

Universidad Autónoma Metropolitana

Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería
Licenciatura 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 título 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 direcciones 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

Resumen	III
Agradecimientos	v
Lista de Figuras	vii
Lista de Tablas	ix
1. Introducción	1
1.1. Planteamiento del problema	1
1.2. Marco Teorico	3
1.3. Objetivo general	4
1.4. Objetivos específicos	4
2. Desarrollo	7
2.1. Dispositivos	7
2.1.1. Computadoras	8
2.1.2. Switch	8
2.1.3. Routers	9
2.2. SNMP	9
2.2.1. MIB's	10
2.2.2. Instalación y configuración de SNMP	11
2.3. Solicitudes SNMP	18
2.4. Descubrimiento de dispositivos	22
2.5. Descubrimiento de conexiones	25
3. Resultados y conclusiones	27
3.1. Despliegue de la topología	27
3.2. Conclusiones	35
A. Código fuente	39
A.1. Código de la clase Main	39
A.2. Código de la clase Lienzo	44
A.3. Código de la clase Propiedades	47

A.4. Código de la clase Topologia	48
A.5. Código de la clase Snmpwalk	50
A.6. Código de la clase Dispositivo	51
A.7. Código de la clase Conexion	56

Índice de figuras

1.1. Ejemplos de redes	2
2.1. Clases de dispositivos.	7
2.2. Representación de una Computadora.	8
2.3. Representación de un Switch.	8
2.4. Representación de un Router.	9
2.5. Jerarquía de los MIB's.	10
3.1. Ejemplo de topología descubierta por el sistema.	27
3.2. Ejemplo de desplazamiento de dispositivos.	28
3.3. Información de una computadora.	29
3.4. Información de un Switch.	30
3.5. Información de un Router.	31
3.6. Cambio de tamaño de la barra de información	33
3.7. Barra de información oculta	34

Índice de tablas

2.1. Objetos utilizados en el proyecto.	18
2.2. Opciones snmp.	19
2.3. Opcion general snmp.	19

Capítulo 1

Introducción

1.1. Planteamiento del problema

Una red es un conjunto de computadoras separadas físicamente 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.

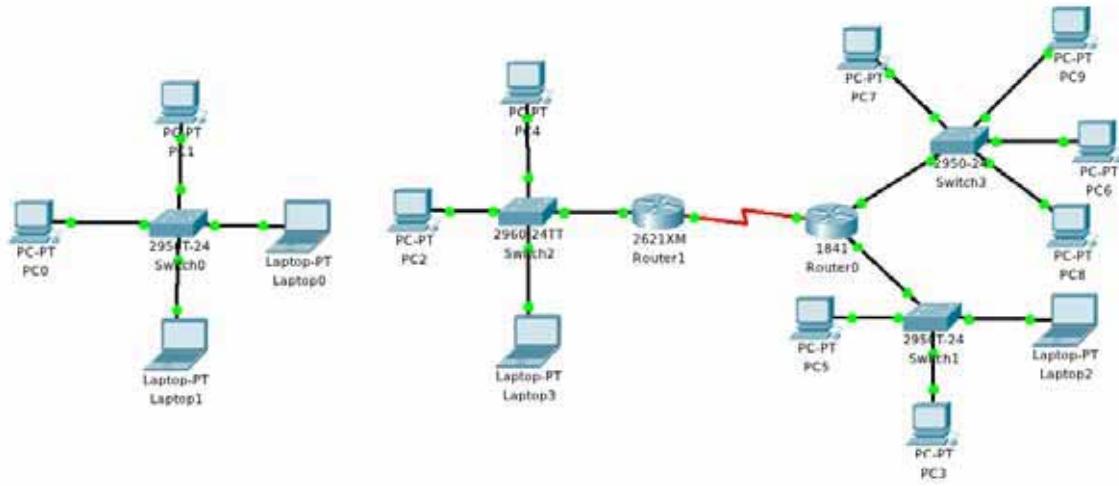


Figura 1.1: Ejemplos de redes.

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 recursos¹ 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 físicamente 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 switch³ o un router⁴, simplemente son sistemas de computo dentro de la red, la diferencia es que cuentan con un hardware⁵ y un sistema operativo⁶ 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.

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

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

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

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

⁵Componentes electrónicos y mecánicos de un dispositivo.

⁶Programa 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 sería 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 interconexión 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 aplicación⁷ 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.

⁷Capa 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.

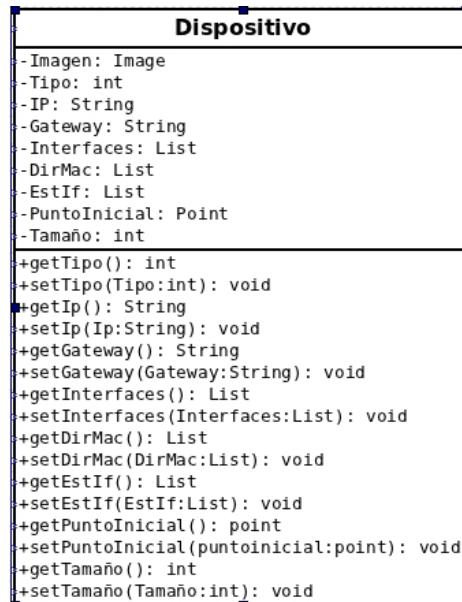


Figura 2.1: Clases de dispositivos.

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 datos¹ 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.

¹Capa 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 enrutamiento³ 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.

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

³Tabla 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.

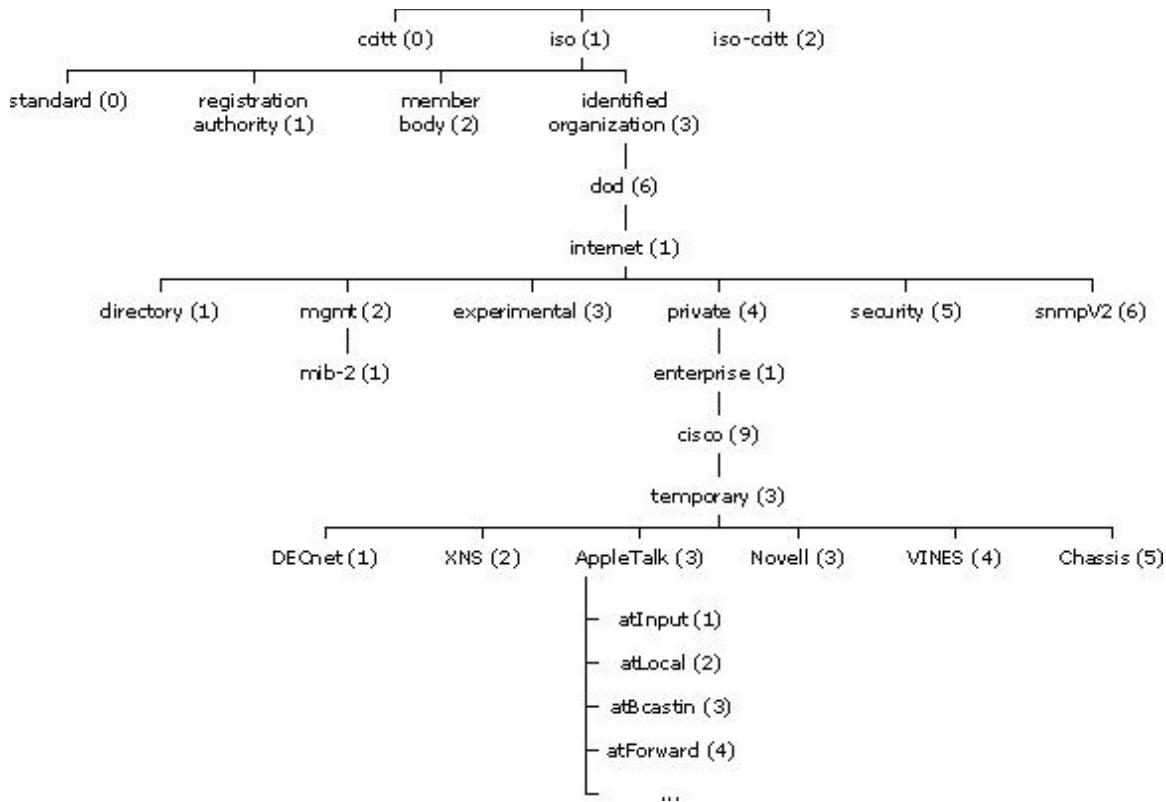


Figura 2.5: Jerarquía de los MIB's.

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 duro 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
#
#view    systemonly included .1.3.6.1.2.1.1          # system + hrSystem groups only
#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 direcciones 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 deberán 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/D.
Switch(config)#snmp-server community
Switch(config)#snmp-server community public RW
Switch(config)#[
```

Podemos verificar que los cambios se hayan realizado, 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 direcciones 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/D
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
1.3.6.1.2.1.1.0	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 [opciones] agente [opciones 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"); /**
10              Descripcion del dispositivo*/
11             Ins.add("snmpwalk -v 2c -c public "+texto+" -Oq ipAdEntAddr"); /**
12                 Ip
13                 del dispositivo*/
14             Ins.add("snmpwalk -v 2c -c public "+texto+" -Oq ipRouteNextHop"); /**
15                 Gateway*/
16             Ins.add("snmpwalk -v 2c -c public "+texto+" -Oq ifDescr"); /**
17                 Interfaces*/
18             Ins.add("snmpwalk -v 2c -c public "+texto+" -Oq ifPhysAddress"); /**
19                 Mac de la interfaz*/
20             Ins.add("snmpwalk -v 2c -c public "+texto+" -Oq ifOperStatus"); /**
21                 Estado de la interfaz*/
22             Ins.add("snmpwalk -v 2c -c public "+texto+" -Oq ifInUcastPkts"); /**
23                 Paquetes recibidos por la interfaz*/
24             Ins.add("snmpwalk -v 2c -c public "+texto+" -Oq ifInErrors"); /**
25                 Errores en los paquetes recibidos*/
26             Ins.add("snmpwalk -v 2c -c public "+texto+" -Oq ifOutUcastPkts"); /**
27                 Paquetes enviados por la interfaz*/
28             Ins.add("snmpwalk -v 2c -c public "+texto+" -Oq ifOutErrors"); /**
29                 Errores en los paquetes enviados*/
30             Ins.add("snmpwalk -v 2c -c public "+texto+" -Oq vtpVlanIfIndex"); /**
31                 Numero de la Vlan a que pertenece la interfaz*/
32             Ins.add("snmpwalk -v 2c -c public "+texto+" -Oq vtpVlanName"); /**
33                 Nombre de la Vlan a que pertenece la interfaz*/
34
35             Snmpwalk t = new Snmpwalk();
36             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      {
5          String line = "";
6          ArrayList<String> resultado = new ArrayList();
7          for ( int i=0; i < instrucciones.size(); i++ ) {
8              String instrucion = (String)instrucciones.get(i);
9              Process p = Runtime.getRuntime().exec(instrucion);
10             BufferedReader res = new BufferedReader(new InputStreamReader(p.getInputStream
11                         ()));
12             while ( (line=res.readLine()) != null ) {
13                 resultado.add(line);
14             }
15         }
16         return resultado;
17     }

```

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,
6                                         EInterfaces,PaqEnv,ErrPE,PaqRec,ErrPR,new Point(x,y),
7                                         tamDisp );
8     }
9     else {
10        topologia.agregarDispositivo(Tipo, Ip[1],Gw[1],Interfaces,
11                                      DirMac,EInterfaces,PaqEnv,ErrPE,PaqRec,ErrPR,new Point(x,
12                                      y), tamDisp );
13    }
14    else{
15        if(Gw==null){
16            topologia.agregarDispositivo(Tipo, Ip[1],Interfaces,DirMac,
17                                          EInterfaces,PaqEnv,ErrPE,PaqRec,ErrPR,Vlan,NVlan,new
18                                          Point(x,y), tamDisp );
19        }
20        else{
21            topologia.agregarDispositivo(Tipo, Ip[1],Gw[1],Interfaces,
22                                          DirMac,EInterfaces,PaqEnv,ErrPE,PaqRec,ErrPR,Vlan,NVlan,
23                                          new Point(x,y), tamDisp );
24        }
25    }
26 }

```

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 únicamente dispositivos como Switches y Routers pueden tener configurada una Vlan y los Routers no tienen gateway, o bien dentro de una red local alguna computadora podría no tenerlo configurado.

A continuación se 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,
8      List<String> mac, List<String> estado, List<String> PaqEnv, List<String> ErrPE,
9      List<String> PaqRec, List<String> ErrPR, List<String> Vlan, List<String> NVlan,
10     Point puntoInicial, int tam){
11     Dispositivo disp = new Dispositivo( Tipo, etiqueta, interfaz,mac,estado,
12         PaqEnv,ErrPE, PaqRec, ErrPR, Vlan, NVlan,puntoInicial,tam);
13     listaDispositivos.add( disp );
14     return disp;
15 }
16
17 public Dispositivo agregarDispositivo(int Tipo, String etiqueta, String gateway,
18     List<String> interfaz, List<String> mac, List<String> estado, List<String> PaqEnv
19     , List<String> ErrPE, List<String> PaqRec, List<String> ErrPR, List<String>
20     Vlan, List<String> NVlan, Point puntoInicial, int tam){
21     Dispositivo disp = new Dispositivo( Tipo, etiqueta, gateway,interfaz,mac,
22         estado,PaqEnv,ErrPE, PaqRec, ErrPR, Vlan, NVlan,puntoInicial,tam);
23     listaDispositivos.add( disp );
24     return disp;
25 }
26
27 public Dispositivo agregarDispositivo(int Tipo, String etiqueta, List<String>
28     interfaz, List<String> mac, List<String> estado, List<String> PaqEnv,
29     List<String> ErrPE, List<String> PaqRec, List<String> ErrPR, Point
      puntoInicial, int tam){
30     Dispositivo disp = new Dispositivo( Tipo, etiqueta, interfaz,mac,estado,
31         PaqEnv,ErrPE, PaqRec, ErrPR, puntoInicial,tam);
32     listaDispositivos.add( disp );
33     return disp;
34 }
```

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                  ();
7              g.drawString( getIp(),pix,(piy+55)+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:

```

1  FileReader lector = new FileReader(archivo);
2  BufferedReader contenido = new BufferedReader(lector);
3      while ((ip = contenido.readLine()) != null) {
4          Ins.add("snmpwalk -v 2c -c public "+ip+" -Oq ipAdEntAddr");
5          Ins.add("snmpwalk -v 2c -c public "+ip+" -Oq ipNetToMediaNetAddress");
6
7      Snmpwalk t = new Snmpwalk();
8      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 conexión entre ambos dispositivos:

```

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

```

La siguiente función es la encargada de agregar la conexión:

```

1  public int agregarConexion(String o, String d){
2      Dispositivo arr[] = {null,null};
3      int cond = 0;
4      Iterator it = listaDispositivos.iterator();
5      Dispositivo disp=null;
6          while ( it.hasNext() & cond != 2 ){/**Existen ambos dispositivos?*/
7              disp = (Dispositivo)it.next();
8              if( disp.getIp().equals(o)){
9                  arr[0]=disp;
10                 cond++;
11             }
12             else if( disp.getIp().equals(d) ){
13                 arr[1]=disp;
14                 cond++;
15             }
16         }
17         if( arr[0] != null & arr[1] != null ){//Se encontraron ambos dispositivos
18             arr[0].getListaAd().add( new Conexion( arr[0],arr[1] ) );
19             setModificado(true);
20             return 1; /**Se agrego el dispositivo a la lista de adyacencia*/
21         }
22         else if( arr[0] != null ){
23             return 3; /**No se encontro el dispositivo destino*/
24         }
25         else {
26             return 4; /**No se encontro el dispositivo origen*/
27         }
28     }

```

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.

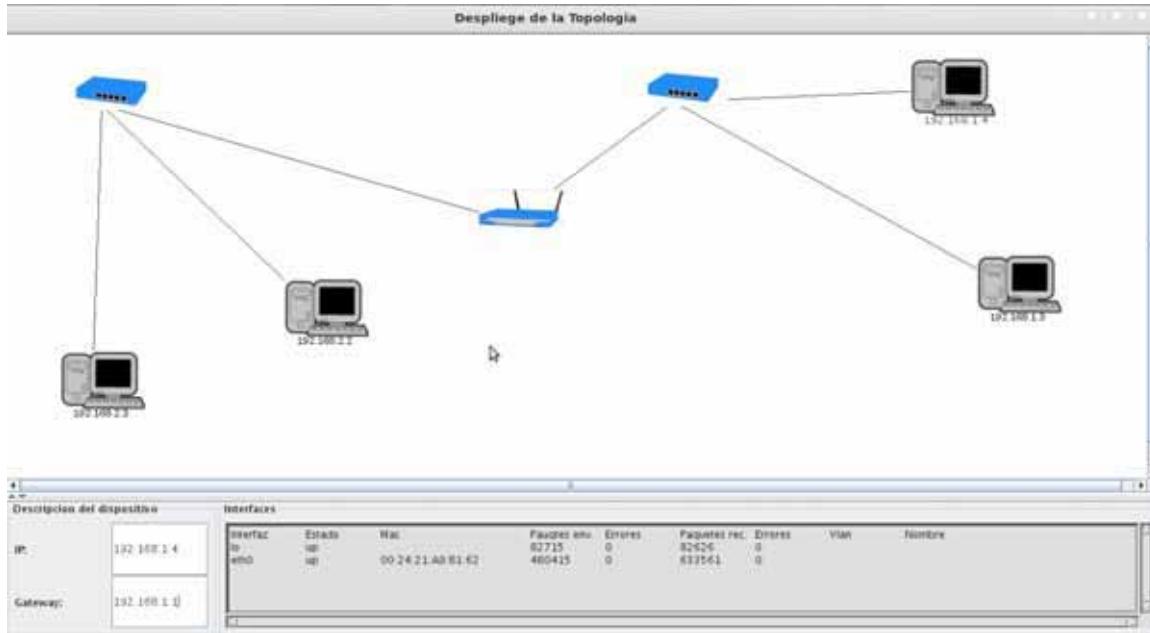


Figura 3.1: Ejemplo de topología descubierta por el sistema.

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.

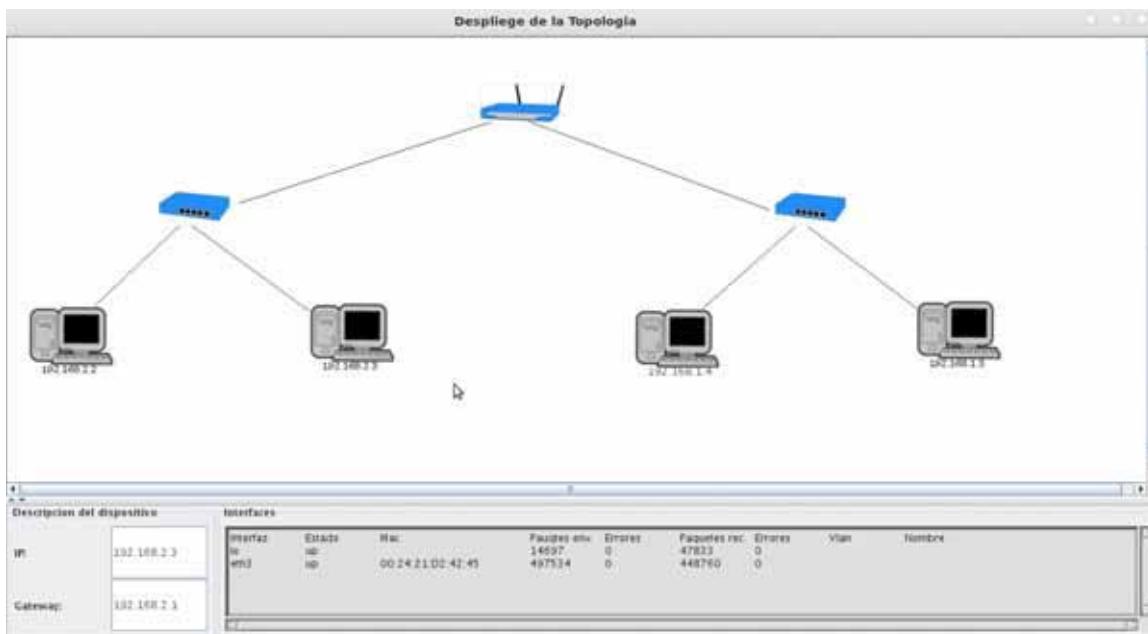


Figura 3.2: Ejemplo de desplazamiento de dispositivos.

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.

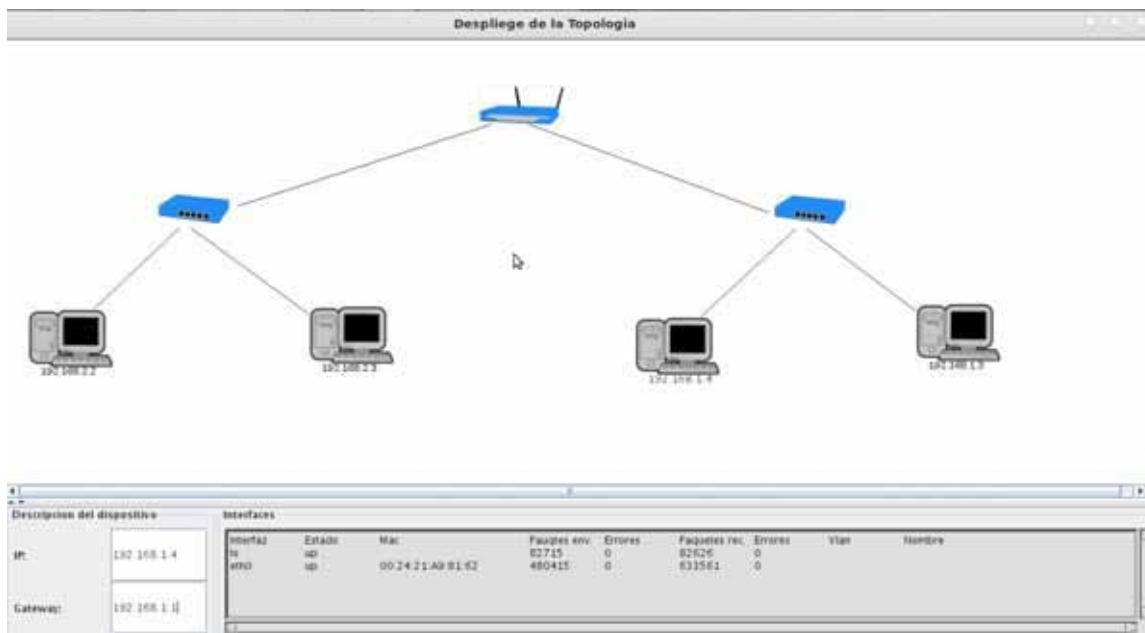


Figura 3.3: Información de una computadora.

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

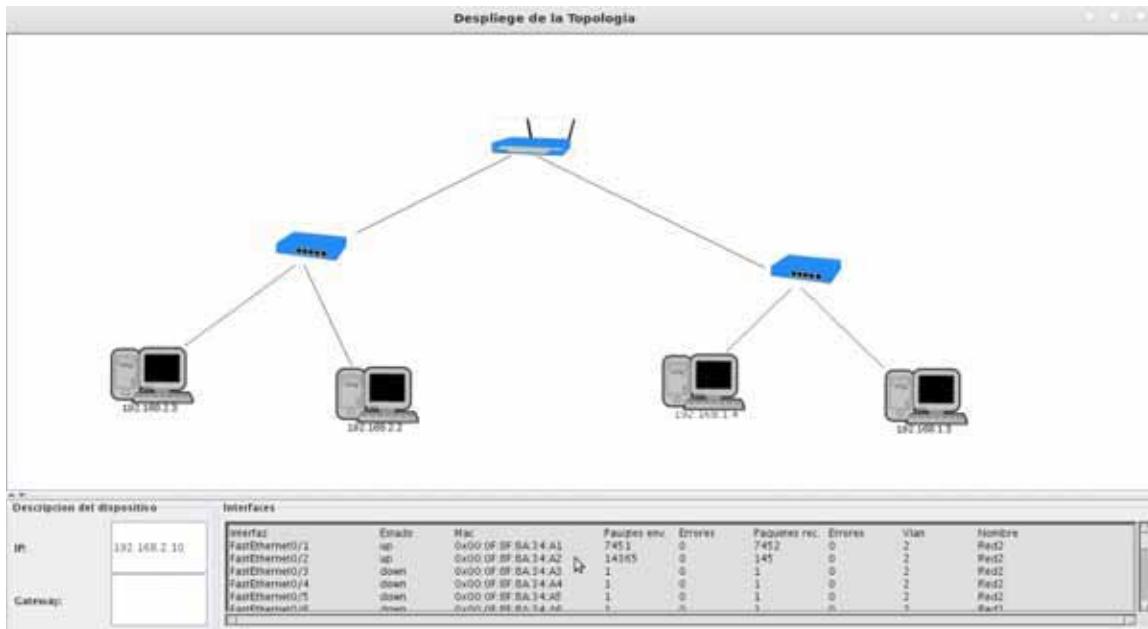


Figura 3.4: Información de un Switch.

Y de un Router. Figura 3.5.

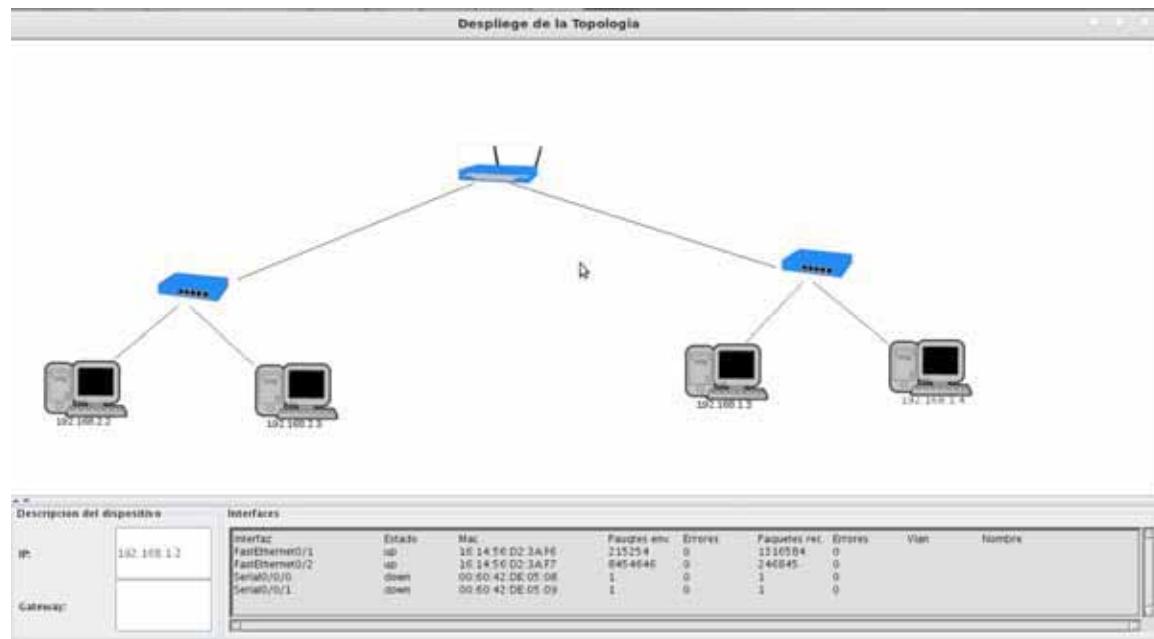


Figura 3.5: Información de un Router.

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             () );
11         while( it.hasPrevious() && encontrado==false ){
12             nod1 = (Dispositivo) it.previous();
13             /*Si hay un clic dentro de un dispositivo*/
14             if( x>=nod1.getPuntoInicial().x && x<nod1.getPuntoInicial().x+nod1.
15                 getTam() && y>=nod1.getPuntoInicial().y && y<nod1.getPuntoInicial().y+nod1.
16                 getTam() ) {
17                 encontrado = true;
18             }
19         }
20     } else{
21         int x = evt.getX(); /*Tomar posicion del mouse*/
22         int y = evt.getY();
23         boolean encontrado=false;
24         ListIterator it = listaDispositivos.listIterator( listaDispositivos.size()
25             () );
26         while( it.hasPrevious() && encontrado==false ){
27             nodA = (Dispositivo) it.previous();
28             /*Si hay un clic dentro de un dispositivo*/
29             if( x>=nodA.getPuntoInicial().x && x<nodA.getPuntoInicial().x+nodA.
30                 getTam() && y>=nodA.getPuntoInicial().y && y<nodA.getPuntoInicial().y+nodA.
31                 getTam() ) {
32                 encontrado = true;
33
34                 arrastrar = true;
35                 despX = x - nodA.getPuntoInicial().x;
36                 despY = y - nodA.getPuntoInicial().y;
37             }
38         }
39     }
40     repaint();
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.llListaInterfaces.setText("");
56             for(int i =0; i<nodA.getInterfaces().size();i++){
57                 Propiedades.llListaInterfaces.append(nodA.getInterfaces().get(i) +"\t"

```

```
58         +nodA.getEstIf().get(i)+"\t"+nodA.getDirMac().get(i)+"\n");
59     }
60
61     if( nodA.getPuntoInicial().x+nodA.getTam() > getWidth() ){/***
62         Redimensionar el lienzo*/
63     setPreferredSize( new Dimension(x+nodA.getTam(),getHeight() ) );
64     revalidate();
65
66     if( nodA.getPuntoInicial().y+nodA.getTam() > getHeight() ){
67         setPreferredSize( new Dimension( getWidth() ,y+nodA.getTam() ) );
68         revalidate();
69     }
70     repaintLienzo();
71 }
```

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.

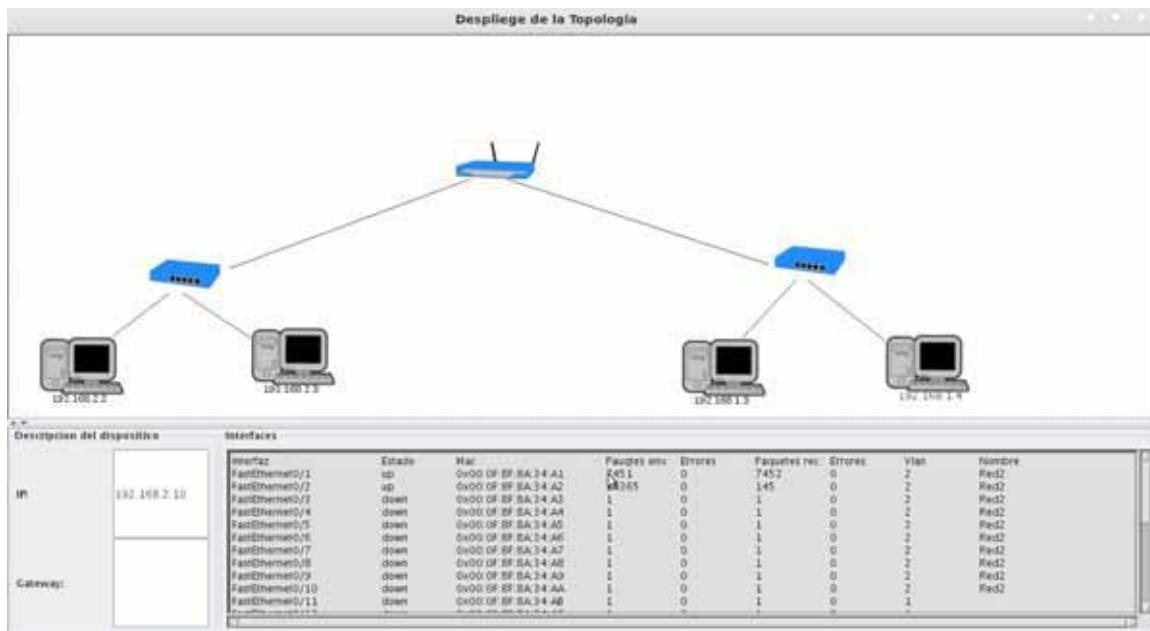


Figura 3.6: Cambio de tamaño de la barra de información

Y también puede ser ocultada. Figura 3.7.

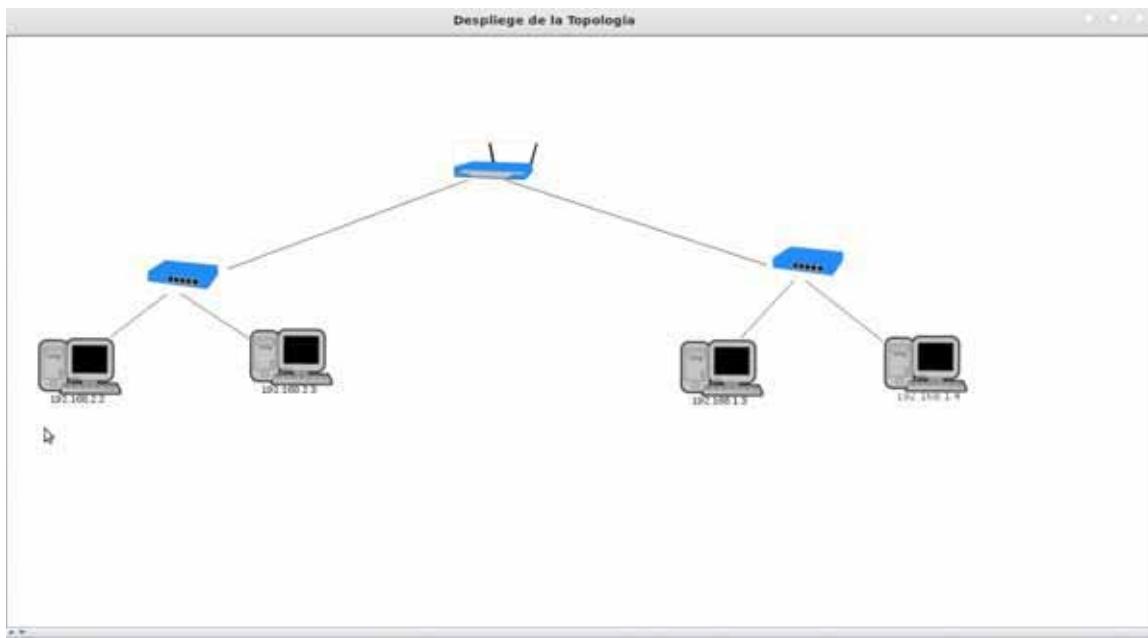


Figura 3.7: Barra de información oculta

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 específica.

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 descripcion 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,
102                                         DirMac,EInterfaces,PaqEnv,ErrPE,PaqRec,ErrPR,new
103                                         Point(x,y), tamDisp );
104                         }
105                     else {
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
878
879
879
880
881
882
883
884
885
886
887
888
888
889
889
890
891
892
893
894
895
896
897
897
898
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
988
989
989
990
991
992
993
994
995
996
997
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1088
1089
1089
1090
1091
1092
1093
1094
1095
1095
1096
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1177
1178
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1191
1192
1193
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1377
1378
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1391
1392
1393
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1417
1418
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1427
1428
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1437
1438
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1447
1448
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1457
1458
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1467
1468
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1477
1478
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1517
1518
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1527
1528
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1537
1538
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1547
1548
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1557
1558
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1567
1568
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1577
1578
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1607
1608
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1617
1618
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1627
1628
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1637
1638
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1647
1648
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1657
1658
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1667
1668
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1677
1678
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1686
1687
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1695
1695
1696
1696
1697
1697
1698
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1707
1708
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1717
1718
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1727
1728
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1737
1738
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1747
1748
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1757
1758
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1767
1768
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1777
1778
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1786
1787
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1795
1796
1796
1797
1797
1798
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1807
1808
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1817
1818
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1827
1828
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1837
1838
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1847
1848
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1857
1858
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1867
1868
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1877
1878
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1886
1887
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1895
1896
1896
1897
1897
1898
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1907
1908
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1917
1918
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1927
1928
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1937
1938
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1947
1948
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1957
1958
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1967
1968
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1977
1978
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1986
1987
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1995
1996
1996
1997
1997
1998
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2007
2008
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2017
2018
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2027
2028
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2037
2038
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
204
```

```

104             topologia.agregarDispositivo(Tipo, Ip[1],Gw[1],
105                                         Interfaces,DirMac,EInterfaces,PaqEnv,ErrPE,PaqRec
106                                         ,ErrPR,new Point(x,y), tamDisp );
107         }
108     }else{
109         if(Gw==null){
110             topologia.agregarDispositivo(Tipo, Ip[1],Interfaces,
111                                         DirMac,EInterfaces,PaqEnv,ErrPE,PaqRec,ErrPR,Vlan
112                                         ,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 catch(Exception e){
219     System.out.println(e);
220 }
221
222 try{
223     FileReader lector2=new FileReader(archivo);
224     BufferedReader contenido2=new BufferedReader(lector2);
225     while((texto=contenido2.readLine())!=null){
226         if (texto.indexOf ("ipAdEntAddr") != -1){
227             if (texto.indexOf ("ipAdEntAddr.127.0.0.1") != -1){}
228             else{
229

```

```
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 catch(Exception e){
239     System.out.println(e);
240 }
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                () );
54            while( it.hasPrevious() && encontrado==false ){
55                nod1 = (Dispositivo) it.previous();
56                /**Si hay un clic dentro de un dispositivo*/
57                if( x>=nod1.getPuntoInicial().x && x<nod1.getPuntoInicial().x+nod1.
58                    getTam()
59                    && y>=nod1.getPuntoInicial().y && y<nod1.getPuntoInicial().y+nod1.
60                    getTam() ) {
61                        encontrado = true;
62                    }
63            }
64        }
65    }
66
67    @Override
68    public void mouseDragged( MouseEvent evt ){
69        if(arrastrar) {
70            int dx = evt.getX() - click.x;
71            int dy = evt.getY() - click.y;
72            click.setLocation( evt.getX(), evt.getY() );
73            if( nodA != null ) {
74                nodA.setPuntoFinal( new Point( click.x + dx, click.y + dy ) );
75            }
76        }
77    }
78
79    @Override
80    public void mouseReleased( MouseEvent evt ){
81        if(arrastrar) {
82            if( nodA != null ) {
83                nodP = nodA;
84                nodA = null;
85            }
86        }
87    }
88
89    public void setGrafo(Topologia graf) {
90        this.graf = graf;
91    }
92
93    public Topologia getGrafo() {
94        return graf;
95    }
96
97    public void setTamNodos(int tam) {
98        this.tamNodos = tam;
99    }
100
101    public int getTamNodos() {
102        return tamNodos;
103    }
104
105    public void setListaDispositivos(List listaDispositivos) {
106        this.listaDispositivos = listaDispositivos;
107    }
108
109    public List getListaDispositivos() {
110        return listaDispositivos;
111    }
112
113    public void setDestino_encontrado(boolean destino_encontrado) {
114        this.destino_encontrado = destino_encontrado;
115    }
116
117    public boolean getDestino_encontrado() {
118        return destino_encontrado;
119    }
120
121    public void setArrostrar(boolean arrastrar) {
122        this.arrastrar = arrastrar;
123    }
124
125    public boolean getArrostrar() {
126        return arrastrar;
127    }
128
129    public void setNodA(Dispositivo nodA) {
130        this.nodA = nodA;
131    }
132
133    public Dispositivo getNodA() {
134        return nodA;
135    }
136
137    public void setNodP(Dispositivo nodP) {
138        this.nodP = nodP;
139    }
140
141    public Dispositivo getNodP() {
142        return nodP;
143    }
144
145    public void setDespX(int despX) {
146        this.despX = despX;
147    }
148
149    public void setDespY(int despY) {
150        this.despY = despY;
151    }
152
153    public void setClick(Point click) {
154        this.click = click;
155    }
156
157    public Point getClick() {
158        return click;
159    }
160
161    public void setTopologia(Topologia topologia) {
162        this.topologia = topologia;
163    }
164
165    public Topologia getTopologia() {
166        return topologia;
167    }
168
169    public void setTam(int tam) {
170        this.tam = tam;
171    }
172
173    public int getTam() {
174        return tam;
175    }
176
177    public void setDestino_encontrado(boolean destino_encontrado) {
178        this.destino_encontrado = destino_encontrado;
179    }
180
181    public boolean getDestino_encontrado() {
182        return destino_encontrado;
183    }
184
185    public void setArrostrar(boolean arrastrar) {
186        this.arrastrar = arrastrar;
187    }
188
189    public boolean getArrostrar() {
190        return arrastrar;
191    }
192
193    public void setNod1(Dispositivo nod1) {
194        this.nod1 = nod1;
195    }
196
197    public Dispositivo getNod1() {
198        return nod1;
199    }
200
201    public void setNod2(Dispositivo nod2) {
202        this.nod2 = nod2;
203    }
204
205    public Dispositivo getNod2() {
206        return nod2;
207    }
208
209    public void setDestino_encontrado(boolean destino_encontrado) {
210        this.destino_encontrado = destino_encontrado;
211    }
212
213    public boolean getDestino_encontrado() {
214        return destino_encontrado;
215    }
216
217    public void setArrostrar(boolean arrastrar) {
218        this.arrastrar = arrastrar;
219    }
220
221    public boolean getArrostrar() {
222        return arrastrar;
223    }
224
225    public void setNodA(Dispositivo nodA) {
226        this.nodA = nodA;
227    }
228
229    public Dispositivo getNodA() {
230        return nodA;
231    }
232
233    public void setNodP(Dispositivo nodP) {
234        this.nodP = nodP;
235    }
236
237    public Dispositivo getNodP() {
238        return nodP;
239    }
240
241    public void setDespX(int despX) {
242        this.despX = despX;
243    }
244
245    public void setDespY(int despY) {
246        this.despY = despY;
247    }
248
249    public void setClick(Point click) {
250        this.click = click;
251    }
252
253    public Point getClick() {
254        return click;
255    }
256
257    public void setTopologia(Topologia topologia) {
258        this.topologia = topologia;
259    }
260
261    public Topologia getTopologia() {
262        return topologia;
263    }
264
265    public void setTam(int tam) {
266        this.tam = tam;
267    }
268
269    public int getTam() {
270        return tam;
271    }
272
273    public void setDestino_encontrado(boolean destino_encontrado) {
274        this.destino_encontrado = destino_encontrado;
275    }
276
277    public boolean getDestino_encontrado() {
278        return destino_encontrado;
279    }
280
281    public void setArrostrar(boolean arrastrar) {
282        this.arrastrar = arrastrar;
283    }
284
285    public boolean getArrostrar() {
286        return arrastrar;
287    }
288
289    public void setNod1(Dispositivo nod1) {
290        this.nod1 = nod1;
291    }
292
293    public Dispositivo getNod1() {
294        return nod1;
295    }
296
297    public void setNod2(Dispositivo nod2) {
298        this.nod2 = nod2;
299    }
300
301    public Dispositivo getNod2() {
302        return nod2;
303    }
304
305    public void setDestino_encontrado(boolean destino_encontrado) {
306        this.destino_encontrado = destino_encontrado;
307    }
308
309    public boolean getDestino_encontrado() {
310        return destino_encontrado;
311    }
312
313    public void setArrostrar(boolean arrastrar) {
314        this.arrastrar = arrastrar;
315    }
316
317    public boolean getArrostrar() {
318        return arrastrar;
319    }
320
321    public void setNodA(Dispositivo nodA) {
322        this.nodA = nodA;
323    }
324
325    public Dispositivo getNodA() {
326        return nodA;
327    }
328
329    public void setNodP(Dispositivo nodP) {
330        this.nodP = nodP;
331    }
332
333    public Dispositivo getNodP() {
334        return nodP;
335    }
336
337    public void setDespX(int despX) {
338        this.despX = despX;
339    }
340
341    public void setDespY(int despY) {
342        this.despY = despY;
343    }
344
345    public void setClick(Point click) {
346        this.click = click;
347    }
348
349    public Point getClick() {
350        return click;
351    }
352
353    public void setTopologia(Topologia topologia) {
354        this.topologia = topologia;
355    }
356
357    public Topologia getTopologia() {
358        return topologia;
359    }
360
361    public void setTam(int tam) {
362        this.tam = tam;
363    }
364
365    public int getTam() {
366        return tam;
367    }
368
369    public void setDestino_encontrado(boolean destino_encontrado) {
370        this.destino_encontrado = destino_encontrado;
371    }
372
373    public boolean getDestino_encontrado() {
374        return destino_encontrado;
375    }
376
377    public void setArrostrar(boolean arrastrar) {
378        this.arrastrar = arrastrar;
379    }
380
381    public boolean getArrostrar() {
382        return arrastrar;
383    }
384
385    public void setNod1(Dispositivo nod1) {
386        this.nod1 = nod1;
387    }
388
389    public Dispositivo getNod1() {
390        return nod1;
391    }
392
393    public void setNod2(Dispositivo nod2) {
394        this.nod2 = nod2;
395    }
396
397    public Dispositivo getNod2() {
398        return nod2;
399    }
400
401    public void setDestino_encontrado(boolean destino_encontrado) {
402        this.destino_encontrado = destino_encontrado;
403    }
404
405    public boolean getDestino_encontrado() {
406        return destino_encontrado;
407    }
408
409    public void setArrostrar(boolean arrastrar) {
410        this.arrastrar = arrastrar;
411    }
412
413    public boolean getArrostrar() {
414        return arrastrar;
415    }
416
417    public void setNodA(Dispositivo nodA) {
418        this.nodA = nodA;
419    }
420
421    public Dispositivo getNodA() {
422        return nodA;
423    }
424
425    public void setNodP(Dispositivo nodP) {
426        this.nodP = nodP;
427    }
428
429    public Dispositivo getNodP() {
430        return nodP;
431    }
432
433    public void setDespX(int despX) {
434        this.despX = despX;
435    }
436
437    public void setDespY(int despY) {
438        this.despY = despY;
439    }
440
441    public void setClick(Point click) {
442        this.click = click;
443    }
444
445    public Point getClick() {
446        return click;
447    }
448
449    public void setTopologia(Topologia topologia) {
450        this.topologia = topologia;
451    }
452
453    public Topologia getTopologia() {
454        return topologia;
455    }
456
457    public void setTam(int tam) {
458        this.tam = tam;
459    }
460
461    public int getTam() {
462        return tam;
463    }
464
465    public void setDestino_encontrado(boolean destino_encontrado) {
466        this.destino_encontrado = destino_encontrado;
467    }
468
469    public boolean getDestino_encontrado() {
470        return destino_encontrado;
471    }
472
473    public void setArrostrar(boolean arrastrar) {
474        this.arrastrar = arrastrar;
475    }
476
477    public boolean getArrostrar() {
478        return arrastrar;
479    }
480
481    public void setNod1(Dispositivo nod1) {
482        this.nod1 = nod1;
483    }
484
485    public Dispositivo getNod1() {
486        return nod1;
487    }
488
489    public void setNod2(Dispositivo nod2) {
490        this.nod2 = nod2;
491    }
492
493    public Dispositivo getNod2() {
494        return nod2;
495    }
496
497    public void setDestino_encontrado(boolean destino_encontrado) {
498        this.destino_encontrado = destino_encontrado;
499    }
500
501    public boolean getDestino_encontrado() {
502        return destino_encontrado;
503    }
504
505    public void setArrostrar(boolean arrastrar) {
506        this.arrastrar = arrastrar;
507    }
508
509    public boolean getArrostrar() {
510        return arrastrar;
511    }
512
513    public void setNodA(Dispositivo nodA) {
514        this.nodA = nodA;
515    }
516
517    public Dispositivo getNodA() {
518        return nodA;
519    }
520
521    public void setNodP(Dispositivo nodP) {
522        this.nodP = nodP;
523    }
524
525    public Dispositivo getNodP() {
526        return nodP;
527    }
528
529    public void setDespX(int despX) {
530        this.despX = despX;
531    }
532
533    public void setDespY(int despY) {
534        this.despY = despY;
535    }
536
537    public void setClick(Point click) {
538        this.click = click;
539    }
540
541    public Point getClick() {
542        return click;
543    }
544
545    public void setTopologia(Topologia topologia) {
546        this.topologia = topologia;
547    }
548
549    public Topologia getTopologia() {
550        return topologia;
551    }
552
553    public void setTam(int tam) {
554        this.tam = tam;
555    }
556
557    public int getTam() {
558        return tam;
559    }
560
561    public void setDestino_encontrado(boolean destino_encontrado) {
562        this.destino_encontrado = destino_encontrado;
563    }
564
565    public boolean getDestino_encontrado() {
566        return destino_encontrado;
567    }
568
569    public void setArrostrar(boolean arrastrar) {
570        this.arrastrar = arrastrar;
571    }
572
573    public boolean getArrostrar() {
574        return arrastrar;
575    }
576
577    public void setNod1(Dispositivo nod1) {
578        this.nod1 = nod1;
579    }
580
581    public Dispositivo getNod1() {
582        return nod1;
583    }
584
585    public void setNod2(Dispositivo nod2) {
586        this.nod2 = nod2;
587    }
588
589    public Dispositivo getNod2() {
590        return nod2;
591    }
592
593    public void setDestino_encontrado(boolean destino_encontrado) {
594        this.destino_encontrado = destino_encontrado;
595    }
596
597    public boolean getDestino_encontrado() {
598        return destino_encontrado;
599    }
600
601    public void setArrostrar(boolean arrastrar) {
602        this.arrastrar = arrastrar;
603    }
604
605    public boolean getArrostrar() {
606        return arrastrar;
607    }
608
609    public void setNodA(Dispositivo nodA) {
610        this.nodA = nodA;
611    }
612
613    public Dispositivo getNodA() {
614        return nodA;
615    }
616
617    public void setNodP(Dispositivo nodP) {
618        this.nodP = nodP;
619    }
620
621    public Dispositivo getNodP() {
622        return nodP;
623    }
624
625    public void setDespX(int despX) {
626        this.despX = despX;
627    }
628
629    public void setDespY(int despY) {
630        this.despY = despY;
631    }
632
633    public void setClick(Point click) {
634        this.click = click;
635    }
636
637    public Point getClick() {
638        return click;
639    }
640
641    public void setTopologia(Topologia topologia) {
642        this.topologia = topologia;
643    }
644
645    public Topologia getTopologia() {
646        return topologia;
647    }
648
649    public void setTam(int tam) {
650        this.tam = tam;
651    }
652
653    public int getTam() {
654        return tam;
655    }
656
657    public void setDestino_encontrado(boolean destino_encontrado) {
658        this.destino_encontrado = destino_encontrado;
659    }
660
661    public boolean getDestino_encontrado() {
662        return destino_encontrado;
663    }
664
665    public void setArrostrar(boolean arrastrar) {
666        this.arrastrar = arrastrar;
667    }
668
669    public boolean getArrostrar() {
670        return arrastrar;
671    }
672
673    public void setNod1(Dispositivo nod1) {
674        this.nod1 = nod1;
675    }
676
677    public Dispositivo getNod1() {
678        return nod1;
679    }
680
681    public void setNod2(Dispositivo nod2) {
682        this.nod2 = nod2;
683    }
684
685    public Dispositivo getNod2() {
686        return nod2;
687    }
688
689    public void setDestino_encontrado(boolean destino_encontrado) {
690        this.destino_encontrado = destino_encontrado;
691    }
692
693    public boolean getDestino_encontrado() {
694        return destino_encontrado;
695    }
696
697    public void setArrostrar(boolean arrastrar) {
698        this.arrastrar = arrastrar;
699    }
700
701    public boolean getArrostrar() {
702        return arrastrar;
703    }
704
705    public void setNodA(Dispositivo nodA) {
706        this.nodA = nodA;
707    }
708
709    public Dispositivo getNodA() {
710        return nodA;
711    }
712
713    public void setNodP(Dispositivo nodP) {
714        this.nodP = nodP;
715    }
716
717    public Dispositivo getNodP() {
718        return nodP;
719    }
720
721    public void setDespX(int despX) {
722        this.despX = despX;
723    }
724
725    public void setDespY(int despY) {
726        this.despY = despY;
727    }
728
729    public void setClick(Point click) {
730        this.click = click;
731    }
732
733    public Point getClick() {
734        return click;
735    }
736
737    public void setTopologia(Topologia topologia) {
738        this.topologia = topologia;
739    }
740
741    public Topologia getTopologia() {
742        return topologia;
743    }
744
745    public void setTam(int tam) {
746        this.tam = tam;
747    }
748
749    public int getTam() {
750        return tam;
751    }
752
753    public void setDestino_encontrado(boolean destino_encontrado) {
754        this.destino_encontrado = destino_encontrado;
755    }
756
757    public boolean getDestino_encontrado() {
758        return destino_encontrado;
759    }
760
761    public void setArrostrar(boolean arrastrar) {
762        this.arrastrar = arrastrar;
763    }
764
765    public boolean getArrostrar() {
766        return arrastrar;
767    }
768
769    public void setNod1(Dispositivo nod1) {
770        this.nod1 = nod1;
771    }
772
773    public Dispositivo getNod1() {
774        return nod1;
775    }
776
777    public void setNod2(Dispositivo nod2) {
778        this.nod2 = nod2;
779    }
780
781    public Dispositivo getNod2() {
782        return nod2;
783    }
784
785    public void setDestino_encontrado(boolean destino_encontrado) {
786        this.destino_encontrado = destino_encontrado;
787    }
788
789    public boolean getDestino_encontrado() {
790        return destino_encontrado;
791    }
792
793    public void setArrostrar(boolean arrastrar) {
794        this.arrastrar = arrastrar;
795    }
796
797    public boolean getArrostrar() {
798        return arrastrar;
799    }
800
801    public void setNodA(Dispositivo nodA) {
802        this.nodA = nodA;
803    }
804
805    public Dispositivo getNodA() {
806        return nodA;
807    }
808
809    public void setNodP(Dispositivo nodP) {
810        this.nodP = nodP;
811    }
812
813    public Dispositivo getNodP() {
814        return nodP;
815    }
816
817    public void setDespX(int despX) {
818        this.despX = despX;
819    }
820
821    public void setDespY(int despY) {
822        this.despY = despY;
823    }
824
825    public void setClick(Point click) {
826        this.click = click;
827    }
828
829    public Point getClick() {
830        return click;
831    }
832
833    public void setTopologia(Topologia topologia) {
834        this.topologia = topologia;
835    }
836
837    public Topologia getTopologia() {
838        return topologia;
839    }
840
841    public void setTam(int tam) {
842        this.tam = tam;
843    }
844
845    public int getTam() {
846        return tam;
847    }
848
849    public void setDestino_encontrado(boolean destino_encontrado) {
850        this.destino_encontrado = destino_encontrado;
851    }
852
853    public boolean getDestino_encontrado() {
854        return destino_encontrado;
855    }
856
857    public void setArrostrar(boolean arrastrar) {
858        this.arrastrar = arrastrar;
859    }
860
861    public boolean getArrostrar() {
862        return arrastrar;
863    }
864
865    public void setNod1(Dispositivo nod1) {
866        this.nod1 = nod1;
867    }
868
869    public Dispositivo getNod1() {
870        return nod1;
871    }
872
873    public void setNod2(Dispositivo nod2) {
874        this.nod2 = nod2;
875    }
876
877    public Dispositivo getNod2() {
878        return nod2;
879    }
880
881    public void setDestino_encontrado(boolean destino_encontrado) {
882        this.destino_encontrado = destino_encontrado;
883    }
884
885    public boolean getDestino_encontrado() {
886        return destino_encontrado;
887    }
888
889    public void setArrostrar(boolean arrastrar) {
890        this.arrastrar = arrastrar;
891    }
892
893    public boolean getArrostrar() {
894        return arrastrar;
895    }
896
897    public void setNodA(Dispositivo nodA) {
898        this.nodA = nodA;
899    }
900
901    public Dispositivo getNodA() {
902        return nodA;
903    }
904
905    public void setNodP(Dispositivo nodP) {
906        this.nodP = nodP;
907    }
908
909    public Dispositivo getNodP() {
910        return nodP;
911    }
912
913    public void setDespX(int despX) {
914        this.despX = despX;
915    }
916
917    public void setDespY(int despY) {
918        this.despY = despY;
919    }
920
921    public void setClick(Point click) {
922        this.click = click;
923    }
924
925    public Point getClick() {
926        return click;
927    }
928
929    public void setTopologia(Topologia topologia) {
930        this.topologia = topologia;
931    }
932
933    public Topologia getTopologia() {
934        return topologia;
935    }
936
937    public void setTam(int tam) {
938        this.tam = tam;
939    }
940
941    public int getTam() {
942        return tam;
943    }
944
945    public void setDestino_encontrado(boolean destino_encontrado) {
946        this.destino_encontrado = destino_encontrado;
947    }
948
949    public boolean getDestino_encontrado() {
950        return destino_encontrado;
951    }
952
953    public void setArrostrar(boolean arrastrar) {
954        this.arrastrar = arrastrar;
955    }
956
957    public boolean getArrostrar() {
958        return arrastrar;
959    }
960
961    public void setNod1(Dispositivo nod1) {
962        this.nod1 = nod1;
963    }
964
965    public Dispositivo getNod1() {
966        return nod1;
967    }
968
969    public void setNod2(Dispositivo nod2) {
970        this.nod2 = nod2;
971    }
972
973    public Dispositivo getNod2() {
974        return nod2;
975    }
976
977    public void setDestino_encontrado(boolean destino_encontrado) {
978        this.destino_encontrado = destino_encontrado;
979    }
980
981    public boolean getDestino_encontrado() {
982        return destino_encontrado;
983    }
984
985    public void setArrostrar(boolean arrastrar) {
986        this.arrastrar = arrastrar;
987    }
988
989    public boolean getArrostrar() {
990        return arrastrar;
991    }
992
993    public void setNodA(Dispositivo nodA) {
994        this.nodA = nodA;
995    }
996
997    public Dispositivo getNodA() {
998        return nodA;
999    }
1000
1001    public void setNodP(Dispositivo nodP) {
1002        this.nodP = nodP;
1003    }
1004
1005    public Dispositivo getNodP() {
1006        return nodP;
1007    }
1008
1009    public void setDespX(int despX) {
1010        this.despX = despX;
1011    }
1012
1013    public void setDespY(int despY) {
1014        this.despY = despY;
1015    }
1016
1017    public void setClick(Point click) {
1018        this.click = click;
1019    }
1020
1021    public Point getClick() {
1022        return click;
1023    }
1024
1025    public void setTopologia(Topologia topologia) {
1026        this.topologia = topologia;
1027    }
1028
1029    public Topologia getTopologia() {
1030        return topologia;
1031    }
1032
1033    public void setTam(int tam) {
1034        this.tam = tam;
1035    }
1036
1037    public int getTam() {
1038        return tam;
1039    }
1040
1041    public void setDestino_encontrado(boolean destino_encontrado) {
1042        this.destino_encontrado = destino_encontrado;
1043    }
1044
1045    public boolean getDestino_encontrado() {
1046        return destino_encontrado;
1047    }
1048
1049    public void setArrostrar(boolean arrastrar) {
1050        this.arrastrar = arrastrar;
1051    }
1052
1053    public boolean getArrostrar() {
1054        return arrastrar;
1055    }
1056
1057    public void setNod1(Dispositivo nod1) {
1058        this.nod1 = nod1;
1059    }
1060
1061    public Dispositivo getNod1() {
1062        return nod1;
1063    }
1064
1065    public void setNod2(Dispositivo nod2) {
1066        this.nod2 = nod2;
1067    }
1068
1069    public Dispositivo getNod2() {
1070        return nod2;
1071    }
1072
1073    public void setDestino_encontrado(boolean destino_encontrado) {
1074        this.destino_encontrado = destino_encontrado;
1075    }
1076
1077    public boolean getDestino_encontrado() {
1078        return destino_encontrado;
1079    }
1080
1081    public void setArrostrar(boolean arrastrar) {
1082        this.arrastrar = arrastrar;
1083    }
1084
1085    public boolean getArrostrar() {
1086        return arrastrar;
1087    }
1088
1089    public void setNodA(Dispositivo nodA) {
1090        this.nodA = nodA;
1091    }
1092
1093    public Dispositivo getNodA() {
1094        return nodA;
1095    }
1096
1097    public void setNodP(Dispositivo nodP) {
1098        this.nodP = nodP;
1099    }
1100
1101    public Dispositivo getNodP() {
1102        return nodP;
1103    }
1104
1105    public void setDespX(int despX) {
1106        this.despX = despX;
1107    }
1108
1109    public void setDespY(int despY) {
1110        this.despY = despY;
1111    }
1112
1113    public void setClick(Point click) {
1114        this.click = click;
1115    }
1116
1117    public Point getClick() {
1118        return click;
1119    }
1120
1121    public void setTopologia(Topologia topologia) {
1122        this.topologia = topologia;
1123    }
1124
1125    public Topologia getTopologia() {
1126        return topologia;
1127    }
1128
1129    public void setTam(int tam) {
1130        this.tam = tam;
1131    }
1132
1133    public int getTam() {
1134        return tam;
1135    }
1136
1137    public void setDestino_encontrado(boolean destino_encontrado) {
1138        this.destino_encontrado = destino_encontrado;
1139    }
1140
1141    public boolean getDestino_encontrado() {
1142        return destino_encontrado;
1143    }
1144
1145    public void setArrostrar(boolean arrastrar) {
1146        this.arrastrar = arrastrar;
1147    }
1148
1149    public boolean getArrostrar() {
1150        return arrastrar;
1151    }
1152
1153    public void setNod1(Dispositivo nod1) {
1154        this.nod1 = nod1;
1155    }
1156
1157    public Dispositivo getNod1() {
1158        return nod1;
1159    }
1160
1161    public void setNod2(Dispositivo nod2) {
1162        this.nod2 = nod2;
1163    }
1164
1165    public Dispositivo getNod2() {
1166        return nod2;
1167    }
1168
1169    public void setDestino_encontrado(boolean destino_encontrado) {
1170        this.destino_encontrado = destino_encontrado;
1171    }
1172
1173    public boolean getDestino_encontrado() {
1174        return destino_encontrado;
1175    }
1176
1177    public void setArrostrar(boolean arrastrar) {
1178        this.arrastrar = arrastrar;
1179    }
1180
1181    public boolean getArrostrar() {
1182        return arrastrar;
1183    }
1184
1185    public void setNodA(Dispositivo nodA) {
1186        this.nodA = nodA;
1187    }
1188
1189    public Dispositivo getNodA() {
1190        return nodA;
1191    }
1192
1193    public void setNodP(Dispositivo nodP) {
1194        this.nodP = nodP;
1195    }
1196
1197    public Dispositivo getNodP() {
1198        return nodP;
1199    }
1200
1201    public void setDespX(int despX) {
1202        this.despX = despX;
1203    }
1204
1205    public void setDespY(int despY) {
1206        this.despY = despY;
1207    }
1208
1209    public void setClick(Point click) {
1210        this.click = click;
1211    }
1212
1213    public Point getClick() {
1214        return click;
1215    }
1216
1217    public void setTopologia(Topologia topologia) {
1218        this.topologia = topologia;
1219    }
1220
1221    public Topologia getTopologia() {
1222        return topologia;
1223    }
1224
1225    public void setTam(int tam) {
1226        this.tam = tam;
1227    }
1228
1229    public int getTam() {
1230        return tam;
1231    }
1232
1233    public void setDestino_encontrado(boolean destino_encontrado) {
1234        this.destino_encontrado = destino_encontrado;
1235    }
1236
1237    public boolean getDestino_encontrado() {
1238        return destino_encontrado;
1239    }
1240
1241    public void setArrostrar(boolean arrastrar) {
1242        this.arrastrar = arrastrar;
1243    }
1244
1245    public boolean getArrostrar() {
1246        return arrastrar;
1247    }
1248
1249    public void setNod1(Dispositivo nod1) {
1250        this.nod1 = nod1;
1251    }
1252
1253    public Dispositivo getNod1() {
1254        return nod1;
1255    }
1256
1257    public void setNod2(Dispositivo nod2) {
1258        this.nod2 = nod2;
1259    }
1260
1261    public Dispositivo getNod2() {
1262        return nod2;
1263    }
1264
1265    public void setDestino_encontrado(boolean destino_encontrado) {
1266        this.destino_encontrado = destino_encontrado;
1267    }
1268
1269    public boolean getDestino_encontrado() {
1270        return destino_encontrado;
1271    }
1272
1273    public void setArrostrar(boolean arrastrar) {
1274        this.arrastrar = arrastrar;
1275    }
1276
1277    public boolean getArrostrar() {
1278        return arrastrar;
1279    }
1280
1281    public void setNodA(Dispositivo nodA) {
1282        this.nodA = nodA;
1283    }
1284
1285    public Dispositivo getNodA() {
1286        return nodA;
1287    }
1288
1289    public void setNodP(Dispositivo nodP) {
1290        this.nodP = nodP;
1291    }
1292
1293    public Dispositivo getNodP() {
1294        return nodP;
1295    }
1296
1297    public void setDespX(int despX) {
1298        this.despX = despX;
1299    }
1300
1301    public void setDespY(int despY) {
1302        this.despY = despY;
1303    }
1304
1305    public void setClick(Point click) {
1306        this.click = click;
1307    }
1308
1309    public Point getClick() {
1310        return click;
1311    }
1312
1313    public void setTopologia(Topologia topologia) {
1314        this.topologia = topologia;
1315    }
1316
1317    public Topologia getTopologia() {
1318        return topologia;
1319    }
1320
1321    public void setTam(int tam) {
1322        this.tam = tam;
1323    }
1324
1325    public int getTam() {
1326        return tam;
1327    }
1328
1329    public void setDestino_encontrado(boolean destino_encontrado) {
1330        this.destino_encontrado = destino_encontrado;
1331    }
1332
1333    public boolean getDestino_encontrado() {
1334        return destino_encontrado;
1335    }
1336
1337    public void setArrostrar(boolean arrastrar) {
1338        this.arrastrar = arrastrar;
1339    }
1340
1341    public boolean getArrostrar() {
1342        return arrastrar;
1343    }
1344
1345    public void setNod1(Dispositivo nod1) {
1346        this.nod1 = nod1;
1347    }
1348
1349    public Dispositivo getNod1() {
1350        return nod1;
1351    }
1352
1353    public void setNod2(Dispositivo nod2) {
1354        this.nod2 = nod2;
1355    }
1356
1357    public Dispositivo getNod2() {
1358        return nod2;
1359    }
1360
1361    public void setDestino_encontrado(boolean destino_encontrado) {
1362        this.destino_encontrado = destino_encontrado;
1363    }
1364
1365    public boolean getDestino_encontrado() {
1366        return destino_encontrado;
1367    }
1368
1369    public void setArrostrar(boolean arrastrar) {
1370        this.arrastrar = arrastrar;
1371    }
1372
1373    public boolean getArrostrar() {
1374        return arrastrar;
1375    }
1376
1377    public void setNodA(Dispositivo nodA) {
1378        this.nodA = nodA;
1379    }
1380
1381    public Dispositivo getNodA() {
1382        return nodA;
1383    }
1384
1385    public void setNodP(Dispositivo nodP) {
1386        this.nodP = nodP;
1387    }
1388
1389    public Dispositivo getNodP() {
1390        return nodP;
1391    }
1392
1393    public void setDespX(int despX) {
1394        this.despX = despX;
1395    }
13
```

```

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            () );
68        while( it.hasPrevious() && encontrado==false ){
69            nodA = (Dispositivo) it.previous();
70            /*Si hay un clic dentro de un dispositivo*/
71            //System.out.println(nodA);
72            if( x>=nodA.getPuntoInicial().x && x<nodA.getPuntoInicial().x+nodA.
73                getTam())
74                && y>=nodA.getPuntoInicial().y && y<nodA.getPuntoInicial().y+nodA.
75                getTam() ){
76                encontrado = true;
77
78                arrastrar = true;
79                despX = x - nodA.getPuntoInicial().x;
80                despY = y - nodA.getPuntoInicial().y;
81            }
82        }
83        repintarLienzo();
84
85    @Override
86    public void mouseDragged( MouseEvent evt ){
87        if(arrastrar ){
88            topologia.setModificado(true);
89            int x = evt.getX()-despX;
90            int y = evt.getY()-despY;
91            if( x<0 ) {
92                x=0;
93            }
94            if( y<0 ) {
95                y=0;
96            }
97            nodA.setPuntoInicial( new Point(x,y) );
98            Propiedades.tfIP.setText(nodA.getIp());
99            Propiedades.tfGateway.setText(nodA.getGateway());
100           Propiedades.lListaInterfaces.setText("Interfaz\tt\tEstado\tMac\tt\tPauqtes
101             env.\tErros\tPaquetes rec.\tErros\tVlan\tNombre\n");
102           for(int i = 0; i<nodA.getInterfaces().size();i++){
103               if(nodA.getVlan().isEmpty()){
104                   Propiedades.lListaInterfaces.append(nodA.getInterfaces().get(i) +
105                     "\t"+nodA.getEstIf().get(i)+"\t"+nodA.getDirMac().get(i)+"\t"
106                     +nodA.getPaqEnv().get(i)+"\t"+nodA.getErrPE().get(i)+"\t"+
107                     nodA.getPaqRec().get(i)+"\t"+nodA.getErrPR().get(i)+"\t"+"\n"
108                     );
109               }
110               else{
111                   Propiedades.lListaInterfaces.append(nodA.getInterfaces().get(i) +
112                     "\t"+nodA.getEstIf().get(i)+"\t"+nodA.getDirMac().get(i)+"\t"
113                     +nodA.getPaqEnv().get(i)+"\t"+nodA.getErrPE().get(i)+"\t"+
114                     nodA.getPaqRec().get(i)+"\t"+nodA.getErrPR().get(i)+"\t"+nodA
115                     .getVlan().get(i)+"\t"+nodA.getNvlan().get(i)+"\n");
116
117               }
118           if( nodA.getPuntoInicial().x+nodA.getTam() > getWidth() ){/*
119             Redimensionar el lienzo*/
120             setPreferredSize( new Dimension(x+nodA.getTam(),getHeight() ) );
121             revalidate();
122         }

```

```

113         if( nodA.getPuntoInicial().y+nodA.getTam() > getHeight() ){
114             setPreferredSize( new Dimension( getWidth() ,y+nodA.getTam() ) );
115             revalidate();
116         }
117         repaintLienzo();
118     }
119 }
120
121 @Override
122 public void mouseReleased( MouseEvent evt ){
123     if( destino_encontrado & nod1 != nod2 ){/**Aregar 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     repaintLienzo();
139 }
140
141 public void crear(Topologia g){
142     setGrafo(g);
143     listaDispositivos = topologia.getListaDispositivos();
144 }
145
146 private void repaintLienzo(){
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<
16        String> interfaz,java.util.List<String> mac,java.util.List<String> estado,
17        java.util.List<String> PaqEnv,java.util.List<String> ErrPE, java.util.List<
18        String> PaqRec,java.util.List<String> ErrPR, java.util.List<String> Vlan,
19        java.util.List<String> NVlan,Point puntoInicial,int tam){
20        Dispositivo disp = new Dispositivo( Tipo, etiqueta, interfaz,mac,estado,
21            PaqEnv,ErrPE, PaqRec, ErrPR, Vlan, NVlan,puntoInicial,tam);
22        listaDispositivos.add( disp );
23        setModificado(true);
24        return disp;
25    }
26
27
28    public Dispositivo agregarDispositivo(int Tipo, String etiqueta,String gateway,
29        java.util.List<String> interfaz,java.util.List<String> mac,java.util.List<String>
30        estado,java.util.List<String>PaqEnv , java.util.List<String> ErrPE,
31        java.util.List<String> PaqRec, java.util.List<String> ErrPR,java.util.List<
32        String> Vlan, java.util.List<String> NVlan, Point puntoInicial,int tam){
33        Dispositivo disp = new Dispositivo( Tipo, etiqueta, gateway,interfaz,mac,
34            estado,PaqEnv,ErrPE, PaqRec, ErrPR, Vlan, NVlan,puntoInicial,tam);
35        listaDispositivos.add( disp );
36        setModificado(true);
37        return disp;
38    }
39
40    public Dispositivo agregarDispositivo(int Tipo, String etiqueta,String gateway,
41        java.util.List<String> interfaz,java.util.List<String> mac,java.util.List<String>
42        estado,java.util.List<String>PaqEnv , java.util.List<String> ErrPE,
43        java.util.List<String> PaqRec, java.util.List<String> ErrPR, Point
44        puntoInicial,int tam){
45        Dispositivo disp = new Dispositivo( Tipo, etiqueta, gateway,interfaz,mac,
46            estado,PaqEnv,ErrPE, PaqRec, ErrPR, puntoInicial,tam);
47        listaDispositivos.add( disp );
48        setModificado(true);
49        return disp;
50    }
51
52
53    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 dispositivos
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 dispositivo 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     {
8         String line = "";
9         ArrayList<String> resultado = new ArrayList();
10        for ( int i=0; i < instrucciones.size(); i++ ) {
11            String instrucion = (String)instrucciones.get(i);
12            Process p = Runtime.getRuntime().exec(instrucion);
13            BufferedReader res = new BufferedReader(new InputStreamReader(p.getInputStream
14                ()));
15            while ( (line=res.readLine()) != null ) {
16                resultado.add(line);
17            }
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>
27        DirMac, List<String>EstIf, List<String> PaqEnv, List<String> ErrPE, List<
28        String> PaqRec, List<String> ErrPR, List<String> Vlan, List<String> NVlan,
29        Point pi,int tam){
30        setTipo(Tipo);
31        setIp(Ip);
32        setInterfaces(Interfaces);
33        setDirMac(DirMac);
34        setEstIf(EstIf);
35        setPaqEnv(PaqEnv);
36        setErrPE(ErrPE);
37        setPaqRec(PaqRec);
38        setErrPR(ErrPR);
39        setVlan(Vlan);
40        setNVlan(NVlan);
41        setPuntoinicial(pi);
42        setTam(tam);
43        setRadio();
44        listaAd = Collections.synchronizedList(new LinkedList());
45    }
46
47    public Dispositivo(int Tipo, String Ip, String gateway, List<String> Interfaces,
48        List<String>DirMac, List<String>EstIf, List<String> PaqEnv, List<String> ErrPE,
49        List<String> PaqRec, List<String> ErrPR, List<String> Vlan, List<String>
50        NVlan,Point pi,int tam){
51        setTipo(Tipo);
52        setIp(Ip);
53        setInterfaces(Interfaces);
54        setDirMac(DirMac);
55        setEstIf(EstIf);
56        setGateway(gateway);
57        setPaqEnv(PaqEnv);
58        setErrPE(ErrPE);
59        setPaqRec(PaqRec);
60        setErrPR(ErrPR);
61        setVlan(Vlan);
62        setNVlan(NVlan);
63    }

```

```

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>
64 DirMac, List<String>EstIf, List<String> PaqEnv, List<String> ErrPE, List<
65 String> PaqRec, List<String> ErrPR, Point pi,int tam){
66     setTipo(Tipo);
67     setIp(Ip);
68     setInterfaces(Interfaces);
69     setDirMac(DirMac);
70     setEstIf(EstIf);
71     setPaqEnv(PaqEnv);
72     setErrPE(ErrPE);
73     setPaqRec(PaqRec);
74     setErrPR(ErrPR);
75     setVlan(new LinkedList<String>());
76     setNVlan(new LinkedList<String>());
77         setPuntoInicial(pi);
78         setTam(tam);
79     setRadio();
80     listaAd = Collections.synchronizedList(new LinkedList());
81 }
82
83 public Dispositivo(int Tipo, String Ip, String gateway,List<String> Interfaces,
84 List<String>DirMac, List<String>EstIf, List<String> PaqEnv, List<String> ErrPE,
85 List<String> PaqRec, List<String> ErrPR, Point pi,int tam){
86     setTipo(Tipo);
87     setIp(Ip);
88     setInterfaces(Interfaces);
89     setDirMac(DirMac);
90     setEstIf(EstIf);
91     setGateway(gateway);
92     setPaqEnv(PaqEnv);
93     setErrPE(ErrPE);
94     setPaqRec(PaqRec);
95     setErrPR(ErrPR);
96     setVlan(new LinkedList<String>());
97     setNVlan(new LinkedList<String>());
98         setPuntoInicial(pi);
99         setTam(tam);
100    setRadio();
101    listaAd = Collections.synchronizedList(new LinkedList());
102 }
103
104 public int getTipo() {
105     return 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     public void setGateway(String gateway) {
120         this.gateway = gateway;
121     }
122
123     public List<String> getInterfaces() {
124         return Interfaces;
125     }
126
127     public void setInterfaces(List<String> Interfaces) {
128         this.Interfaces = Interfaces;
129     }
130
131     public List<String> getDirMac() {
132         return DirMac;
133     }
134
135     public void setDirMac(List<String> DirMac) {
136         this.DirMac = DirMac;
137     }
138
139     public List<String> getEstIf() {
140         return EstIf;
141     }
142
143     public void setEstIf(List<String> EstIf) {
144         this.EstIf = EstIf;
145     }
146     public Point getPuntoInicial() {
147         return puntoInicial;
148     }
149
150     public void setPuntoInicial(Point puntoInicial) {
151         this.puntoInicial = puntoInicial;
152     }
153
154     public int getTam() {
155         return tam;
156     }
157
158     public void setTam(int tam) {
159         this.tam = tam;
160     }
161
162     public double getRadio() {
163         return radio;
164     }
165
166     public void setRadio(double radio) {
167         this.radio = radio;
168     }
169
170     public java.util.List getListaAd() {
171         return listaAd;
172     }
173
174     public void setListaAd(java.util.List listaAd) {
175         this.listaAd = listaAd;
176     }
177
178
179     public void setRadio(){
180         this.radio = getTam() / 2;
181     }
182
183     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             ();
246             g.drawString( getIp() ,pix ,(piy+55)+7 );
247         }
248         else if (tipo==2){
249             image = new ImageIcon(getClass().getResource("switch.jpg")).getImage();
```

```
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

```

1 import java.awt.*;
2 import java.io.*;
3
4 public final class Conexion implements Serializable {
5
6     private Dispositivo dispositivoOrigen;
7     private Dispositivo dispositivoDestino;
8     private Point puntoIn;
9     private Point puntoFi;
10
11    public Conexion( Dispositivo o, Dispositivo d ){
12        setDispositivoOrigen(o);
13        setDispositivoDestino(d);
14        setPuntoIn( null );
15        setPuntoFi( null );
16    }
17
18    public Dispositivo getDispositivoOrigen(){
19        return dispositivoOrigen;
20    }
21    public Dispositivo getDispositivoDestino(){
22        return dispositivoDestino;
23    }
24    public Point getPuntoIn(){
25        return puntoIn;
26    }
27    public Point getPuntoFi(){
28        return puntoFi;
29    }
30    public void setDispositivoOrigen( Dispositivo n ){
31        this.dispositivoOrigen = n;
32    }
33    public void setDispositivoDestino( Dispositivo n ){
34        this.dispositivoDestino = n;
35    }
36    public void setPuntoIn( Point p ){
37        this.puntoIn = p;
38    }
39    public void setPuntoFi( Point p ){
40        this.puntoFi = p;
41    }
42
43    public void pintar( Graphics g )      {
44        /**coordenadas iniciales*/
45        int xi = dispositivoOrigen.getPuntoInicial().x;
46        int yi = dispositivoOrigen.getPuntoInicial().y;
47        /**coordenadas finales*/
48        int xf = dispositivoDestino.getPuntoInicial().x;
49        int yf = dispositivoDestino.getPuntoInicial().y;
50        /**Puntos que representan el centro del icono*/
51        int xim = xi+(dispositivoOrigen.getTam()/2); /*centro del nodo origen*/
52        int yim = yi+(dispositivoOrigen.getTam()/2);
53        int xfm = xf+(dispositivoDestino.getTam()/2); /*centro del nodo destino*/
54        int yfm = yf+(dispositivoDestino.getTam()/2);
55
56        int pfx = (int) (xfm ); /*punto final en x donde termina la flecha*/
57        int pfy = (int) (yfm ); /*punto final en y donde termina la flecha*/
58
59        if( dispositivoOrigen != dispositivoDestino ){
60            dibujarConexion( g, xim, yim, pfx, pfy );
61        }
62

```

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