adecuada. Esta ruta se decide en función de la tabla de enrutamiento[3] y esta a su vez es creada por medio de protocolos que deciden cual es la ruta mas corta.[1]

Los Routers serán representados por el icono mostrado en la figura 2.4.

Figura 2.4: Representación de un Router.

2.2. SNMP

El protocolo SNMP (*Simple Network Management Protocol*) es utilizado para supervisar y administrar el funcionamiento de una red. Una red que es administrada por medio de este protocolo cuenta con tres componentes:

- Dispositivos administrados. Dispositivos pertenecientes a la red administrada que contienen un agente SNMP, el cual se encarga de recolectar la información y proporcionarla al sistema administrador de red.
- Agente. Módulo de software de administración de red que se encuentra en los dispositivos administrados, posee un conocimiento local de información de administración, cuando esta información es solicitada se traduce a un formato compatible con SNMP para ser enviado a través de la red de área local.
- Sistema Administrador de Red. Sistema encargado de ejecutar aplicaciones que supervisan y controlan la información de los dispositivos administrados.

[2]Capa encargada de proporcionar conectividad y seleccionar la mejor ruta para conseguir que los datos lleguen del origen al destino.

[3]Tabla que almacena las rutas para llegar a los nodos de una red.

2.2.1. MIB's

Un MIB (*Management Information Base*) o base de información administrada, es una colección de información que está organizada jerárquicamente en forma de árbol, (Figura 2.5) contienen características específicas de un dispositivo administrado almacenados es variables tales como un entero, una cadena una dirección ip, etc.

Por medio del protocolo SNMP se puede acceder a los MIB's, así se recolecta la información de cada dispositivo de la topología, se analiza y es posible conocer la relación de los dispositivos dentro de la red.

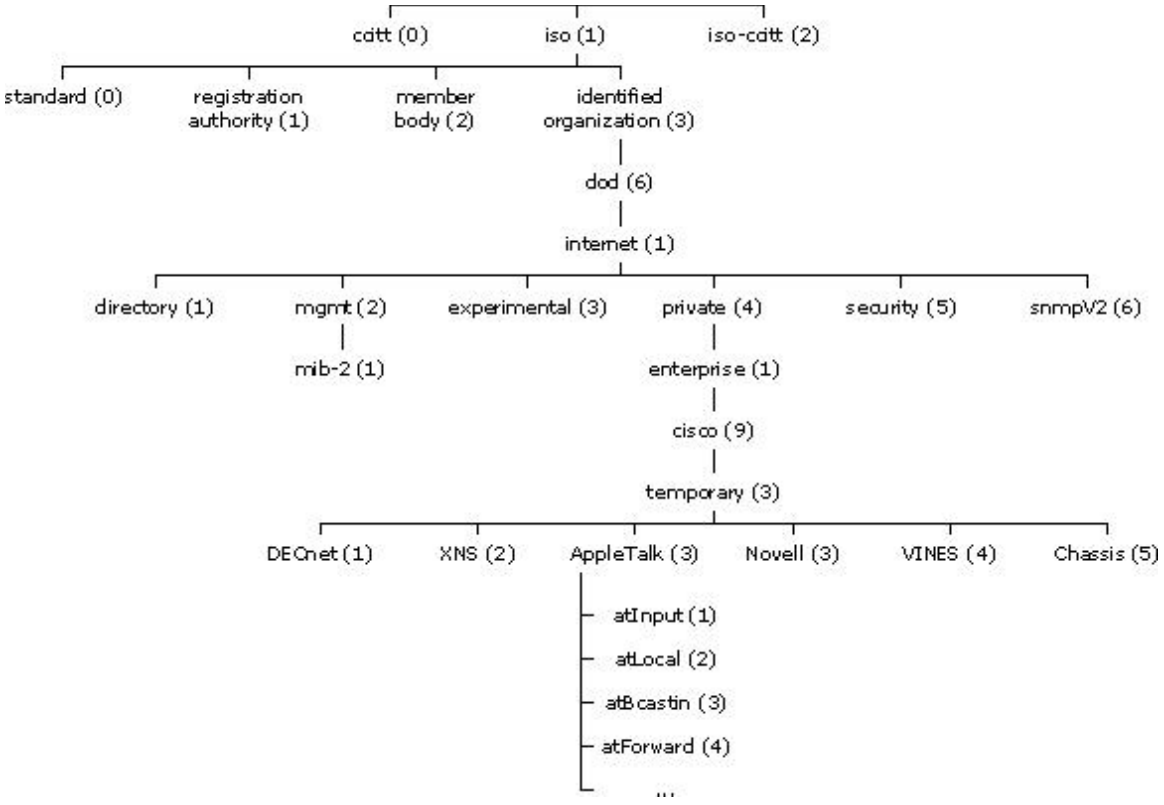

Figura 2.5: Jerarquia de los MIB's.

2.2.2. Instalación y configuración de SNMP

Computadoras

La instalación de SNMP se realiza mediante el siguiente comando.

```
root@lubuntu:/home/lubuntu# apt-get install snmp snmpd
```

El sistema nos indicará el espacio a utilizar en el disco dura para la instalación, se acepta y la instalación prosigue.

```
root@lubuntu:/home/lubuntu# apt-get install snmp snmpd
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes extras:
  libsensors4 libsnmp-base libsnmp15
Paquetes sugeridos:
  lm-sensors snmp-mibs-downloader
Se instalarán los siguientes paquetes NUEVOS:
  libsensors4 libsnmp-base libsnmp15 snmp snmpd
0 actualizados, 5 se instalarán, 0 para eliminar y 0 no actualizados.
Necesito descargar 1.732 kB de archivos.
Se utilizarán 4.705 kB de espacio de disco adicional después de esta operación.
¿Desea continuar [S/n]?
```

Una vez terminada la instalación, mediante el siguiente comando, podemos verificar que el servicio se encuentre funcionando.

```
root@lubuntu:/home/lubuntu# service snmpd status
 * snmpd is running
root@lubuntu:/home/lubuntu#
```

El siguiente paso sera configurar el protocolo SNMP para poder acceder a la información requerida. Los archivos de configuración se encuentran en la siguiente ruta.

```
root@lubuntu:/home/lubuntu# cd /etc/snmp
root@lubuntu:/etc/snmp# ls
snmp.conf  snmpd.conf  snmptrapd.conf
root@lubuntu:/etc/snmp#
```

El archivo a modificar es `snmpd.conf`, para poder modificarlo se deben tener permisos de administrador y se puede realizar desde el editor de texto de nuestro agrado, en este caso se utiliza `nano`.

```
root@lubuntu:/etc/snmp# nano snmpd.conf
```

El primer aspecto a modificar será modificar el comportamiento del agente, se debe permitir que el agente escuche las conexiones de todas las interfaces, esto se logra comentando la primera opción y habilitando la segunda.

```
#  AGENT BEHAVIOUR
#

#  Listen for connections from the local system only
agentAddress  udp:127.0.0.1:161
#  Listen for connections on all interfaces (both IPv4 *and* IPv6)
#agentAddress udp:161,udp6:[::1]:161
```

```
#  AGENT BEHAVIOUR
#

#  Listen for connections from the local system only
#agentAddress  udp:127.0.0.1:161
#  Listen for connections on all interfaces (both IPv4 *and* IPv6)
agentAddress udp:161,udp6:[::1]:161
```

El siguiente aspecto sera el control de acceso, deberán comentarse las 2 primeras líneas.

```
#  ACCESS CONTROL
#

                                                #  system + hrSystem groups on$
view   systemonly  included   .1.3.6.1.2.1.1
view   systemonly  included   .1.3.6.1.2.1.25.1

                                                #  Full access from the local $
```

Y agregar las siguientes.

```
###############################################################################
#
#  ACCESS CONTROL
#

                                                #  system + hrSystem groups only
#view   systemonly  included   .1.3.6.1.2.1.1
#view   systemonly  included   .1.3.6.1.2.1.25.1

com2sec notConfigUser  127.0.0.1       public
group   notConfigGroup v1           notConfigUser
group   notConfigGroup v2c           notConfigUser
view    systemview    included   .1.3.6.1.2.1.1
view    systemview    included   .1.3.6.1.2.1.25.1.1
access  notConfigGroup ""      any       noauth    exact  all none none
view all    included  .1                               80
```

Por ultimo se deberá habilitar el acceso a todos los agentes de la red habilitando la siguiente línea.

```
#rocommunity secret  10.0.0.0/16
```

Y modificarla para que coincida con las direcciónes de la red a dibujar. Esto permitirá que los dispositivos prevean de la información necesaria al sistema encargado de dibujar la topología.

```
rocommunity public  192.168.1.0/24
```

Las palabras “secret”, “public” se refieren al nombre de la comunidad, la comunidad puede recibir cualquier nombre, pero todos los dispositivos deberan pertenecer a la misma deberán para poder ser identificados.

Una vez modificado este archivo, se deberá reiniciar el servicio para que los cambios se vean reflejados.

```
root@lubuntu:/etc/snmp# service snmpd restart
 * Restarting network management services:
root@lubuntu:/etc/snmp# service snmpd status
 * snmpd is running
```

Switches

El primer paso para levantar el servicio SNMP en un switch, es establecer una ip, esto nos permitirá solicitar la información necesaria a este tipo de dispositivos mediante su ip, adicionalmente algunos tipos de switch no permiten iniciar el servicio SNMP si no se tiene alguna ip establecida en el dispositivo. También como en otro tipo de dispositivos de deberá establecer un Gateway para permitir la comunicación con otras redes.

```
Switch>en
Switch#configure terminal
Enter configuration commands, one per line.  End with CNTL/
Switch(config)#interface Vlan1
Switch(config-if)#ip address 192.168.1.25 255.255.255.0
Switch(config-if)#no shutdown

Switch(config-if)#
%LINK-5-CHANGED: Interface Vlan1, changed state to up

Switch(config-if)#ex
Switch(config)#ip default-gateway 192.168.1.1
Switch(config)#
```

Una vez configurada la ip y el gateway, se inicia el protocolo snmp.

```
Switch>en
Switch#configure terminal
Enter configuration commands, one per line.  End with CNTL/
Switch(config)#snmp-server community
Switch(config)#snmp-server community public RW
Switch(config)#
```

Podemos verificar que los cambios se hayan realizando, viendo la configuración actual del switch.

```
Switch(config)#ex
Switch#
%SYS-5-CONFIG_I: Configured from console by console

Switch#show running-config
Building configuration...

Current configuration : 1036 bytes
!
version 12.1
no service timestamps log datetime msec
no service timestamps debug datetime msec
no service password-encryption
!
hostname Switch
!
!
interface Vlan1
 ip address 192.168.1.25 255.255.255.0
!
ip default-gateway 192.168.1.1
!
snmp-server community public RW
!
```

Routers

La solicitud de información a un router puede realizarse a cualquiera de las direcciónes ip establecidas en alguna de sus interfaces. El protocolo snmp es iniciado de la misma manera que en un switch.

```
Router>en
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/
Router(config)#snmp-server community public RW
%SNMP-5-WARMSTART: SNMP agent on host Router is undergoing
Router(config)#
```

Al igual que la verificación de los cambios mediante la visualización de la configuración del router.

```
Router(config)#ex
Router#
%SYS-5-CONFIG_I: Configured from console by console

Router#show running-config
Building configuration...

Current configuration : 482 bytes
!
!
snmp-server community public RW
!
```

2.3. Solicitudes SNMP

Snmp permite la administración de los dispositivos mediante comandos[4] que solicitan la información requerida al dispositivo y, en caso de éxito, el mensaje regresado contendrá la información solicitada. Estos comandos son:

- Snmpget. Permite obtener información de un host remoto, dada su ip y el identificador de objeto(OID) que deseamos obtener.
- Snmpgetnetx. Es muy similar a snmpget, la diferencia consiste en que en vez de obtener la información del objeto que se indica en el comando, obtiene la información contenida en el siguiente objeto encontrado en el árbol jerárquico de los OID's.
- Snmpwalk. Realiza automáticamente toda una secuencia de getnext, trayendo todos los resultados de los objetos encontrados en el rango del IOD especificado en el comando.
- Snmptable. Cumple la misma función que snmpwalk, con la diferencia de que el resultado es mostrado en forma de tabla, permitiendo que su visualización y comprensión sea mas sencilla.

Los objetos [5], [6] a utilizar en el proyecto son:

OID	Representación textual	Descripción
1.3.6.1.2.1.1.1.0	sysDescr	Descripción textual del dispositivo.
1.3.6.1.2.1.4.20.1.1	ipAdEntAddr	Ip a la que pertenece la información.
1.3.6.1.2.1.4.21.1.7	ipRouteNextHop	Ip que representa el siguiente salto.
1.3.6.1.2.1.2.2.1.2	ifDescr	Tipo de interfaz del dispositivo.
1.3.6.1.2.1.2.2.1.6	ifPhysAddress	Dirección física de la interfaz.
1.3.6.1.2.1.2.2.1.8	ifOperStatus	Estado operacional de la interfaz.
1.3.6.1.2.1.4.22.1.3	ipNetToMediaNetAddress	Ip's conectadas al dispositivo.
1.3.6.1.2.1.2.2.1.11	ifInUcastPkts	Paquetes recibidos en la interfaz.
1.3.6.1.2.1.2.2.1.14	ifInErrors	Errores en los paquetes recibidos.
1.3.6.1.2.1.2.2.1.17	ifOutUcastPkts	Paquetes enviados por la interfaz.
1.3.6.1.2.1.2.2.1.20	ifOutErrors	Errores en los paquetes enviados.
1.3.6.1.4.1.9.9.46.1.3.1.1.18	vtpVlanIfIndex	Numero de Vlan de la interfaz.
1.3.6.1.4.1.9.9.46.1.3.1.1.4	vtpVlanName	Nombre de la Vlan de la interfaz.

Tabla 2.1: Objetos utilizados en el proyecto.

El comando que se utiliza es snmpwalk. El uso de este comando debe ser la de la siguiente manera:

snmpwalk [opciónes] agente [opciónes generales] [oid]

opción	Funcion
-v	Especificar la versión de SNMP que se utilizara.
-c	Especificar la comunidad a que pertenece el dispositivo.

Tabla 2.2: Opciones snmp.

Las opciones que siempre se deben de especificar son:
Y dentro de las opciones generales se utilizara:

opción	Funcion
-Oq	Modifica el formato en que se muestra el resultado.

Tabla 2.3: Opcion general snmp.

Esta opción se utiliza ya que el formato que regresa la información solicitada, permite un fácil análisis del resultado al momento de procesar la información para descubrir los dispositivos y dibujar la topología.

En los comandos utilizados en el proyecto, se utiliza la versión 2c del protocolo snmp y se establece la comunidad con el nombre de public, por ejemplo el comando para obtener la descripción del dispositivo con dirección ip 192.168.1.16 quedaría de la siguiente manera.

snmpwalk -v 2c -c public 192.168.1.16 -Oq 1.3.6.1.2.1.1.1.0

Snmp permite cambiar el oid por la Descripción textual de este, por lo que ejecutar el comando anterior es equivalente a ejecutar el siguiente:

snmpwalk -v 2c -c public 192.168.1.16 -Oq sysDescr

Una vez que en todos los dispositivos que se quieran dibujar, se tenga instalado y configurado el protocolo snmp, se deberá crear un archivo llamado Ip.txt que contenga las direcciones IP de estos dispositivos, como ya se había mencionado en la sección de instalación y configuración de snmp, en el caso de los switch se deberá poner la Ip que se estableció a la Vlan1 y en el caso de los Routers solo se debe poner la Ip de una de sus interfaces.

Una vez creado este archivo, el sistema lo lee mediante el siguiente código:

```
1  String archivo = "Ip.txt";
2  String ip = null;
3  ArrayList<String> Ins = new ArrayList<String>();
4
```

```
5  try {
6              FileReader lector = new FileReader(archivo);
7              BufferedReader contenido = new BufferedReader(lector);
8              while ((texto = contenido.readLine()) != null) {
9                  Ins.add("snmpwalk -v 2c -c public "+texto+" -Oq sysDescr");/**
                       Descripcion del dispositivo*/
10                 Ins.add("snmpwalk -v 2c -c public "+texto+" -Oq ipAdEntAddr");/**Ip
                       del dispositivo*/
11                 Ins.add("snmpwalk -v 2c -c public "+texto+" -Oq ipRouteNextHop");/**
                       Gateway*/
12                 Ins.add("snmpwalk -v 2c -c public "+texto+" -Oq ifDescr");/**
                       Interfaces*/
13                 Ins.add("snmpwalk -v 2c -c public "+texto+" -Oq ifPhysAddress");/**
                       Mac de la interfaz*/
14                 Ins.add("snmpwalk -v 2c -c public "+texto+" -Oq ifOperStatus");/**
                       Estado de la interfaz*/
15                 Ins.add("snmpwalk -v 2c -c public "+texto+" -Oq ifInUcastPkts");/**
                       Paquetes recibidos por la interfaz*/
16                 Ins.add("snmpwalk -v 2c -c public "+texto+" -Oq ifInErrors");/**
                       Errores en los paquetes recibidos*/
17                 Ins.add("snmpwalk -v 2c -c public "+texto+" -Oq ifOutUcastPkts");/**
                       Paquetes enviados por la interfaz*/
18                 Ins.add("snmpwalk -v 2c -c public "+texto+" -Oq ifOutErrors");/**
                       Errores en los paquetes enviados*/
19                 Ins.add("snmpwalk -v 2c -c public "+texto+" -Oq vtpVlanIfIndex");/**
                       Numero de la Vlan a que pertenece la interfaz*/
20                 Ins.add("snmpwalk -v 2c -c public "+texto+" -Oq vtpVlanName");/**
                       Nombre de la Vlan a que pertenece la interfaz*/
21
22
23                 Snmpwalk t = new Snmpwalk();
24                 ArrayList<String> snmpwalk = t.snmpwalk(Ins);
```

Por cada dirección Ip en el archivo, este código crea una lista de doce instrucciones a ejecutar para obtener la información requerida para descubrir y dibujar la topología, una vez llena esta lista, mediante la instrucción de la línea 23 es creado un nuevo objeto de nombre t y de tipo Snmpwalk.

Este objeto recibe una lista de instrucciones, las ejecuta y devuelve el resultado dentro de otra lista, en la línea 24 se puede observar que como se envían las instrucciones contenidas en la lista de nombre Ins y la lista que devuelve se guarda en otra lista de nombre snmpwalk. A continuación se describe el código de la clase Snmpwalk.

```
1  public class Snmpwalk {
2
3    public ArrayList<String> snmpwalk(ArrayList<String> instrucciones) throws Exception
         {
4      String line = "";
5      ArrayList<String> resultado = new ArrayList();
6      for ( int i=0; i < instrucciones.size(); i++ ) {
7        String instruccion = (String)instrucciones.get(i);
8        Process p = Runtime.getRuntime().exec(instruccion);
9        BufferedReader res = new BufferedReader(new InputStreamReader(p.getInputStream
             ()));
10       while ( (line=res.readLine()) != null ) {
11         resultado.add(line);
12       }
13     }
14     return resultado;
15   }
```

```
16 }
```

Es un código bastante sencillo, en el que se recorre la lista de instrucciones que se recibió, cada una de ellas es ejecutada mediante la línea 8 y el resultado es guardado en una lista llamada resultado, una vez ejecutadas todas las instrucciones contenidas en la lista se devuelve la lista con los resultados.

2.4. Descubrimiento de dispositivos

La lista devuelta es recorrida y se obtienen los datos del dispositivo tales como:

- Tipo de dispositivo.
- Dirección Ip.
- Gateway.
- Interfaces.
- Dirección física de las interfaces.
- Estado operacional de las interfaces.
- Número de paquetes enviados.
- Errores en los paquetes enviados.
- Número de paquetes recibidos.
- Errores en los paquetes recibidos.
- Vlan a la que pertenece la interfaz.
- Nombre de la Vlan a la que pertenece la interfaz.

Una vez obtenidos estos datos, se crea un objeto de tipo Dispositivo, que es agregado a una lista y se dibuja el dispositivo.

```
1  int x = (int)(Math.random()*lienzo.getWidth());
2  int y = (int)(Math.random()*lienzo.getHeight());
3  if(Vlan.isEmpty()){
4                       if(Gw==null){
5                           topologia.agregarDispositivo(Tipo, Ip[1],Interfaces,DirMac,
                                EInterfaces,PaqEnv,ErrPE,PaqRec,ErrPR,new Point(x,y),
                                tamDisp );
6                       }
7                       else {
8                           topologia.agregarDispositivo(Tipo, Ip[1],Gw[1],Interfaces,
                                DirMac,EInterfaces,PaqEnv,ErrPE,PaqRec,ErrPR,new Point(x,
                                y), tamDisp );
9                       }
10                  }else{
11                      if(Gw==null){
12                          topologia.agregarDispositivo(Tipo, Ip[1],Interfaces,DirMac,
                                EInterfaces,PaqEnv,ErrPE,PaqRec,ErrPR,Vlan,NVlan,new
                                Point(x,y), tamDisp );
13                      }
14                      else{
15                          topologia.agregarDispositivo(Tipo, Ip[1],Gw[1],Interfaces,
                                DirMac,EInterfaces,PaqEnv,ErrPE,PaqRec,ErrPR,Vlan,NVlan,
                                new Point(x,y), tamDisp );
16                      }
17                  }
```

X y Y son las coordenadas en que se dibuja el dispositivo dentro del lienzo, y tamDisp es el tamaño del icono que se dibujara. La sobrecarga de constructores se debe a que unicamente dispositivos como Switches y Routers pueden tener configurada una Vlan y los Routers no tienen gateway, o bien dentro de una red local alguna compputadora podria no tenerlo configurado.

A continuación de describe el código de topología, que es utilizado en la línea 5, 8, 12 y 15 del código mostrado anteriormente

```
1  private java.util.List listaDispositivos; /**Lista de Dispositivos*/
2
3  public topologia(){
4          listaDispositivos = Collections.synchronizedList(new LinkedList());
5  }
6
7  public Dispositivo agregarDispositivo(int Tipo, String etiqueta,List<String> interfaz
       ,List<String> mac,List<String> estado, List<String> PaqEnv,List<String> ErrPE,
       List<String> PaqRec,List<String> ErrPR, List<String> Vlan, List<String> NVlan,
       Point puntoInicial,int tam){
8          Dispositivo disp = new Dispositivo( Tipo, etiqueta, interfaz,mac,estado,
               PaqEnv,ErrPE, PaqRec, ErrPR, Vlan, NVlan,puntoInicial,tam);
9          listaDispositivos.add( disp );
10         return disp;
11     }
12
13     public Dispositivo agregarDispositivo(int Tipo, String etiqueta,String gateway,
           List<String> interfaz,List<String> mac,List<String> estado,List<String>PaqEnv
           , List<String> ErrPE, List<String> PaqRec, List<String> ErrPR,List<String>
           Vlan, List<String> NVlan, Point puntoInicial,int tam){
14         Dispositivo disp = new Dispositivo( Tipo, etiqueta, gateway,interfaz,mac,
               estado,PaqEnv,ErrPE, PaqRec, ErrPR, Vlan, NVlan,puntoInicial,tam);
15         listaDispositivos.add( disp );
16         return disp;
17     }
18
19     public Dispositivo agregarDispositivo(int Tipo, String etiqueta,List<String>
           interfaz,List<String> mac,List<String> estado, List<String> PaqEnv,List<
           String> ErrPE, List<String> PaqRec,List<String> ErrPR,Point puntoInicial,int
           tam){
20         Dispositivo disp = new Dispositivo( Tipo, etiqueta, interfaz,mac,estado,
               PaqEnv,ErrPE, PaqRec, ErrPR, puntoInicial,tam);
21         listaDispositivos.add( disp );
22         return disp;
23     }
24
25     public Dispositivo agregarDispositivo(int Tipo, String etiqueta,String gateway,
           List<String> interfaz,List<String> mac,List<String> estado,List<String>PaqEnv
           , List<String> ErrPE, List<String> PaqRec, List<String> ErrPR, Point
           puntoInicial,int tam){
26         Dispositivo disp = new Dispositivo( Tipo, etiqueta, gateway,interfaz,mac,
               estado,PaqEnv,ErrPE, PaqRec, ErrPR, puntoInicial,tam);
27         listaDispositivos.add( disp );
28         return disp;
29     }
```

La lista listaDispositivos es utilizada para crear las conexiones entre dos dispositivos y también para permitir que al dar clic sobre un dispositivo se permita arrastrarlo dentro del lienzo y mostrar las propiedades de este dispositivo.

La clase dispositivo utilizada en las líneas 8, 14, 20 y 26 del código anterior, se encarga de crear los objetos de tipo Dispositivo y de dibujarlos en el lienzo mediante la siguiente función:

```
1       public void pintar( Graphics g ){
2           int pix = getPuntoInicial().x;
3                   int piy = getPuntoInicial().y;
4                   if (tipo==1){
5               image = new ImageIcon(getClass().getResource("Computadora.jpg")).getImage
                    ();
6               g.drawString( getIp(),pix,(piy+55)+7 );
7           }
8           else if (tipo==2){
9               image = new ImageIcon(getClass().getResource("switch.jpg")).getImage();
10          }
11          else if (tipo==3){
12              image = new ImageIcon(getClass().getResource("Router.jpg")).getImage();
13          }
14          g.drawImage(image, pix-15, piy-15, Main.lienzo);
15          g.setColor(Color.BLACK);
16                  g.setFont( new Font( "TimesRoman",Font.PLAIN, getTam()*25/100) );
17      }
```

2.5. Descubrimiento de conexiones

Hasta ahora, se han descubierto y dibujado los dispositivos, pero no sus conexiones, para encontrar la relación entre los dispositivos que conforman la red, el sistema vuelve a acceder al archivo que contiene las direcciones IP y solicita la ejecución de dos comandos snmpwalk:

```
FileReader lector = new FileReader(archivo);
BufferedReader contenido = new BufferedReader(lector);
        while ((ip = contenido.readLine()) != null) {
        Ins.add("snmpwalk -v 2c -c public "+ip+" -Oq ipAdEntAddr");
        Ins.add("snmpwalk -v 2c -c public "+ip+" -Oq ipNetToMediaNetAddress");

        Snmpwalk t = new Snmpwalk();
        ArrayList<String> snmpwalk = t.snmpwalk(Ins);
```

Esta vez la lista obtenida contiene:

- Dirección Ip.
- Direcciones Ip conectadas directamente al dispositivo.

Una vez obtenida esta información se agrega la conexione entre ambos dispositivos:

```
topologia.agregarConexion( Ip[1],Con[1]);
```

La siguiente función es la encargada de agregar la conexion:

```
public int agregarConexion(String o, String d){
        Dispositivo arr[] = {null,null};
    int cond = 0;
    Iterator it = listaDispositivos.iterator();
    Dispositivo disp=null;
        while ( it.hasNext() & cond != 2 ){/**Existen ambos dispositivos?*/
        disp = (Dispositivo)it.next();
        if( disp.getIp().equals(o)){
                        arr[0]=disp;
                        cond++;
        }
        else if( disp.getIp().equals(d) ){
                        arr[1]=disp;
                cond++;
        }
    }
    if( arr[0] != null & arr[1] != null ){//Se encontraron ambos dispostivos
        arr[0].getListaAd().add( new Conexion( arr[0],arr[1] ) );
        setModificado(true);
        return 1; /**Se agrego el dispositivo a la lista de adyacencia*/
    }
    else if( arr[0] != null ){
                return 3; /**No se encontro el dispositivo destino*/
        }
    else {
                return 4; /**No se encontro el despositivo origen*/
        }
}
```

La clase Conexión utilizada en la línea 18 es la encargada de crear la nueva conexión y dibujarla en el lienzo mediante la siguiente función:

```
1  public void pintar( Graphics g )        {
2          /**coordenadas iniciales*/
3          int xi = dispositivoOrigen.getPuntoInicial().x;
4          int yi = dispositivoOrigen.getPuntoInicial().y;
5          /**coordenadas finales*/
6          int xf = dispositivoDestino.getPuntoInicial().x;
7          int yf = dispositivoDestino.getPuntoInicial().y;
8          /**Puntos que representan el centro del icono*/
9      int xim = xi+(dispositivoOrigen.getTam()/2); /**centro del nodo origen*/
10     int yim = yi+(dispositivoOrigen.getTam()/2);
11     int xfm = xf+(dispositivoDestino.getTam()/2); /**centro del nodo destino*/
12     int yfm = yf+(dispositivoDestino.getTam()/2);
13
14     int pfx = (int) (xfm ); /**punto final en x donde termina la conexion*/
15     int pfy = (int) (yfm ); /**punto final en y donde termina la conexion*/
16
17     if( dispositivoOrigen != dispositivoDestino ){
18         dibujarConexion( g, xim, yim, pfx, pfy );
19         }
20 }
21
22 public void dibujarConexion(Graphics g,double x1, double y1, double x2, double y2){
23     g.drawLine((int)x1,(int)y1,(int)x2,(int)y2);
24 }
```

Capítulo 3

Resultados y conclusiones

3.1. Despliegue de la topología

Una vez terminado todo el proceso de descubrir y dibujar todos los dispositivos y sus conexiones, el sistema mostrara la topología descubierta. En la figura 3.1 se muestran el resultado de una ejecución sobre una pequeña red, que contiene cuatro computadoras, dos switch y un router.

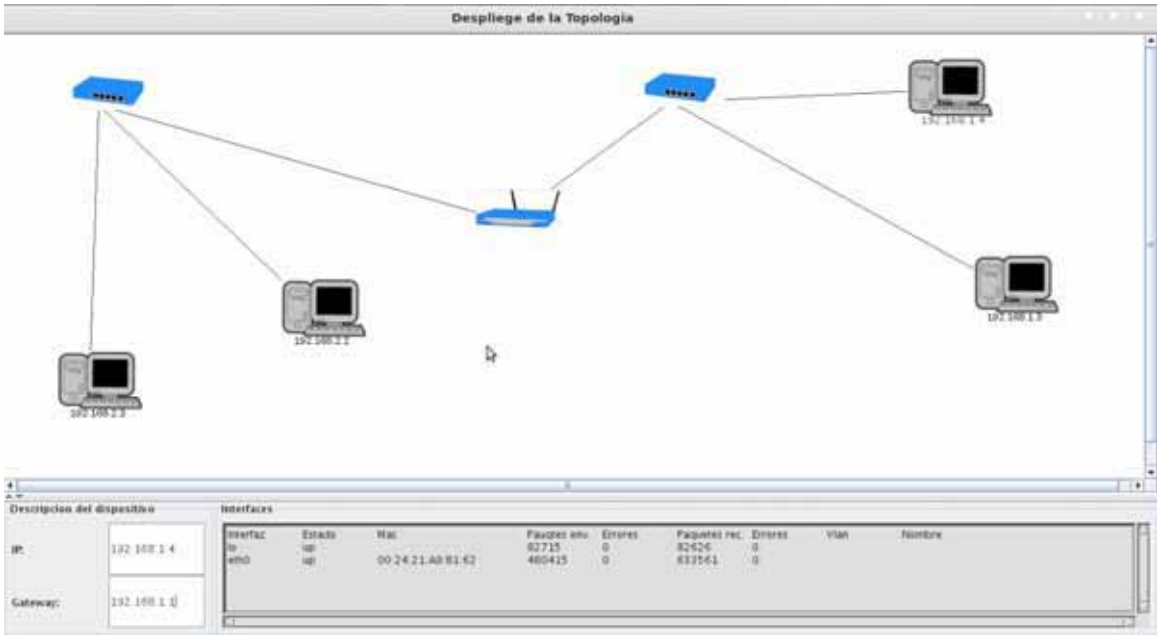

Figura 3.1: Ejemplo de topología descubierta por el sistema.

La ventana desplegada nos permite realizar varias funciones, una de ellas, es arrastrar los dispositivos para acomodarlos dentro el lienzo, permitiendo mejorar la visualización. En la figura 3.2 se muestra la misma topología acomodada de manera distinta.

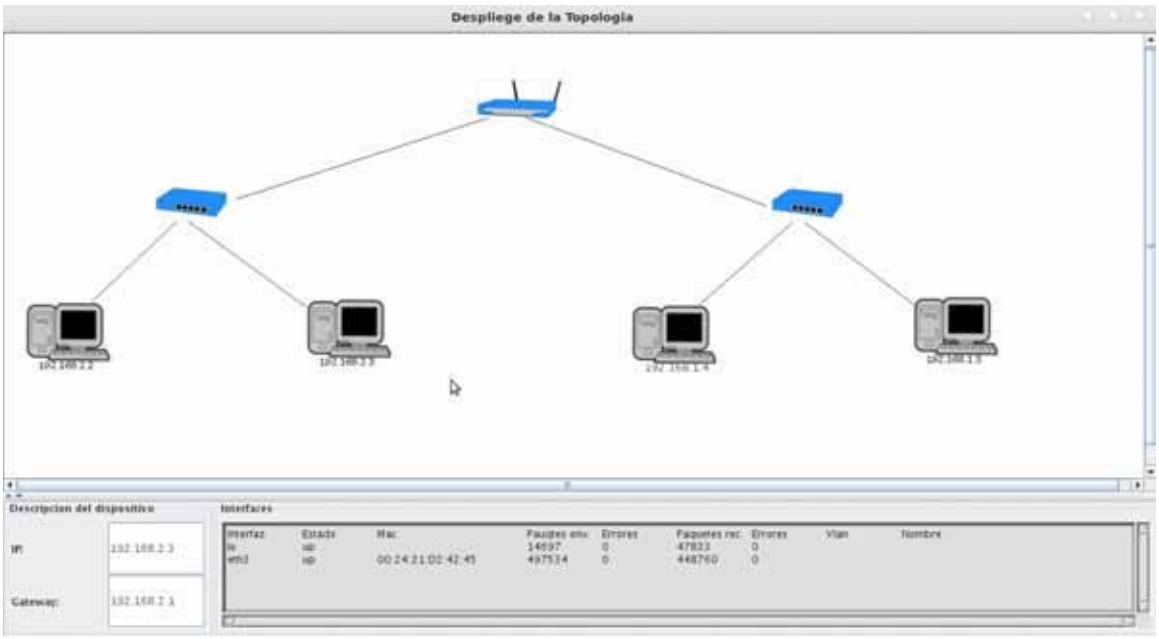

Figura 3.2: Ejemplo de desplazamiento de dispositivos.

Adicionalmente, al seleccionar un dispositivo, en la parte inferior de la pantalla se muestra la información relacionada con este dispositivo, tal como su dirección IP, su gateway, en caso de que tenga uno, las interfaces con que cuenta el dispositivo, así como su estado operacional y dirección física de cada una de ellas, numero de paquetes enviados, recibidos y numero de errores generados en cada interfaz, si el dispositivo es un switch o un router se mostrara a que Vlan pertenece cada puerto, y el nombre de la Vlan,en caso de tener uno establecido. En la figura 3.3 se muestra la información de la computadora con dirección Ip 192.168.1.4.

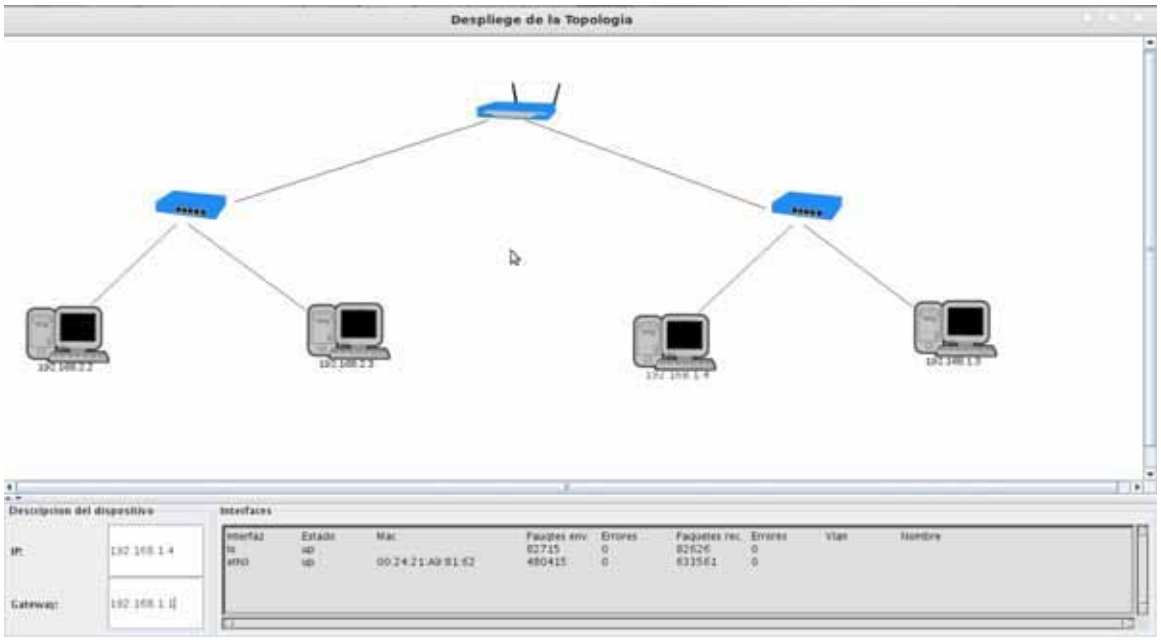

Figura 3.3: Información de una computadora.

Del mismo modo se muetra la información de un Switch. Figura 3.4.

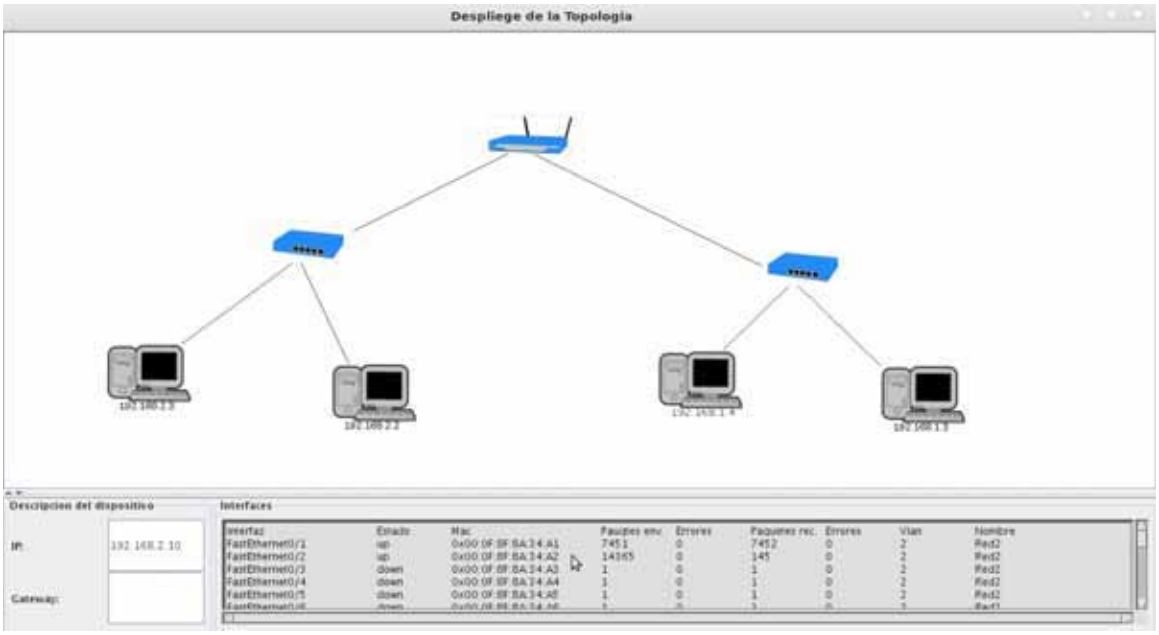

Figura 3.4: Información de un Switch.

Y de un Router. Figura 3.5.

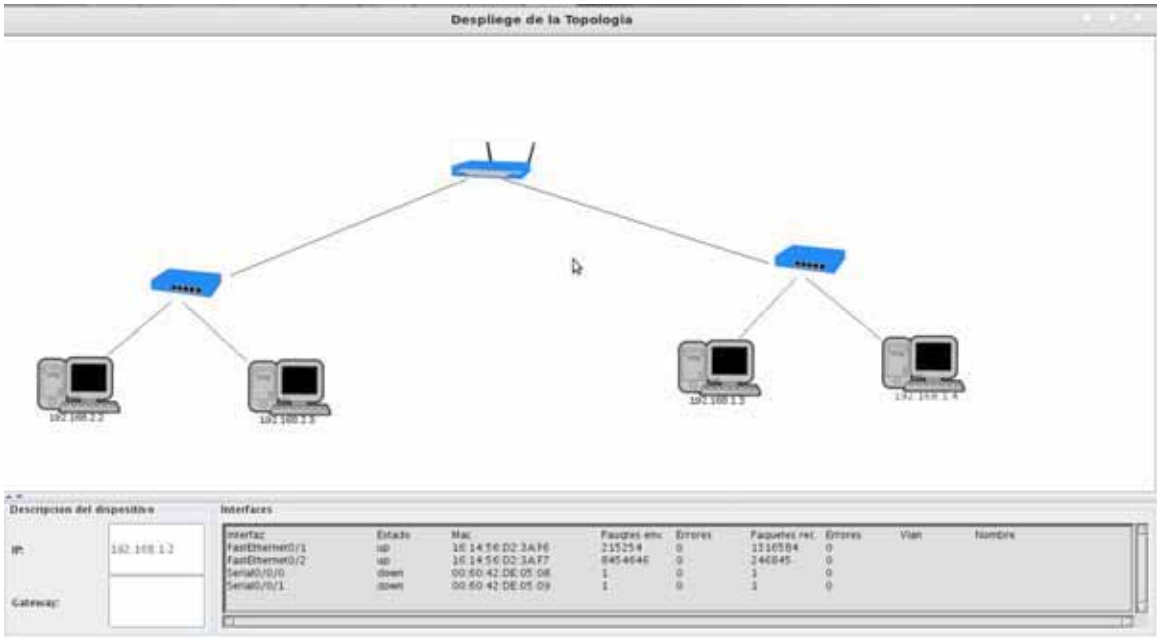

Figura 3.5: Información de un Router.

Estas funcionalidades son permitidas gracias al siguiente código:

```
1  public void mousePressed( MouseEvent evt ){
2          if(arrastrar) {
3              return;
4          }
5          if( evt.isMetaDown() ){
6              int x = evt.getX(); /**Tomar posicion del mouse*/
7              int y = evt.getY();
8              boolean encontrado=false;
9              ListIterator it = listaDispositivos.listIterator( listaDispositivos.size
                   () );
10             while( it.hasPrevious() && encontrado==false ){
11                 nod1 = (Dispositivo) it.previous();
12                 /**Si hay un clic dentro de un dispositivo*/
13                 if( x>=nod1.getPuntoInicial().x && x<nod1.getPuntoInicial().x+nod1.
                       getTam()
14                 && y>=nod1.getPuntoInicial().y && y<nod1.getPuntoInicial().y+nod1.
                       getTam() ) {
15                     encontrado = true;
16                   }
17             }
18         }
19         else{
20             int x = evt.getX(); /**Tomar posicion del mouse*/
21             int y = evt.getY();
22             boolean encontrado=false;
23             ListIterator it = listaDispositivos.listIterator( listaDispositivos.size
                   () );
24             while( it.hasPrevious() && encontrado==false ){
25                 nodA = (Dispositivo) it.previous();
26                 /**Si hay un clic dentro de un dispositivo*/
27                 if( x>=nodA.getPuntoInicial().x && x<nodA.getPuntoInicial().x+nodA.
                       getTam()
28                 && y>=nodA.getPuntoInicial().y && y<nodA.getPuntoInicial().y+nodA.
                       getTam() ){
29                     encontrado = true;
30
31                     arrastrar = true;
32                     despX = x - nodA.getPuntoInicial().x;
33                     despY = y - nodA.getPuntoInicial().y;
34                 }
35             }
36         }
37         repintarLienzo();
38     }
39
40     @Override
41     public void mouseDragged( MouseEvent evt ){
42         if(arrastrar ){
43             topologia.setModificado(true);
44             int x = evt.getX()-despX;
45             int y = evt.getY()-despY;
46             if( x<0 ) {
47                 x=0;
48             }
49             if( y<0 ) {
50                 y=0;
51             }
52             nodA.setPuntoinicial( new Point(x,y) );
53             Propiedades.tfIP.setText(nodA.getIp());
54             Propiedades.tfGateway.setText(nodA.getGateway());
55             Propiedades.lListaInterfaces.setText("");
56             for(int i =0; i<nodA.getInterfaces().size();i++){
57                 Propiedades.lListaInterfaces.append(nodA.getInterfaces().get(i) +"\t"
```

```
                        +nodA.getEstIf().get(i)+"\t"+nodA.getDirMac().get(i)+"\n");
58              }
59
60              if( nodA.getPuntoInicial().x+nodA.getTam() > getWidth() ){/**
                    Redimenzionar el lienzo*/
61                  setPreferredSize( new Dimension(x+nodA.getTam(),getHeight() ) );
62                  revalidate();
63              }
64              if( nodA.getPuntoInicial().y+nodA.getTam() > getHeight() ){
65                  setPreferredSize( new Dimension( getWidth() ,y+nodA.getTam() ) );
66                  revalidate();
67              }
68              repintarLienzo();
69          }
70      }
```

Para permitir una mejor visualización, tanto de la topología como de la Información, la barra en la que se muestra la información de los dispositivos, puede ampliarse. Figura 3.6.

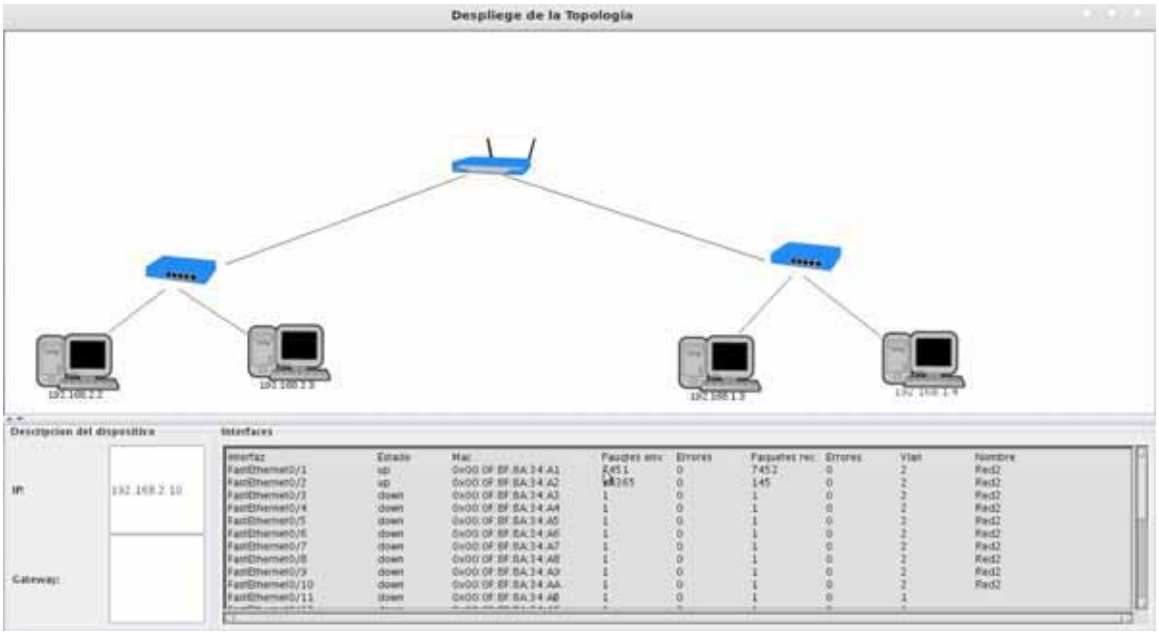

Figura 3.6: Cambio de tamaño de la barra de información

Y también puede ser ocultada. Figura 3.7.

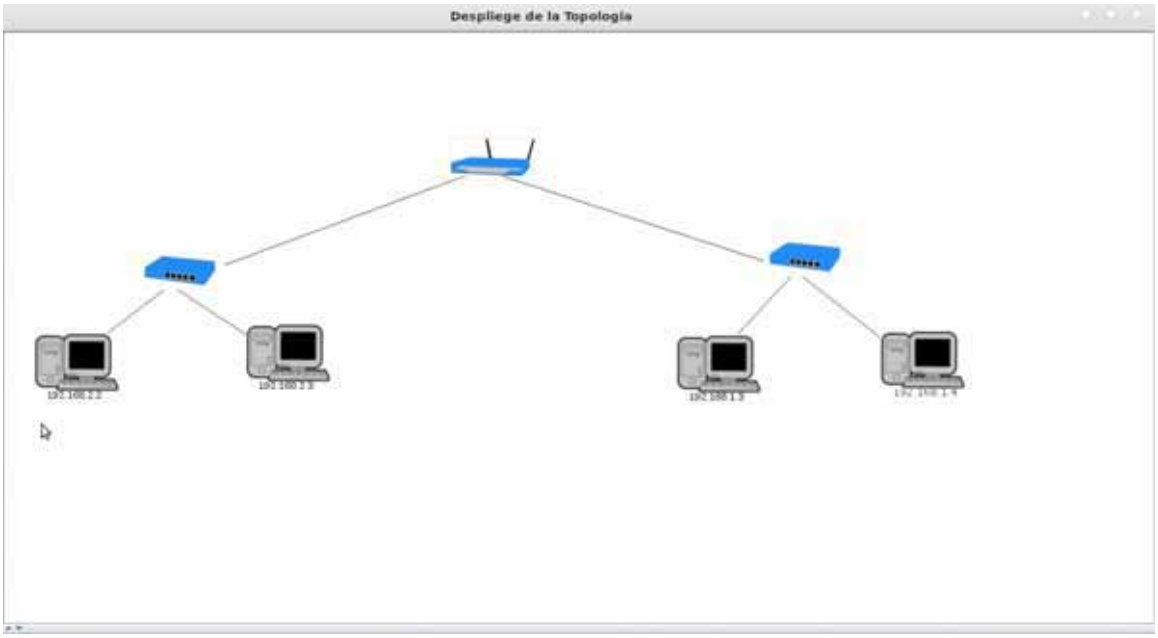

Figura 3.7: Barra de información oculta

3.2. Conclusiones

El producto final se presenta como una aplicación, que lo único que requiere es la previa instalación y configuración de protocolo snmp en los dispositivos que se desean dibujar y las direcciones Ip de estos. El resultado es mostrado en una interfaz intuitiva, que muestra la topología encontrada por la aplicación y la información relacionada con los dispositivos pertenecientes a la topología.

Una herramienta de este tipo puede ser de gran utilidad para un administrador de red, pues le permite conocer la topología de la red que administra y tener este conocimiento le permitirá cumplir con sus funciones de manera mas optima.

Si bien es una aplicación limitada en comparación con productos de software privado existentes en el mercado, permite el descubrimiento de la topología dentro de redes de tamaño pequeño o mediano, como las que se pueden encontrar en medianas empresas. De la misma manera esta aplicación podría ser de gran ayuda en cursos básico de diseño y administración de redes de computadoras.

Esta herramienta se podría expandir o utilizar como base para desarrollar un proyecto mas grande y complejo en el futuro, que requiera un nivel de detalle mayor, o simplemente que deba adecuarse a las condiciones existentes dentro de una red especifica.

Bibliografía

[1] Andrew S. Tanenbaum. Redes de computadoras, cuarta edición. Pearson ,2003.

[2] Xiangyu Li. (24 de Marzo de 2014). "A method of Network topology visualization based on SNMP", 2011. [En lÃnea]. Disponible en: http://ieeexplore.ieee.org/xpl/articleDetails.jsp? tp=&arnumber=6154046

[3] Secure Internet Management and Snmp. (24 de Marzo de 2014). [En lÃnea]. Disponible en: http://www.snmp.com/

[4] Net-Snmp Man Pages. (24 de Marzo de 2014). [En lÃnea]. Disponible en: http://www.net-snmp.org/docs/man/

[5] RFC 1213. (24 de Marzo de 2014)Management Information Base for Network Management of TCP/IP-based internets: MIB-II. IETF, 1991. [En lÃnea]. Disponible en: http://www.ietf.org/rfc/rfc1213.txt

[6] Cisco-Vtp-Mib. (24 de Marzo de 2014)[En lÃnea]. Disponible en: ftp://ftp.cisco.com/pub/mibs/v2/CISCO-VTP-MIB.my

Apéndice A

Código fuente

A.1. Código de la clase Main

```
1  import java.awt.*;
2  import java.io.BufferedReader;
3  import java.io.FileNotFoundException;
4  import java.io.FileReader;
5  import java.io.IOException;
6  import java.util.ArrayList;
7  import javax.swing.*;
8
9  public class Main extends JFrame {
10
11     static private int tamDisp = 45;
12     private java.util.List listaDispositivos;
13     static private Topologia topologia;
14
15     static public Lienzo lienzo;    /**Area  de dibujo*/
16
17     /**Menu de descirpcion del dispositivo*/
18     private Propiedades mpProps;
19
20
21     private JSplitPane spSeparador;
22
23     public Main(){
24         super("Despliege de la Topologia");
25
26         Container c = getContentPane();
27         c.setLayout( new BorderLayout() );
28
29         /**Descripcion del dispositivo*/
30         mpProps = new Propiedades();
31         Dimension minimumSize2 = new Dimension(0, 100);
32         mpProps.setMinimumSize(minimumSize2);
33
34         /**Inicializar topologia*/
35         topologia = new Topologia();
36         listaDispositivos = topologia.getListaDispositivos();
37
38         lienzo = new Lienzo(topologia, tamDisp);
39         JScrollPane spLienzo = new JScrollPane(lienzo);
40         Dimension minimumSize = new Dimension(0,400);
```

```
41          spLienzo.setMinimumSize( minimumSize );
42
43          spSeparador = new JSplitPane( JSplitPane.VERTICAL_SPLIT,spLienzo,mpProps );
44          spSeparador.setOneTouchExpandable(true);
45          spSeparador.setDividerLocation(920);
46
47          c.add(spSeparador);
48
49          setSize(1200,600);
50
51          setVisible(true);
52      }
53
54
55
56      public static void main(String args[]) throws FileNotFoundException, IOException{
57          Main apl = new Main();
58          apl.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
59          int n;
60          char ult;
61          int Tipo=0;
62          int c = 0;
63          String archivo="Mibs2.txt";
64          String texto;
65          String Ip[] = null;
66          String Gw[] = null;
67          String Con[] = null;
68          String Interfaz[] = null;
69          String DMac[] = null;
70          String Estado[] = null;
71          String PaqE[] = null;
72          String Epe[] = null;
73          String PaqR[] = null;
74          String Epr[] = null;
75          String Vl[] = null;
76          String Nvl[] = null;
77
78          java.util.List<String> Interfaces = new java.util.LinkedList<String>();
79          java.util.List<String> DirMac = new java.util.LinkedList<String>();
80          java.util.List<String> EInterfaces = new java.util.LinkedList<String>();
81          java.util.List<String> PaqEnv = new java.util.LinkedList<String>();
82          java.util.List<String> ErrPE = new java.util.LinkedList<String>();
83          java.util.List<String> PaqRec = new java.util.LinkedList<String>();
84          java.util.List<String> ErrPR = new java.util.LinkedList<String>();
85          java.util.List<String> Vlan = new java.util.LinkedList<String>();
86          java.util.List<String> NVlan = new java.util.LinkedList<String>();
87
88
89          try{
90              FileReader lector=new FileReader(archivo);
91              BufferedReader contenido=new BufferedReader(lector);
92
93              while((texto=contenido.readLine())!=null){
94                  if (texto.indexOf ("sysDescr") != -1){
95                      if(c==0){c +=1;}
96                      else{
97                          int x = (int)(Math.random()*lienzo.getWidth());
98                          int y = (int)(Math.random()*lienzo.getHeight());
99                          if(Vlan.isEmpty()){
100                             if(Gw==null){
101                                 topologia.agregarDispositivo(Tipo, Ip[1],Interfaces,
                                        DirMac,EInterfaces,PaqEnv,ErrPE,PaqRec,ErrPR,new
                                        Point(x,y), tamDisp );
102                             }
103                             else {
```

```
104                                 topologia.agregarDispositivo(Tipo, Ip[1],Gw[1],
                                        Interfaces,DirMac,EInterfaces,PaqEnv,ErrPE,PaqRec
                                        ,ErrPR,new Point(x,y), tamDisp );
105                             }
106                         }else{
107                             if(Gw==null){
108                                 topologia.agregarDispositivo(Tipo, Ip[1],Interfaces,
                                        DirMac,EInterfaces,PaqEnv,ErrPE,PaqRec,ErrPR,Vlan
                                        ,NVlan,new Point(x,y), tamDisp );
109
110                             }
111                             else{
112                                 topologia.agregarDispositivo(Tipo, Ip[1],Gw[1],
                                        Interfaces,DirMac,EInterfaces,PaqEnv,ErrPE,PaqRec
                                        ,ErrPR,Vlan,NVlan,new Point(x,y), tamDisp );
113                             }
114                         }
115                         Gw=null;
116                         Vlan=null;
117                         Interfaces= new java.util.LinkedList<String>();
118                         DirMac= new java.util.LinkedList<String>();
119                         EInterfaces = new java.util.LinkedList<String>();
120                         PaqEnv = new java.util.LinkedList<String>();
121                         ErrPE = new java.util.LinkedList<String>();
122                         PaqRec = new java.util.LinkedList<String>();
123                         ErrPR = new java.util.LinkedList<String>();
124                         Vlan = new java.util.LinkedList<String>();
125                         NVlan = new java.util.LinkedList<String>();
126
127                     }
128                     if (texto.indexOf ("Linux") != -1){
129                         Tipo=1;
130                     }
131                     else if (texto.indexOf ("Windows") != -1){
132                         Tipo=1;
133                     }
134                     else if (texto.indexOf ("2960") != -1){
135                         Tipo=2;
136                     }
137                     else if (texto.indexOf ("2950-24") != -1){
138                         Tipo=2;
139                     }
140                     else if (texto.indexOf ("2950-T") != -1){
141                         Tipo=2;
142                     }
143                     else if (texto.indexOf ("1841") != -1){
144                         Tipo=3;
145                     }
146                     else if (texto.indexOf ("2620XM") != -1){
147                         Tipo=3;
148                     }
149                     else if (texto.indexOf ("2621XM") != -1){
150                         Tipo=3;
151                     }
152                     else if (texto.indexOf ("2811") != -1){
153                         Tipo=3;
154                     }
155                 }
156                 if (texto.indexOf ("ipAdEntAddr") != -1){
157                     if (texto.indexOf ("ipAdEntAddr.127.0.0.1") != -1){}
158                     else{
159                         Ip = texto.split(" ");
160                     }
161                 }
162                 if (texto.indexOf ("ipRouteNextHop.0.0.0.0") != -1){
163                     Gw = texto.split(" ");
```

```
164                 }
165                 if (texto.indexOf ("ifDescr") != -1){
166                     Interfaz = texto.split(" ");
167                     Interfaces.add(Interfaz[1]);
168                 }
169                 if (texto.indexOf ("ifPhysAddress") != -1){
170                     n=texto.length();
171                     ult=texto.charAt(n-1);
172                     if (ult==' '){DirMac.add("\t");}
173                     else{
174                         DMac = texto.split(" ");
175                         DirMac.add(DMac[1]);
176                     }
177                 }
178                 if (texto.indexOf ("ifOperStatus") != -1){
179                     Estado = texto.split(" ");
180                     EInterfaces.add(Estado[1]);
181                 }
182                 if (texto.indexOf ("ifInUcastPkts") != -1){
183                     PaqR = texto.split(" ");
184                     PaqRec.add(PaqR[1]);
185                 }
186                 if (texto.indexOf ("ifInErrors") != -1){
187                     Epr = texto.split(" ");
188                     ErrPR.add(Epr[1]);
189                 }
190                 if (texto.indexOf ("ifOutUcastPkts") != -1){
191                     PaqE = texto.split(" ");
192                     PaqEnv.add(PaqE[1]);
193                 }
194                 if (texto.indexOf ("ifOutErrors") != -1){
195                     Epe = texto.split(" ");
196                     ErrPE.add(Epe[1]);
197                 }
198                 if (texto.indexOf ("vtpVlanIfIndex") != -1){
199                     n=texto.length();
200                     ult=texto.charAt(n-1);
201                     if (ult==' '){Vlan.add("\t");}
202                     else{
203                         Vl = texto.split(" ");
204                         Vlan.add(Vl[1]);
205                     }
206                 }
207                 if (texto.indexOf ("vtpVlanName") != -1){
208                     n=texto.length();
209                     ult=texto.charAt(n-1);
210                     if (ult==' '){NVlan.add("\t");}
211                     else {
212                         Nvl = texto.split(" ");
213                         NVlan.add(Nvl[1]);
214                     }
215                 }
216             }
217 
218         }
219         catch(Exception e){
220             System.out.println(e);
221         }
222 
223         try{
224             FileReader lector2=new FileReader(archivo);
225             BufferedReader contenido2=new BufferedReader(lector2);
226             while((texto=contenido2.readLine())!=null){
227                 if (texto.indexOf ("ipAdEntAddr") != -1){
228                     if (texto.indexOf ("ipAdEntAddr.127.0.0.1") != -1){}
229                     else{
```

```
230                         Ip = texto.split(" ");
231                     }
232                 }
233                 if (texto.indexOf ("ipNetToMediaNetAddress") != -1){
234                     Con = texto.split(" ");
235                     topologia.agregarConexion( Ip[1],Con[1]);
236                 }
237             }
238         }
239         catch(Exception e){
240             System.out.println(e);
241         }
242 
243     }
244 }
```

A.2. Código de la clase Lienzo

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import java.util.*;
4  import javax.swing.*;
5
6  final class Lienzo extends JPanel implements MouseListener,MouseMotionListener{
7
8      private Topologia topologia;
9      private java.util.List listaDispositivos;
10
11     private int tamNodos;
12
13     private Dispositivo nodA = null;
14     private Dispositivo nodP = null;
15     private boolean arrastrar = false;
16
17     private Dispositivo nod1 = null;
18     private Dispositivo nod2 = null;
19     boolean destino_encontrado = false;
20
21     private int despX,despY;
22     private Point click = new Point();
23
24    public Lienzo( Topologia graf, int tam  ){
25         setGrafo(graf);
26         this.listaDispositivos = graf.getListaDispositivos();
27         this.tamNodos = tam;
28
29         addMouseListener(this);
30         addMouseMotionListener(this);
31         setBackground(Color.WHITE);
32     }
33
34     @Override
35     public void mouseExited( MouseEvent evt ){}
36     @Override
37     public void mouseEntered( MouseEvent evt ){}
38     @Override
39     public void mouseClicked( MouseEvent evt ){}
40     @Override
41     public void mouseMoved( MouseEvent evt ){}
42
43     @Override
44     public void mousePressed( MouseEvent evt ){
45         if(arrastrar) {
46             return;
47         }
48         if( evt.isMetaDown() ){
49             int x = evt.getX(); /**Tomar posicion del mouse*/
50             int y = evt.getY();
51             boolean encontrado=false;
52             ListIterator it = listaDispositivos.listIterator( listaDispositivos.size
                   () );
53             while( it.hasPrevious() && encontrado==false ){
54                 nod1 = (Dispositivo) it.previous();
55                 /**Si hay un clic dentro de un dispositivo*/
56                 if( x>=nod1.getPuntoInicial().x && x<nod1.getPuntoInicial().x+nod1.
                       getTam()
57                 && y>=nod1.getPuntoInicial().y && y<nod1.getPuntoInicial().y+nod1.
                       getTam() ) {
58                     encontrado = true;
59                   }
```

```
60              }
61          }
62          else{
63              int x = evt.getX(); /**Tomar posicion del mouse*/
64              int y = evt.getY();
65              boolean encontrado=false;
66              ListIterator it = listaDispositivos.listIterator( listaDispositivos.size
                    () );
67              while( it.hasPrevious() && encontrado==false ){
68                  nodA = (Dispositivo) it.previous();
69                  /**Si hay un clic dentro de un dispositivo*/
70                  //System.out.println(nodA);
71                  if( x>=nodA.getPuntoInicial().x && x<nodA.getPuntoInicial().x+nodA.
                        getTam()
72                  && y>=nodA.getPuntoInicial().y && y<nodA.getPuntoInicial().y+nodA.
                        getTam() ){
73                      encontrado = true;
74
75                      arrastrar = true;
76                      despX = x - nodA.getPuntoInicial().x;
77                      despY = y - nodA.getPuntoInicial().y;
78                  }
79              }
80          }
81          repintarLienzo();
82      }
83
84      @Override
85      public void mouseDragged( MouseEvent evt ){
86          if(arrastrar ){
87              topologia.setModificado(true);
88              int x = evt.getX()-despX;
89              int y = evt.getY()-despY;
90              if( x<0 ) {
91                  x=0;
92              }
93              if( y<0 ) {
94                  y=0;
95              }
96              nodA.setPuntoinicial( new Point(x,y) );
97              Propiedades.tfIP.setText(nodA.getIp());
98              Propiedades.tfGateway.setText(nodA.getGateway());
99              Propiedades.lListaInterfaces.setText("Interfaz\t\tEstado\tMac\t\tPauqtes
                    env.\tErrores\tPaquetes rec.\tErrores\tVlan\tNombre\n");
100             for(int i =0; i<nodA.getInterfaces().size();i++){
101                 if(nodA.getVlan().isEmpty()){
102                     Propiedades.lListaInterfaces.append(nodA.getInterfaces().get(i) +
                            "\t"+nodA.getEstIf().get(i)+"\t"+nodA.getDirMac().get(i)+"\t"
                            +nodA.getPaqEnv().get(i)+"\t"+nodA.getErrPE().get(i)+"\t"+
                            nodA.getPaqRec().get(i)+"\t"+nodA.getErrPR().get(i)+"\t"+"\n"
                            );
103                 }
104                 else{
105                     Propiedades.lListaInterfaces.append(nodA.getInterfaces().get(i) +
                            "\t"+nodA.getEstIf().get(i)+"\t"+nodA.getDirMac().get(i)+"\t"
                            +nodA.getPaqEnv().get(i)+"\t"+nodA.getErrPE().get(i)+"\t"+
                            nodA.getPaqRec().get(i)+"\t"+nodA.getErrPR().get(i)+"\t"+nodA
                            .getVlan().get(i)+"\t"+nodA.getNVlan().get(i)+"\n");
106
107                 }
108             }
109             if( nodA.getPuntoInicial().x+nodA.getTam() > getWidth() ){/**
                    Redimenzionar el lienzo*/
110                 setPreferredSize( new Dimension(x+nodA.getTam(),getHeight() ) );
111                 revalidate();
112             }
```

```
113             if( nodA.getPuntoInicial().y+nodA.getTam() > getHeight() ){
114                 setPreferredSize( new Dimension( getWidth() ,y+nodA.getTam() ) );
115                 revalidate();
116             }
117             repintarLienzo();
118         }
119     }
120 
121     @Override
122     public void mouseReleased( MouseEvent evt ){
123         if( destino_encontrado & nod1 != nod2 ){/**Agregar conexion*/
124             try{
125                 topologia.agregarConexion( nod1.getIp(),nod2.getIp() );
126             }
127             catch(NumberFormatException ex){
128                 JOptionPane.showMessageDialog(this,"Teclee solo numeros");
129             }
130             catch(NullPointerException ex){}
131         }
132 
133         arrastrar = false;
134         nodP = nodA;
135         nod1=null;
136         nod2=null;
137         destino_encontrado=false;
138         repintarLienzo();
139     }
140 
141     public void crear(Topologia g){
142         setGrafo(g);
143         listaDispositivos = topologia.getListaDispositivos();
144     }
145 
146     private void repintarLienzo(){
147         SwingUtilities.invokeLater(
148             new Runnable(){
149                 @Override
150                 public void run(){
151                     repaint();
152                 }
153             }
154         );
155     }
156 
157     @Override
158     protected void paintComponent( Graphics g ){
159         super.paintComponent(g);
160         if(topologia!=null) {
161             topologia.pintar(g);
162         }
163     }
164 
165     public void setGrafo( Topologia graf ){
166         this.topologia = graf;
167     }
168 }
```

A.3. Código de la clase Propiedades

```
1  import java.awt.*;
2  import javax.swing.*;
3
4
5  public class Propiedades extends JPanel {
6      private JPanel pDispositivo;
7      public static JTextField tfIP,tfGateway;
8
9      private JPanel pPanelInterfaces;
10     public static TextArea lListaInterfaces;
11
12     public Propiedades(){
13         setLayout( new BorderLayout() );
14
15         pDispositivo = new JPanel( new GridLayout(2,1) );
16         pDispositivo.setBorder(BorderFactory.createCompoundBorder(
17         BorderFactory.createTitledBorder("Descripcion del dispositivo"),
18         BorderFactory.createEmptyBorder(5,5,5,5)));
19
20         pDispositivo.add( new JLabel("IP:") );
21         tfIP = new JTextField(10);
22         pDispositivo.add(tfIP);
23
24         pDispositivo.add( new JLabel("Gateway:") );
25         tfGateway = new JTextField(10);
26         pDispositivo.add(tfGateway);
27
28         add(pDispositivo,BorderLayout.WEST);
29
30         pPanelInterfaces = new JPanel( new GridLayout(1,2,5,5) );
31         pPanelInterfaces.setBorder(BorderFactory.createCompoundBorder(
32         BorderFactory.createTitledBorder("Interfaces"),
33         BorderFactory.createEmptyBorder(5,5,5,5)));
34         add(pPanelInterfaces,BorderLayout.CENTER);
35
36         lListaInterfaces = new TextArea();
37         lListaInterfaces.setEditable(false);
38
39         pPanelInterfaces.add( lListaInterfaces, BorderLayout.CENTER );
40         add(pPanelInterfaces,BorderLayout.CENTER);
41     }
42 }
```

A.4. Código de la clase Topologia

```
1  import java.awt.*;
2  import java.io.*;
3  import java.util.*;
4
5  public class Topologia implements Serializable {
6
7      private java.util.List listaDispositivos; /**Lista de Adyacencia*/
8      private boolean modificado;
9
10     public Topologia(){
11         listaDispositivos = Collections.synchronizedList(new LinkedList());
12         this.modificado = false;
13     }
14
15     public Dispositivo agregarDispositivo(int Tipo, String etiqueta,java.util.List<
           String> interfaz,java.util.List<String> mac,java.util.List<String> estado,
           java.util.List<String> PaqEnv,java.util.List<String> ErrPE, java.util.List<
           String> PaqRec,java.util.List<String> ErrPR, java.util.List<String> Vlan,
           java.util.List<String> NVlan,Point puntoInicial,int tam){
16         Dispositivo disp = new Dispositivo( Tipo, etiqueta, interfaz,mac,estado,
               PaqEnv,ErrPE, PaqRec, ErrPR, Vlan, NVlan,puntoInicial,tam);
17         listaDispositivos.add( disp );
18         setModificado(true);
19         return disp;
20     }
21
22     public Dispositivo agregarDispositivo(int Tipo, String etiqueta,String gateway,
           java.util.List<String> interfaz,java.util.List<String> mac,java.util.List<
           String> estado,java.util.List<String>PaqEnv, java.util.List<String> ErrPE,
           java.util.List<String> PaqRec, java.util.List<String> ErrPR,java.util.List<
           String> Vlan, java.util.List<String> NVlan, Point puntoInicial,int tam){
23         Dispositivo disp = new Dispositivo( Tipo, etiqueta, gateway,interfaz,mac,
               estado,PaqEnv,ErrPE, PaqRec, ErrPR, Vlan, NVlan,puntoInicial,tam);
24         listaDispositivos.add( disp );
25         setModificado(true);
26         return disp;
27     }
28
29     public Dispositivo agregarDispositivo(int Tipo, String etiqueta,java.util.List<
           String> interfaz,java.util.List<String> mac,java.util.List<String> estado,
           java.util.List<String> PaqEnv,java.util.List<String> ErrPE, java.util.List<
           String> PaqRec,java.util.List<String> ErrPR,Point puntoInicial,int tam){
30         Dispositivo disp = new Dispositivo( Tipo, etiqueta, interfaz,mac,estado,
               PaqEnv,ErrPE, PaqRec, ErrPR, puntoInicial,tam);
31         listaDispositivos.add( disp );
32         setModificado(true);
33         return disp;
34     }
35
36     public Dispositivo agregarDispositivo(int Tipo, String etiqueta,String gateway,
           java.util.List<String> interfaz,java.util.List<String> mac,java.util.List<
           String> estado,java.util.List<String>PaqEnv, java.util.List<String> ErrPE,
           java.util.List<String> PaqRec, java.util.List<String> ErrPR, Point
           puntoInicial,int tam){
37         Dispositivo disp = new Dispositivo( Tipo, etiqueta, gateway,interfaz,mac,
               estado,PaqEnv,ErrPE, PaqRec, ErrPR, puntoInicial,tam);
38         listaDispositivos.add( disp );
39         setModificado(true);
40         return disp;
41     }
42
43     public int agregarConexion(String o, String d){
```

```
44          Dispositivo arr[] = {null,null};
45          int cond = 0;
46          Iterator it = listaDispositivos.iterator();
47          Dispositivo disp=null;
48          while ( it.hasNext() & cond != 2 ){/**Existen ambos dispositivos?*/
49              disp = (Dispositivo)it.next();
50              if( disp.getIp().equals(o)){
51                  arr[0]=disp;
52                  cond++;
53              }
54              else if( disp.getIp().equals(d) ){
55                  arr[1]=disp;
56                  cond++;
57              }
58          }
59          if( arr[0] != null & arr[1] != null ){//Se encontraron ambos dispostivos
60              arr[0].getListaAd().add( new Conexion( arr[0],arr[1] ) );
61              setModificado(true);
62              return 1; /**Se agrego el dispositivo a la lista de adyacencia*/
63          }
64          else if( arr[0] != null ){
65                  return 3; /**No se encontro el dispositivo destino*/
66              }
67          else {
68                  return 4; /**No se encontro el despositivo origen*/
69          }
70      }
71
72      public void pintar(Graphics g){
73          Iterator disps = listaDispositivos.iterator();
74          Iterator arcos;
75
76          while( disps.hasNext() ){
77              Dispositivo n = (Dispositivo)disps.next();
78              arcos = n.getListaAd().iterator();
79
80              while( arcos.hasNext() ){
81                  Conexion a = (Conexion)arcos.next();
82                  a.pintar(g);
83              }
84          }
85
86          disps = listaDispositivos.iterator();
87          while( disps.hasNext() ){
88              Dispositivo n = (Dispositivo)disps.next();
89              n.pintar(g);
90          }
91
92      }
93
94      public java.util.List getListaDispositivos(){
95          return listaDispositivos;
96      }
97      public boolean getModificado(){
98          return modificado;
99      }
100     public void setModificado( boolean m ){
101         this.modificado = m;
102     }
103 }
```

A.5. Código de la clase Snmpwalk

```
1  import java.io.*;
2  import java.util.*;
3
4  public class Snmpwalk {
5
6    public ArrayList<String> snmpwalk(ArrayList<String> instrucciones) throws Exception
         {
7      String line = "";
8      ArrayList<String> resultado = new ArrayList();
9      for ( int i=0; i < instrucciones.size(); i++ ) {
10       String instruccion = (String)instrucciones.get(i);
11       Process p = Runtime.getRuntime().exec(instruccion);
12       BufferedReader res = new BufferedReader(new InputStreamReader(p.getInputStream
             ()));
13       while ( (line=res.readLine()) != null ) {
14         resultado.add(line);
15       }
16     }
17     return resultado;
18   }
19 }
```

A.6. Código de la clase Dispositivo

```
1  import java.awt.*;
2  import java.util.*;
3  import java.util.List;
4  import javax.swing.ImageIcon;
5
6  public final class Dispositivo {
7
8      Image image;
9      private int tipo;
10     private String ip;
11     private String gateway;
12     private List<String> Interfaces;
13     private List<String> DirMac;
14     private List<String> EstIf;
15     private List<String> PaqEnv;
16     private List<String> ErrPE;
17     private List<String> PaqRec;
18     private List<String> ErrPR;
19     private List<String> Vlan;
20     private List<String> NVlan;
21     private Point puntoinicial;
22     private int tam;
23     private double radio;
24     public java.util.List listaAd;
25
26     public Dispositivo(int Tipo, String Ip, List<String> Interfaces, List<String>
           DirMac, List<String>EstIf, List<String> PaqEnv,List<String> ErrPE, List<
           String> PaqRec,List<String> ErrPR, List<String> Vlan, List<String> NVlan,
           Point pi,int tam){
27         setTipo(Tipo);
28         setIp(Ip);
29         setInterfaces(Interfaces);
30         setDirMac(DirMac);
31         setEstIf(EstIf);
32         setPaqEnv(PaqEnv);
33         setErrPE(ErrPE);
34         setPaqRec(PaqRec);
35         setErrPR(ErrPR);
36         setVlan(Vlan);
37         setNVlan(NVlan);
38                 setPuntoinicial(pi);
39                 setTam(tam);
40         setRadio();
41         listaAd = Collections.synchronizedList(new LinkedList());
42     }
43
44     public Dispositivo(int Tipo, String Ip, String gateway,List<String> Interfaces,
           List<String>DirMac, List<String>EstIf,List<String> PaqEnv,List<String> ErrPE,
            List<String> PaqRec,List<String> ErrPR, List<String> Vlan, List<String>
           NVlan,Point pi,int tam){
45         setTipo(Tipo);
46         setIp(Ip);
47         setInterfaces(Interfaces);
48         setDirMac(DirMac);
49         setEstIf(EstIf);
50         setGateway(gateway);
51         setPaqEnv(PaqEnv);
52         setErrPE(ErrPE);
53         setPaqRec(PaqRec);
54         setErrPR(ErrPR);
55         setVlan(Vlan);
56         setNVlan(NVlan);
```

```
57                  setPuntoinicial(pi);
58                  setTam(tam);
59          setRadio();
60          listaAd = Collections.synchronizedList(new LinkedList());
61      }
62
63      public Dispositivo(int Tipo, String Ip, List<String> Interfaces, List<String>
            DirMac, List<String>EstIf, List<String> PaqEnv,List<String> ErrPE, List<
            String> PaqRec,List<String> ErrPR, Point pi,int tam){
64          setTipo(Tipo);
65          setIp(Ip);
66          setInterfaces(Interfaces);
67          setDirMac(DirMac);
68          setEstIf(EstIf);
69          setPaqEnv(PaqEnv);
70          setErrPE(ErrPE);
71          setPaqRec(PaqRec);
72          setErrPR(ErrPR);
73          setVlan(new LinkedList<String>());
74          setNVlan(new LinkedList<String>());
75                  setPuntoinicial(pi);
76                  setTam(tam);
77          setRadio();
78          listaAd = Collections.synchronizedList(new LinkedList());
79      }
80
81      public Dispositivo(int Tipo, String Ip, String gateway,List<String> Interfaces,
            List<String>DirMac, List<String>EstIf,List<String> PaqEnv,List<String> ErrPE,
             List<String> PaqRec,List<String> ErrPR,Point pi,int tam){
82          setTipo(Tipo);
83          setIp(Ip);
84          setInterfaces(Interfaces);
85          setDirMac(DirMac);
86          setEstIf(EstIf);
87          setGateway(gateway);
88          setPaqEnv(PaqEnv);
89          setErrPE(ErrPE);
90          setPaqRec(PaqRec);
91          setErrPR(ErrPR);
92          setVlan(new LinkedList<String>());
93          setNVlan(new LinkedList<String>());
94                  setPuntoinicial(pi);
95                  setTam(tam);
96          setRadio();
97          listaAd = Collections.synchronizedList(new LinkedList());
98      }
99
100     public int getTipo() {
101         return tipo;
102     }
103
104     public void setTipo(int tipo) {
105         this.tipo = tipo;
106     }
107
108     public String getIp() {
109         return ip;
110     }
111
112     public void setIp(String ip) {
113         this.ip = ip;
114     }
115
116     public String getGateway() {
117         return gateway;
118     }
```

```
119
120     public void setGateway(String gateway) {
121         this.gateway = gateway;
122     }
123
124     public List<String> getInterfaces() {
125         return Interfaces;
126     }
127
128     public void setInterfaces(List<String> Interfaces) {
129         this.Interfaces = Interfaces;
130     }
131
132     public List<String> getDirMac() {
133         return DirMac;
134     }
135
136     public void setDirMac(List<String> DirMac) {
137         this.DirMac = DirMac;
138     }
139
140     public List<String> getEstIf() {
141         return EstIf;
142     }
143
144     public void setEstIf(List<String> EstIf) {
145         this.EstIf = EstIf;
146     }
147     public Point getPuntoinicial() {
148         return puntoinicial;
149     }
150
151     public void setPuntoinicial(Point puntoinicial) {
152         this.puntoinicial = puntoinicial;
153     }
154
155     public int getTam() {
156         return tam;
157     }
158
159     public void setTam(int tam) {
160         this.tam = tam;
161     }
162
163     public double getRadio() {
164         return radio;
165     }
166
167     public void setRadio(double radio) {
168         this.radio = radio;
169     }
170
171     public java.util.List getListaAd() {
172         return listaAd;
173     }
174
175     public void setListaAd(java.util.List listaAd) {
176         this.listaAd = listaAd;
177     }
178
179
180     public void setRadio(){
181         this.radio = getTam() / 2;
182     }
183
184     public Point getPuntoInicial(){
```

```
185             return puntoinicial;
186         }
187 
188         public List<String> getPaqEnv() {
189             return PaqEnv;
190         }
191 
192         public void setPaqEnv(List<String> PaqEnv) {
193             this.PaqEnv = PaqEnv;
194         }
195 
196         public List<String> getErrPE() {
197             return ErrPE;
198         }
199 
200         public void setErrPE(List<String> ErrPE) {
201             this.ErrPE = ErrPE;
202         }
203 
204         public List<String> getPaqRec() {
205             return PaqRec;
206         }
207 
208         public void setPaqRec(List<String> PaqRec) {
209             this.PaqRec = PaqRec;
210         }
211 
212         public List<String> getErrPR() {
213             return ErrPR;
214         }
215 
216         public void setErrPR(List<String> ErrPR) {
217             this.ErrPR = ErrPR;
218         }
219 
220 
221 
222         public List<String> getVlan() {
223             return Vlan;
224         }
225 
226         public void setVlan(List<String> Vlan) {
227             this.Vlan = Vlan;
228         }
229 
230         public List<String> getNVlan() {
231             return NVlan;
232         }
233 
234         public void setNVlan(List<String> NVlan) {
235             this.NVlan = NVlan;
236         }
237 
238 
239 
240         public void pintar( Graphics g ){
241             int pix = getPuntoInicial().x;
242             int piy = getPuntoInicial().y;
243             if (tipo==1){
244                 image = new ImageIcon(getClass().getResource("Computadora.jpg")).getImage
                        ();
245                 g.drawString( getIp(),pix,(piy+55)+7 );
246             }
247             else if (tipo==2){
248                 image = new ImageIcon(getClass().getResource("switch.jpg")).getImage();
249             }
```

```
250             else if (tipo==3){
251                 image = new ImageIcon(getClass().getResource("Router.jpg")).getImage();
252             }
253             g.drawImage(image, pix-15, piy-15, Main.lienzo);
254             g.setColor(Color.BLACK);
255             g.setFont( new Font( "TimesRoman",Font.PLAIN, getTam()*25/100) );
256         }
257 
258     }
```

A.7. Código de la clase Conexion

```
import java.awt.*;
import java.io.*;

public final class Conexion implements Serializable {

    private Dispositivo dispositivoOrigen;
    private Dispositivo dispositivoDestino;
    private Point puntoIn;
    private Point puntoFi;

    public Conexion( Dispositivo o, Dispositivo d ){
        setDispositivoOrigen(o);
        setDispositivoDestino(d);
        setPuntoIn( null );
        setPuntoFi( null );
    }

    public Dispositivo getDispositivoOrigen(){
        return dispositivoOrigen;
    }
    public Dispositivo getDispositivoDestino(){
        return dispositivoDestino;
    }
    public Point getPuntoIn(){
        return puntoIn;
    }
    public Point getPuntoFi(){
        return puntoFi;
    }
    public void setDispositivoOrigen( Dispositivo n ){
        this.dispositivoOrigen = n;
    }
    public void setDispositivoDestino( Dispositivo n ){
        this.dispositivoDestino = n;
    }
    public void setPuntoIn( Point p ){
        this.puntoIn = p;
    }
    public void setPuntoFi( Point p ){
        this.puntoFi = p;
    }

    public void pintar( Graphics g )    {
        /**coordenadas iniciales*/
        int xi = dispositivoOrigen.getPuntoInicial().x;
        int yi = dispositivoOrigen.getPuntoInicial().y;
        /**coordenadas finales*/
        int xf = dispositivoDestino.getPuntoInicial().x;
        int yf = dispositivoDestino.getPuntoInicial().y;
        /**Puntos que representan el centro del icono*/
        int xim = xi+(dispositivoOrigen.getTam()/2); /**centro del nodo origen*/
        int yim = yi+(dispositivoOrigen.getTam()/2);
        int xfm = xf+(dispositivoDestino.getTam()/2); /**centro del nodo destino*/
        int yfm = yf+(dispositivoDestino.getTam()/2);

        int pfx = (int) (xfm ); /**punto final en x donde termina la flecha*/
        int pfy = (int) (yfm ); /**punto final en y donde termina la flecha*/

        if( dispositivoOrigen != dispositivoDestino ){
            dibujarConexion( g, xim, yim, pfx, pfy );
        }

```

```
63        }
64
65        public void dibujarConexion(Graphics g,double x1, double y1, double x2, double y2
              ){
66            g.drawLine((int)x1,(int)y1,(int)x2,(int)y2);
67        }
68    }
```