



**Instituto Politécnico Nacional
Escuela Superior de Cómputo
Academia de Ingeniería de Software**



Práctica 1

Integrantes del equipo:

**Castro Flores Marcela
Sánchez Cruz Rosa María
Santiago Mancera Arturo Samuel**

M. en C. Tanibet Pérez de los Santos Mondragón

México, Ciudad de México a 5 de septiembre de 2018

Índice general

1. Introducción	6
2. Instalación y configuración de gestor y agentes	7
2.1. Observium	7
2.2. Configuración de agente en Linux	9
2.3. Configuración de agente en Windows	16
3. Cuestionario	20
3.1. Cuestionario	20
4. Análisis de tráfico	32
4.1. Análisis de tráfico con Wireshark	32
4.1.1. snmpget	32
4.1.2. snmpgetnext	34
4.1.3. snmpwalk	35
4.1.4. snmpset	36
5. Implementación de un modelo de administración de red de SNMP	39
5.1. Pantalla de inicio	39
5.2. Agregar agente	44
5.3. Eliminar agente	46
5.4. Estado del dispositivo	48
5.5. Gráficas de dispositivos	48
6. Conclusiones	53
6.0.1. Castro Flores Marcela	53
6.0.2. Sánchez Cruz Rosa María	53
6.0.3. Santiago Mancera Arturo Samuel	53

Índice de figuras

2.1.	Finalización de la instalación del S.O	7
2.2.	Asignación de dirección IP	8
2.3.	Pág. de Observium	8
2.4.	Conectividad	9
2.5.	Agentes que contiene Observium	9
2.6.	Configuración de snmp (1)	11
2.7.	Configuración de snmp (2)	12
2.8.	Archivo de configuración finalizado.	13
2.9.	Archivo de hosts Observium.	14
2.10.	Ping a Linux.	14
2.11.	Agente añadido.	15
2.12.	Información del agente.	15
2.13.	Características de Windows.	16
2.14.	Protocolo SNMP.	16
2.15.	Captura SNMP.	17
2.16.	Comunidad SNMP.	17
2.17.	Permisos de la comunidad.	18
2.18.	Servicio SNMP.	18
2.19.	Firewall de Windows.	19
3.1.	Último reinicio del agente en Linux con comando snmpget.	20
3.2.	Último reinicio del agente en Linux con comando snmpgetnext.	20
3.3.	Último reinicio del agente en Linux con comando snmpgetwalk.	21
3.4.	Último reinicio del agente en Windows.	21
3.5.	Número de interfaces Ethernet en Linux con comando snmpget.	21
3.6.	Número de interfaces Ethernet en Linux con comando snmpgetnext.	21
3.7.	Número de interfaces Ethernet en Linux con comando snmpwalk.	21
3.8.	Número de interfaces Ethernet en Windows.	22
3.9.	Velocidad de las interfaces con comando snmpget.	22
3.10.	Velocidad de las interfaces con comando snmpgetnext.	22
3.11.	Velocidad de las interfaces en Linux con comando snmpwalk.	22
3.12.	Velocidad de las interfaces en Windows.	23
3.13.	El OID para ver los Octetos es 1.3.6.1.2.1.2.2.1.10	23
3.14.	tabla de las interfaces en Windows con sus Octetos	24

3.15. Interfaces	24
3.16. Octetos en enp0s3	25
3.17. Octetos en Windows	25
3.18. Dirección Mac de Enp0s3	26
3.19. Tabla de direcciones Mac en Windows	26
3.20. Octetos por interfaz.	27
3.21. Direcciones IP de cada interfaz.	27
3.22. Octetos por interfaz.	27
3.23. Direcciones IP de cada interfaz.	27
3.24. Mensajes ICMP recibidos en Linux.	27
3.25. Mensajes ICMP recibidos en Windows.	27
3.26. Entradas en ipRouteTable en Linux.	28
3.27. Entradas en ipRouteTable en Windows.	28
3.28. Número de mensajes TCP recibidos en Linux con comando snmpget.	28
3.29. Número de mensajes TCP recibidos en Linux con comando snmpgetnext.	28
3.30. Número de mensajes TCP recibidos en Linux con comando snmpwalk.	28
3.31. S.O Windows no tiene mensajes EGP	29
3.32. S.O Centos no cuenta con mensajes EGP	29
3.33. Información del Sistema Operativo Windows	29
3.34. Información del Sistema Operativo Centos	29
3.35. valores de sysContact y sysUbcation, se cambio la ubicación del agente a "MyHouse"	30
3.36. Se cambio el valor de sysContact "WindowsAgentz se muestran los cambios hechos en sys- Contact y sysUbcation	30
3.37. Se cambio el valor de sysUbcation(6) "LaptopAcerz se muestran los cambios hechos del valor inicial al final	30
3.38. Muestra el valor de sysContact(4) y se modifica por CentOS"finalmente muestra el cambio.	30
3.39. Dibujo de la MiB.	31
4.1. Comando snmpget.	32
4.2. Captura snmpget.	33
4.3. Solicitud snmpget.	33
4.4. Respuesta snmpget.	34
4.5. Comando snmpgetnext.	34
4.6. Captura snmpgetnext.	34
4.7. Solicitud snmpgetnext.	35
4.8. Respuesta snmpgetnext.	35
4.9. Comando snmpwalk.	36
4.10. Capturas snmpwalk.	36
4.11. Comando snmpset.	37
4.12. Captura snmpset.	37
4.13. Solicitud snmpset.	37
4.14. Respuesta snmpset.	38
5.1. Pantalla principal.	40
5.2. Método getHostInfo.	40
5.3. Método getHostInfo, líneas importantes.	41
5.4. Método getAgentInfo.	41
5.5. Método getAgentInfo, líneas importantes.	42
5.6. Método ping.	42
5.7. Método consultaSNMP.	43
5.8. Botón agregar agente.	44
5.9. Método addClient.	44
5.10. Pantalla desplegada para añadir un cliente nuevo.	44

5.11. Método makeform.	44
5.12. Pantalla desplegada para añadir un cliente nuevo.	45
5.13. Alerta desplegada al crearse un nuevo agente.	45
5.14. Pantalla de inicio que muestra nuevo agente agregado.	45
5.15. Agente agregado en archivo hosts.	45
5.16. Pantalla de inicio que muestra botón para eliminar un agente.	46
5.17. Botones de pantalla de principal.	46
5.18. Método eliminarAgente.	47
5.19. Alerta de agente eliminado.	47
5.20. Método main.	47
5.21. Pantalla principal sin el agente eliminado.	47
5.22. Archivo hosts sin el agente eliminado.	48
5.23. Inicio de captura de datos.	48
5.24. Actualización de la base de datos RRD del host.	49
5.25. Actualización de la base de datos RRD del host.	49
5.26. Consulta SNMP.	49
5.27. Llamada a la pantalla de selección de gráficas.	50
5.28. Clase para la selección de gráficas.	50
5.29. Clase para la graficación.	51
5.30. Graficación con rrdtool.	52

CAPÍTULO 1

Introducción

Para la realización de esta práctica se utilizó el protocolo SNMP(Simple Network Management Protocol) el cuál permite a los administradores de red administrar dispositivos y diagnosticar sus problemas [1].

El protocolo SNMP está compuesto por dos elementos: el agente, y el gestor. Es una arquitectura cliente-servidor, en la cual el agente desempeña el papel de servidor y el gestor hace el de cliente.

El agente es un programa que ha de ejecutarse en cada nodo de red que se desea gestionar o monitorizar. Ofrece un interfaz de todos los elementos que se pueden configurar. Estos elementos se almacenan en unas estructuras de datos llamadas "Management Information Base"(MIB), se explicarán más adelante. Representa la parte del servidor, en la medida que tiene la información que se desea gestionar y espera comandos por parte del cliente. El gestor es el software que se ejecuta en la estación encargada de monitorizar la red, y su tarea consiste en consultar los diferentes agentes que se encuentran en los nodos de la red los datos que estos han ido obteniendo[2].

Esta práctica se dividió en las tres partes siguientes:

1. La primera parte se enfocó a la instalación de dos sistemas operativos que pudo ser Linux o Windows en caso de no tener alguno de los dos de forma nativa y una máquina virtual para la instalación de Observium mediante los cuáles se manejaron dos agentes y un gestor.
2. La segunda parte se enfocó a la utilización de ambos agentes y el gestor quien por medio del protocolo SNMP obtenían mediante el comando snmpget la diferente información de cada agente como por ejemplo el número de interfaces o la IP que corresponde a cierta interfaz en específico.
3. Por último, la tercera parte fue enfocada a la persistencia de la información por medio del uso de la herramienta rrdtool con la cuál se generan gráficas y se almacena la información de cada punto medido en un cierto lapso de tiempo.

En los capítulos mostrados a continuación se observa el desarrollo de la práctica desde la instalación del dichos S.O. hasta la implementación del código desarrollado.

CAPÍTULO 2

Instalación y configuración de gestor y agentes

2.1. Observium

Para llevar a cabo la comunicación de un Gestor-Agente, se necesitó se la configuración de 3 distintos Sistemas Operativos. El Sistema Operativo que toma el rol de Gestor es Observium, el cual estará encargado de monitorear la comunicación entre los mismo Agentes y Gestor. El primer paso es crear una máquina virtual con dicho sistema operativo, es esencial que se le agregan los requerimientos necesarios como es la cantidad de memoria, tipo de archivo del disco duro, el tamaño en bits etc. Pero sobre todo una vez creada la máquina virtual fue necesario entrar en su **configuración** y cambiar el adaptador de red para que se pudiera conectar en **Adaptador Puente**.

Una vez que se configuró la máquina virtual se inicia para su instalación del sistema, que es básicamente aceptar el modo de instalación que se hace mediante un disco y finalmente el reinicio del sistema operativo como se muestra en la figura 2.1.

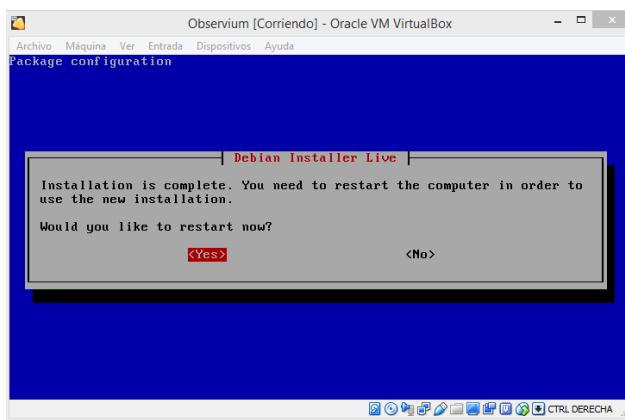


Figura 2.1: Finalización de la instalación del S.O .

Posteriormente se requiere ingresar un usuario con contraseña , después solo será necesario poner **Skip** a las acciones para que al final nos encontremos con la **dirección ip** que fue asignada. **Nota: Es importante saber que la dirección ip varía dependiendo de la red a la que se encuentre conectada la máquina virtual,**

esto se puede visualizar en la figura 2.2.

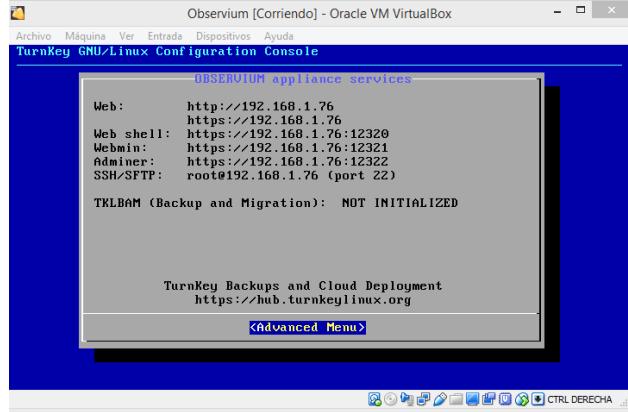


Figura 2.2: Asignación de dirección IP.

En este momento podemos decir que ya terminamos el punto de las configuraciones y tenemos a nuestro **Gestor**. Si se quiere comprobar la conectividad se abre el navegador de nuestro SO nativo y se ingresa la **dirección IP** que muestra Observium e ingresamos en modo **administrador** con la contraseña que se introdujo al inicio de la configuración (figura 2.3).

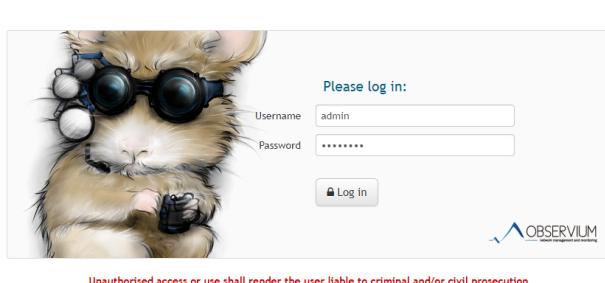


Figura 2.3: Pág. de Observium

Como se observa en la pagina, la figura 2.4 muestra que no se tiene ninguna conectividad de los agentes, esto debido a que no se han registrado en Observium.

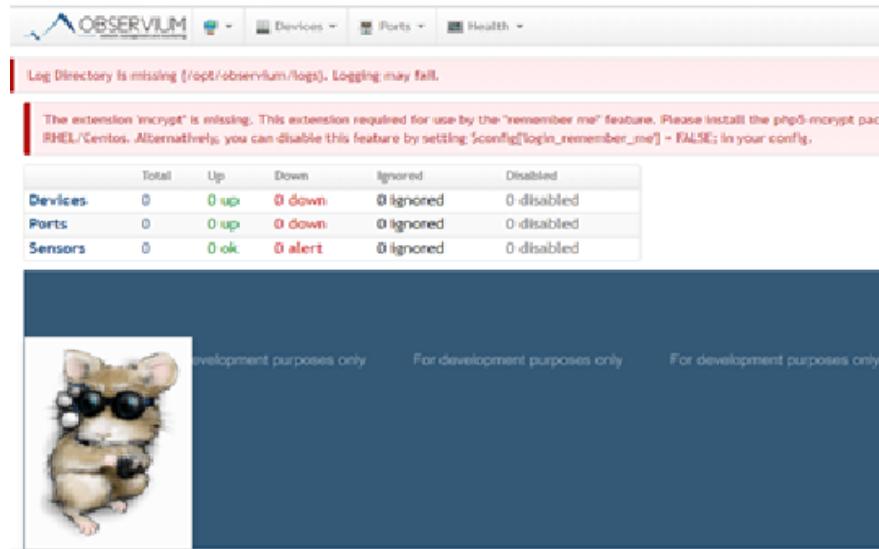


Figura 2.4: Conectividad

Para poder agregar un agente en Observium solo se necesita poner en la consola el comando **nano /etc/hosts** con esto ingresamos a nuestro editor de texto nano y solo ingresamos la dirección ip de nuestros sistemas operativos y el nombre de la comunidad que le asignamos a cada S.O. (figura 2.5).

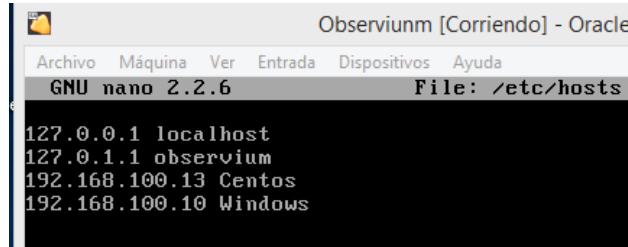


Figura 2.5: Agentes que contiene Observium

Una vez finalizado todo este procedimiento, tenemos listo el S.O. Observium como agente.

2.2. Configuración de agente en Linux

Una vez que se realizó la instalación de Observium, continuamos con la instalación de la máquina virtual en Linux, en este caso, se utilizó Linux de forma nativa por lo cual pasamos directamente a la instalación de los paquetes “SNMP” y “SNMPD” por medio de la instrucción en consola:

- **sudo apt-get install snmp snmpd**

Posteriormente, se realizó la configuración del protocolo SNMP por medio del comando:

- **snmpconf – r none – g basic – setup**

Se puede observar en la figura 2.6 el procedimiento que nos apareció al ejecutar el comando anterior. A continuación se enlistarán las opciones que fueron seleccionadas en el transcurso de dicha configuración:

- Configurar la información devuelta en el sistema del grupo de la MIB.
- Ingresamos un nombre para el almacenamiento del sistema.

- Agregamos un correo electrónico.
- Seleccionamos que no deseábamos configurar el valor de sysService.
- Sí configuramos el agente de control de acceso.
- No permitimos el acceso basado en usuario SNMPv3 de solo escritura.
- No permitimos el acceso basado en usuario SNMPv3 de solo lectura.
- Sí permitimos el acceso de la comunidad SNMPv1/v2c de lectura–escritura.
- Añadimos un nombre a la comunidad de acceso de lectura–escritura.
- Seleccionamos que no deseábamos agregar otra línea a rwcommunity.
- Por último, no permitimos que la comunidad SNMPv1/v2c tuviera acceso de solo lectura.

```
* ~ snmpconf -r none -g basic setup
*****
*** Beginning basic system information setup ***
*****
Do you want to configure the information returned in the system MIB group (conta
ct info, etc)? (default = y): y

Configuring: syslocation
Description:
The [typically physical] location of the system.
Note that setting this value here means that when trying to
perform an snmp SET operation to the sysLocation.0 variable will make
the agent return the "notWritable" error code. IE, including
this token in the snmpd.conf file will disable write access to
the variable.
arguments: location_string
Properties
The location of the system: Laboratorio progra 1
Tak
Finished Output: syslocation "Laboratorio progra 1"
-> Current State

Configuring: syscontact
Description:
The contact information for the administrator
Note that setting this value here means that when trying to
perform an snmp SET operation to the sysContact.0 variable will make
the agent return the "notWritable" error code. IE, including
this token in the snmpd.conf file will disable write access to
the variable.
arguments: contact_string

The contact information: march.castrof@gmail.com

Finished Output: syscontact march.castrof@gmail.com
Do you want to properly set the value of the sysServices.0 OID (if you don't kno
w, just say no)? (default = y): n
*****
*** BEGINNING ACCESS CONTROL SETUP ***
*****
Do you want to configure the agent's access control? (default = y): y
Do you want to allow SNMPv3 read-write user based access (default = y): n
Do you want to allow SNMPv3 read-only user based access (default = y): n
Do you want to allow SNMPv1/v2c read-write community access (default = y): y

Configuring: rwcommunity
Description:
a SNMPv1/SNMPv2c read-write access community name
arguments: community [default|hostname|network/bits] [oid]

Enter the community name to add read-write access for: comunidadMarcela
The hostname or network address to accept this community name from [RETURN for a
ll]:
The OID that this community should be restricted to [RETURN for no-restriction]:
```

```
Finished Output: rwcommunity comunidadMarcela
Do another rwcommunity line? (default = y): n
Do you want to allow SNMPv1/v2c read-only community access (default = y): n
*****
```

Figura 2.6: Configuración de snmp (1).

Una vez finalizada toda la configuración básica, continuamos con la siguiente parte de la configuración mostrada en la figura 2.7, en la cual se indicaron únicamente dos partes:

- No se configuró si el agente enviaría traps (trampas).
- No se configuró la habilidad al agente para monitorear el sistema.

Es importante recalcar que una vez finalizadas estas dos acciones, se muestra que el archivo nombrado como **snmpd.conf** fue creado pues fue el utilizado posteriormente.

```
*****  
*** Beginning trap destination setup ***  
*****  
Do you want to configure where and if the agent will send traps? (default = y):  
n  
*****  
*** Beginning monitoring setup ***  
*****  
Do you want to configure the agent's ability to monitor various aspects of your  
system? (default = y): n  
  
The following files were created:  
  
    snmpd.conf  
  
These files should be moved to /usr/share/snmp if you  
want them used by everyone on the system. In the future, if you add  
the -i option to the command line I'll copy them there automatically for you.  
  
Or, if you want them for your personal use only, copy them to  
/home/marce/.snmp . In the future, if you add the -p option to the  
command line I'll copy them there automatically for you.
```

Figura 2.7: Configuración de snmp (2).

Como se mencionó anteriormente, ya que se generó nuestro archivo de la configuración, se cambió el lugar de almacenamiento a la carpeta correcta por medio del comando:

- **sudo mv snmpd.conf /etc/snmp/snmpd.conf**

Y una vez que este fue almacenado debidamente, se reinició el servicio snmpd mediante la instrucción:

- **sudo service snmpd restart**

Y finalmente, por medio del comando:

- **nano /etc/snmp/snmpd.conf**

pudimos acceder al archivo mostrado en la figura 2.8 en el cual podemos observar todo lo que se fue configurando y el cual es de mucha utilidad en caso de que hayamos olvidado el nombre de nuestra comunidad por ejemplo.

```
GNU nano 2.5.3          File: /etc/snmp/snmpd.conf

#####
#      Arduino IDE
# snmpd.conf
#
# - created by the snmpconf configuration program
#



#####
# SECTION: Access Control Setup
#
# This section defines who is allowed to talk to your running
# snmp agent.

# rwcommunity: a SNMPv1/SNMPv2c read-write access community name
#   arguments: community [default|hostname|network/bits] [oid]

rwcommunity  comunidadMarcela


#####
# SECTION: System Information Setup
#
# This section defines some of the information reported in
# the "system" mib group in the mibII tree.

# syslocation: The [typically physical] location of the system.
#   Note that setting this value here means that when trying to
#   perform an snmp SET operation to the sysLocation.0 variable will make
#   the agent return the "notWritable" error code. IE, including
#   this token in the snmpd.conf file will disable write access to
#   the variable.
#   arguments: location_string

syslocation "Laboratorio progra 1"

# syscontact: The contact information for the administrator
#   Note that setting this value here means that when trying to
#   perform an snmp SET operation to the sysContact.0 variable will make
#   the agent return the "notWritable" error code. IE, including
#   this token in the snmpd.conf file will disable write access to
#   the variable.
#   arguments: contact_string

syscontact march.castrof@gmail.com
```

Figura 2.8: Archivo de configuración finalizado.

Después regresamos a nuestro gestor de Observium en el cual abrimos nuestro archivo de hosts haciendo uso de la instrucción:

- **nano /etc/hosts**

mismo que nos abrirá el archivo mostrado en la figura 2.9 en el cual agregamos la ip de nuestro sistema operativo Linux y un nombre identificador.



```
GNU nano 2.7.4                               File: /etc/hosts                         Modified

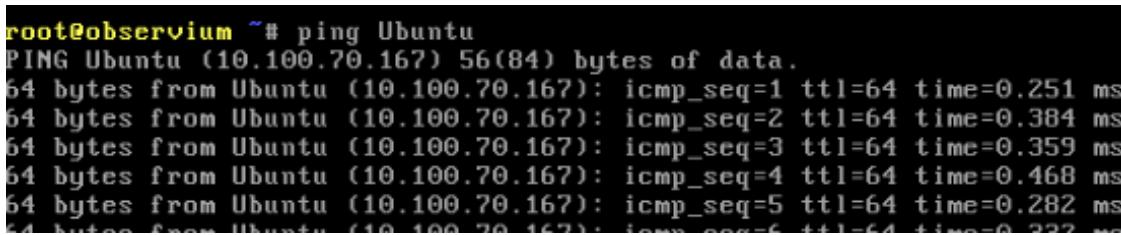
127.0.0.1 localhost
127.0.1.1 observium
10.100.70.167 Ubuntu
10.100.70.195 Windows

#Required for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text ^T To Spell ^_ Go To Line
```

Figura 2.9: Archivo de hosts Observium.

Guardamos y salimos para finalmente probar el funcionamiento de nuestra conexión mediante un ping más el nombre identificador escrito que en este caso fue Ubuntu para obtener una la respuesta mostrada en la figura 5.6



```
root@observium ~# ping Ubuntu
PING Ubuntu (10.100.70.167) 56(84) bytes of data.
64 bytes from Ubuntu (10.100.70.167): icmp_seq=1 ttl=64 time=0.251 ms
64 bytes from Ubuntu (10.100.70.167): icmp_seq=2 ttl=64 time=0.384 ms
64 bytes from Ubuntu (10.100.70.167): icmp_seq=3 ttl=64 time=0.359 ms
64 bytes from Ubuntu (10.100.70.167): icmp_seq=4 ttl=64 time=0.468 ms
64 bytes from Ubuntu (10.100.70.167): icmp_seq=5 ttl=64 time=0.282 ms
64 bytes from Ubuntu (10.100.70.167): icmp_seq=6 ttl=64 time=0.322 ms
```

Figura 2.10: Ping a Linux.

Por último, entramos a nuestra dirección de Observium en el navegador para añadir un dispositivo para monitorearlo como se observa en la figura 2.11, esto añadiendo un hostname que en este caso fue **Ubuntu** y una comunidad SNMP, misma que debe ser el nombre de la comunidad que elegimos poner en nuestro archivo de configuración que fue **comunidadMarcela**.

The screenshot shows the OBSERVIUM interface for adding a new device. On the left, under 'Basic Configuration', fields include Hostname (Ubuntu), Skip PING (unchecked), Protocol Version (v2c), Transport (UDP), Port (161), Timeout (1), Retries (5), and Ignore existing RRDs (checked). On the right, under 'Authentication Configuration', the SNMP Community is set to 'comunidadMarcela'. At the bottom center is a large 'Add device' button.

Figura 2.11: Agente añadido.

Posteriormente, volvimos a la pestaña de Devices, seleccionamos All devices y aquí encontramos nuestro agente de Ubuntu como vemos en la figura 2.12, mismo que al seleccionar nos muestra las diferentes gráficas e información de este.

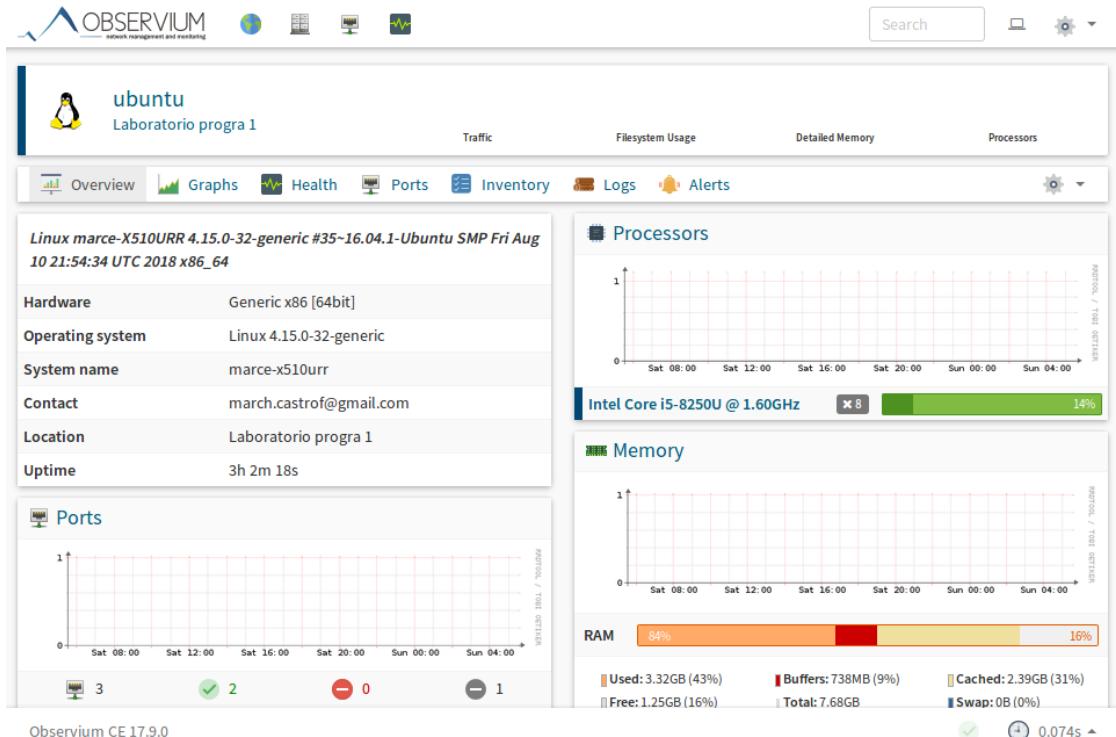


Figura 2.12: Información del agente.

2.3. Configuración de agente en Windows

Para la configuración de un agente en el sistema operativo Windows, se debe agregar una característica del sistema operativo. Esto con la finalidad de habilitar el servicio de "SNMP". Para habilitar la característica nos dirigimos al **Panel de control** de Windows y después a la sección de **Programas y características** como se muestra en la figura 2.13.

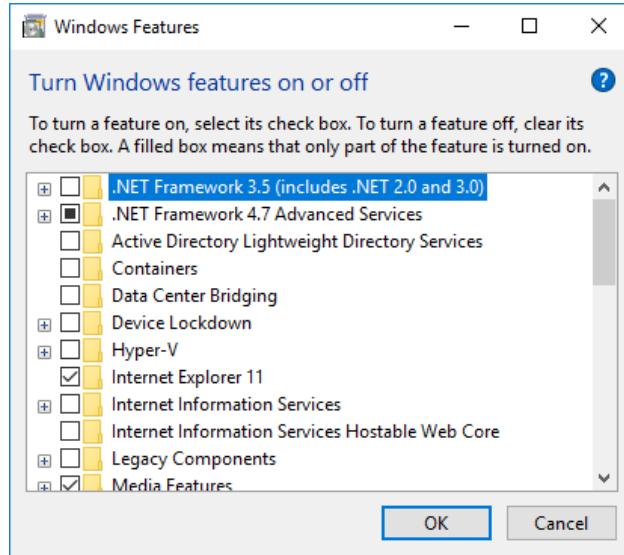


Figura 2.13: Características de Windows.

Una vez aquí debemos buscar el Protocolo Simple de Administración de Redes (SNMP) o *Simple Network Management Protocol (SNMP)* y activar su casilla correspondiente así como la de del nodo que se origina a partir de él tal y como se indica en la 2.14.

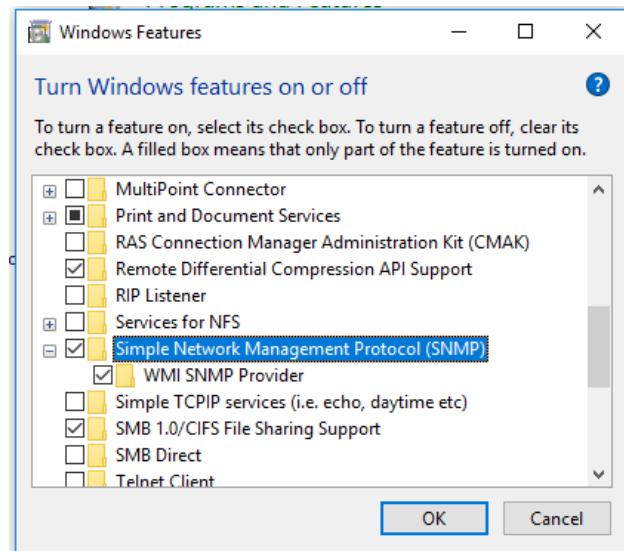


Figura 2.14: Protocolo SNMP.

El siguiente paso será iniciar el servicio de SNMP y de captura SNMP. Para ello entramos a los **Servicios**

de Windows y buscamos **Captura SNMP** o **SNMP Trap** como se indica en la figura 2.15. Hacemos clic derecho sobre él y lo iniciamos:

 SNMP Service	Enables Sim...	Running	Automatic	Loc
 SNMP Trap	Receives tra...	Running	Automatic	Loc
 Software Protection	Enables the ...		Automatic (D...)	Net

Figura 2.15: Captura SNMP.

Después, buscamos el servicio **SNMP** o **SNMP Service**, hacemos clic derecho sobre él y en la pestaña de **Capturas** o **Traps** ingresamos el nombre de la comunidad a la que pertenecerá el agente como se observa en la figura 2.16.

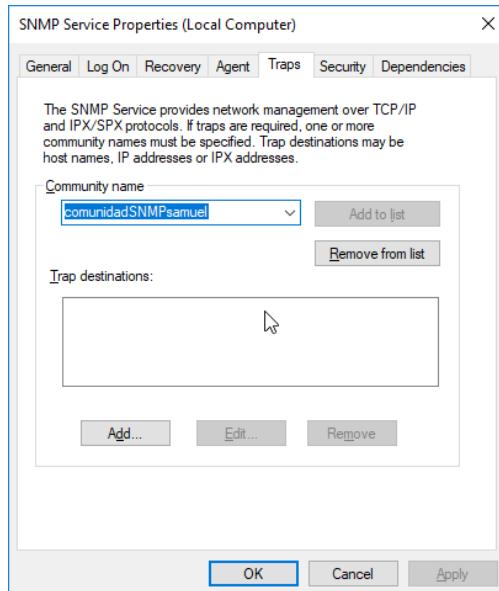


Figura 2.16: Comunidad SNMP.

Posteriormente, como observamos en la figura 2.17, debemos establecer los permisos que tendrá la comunidad anterior sobre el agente. Para ello, nos dirigimos a la pestaña de **Seguridad** o **Security**, hacemos clic en **Agregar** o **Add** y establecemos los permisos de **Lectura y Escritura** o **Read and Write**.

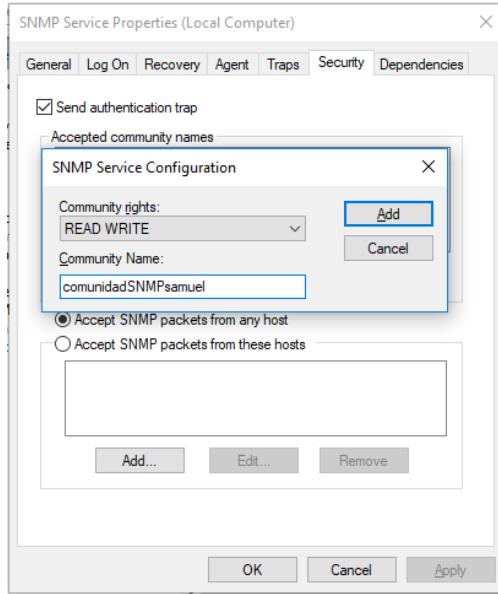


Figura 2.17: Permisos de la comunidad.

Finalmente, escribimos el nombre de la comunidad, tal y como se observa en la figura 2.18; y habilitamos la opción de **Aceptar paquetes de cualquier host**. Hacemos clic en **Aplicar**, **Aceptar** y reiniciamos el servicio de SNMP.

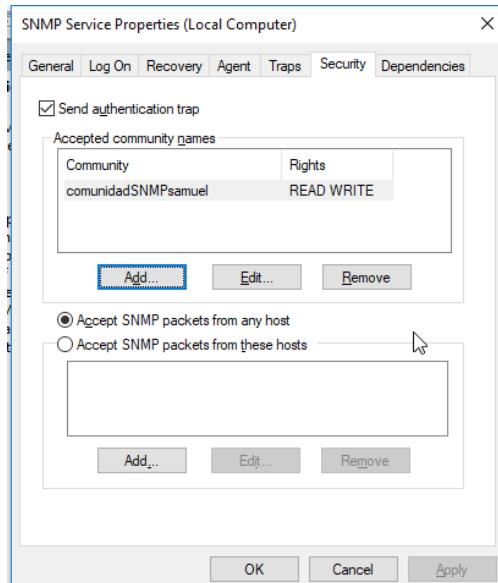


Figura 2.18: Servicio SNMP.

Como paso adicional, se deben agregar las reglas de firewall de Windows que permitan la transmisión y recepción de paquetes SNMP. Sin embargo, para este caso de prueba procederemos a desactivar completamente el firewall de Windows. En este caso, al ser una versión de Windows 10 nos dirigimos a **Windows Defender** y lo deshabilitamos:

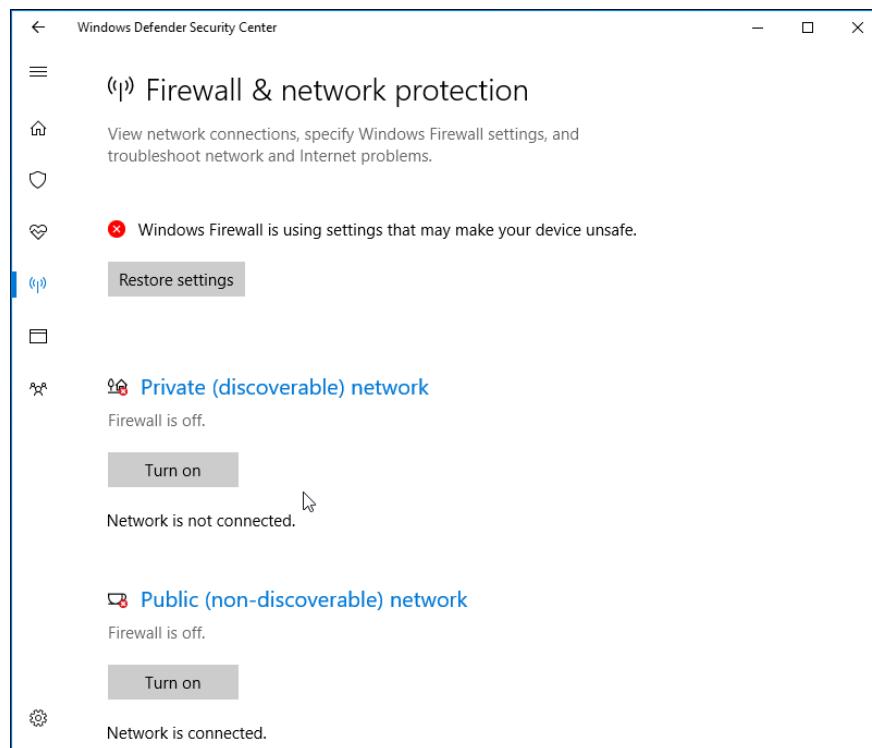


Figura 2.19: Firewall de Windows.

CAPÍTULO 3

Cuestionario

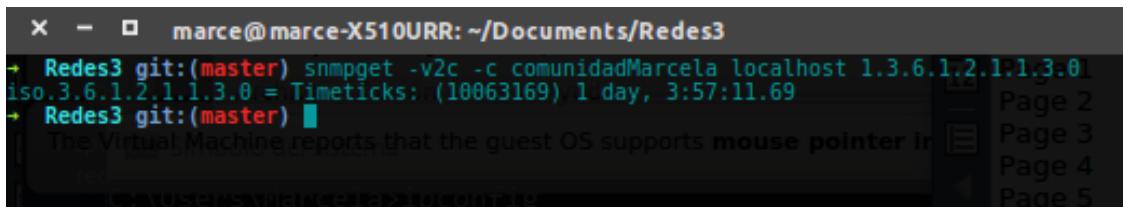
En este capítulo, se observan las diferentes pantallas que responden a las consultas realizadas a la MIB de Linux y de Windows y que de igual manera, muestra el segundo punto de la práctica que se refiere a la utilización del comando snmpget y algunos otros.

3.1. Cuestionario

1. ¿Cuándo fue el último reinicio (Dia, hora y minuto) de los agentes?

El resultado del último reinicio en Linux como se observa a continuación fue:

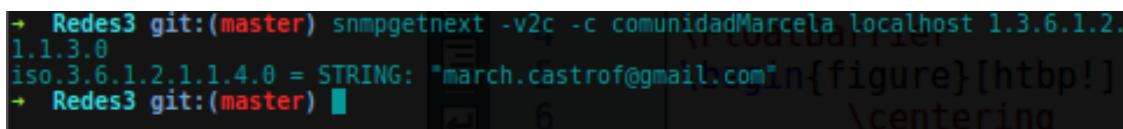
Comando: snmpget (figura 3.1)



```
X - marce@marce-X510URR: ~/Documents/Redes3
+ Redes3 git:(master) snmpget -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.1.3.01
iso.3.6.1.2.1.1.3.0 = Timeticks: 10063169, 1 day, 3:57:11.69
+ Redes3 git:(master) 
  The Virtual Machine reports that the guest OS supports mouse pointer in
C:\users\marcela>ipconfig
```

Figura 3.1: Último reinicio del agente en Linux con comando snmpget.

Comando: snmpgetnext (figura 3.2)



```
+ Redes3 git:(master) snmpgetnext -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.1.3.0
iso.3.6.1.2.1.1.4.0 = STRING: "marc.castrof@gmail.com\{figure}[htbp!]"
+ Redes3 git:(master) 
  6 \centering
```

Figura 3.2: Último reinicio del agente en Linux con comando snmpgetnext.

Comando: snmpwalk (figura 3.3)

```
+ Redes3 git:(master) ✘ snmpwalk -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.1.3.0 iso.3.6.1.2.1.1.3.0 = Timeticks: (8797875) 1 day, 0:26:18.75 Page 18
+ Redes3 git:(master) ✘ snmpwalk -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.1 iso.3.6.1.2.1.1.1.0 = STRING: "Linux marce-X510URR 4.15.0-32-generic #35~16.04.1-Ubuntu SMP Fri Aug 10 21:54:34 UTC 2018 x86_64" Page 19
+ Redes3 git:(master) ✘ snmpwalk -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.1.2.0 iso.3.6.1.2.1.1.2.0 = OID: iso.3.6.1.4.1.8072.3.2.10 Page 20
+ Redes3 git:(master) ✘ snmpwalk -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.1.3.0 iso.3.6.1.2.1.1.3.0 = Timeticks: (8798361) 1 day, 0:26:23.61 Page 21
+ Redes3 git:(master) ✘ snmpwalk -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.1.4.0 iso.3.6.1.2.1.1.4.0 = STRING: "march.castrof@gmail.com" Page 22
+ Redes3 git:(master) ✘ snmpwalk -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.1.5.0 iso.3.6.1.2.1.1.5.0 = STRING: "marce-X510URR" Page 23
+ Redes3 git:(master) ✘ snmpwalk -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.1.6.0 iso.3.6.1.2.1.1.6.0 = STRING: "\"Laboratorio programa 1\""
+ Redes3 git:(master) ✘ snmpwalk -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.1.8.0 iso.3.6.1.2.1.1.8.0 = Timeticks: (65) 0:00:00.65 Page 24
+ Redes3 git:(master) ✘ snmpwalk -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.1.9.0 iso.3.6.1.2.1.1.9.0 = Integer: 2 Page 25
```

Figura 3.3: Último reinicio del agente en Linux con comando `snmpgetwalk`.

Comando: snmpset Comando:snmptranslate

Por otro lado, el resultado del último reinicio en Windows como se observa en la figura 3.4 fue:

```
C:\Windows\system32>snmpget -v2c -c comunidad3 192.168.1.66 1.3.6.1.2.1.1.3.0  
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (27236446) 3 days, 3:39:24.46
```

Figura 3.4: Último reinicio del agente en Windows.

- ## 2. ¿Cuántas interfaces Ethernet tienen?

Se puede observar que resultado en Linux fue de una interfaz Ethernet.

Comando: snmpget (figura 3.5)

```
+ Redes3 git:(master) ✘ snmpget -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.2.2.1.2.1 iso.3.6.1.2.1.2.2.1.2.1 = STRING: "lo"
+ Redes3 git:(master) ✘ snmpget -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.2.2.1.2.2 iso.3.6.1.2.1.2.2.1.2.2 = STRING: "Intel Corporation Device 24fd" 3.14159265-2.6-1.40
```

Figura 3.5: Número de interfaces Ethernet en Linux con comando snmpget.

Comando: snmpgetnext (figura 3.6)

```
+ Redes3 git:(master) x snmpwalk -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.2.2.1.2 16  
iso.3.6.1.2.1.2.2.1.2.1 = STRING: "lo"  Live 2015/Debian) (preloaded format=pdflatex  
+ Redes3 git:(master) x snmpgetnext -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.2.2.1.2.  
1          entering extended mode  
iso.3.6.1.2.1.2.2.1.2.2 = STRING: "Intel Corporation Device 24fd"
```

Figura 3.6: Número de interfaces Ethernet en Linux con comando `snmpgetnext`.

Comando: snmpwalk (figura 3.7)

```
→ Redes3 git:(master) ✘ snmpwalk -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.2.2.1.2  
iso.3.6.1.2.1.2.2.1.2.1 = STRING: "lo"  
iso.3.6.1.2.1.2.2.1.2.2 = STRING: "Intel Corporation Device 24fd"  
iso.3.6.1.2.1.2.2.1.2.3 = STRING: "vboxnet0"  
→ Redes3 git:(master) ✘
```

Figura 3.7: Número de interfaces Ethernet en Linux con comando snmpwalk.

De igual manera, se puede observar en la figura 3.8 que el resultado en Windows fue de 4 interfaces Ethernet.

```
C:\Windows\system32>snmpwalk -v2c -c comunidad3 192.168.1.66 1.3.6.1.2.1.2.2.1.2
IF-MIB::ifDescr.1 = STRING: Software Loopback Interface 1
IF-MIB::ifDescr.2 = STRING: Microsoft 6to4 Adapter
IF-MIB::ifDescr.3 = STRING: Microsoft IP-HTTPS Platform Adapter
IF-MIB::ifDescr.4 = STRING: Microsoft Kernel Debug Network Adapter
IF-MIB::ifDescr.5 = STRING: Microsoft Teredo Tunnelling Adapter
IF-MIB::ifDescr.6 = STRING: Intel(R) PRO/1000 MT Desktop Adapter
IF-MIB::ifDescr.7 = STRING: Intel(R) PRO/1000 MT Desktop Adapter-WFP Native MAC Layer LightWeight Filter-0000
IF-MIB::ifDescr.8 = STRING: Intel(R) PRO/1000 MT Desktop Adapter-QoS Packet Scheduler-0000
IF-MIB::ifDescr.9 = STRING: Intel(R) PRO/1000 MT Desktop Adapter-WFP 802.3 MAC Layer LightWeight Filter-0000
```

Figura 3.8: Número de interfaces Ethernet en Windows.

3. ¿Cuál es la velocidad (en MBPS) de esas interfaces?

El resultado en Linux mostrado fue:

Comando: snmpget (figura 3.10)

```
+ Redes3 git:(master) x snmpget -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.2.2.1.5.1
iso.3.6.1.2.1.2.2.1.5.1 = Gauge32: 10000000
+ Redes3 git:(master) x snmpget -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.2.2.1.5.2
iso.3.6.1.2.1.2.2.1.5.2 = Gauge32: 0
```

Figura 3.9: Velocidad de las interfaces con comando snmpget.

Es importante recalcar que en este caso, aunque la interfaz Ethernet corresponde a la llamada “Intel Corporation Device 24fd”, su velocidad aparece ser de 0 mbps debido a que esta está obteniendo el ancho de banda vía wi-fi y no de forma alámbrica.

Comando: snmpgetnext (figura ??)

```
+ Redes3 git:(master) x snmpgetnext -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.2.2.1.5.1
iso.3.6.1.2.1.2.2.1.5.2 = Gauge32: 0
+ Redes3 git:(master) x snmpgetnext -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.2.2.1.5.2
iso.3.6.1.2.1.2.2.1.6.1 = ""
```

Figura 3.10: Velocidad de las interfaces con comando snmpgetnext.

Comando: snmpwalk (figura 3.11)

```
+ Redes3 git:(master) x snmpwalk -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.2.2.1.2
iso.3.6.1.2.1.2.2.1.2.1 = STRING: "lo"
iso.3.6.1.2.1.2.2.1.2.2 = STRING: "Intel Corporation Device 24fd"
iso.3.6.1.2.1.2.2.1.2.3 = STRING: "vboxnet0"
+ Redes3 git:(master) x snmpwalk -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.2.2.1.5
iso.3.6.1.2.1.2.2.1.5.1 = Gauge32: 10000000
iso.3.6.1.2.1.2.2.1.5.2 = Gauge32: 0
iso.3.6.1.2.1.2.2.1.5.3 = Gauge32: 10000000
```

Figura 3.11: Velocidad de las interfaces en Linux con comando snmpwalk.

En el caso de Windows, el resultado mostrado en la figura 3.12 fue:

- Software Loopback Interface 1 = 1073741824
- Microsoft 6to4 Adapter = 0
- Microsoft IP-HTTPS Platform Adapter = 0
- Microsoft Kernel Debug Network Adapter = 0
- Microsoft Teredo Tunnelling Adapter = 0
- Intel(R) PRO/1000 MT Desktop Adapter = 1000000000
- Intel(R) PRO/1000 MT Desktop Adapter-WFP Native MAC Layer LightWeight Filter-0000 = 1000000000

- Intel(R) PRO/1000 MT Desktop Adapter–QoS Packet Scheduler–0000 = 1000000000
- Intel(R) PRO/1000 MT Desktop Adapter–WFP 802.3 MAC Layer LightWeight Filter–0000 = 1000000000

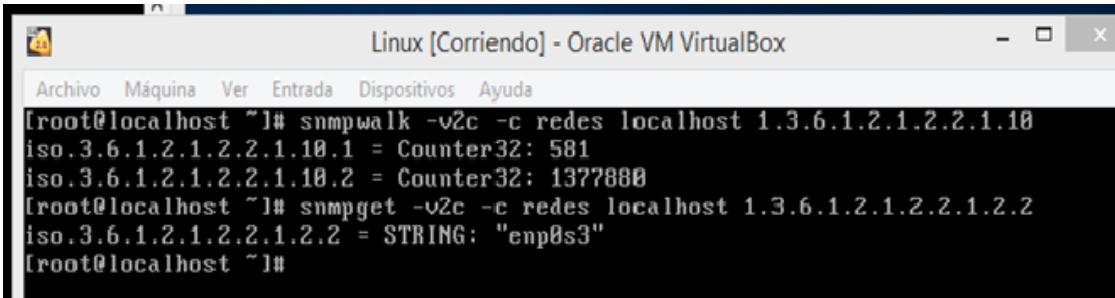
```
C:\Windows\system32>snmpwalk -v2c -c comunidad3 192.168.1.66 1.3.6.1.2.1.2.2.1.2
IF-MIB::ifDescr.1 = STRING: Software Loopback Interface 1
IF-MIB::ifDescr.2 = STRING: Microsoft 6to4 Adapter
IF-MIB::ifDescr.3 = STRING: Microsoft IP-HTTPS Platform Adapter
IF-MIB::ifDescr.4 = STRING: Microsoft Kernel Debug Network Adapter
IF-MIB::ifDescr.5 = STRING: Microsoft Teredo Tunneling Adapter
IF-MIB::ifDescr.6 = STRING: Intel(R) PRO/1000 MT Desktop Adapter
IF-MIB::ifDescr.7 = STRING: Intel(R) PRO/1000 MT Desktop Adapter-WFP Native MAC Layer LightWeight Filter-0000
IF-MIB::ifDescr.8 = STRING: Intel(R) PRO/1000 MT Desktop Adapter-QoS Packet Scheduler-0000
IF-MIB::ifDescr.9 = STRING: Intel(R) PRO/1000 MT Desktop Adapter-WFP 802.3 MAC Layer LightWeight Filter-0000

C:\Windows\system32>snmpwalk -v2c -c comunidad3 192.168.1.66 1.3.6.1.2.1.2.2.1.5
IF-MIB::ifSpeed.1 = Gauge32: 1073741824
IF-MIB::ifSpeed.2 = Gauge32: 0
IF-MIB::ifSpeed.3 = Gauge32: 0
IF-MIB::ifSpeed.4 = Gauge32: 0
IF-MIB::ifSpeed.5 = Gauge32: 0
IF-MIB::ifSpeed.6 = Gauge32: 1000000000
IF-MIB::ifSpeed.7 = Gauge32: 1000000000
IF-MIB::ifSpeed.8 = Gauge32: 1000000000
IF-MIB::ifSpeed.9 = Gauge32: 1000000000
```

Figura 3.12: Velocidad de las interfaces en Windows.

4. ¿Cuál es la interfaz que ha recibido el mayor número de octetos?

SO Centos Con ifInOctets vemos que interfaz tiene el mayor número de octetos en este caso es la 2, con ifDescr(2) vemos que la interfaz 2 es enpos3. figura 3.13



```
Linux [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
[root@localhost ~]# snmpwalk -v2c -c redes localhost 1.3.6.1.2.1.2.2.1.10
iso.3.6.1.2.1.2.2.1.10.1 = Counter32: 581
iso.3.6.1.2.1.2.2.1.10.2 = Counter32: 1377880
[root@localhost ~]# snmpget -v2c -c redes localhost 1.3.6.1.2.1.2.2.1.2.2
iso.3.6.1.2.1.2.2.1.2.2 = STRING: "enp0s3"
[root@localhost ~]#
```

Figura 3.13: El OID para ver los Octetos es 1.3.6.1.2.1.2.2.1.10

SO Windows Con snmpwalk podemos visualizar la tabla de las 26 interfaces con las que cuenta el so Windows y con la OID 1.3.6.1.2.1.2.2.1.10 (ifInOctets) se observan los octetos de cada una.

figura 3.14

```
C:\>Users\arturo>snmpwalk -v2c -c ComunidadWindows localhost 1.3.6.1.2.1.2.1.1.10
IF-MIB::ifInOctets.1 = Counter32: 0
IF-MIB::ifInOctets.2 = Counter32: 0
IF-MIB::ifInOctets.3 = Counter32: 0
IF-MIB::ifInOctets.4 = Counter32: 49243874
IF-MIB::ifInOctets.5 = Counter32: 0
IF-MIB::ifInOctets.6 = Counter32: 0
IF-MIB::ifInOctets.7 = Counter32: 0
IF-MIB::ifInOctets.8 = Counter32: 0
IF-MIB::ifInOctets.9 = Counter32: 0
IF-MIB::ifInOctets.10 = Counter32: 0
IF-MIB::ifInOctets.11 = Counter32: 0
IF-MIB::ifInOctets.12 = Counter32: 0
IF-MIB::ifInOctets.13 = Counter32: 0
IF-MIB::ifInOctets.14 = Counter32: 0
IF-MIB::ifInOctets.15 = Counter32: 0
IF-MIB::ifInOctets.16 = Counter32: 0
IF-MIB::ifInOctets.17 = Counter32: 0
IF-MIB::ifInOctets.18 = Counter32: 0
IF-MIB::ifInOctets.19 = Counter32: 0
IF-MIB::ifInOctets.20 = Counter32: 0
IF-MIB::ifInOctets.21 = Counter32: 49243874
IF-MIB::ifInOctets.22 = Counter32: 49243874
IF-MIB::ifInOctets.23 = Counter32: 49243874
IF-MIB::ifInOctets.24 = Counter32: 49243874
IF-MIB::ifInOctets.25 = Counter32: 49243874
IF-MIB::ifInOctets.26 = Counter32: 49243874
IF-MIB::ifInOctets.27 = Counter32: 0
IF-MIB::ifInOctets.28 = Counter32: 0
IF-MIB::ifInOctets.29 = Counter32: 0
IF-MIB::ifInOctets.30 = Counter32: 0
IF-MIB::ifInOctets.31 = Counter32: 0
IF-MIB::ifInOctets.32 = Counter32: 0
IF-MIB::ifInOctets.33 = Counter32: 0
IF-MIB::ifInOctets.34 = Counter32: 0
IF-MIB::ifInOctets.35 = Counter32: 0
IF-MIB::ifInOctets.36 = Counter32: 0
```

Figura 3.14: tabla de las interfaces en Windows con sus Octetos

Tenemos 7 interfaces que cuentan con la misma cantidad de octetos (.4, .21, .22, .23, .24, .25, .26) y podemos ver que esas interfaces son:

figura 3.15

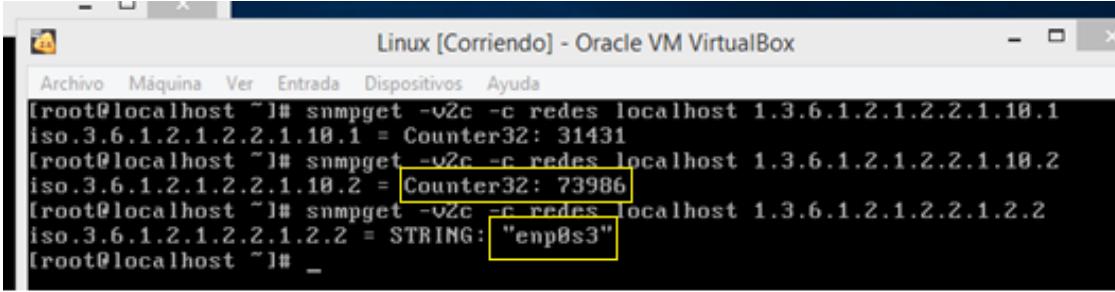
```
C:\>Users\arturo>snmpget -v2c -c ComunidadWindows localhost 1.3.6.1.2.1.2.1.2.4
IF-MIB::ifDescr.4 = STRING: Qualcomm Atheros AR956x Wireless Network Adapter
C:\>Users\arturo>snmpget -v2c -c ComunidadWindows localhost 1.3.6.1.2.1.2.2.1.2.2.1
IF-MIB::ifDescr.21 = STRING: Qualcomm Atheros AR956x Wireless Network Adapter-WF Native MAC Layer LightWeight Filter-0000
C:\>Users\arturo>snmpget -v2c -c ComunidadWindows localhost 1.3.6.1.2.1.2.2.1.2.2.2
IF-MIB::ifDescr.22 = STRING: Qualcomm Atheros AR956x Wireless Network Adapter-Virtual WiFi Filter Driver-0000
C:\>Users\arturo>snmpget -v2c -c ComunidadWindows localhost 1.3.6.1.2.1.2.2.1.2.2.3
IF-MIB::ifDescr.23 = STRING: Qualcomm Atheros AR956x Wireless Network Adapter-Native WiFi Filter Driver-0000
C:\>Users\arturo>snmpget -v2c -c ComunidadWindows localhost 1.3.6.1.2.1.2.2.1.2.2.5
IF-MIB::ifDescr.25 = STRING: Qualcomm Atheros AR956x Wireless Network Adapter-QoS Packet Scheduler-0000
C:\>Users\arturo>snmpget -v2c -c ComunidadWindows localhost 1.3.6.1.2.1.2.2.1.2.2.6
IF-MIB::ifDescr.26 = STRING: Qualcomm Atheros AR956x Wireless Network Adapter-WFP 802.3 MAC Layer LightWeight Filter-0000
C:\>Users\arturo>
```

Figura 3.15: Interfaces

5. Indica cuál interfaz de red ha recibido el mayor número de octetos

SO Centos enpos3 cuenta con 73986 octetos ifInOctets (OID 1.3.6.1.2.1.2.2.1.10.2) el cuál es **73986**.

figura 3.16

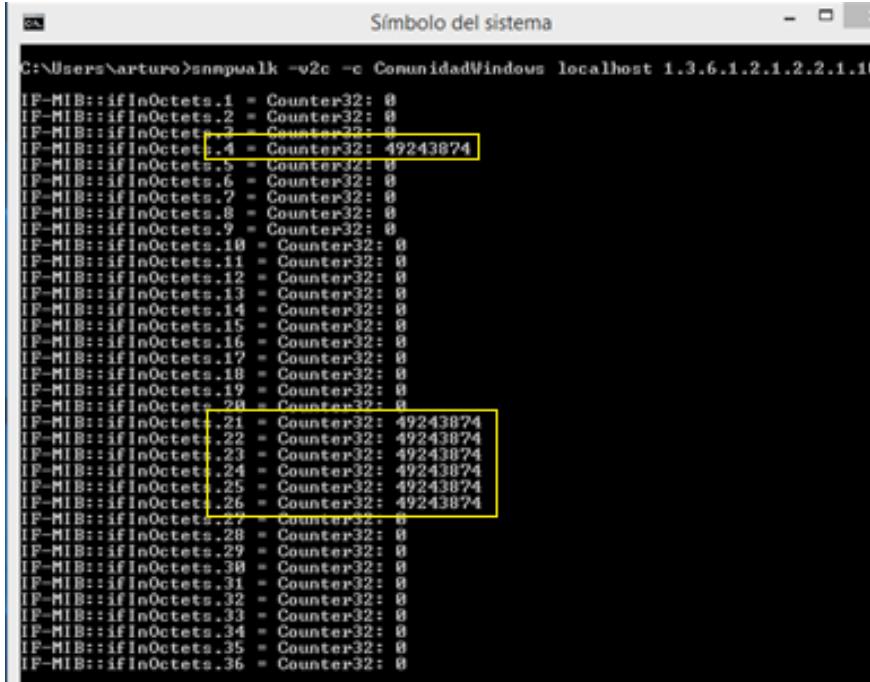


```
[root@localhost ~]# snmpget -v2c -c redes localhost 1.3.6.1.2.1.2.2.1.10.2
iso.3.6.1.2.1.2.2.1.10.2 = Counter32: 73986
```

Figura 3.16: Octetos en enp0s3

SO Windows El número de octetos que han recibido las 7 interfaces por igual es **49243874**.

figura 3.17



```
C:\Users\arturo>snmpwalk -v2c -c ComunidadWindows localhost 1.3.6.1.2.1.2.2.1.10
IF-MIB::ifInOctets.1 = Counter32: 49243874
IF-MIB::ifInOctets.2 = Counter32: 49243874
IF-MIB::ifInOctets.3 = Counter32: 49243874
IF-MIB::ifInOctets.4 = Counter32: 49243874
IF-MIB::ifInOctets.5 = Counter32: 49243874
IF-MIB::ifInOctets.6 = Counter32: 49243874
IF-MIB::ifInOctets.7 = Counter32: 49243874
IF-MIB::ifInOctets.8 = Counter32: 49243874
IF-MIB::ifInOctets.9 = Counter32: 49243874
IF-MIB::ifInOctets.10 = Counter32: 49243874
IF-MIB::ifInOctets.11 = Counter32: 49243874
IF-MIB::ifInOctets.12 = Counter32: 49243874
IF-MIB::ifInOctets.13 = Counter32: 49243874
IF-MIB::ifInOctets.14 = Counter32: 49243874
IF-MIB::ifInOctets.15 = Counter32: 49243874
IF-MIB::ifInOctets.16 = Counter32: 49243874
IF-MIB::ifInOctets.17 = Counter32: 49243874
IF-MIB::ifInOctets.18 = Counter32: 49243874
IF-MIB::ifInOctets.19 = Counter32: 49243874
IF-MIB::ifInOctets.20 = Counter32: 49243874
IF-MIB::ifInOctets.21 = Counter32: 49243874
IF-MIB::ifInOctets.22 = Counter32: 49243874
IF-MIB::ifInOctets.23 = Counter32: 49243874
IF-MIB::ifInOctets.24 = Counter32: 49243874
IF-MIB::ifInOctets.25 = Counter32: 49243874
IF-MIB::ifInOctets.26 = Counter32: 49243874
IF-MIB::ifInOctets.27 = Counter32: 49243874
IF-MIB::ifInOctets.28 = Counter32: 49243874
IF-MIB::ifInOctets.29 = Counter32: 49243874
IF-MIB::ifInOctets.30 = Counter32: 49243874
IF-MIB::ifInOctets.31 = Counter32: 49243874
IF-MIB::ifInOctets.32 = Counter32: 49243874
IF-MIB::ifInOctets.33 = Counter32: 49243874
IF-MIB::ifInOctets.34 = Counter32: 49243874
IF-MIB::ifInOctets.35 = Counter32: 49243874
IF-MIB::ifInOctets.36 = Counter32: 49243874
```

Figura 3.17: Octetos en Windows

6. ¿Cuál es la MAC de esa interfaz? **SO Centos** Con ifphysAddres (OID 1.3.6.1.2.1.2.2.1.6.2)podemos visualizar le Mac de enpos3 8:0:27:91:60:40.

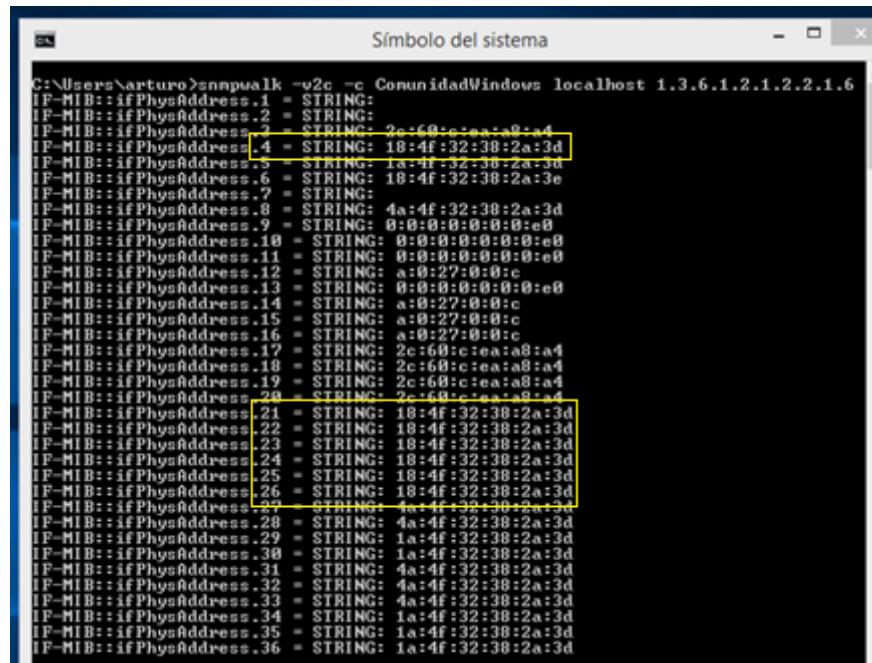
figura 3.18

```
[root@localhost ~]# snmpget -v2c -c redes localhost 1.3.6.1.2.1.2.2.1.2.2
iso.3.6.1.2.1.2.2.1.2.2 = STRING: "enp0s3"
[root@localhost ~]# snmpget -v2c -c redes localhost 1.3.6.1.2.1.2.2.1.6.2
iso.3.6.1.2.1.2.2.1.6.2 = Hex-STRING: 08 00 27 91 60 40
[root@localhost ~]#
```

Figura 3.18: Dirección Mac de Enp0s3

SO Windows Con ifPhysAddress se puede observar que las 7 interfaces que tiene el número igualitario de octetos cuentan con la misma dirección Mac, para esto nos apoyamos del comando snmpwalk para poder visualizar en forma de lista las diferentes direcciones de cada interfaz con la que cuenta Windows. (OID 1.3.6.1.2.1.2.2.1.6) Mac **18:4f:32:38:2a:3d**.

figura 3.19



```
C:\Users\arturo>snmpwalk -v2c -c ComunidadWindows localhost 1.3.6.1.2.1.2.2.1.6
IP-MIB::ifPhysAddress.1 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.2 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.3 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.4 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.5 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.6 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.7 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.8 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.9 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.10 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.11 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.12 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.13 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.14 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.15 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.16 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.17 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.18 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.19 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.20 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.21 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.22 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.23 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.24 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.25 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.26 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.27 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.28 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.29 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.30 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.31 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.32 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.33 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.34 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.35 = STRING: 18:4f:32:38:2a:3d
IP-MIB::ifPhysAddress.36 = STRING: 18:4f:32:38:2a:3d
```

Figura 3.19: Tabla de direcciones Mac en Windows

7. ¿Cuál es la ip de la Interfaz que ha recibido el mayor número de octetos?

Primero listamos en la figura 3.20 los octetos que han recibido cada una de las interfaces. Esto nos devuelve en índice del objeto correspondiente a cada interfaz.

```
root@ss:~/redes3/Redes3# snmpwalk -v2c -c comunidadSNMPsamuel localhost 1.3.6.1.2.1.31.1.1.1.6
iso.3.6.1.2.1.31.1.1.1.6.1 = Counter64: 173102
iso.3.6.1.2.1.31.1.1.1.6.2 = Counter64: 4260118
```

Figura 3.20: Octetos por interfaz.

Observamos que la interfaz que ha recibido más octetos es la que tiene el índice 2. Por lo tanto, procedemos listar las interfaces por índice mostrando su dirección IP, tal y como se muestra en la figura 3.21.

```
root@ss:~/redes3/Redes3# snmpwalk -v2c -c comunidadSNMPsamuel localhost 1.3.6.1.2.1.4.20.1.2
iso.3.6.1.2.1.4.20.1.2.127.0.0.1 = INTEGER: 1
iso.3.6.1.2.1.4.20.1.2.192.168.61.129 = INTEGER: 2
```

Figura 3.21: Direcciones IP de cada interfaz.

Obtenemos que, la dirección IP de la interfaz que ha recibido más octetos es la dirección 192.168.61.129. Repetimos el procedimiento anterior para el caso de Windows. Con respecto al número de octetos recibidos por cada interfaz obtenemos lo mostrado en la figura 3.22, mientras que la lista de direcciones IP se muestra en la figura 3.23.

```
C:\Windows\system32>snmpwalk -v2c -c comunidadSNMPsamuel localhost 1.3.6.1.2.1.31.1.1.1.6
C:\Windows\system32>
```

Figura 3.22: Octetos por interfaz.

```
C:\Windows\system32>snmpwalk -v2c -c comunidadSNMPsamuel localhost 1.3.6.1.2.1.4.20.1.2
IP-MIB::ipAdEntIfIndex.127.0.0.1 = INTEGER: 1
IP-MIB::ipAdEntIfIndex.192.168.61.135 = INTEGER: 3
```

Figura 3.23: Direcciones IP de cada interfaz.

Sin embargo, obtenemos que, el número de octetos recibidos por cada interfaz no está definido.

8. ¿Cuántos mensajes ICMP ha recibido el agente?

En la figura 3.24 observamos que el número de mensajes que ha recibido el agente Linux es de 1. Mientras que, en la figura 3.25 observamos que el número de mensajes que ha recibido el agente Windows es de 0.

```
root@ss:~/redes3/Redes3# snmpget -v2c -c comunidadSNMPsamuel localhost 1.3.6.1.2.1.5.1.0
iso.3.6.1.2.1.5.1.0 = Counter32: 1
```

Figura 3.24: Mensajes ICMP recibidos en Linux.

```
C:\Windows\system32>snmpwalk -v2c -c comunidadSNMPsamuel localhost 1.3.6.1.2.1.5.1.0
IP-MIB::icmpInMsgs.0 = Counter32: 0
```

Figura 3.25: Mensajes ICMP recibidos en Windows.

9. ¿Cuántas entradas tiene la tabla de enrutamiento IP?

Para responder esta pregunta, basta con listar y contar el número de entradas en alguna de las columnas de la tabla ipRouteTable. En este caso, utilizamos el objeto ipRouteDest.

En la figura 3.26 observamos que el número de entradas de ipRouteTable del agente Linux es de 2. Mientras que, en la figura 3.27 observamos que el número de entradas de ipRouteTable del agente Windows es de 9.

```
root@ss:~/redes3/Redes3# snmpwalk -v2c -c comunidadSNMPsamuel localhost 1.3.6.1.2.1.4.21.1.1
iso.3.6.1.2.1.4.21.1.1.0.0.0.0 = IpAddress: 0.0.0.0
iso.3.6.1.2.1.4.21.1.1.192.168.61.0 = IpAddress: 192.168.61.0
```

Figura 3.26: Entradas en ipRouteTable en Linux.

```
C:\Windows\system32>snmpwalk -v2c -c comunidadSNMPsamuel localhost 1.3.6.1.2.1.4.21.1.1
RFC1213-MIB::ipRouteDest.0.0.0.0 = IpAddress: 0.0.0.0
RFC1213-MIB::ipRouteDest.127.0.0.0 = IpAddress: 127.0.0.0
RFC1213-MIB::ipRouteDest.127.0.0.1 = IpAddress: 127.0.0.1
RFC1213-MIB::ipRouteDest.127.255.255.255 = IpAddress: 127.255.255.255
RFC1213-MIB::ipRouteDest.192.168.61.0 = IpAddress: 192.168.61.0
RFC1213-MIB::ipRouteDest.192.168.61.135 = IpAddress: 192.168.61.135
RFC1213-MIB::ipRouteDest.192.168.61.255 = IpAddress: 192.168.61.255
RFC1213-MIB::ipRouteDest.224.0.0.0 = IpAddress: 224.0.0.0
RFC1213-MIB::ipRouteDest.255.255.255.255 = IpAddress: 255.255.255.255
```

Figura 3.27: Entradas en ipRouteTable en Windows.

10. ¿El agente ha recibido mensajes TCP? ¿Cuántos?

Comando: **snmpget** (figura 3.28)

```
+ Redes3 git:(master) x snmpget -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.6.10.0
iso.3.6.1.2.1.6.10.0 = Counter32: 2178217
```

Figura 3.28: Número de mensajes TCP recibidos en Linux con comando snmpget.

Comando: **snmpgetnext** (figura 3.29)

```
+ Redes3 git:(master) x snmpgetnext -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.6.10.0
iso.3.6.1.2.1.6.11.0 = Counter32: 2166306
```

Figura 3.29: Número de mensajes TCP recibidos en Linux con comando snmpgetnext.

Comando: **snmpwalk** (figura 3.30)

```
+ Redes3 git:(master) x snmpwalk -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.6.10.0
iso.3.6.1.2.1.6.10.0 = Counter32: 2179163
+ Redes3 git:(master) x snmpwalk -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.6.11.0 = Counter32: 2166306
```

	Type	Line	M	Page
\Label				18
\end{Figure}				19
\Float				20
\textbf{(figur				21
\Float				22
\textbf{(figur				23
\textbf{(figur				24
\Float				25
\textbf{(figur				26
\textbf{(figur				27
				28

Figura 3.30: Número de mensajes TCP recibidos en Linux con comando snmpwalk.

11. ¿Cuántos mensajes EGP ha recibido el agente?

Debido a que el grupo **EGP** está asociado a la comunicación de la interconexión de redes externas. El alcance de la implementación se **SNMP** se limita a equipos de la misma red, por esa razón no se encuentran mensajes EGP en ambos agentes, a continuación se muestran los resultados de la consulta **egpInMsgs**.

Consulta SNMP: **egpInMsgs** Windows OID: 1.3.6.1.2.1.8.1.0 (figura 3.31)

```
C:\Users\arturo>snmpget -v2c -c comunidadWindows localhost 1.3.6.1.2.1.8.1
RFC1213-MIB::egpInMsgs = No Such Object available on this agent at this OID
```

Figura 3.31: S.O Windows no tiene mensajes EGP

Consulta SNMP: egpInMsgs Centos OID: 1.3.6.1.2.1.8.1.0 (figura 3.32)

```
[root@localhost ~]# snmpget -v2c -c redes localhost 1.3.6.1.2.1.8.0
iso.3.6.1.2.1.8.0 = No Such Object available on this agent at this OID
[root@localhost ~]#
```

Figura 3.32: S.O Centos no cuenta con mensajes EGP

12. Indica el Sistema Operativo que maneja el agente.

Con el grupo System de la MIB SNMP encontramos la información relacionada a los agentes, con la **consulta SNMP sysDescr** podemos ver la información del agente tanto a nivel Hardware como Software, los puntos más importantes de la consulta SNMP es el sistema operativo con el cual cuenta el agente y la versión, como se puede ver en la imagen 3.33 muestra la descripción del Sistema Windows y en la imagen ?? del Sistema Operativo Linux.

Consulta SNMP: sysDescr Windows OID: 1.3.6.1.2.1.1.0 (figura 3.33)

```
C:\Users\arturo>snmpget -v2c -c comunidadWindows localhost 1.3.6.1.2.1.1.0
SNMPv2-MIB::sysDescr.0 = STRING: Hardware: Intel64 Family 6 Model 76 Stepping 3
AT/AT COMPATIBLE - Software: Windows Version 6.3 <Build 9600 Multiprocessor Free
>
```

Figura 3.33: Información del Sistema Operativo Windows

Consulta SNMP: sysDescr Linux OID: 1.3.6.1.2.1.1.0 (figura 3.34)

```
[root@localhost ~]# snmpget -v2c -c redes localhost 1.3.6.1.2.1.1.0
iso.3.6.1.2.1.1.0 = STRING: "Linux localhost.localdomain 3.10.0-862.11.6.el7.x86_64 #1 SMP Tue Aug 14 21:49:04 UTC 2018 x86_64"
[root@localhost ~]#
```

Figura 3.34: Información del Sistema Operativo Centos

13. Modifica el nombre del contacto o la ubicación del sistema de un agente.

Con la consulta **snmpset** podemos modificar el valor de un OID, esto se hace mediante el comando **snmpset versión -c comunidad localhost OID s .El nuevo valor**. A continuación se muestran los resultados de la consulta en el Sistema Operativo Windows 3.35 y Sistema Operativo Linux 3.36

Consulta snmpset: sysContact(4) sysUbication(6) Windows (figura 3.35) y (figura 3.36)

```
C:\Users\arturo>snmpget -v2c -c comunidadWindows localhost 1.3.6.1.2.1.1.4.0
SNMPv2-MIB::sysContact.0 = STRING:
C:\Users\arturo>snmpget -v2c -c comunidadWindows localhost 1.3.6.1.2.1.1.6.0
SNMPv2-MIB::sysLocation.0 = STRING: holai
C:\Users\arturo>snmpset -v1 -c comunidadWindows localhost 1.3.6.1.2.1.1.6.0 s "MyHouse"
SNMPv2-MIB::sysLocation.0 = STRING: MyHouse
```

Figura 3.35: valores de sysContact y sysUbication, se cambio la ubicación del agente a "MyHouse"

```
C:\Users\arturo>snmpget -v2c -c comunidadWindows localhost 1.3.6.1.2.1.1.6.0
SNMPv2-MIB::sysLocation.0 = STRING: MyHouse
C:\Users\arturo>snmpget -v2c -c comunidadWindows localhost 1.3.6.1.2.1.1.4.0
SNMPv2-MIB::sysContact.0 = STRING:
C:\Users\arturo>snmpset -v1 -c comunidadWindows localhost 1.3.6.1.2.1.1.4.0 s "WindowsAgent"
SNMPv2-MIB::sysContact.0 = STRING: WindowsAgent
C:\Users\arturo>snmpget -v2c -c comunidadWindows localhost 1.3.6.1.2.1.1.4.0
SNMPv2-MIB::sysContact.0 = STRING: WindowsAgent
C:\Users\arturo>_
```

Figura 3.36: Se cambio el valor de sysContact "WindowsAgentz se muestran los cambios hechos en sysContact y sysUbication

Consulta snmpset: sysContact(4) sysUbication(6) Centos (figura 3.38) y (figura ??)

```
[root@myserver ~]# snmpget -v1 -c comunidad localhost 1.3.6.1.2.1.1.6.0
SNMPv2-MIB::sysLocation.0 = STRING: lab de redes
[root@myserver ~]# snmpset -v1 -c comunidad localhost 1.3.6.1.2.1.1.6.0 s "LaptopAcer"
SNMPv2-MIB::sysLocation.0 = STRING: LaptopAcer
[root@myserver ~]# snmpget -v1 -c comunidad localhost 1.3.6.1.2.1.1.6.0
SNMPv2-MIB::sysLocation.0 = STRING: LaptopAcer
[root@myserver ~]# _
```

Figura 3.37: Se cambio el valor de sysUbication(6) "LaptopAcerz se muestran los cambios hechos del valor inicial al final

```
[root@myserver ~]# snmpget -v1 -c comunidad localhost 1.3.6.1.2.1.1.4.0
SNMPv2-MIB::sysContact.0 = STRING: root@localhost
[root@myserver ~]# snmpset -v1 -c comunidad localhost 1.3.6.1.2.1.1.4.0 s "CentOS"
SNMPv2-MIB::sysContact.0 = STRING: CentOS
[root@myserver ~]# snmpget -v1 -c comunidad localhost 1.3.6.1.2.1.1.4.0
SNMPv2-MIB::sysContact.0 = STRING: CentOS
[root@myserver ~]# _
```

Figura 3.38: Muestra el valor de sysContact(4) y se modifica por CentOS"finalmente muestra el cambio.

14. Dibuja la MIB del agente.

Observamos en la figura 3.39, la ejecución del comando **snmpwalk** al OID 1.3.6.1.2.1 correspondiente al objeto **mib-2**.

```

root@ss:~/redes3/Redes3/ReporteLatex/MarcoTeorico# snmpwalk -v2c -c comunidadEquipo12_4CM3 localhost 1.3.6.1.2.1
iso.3.6.1.2.1.1.1.0 = STRING: "Linux ss 4.17.0-kalil-amd64 #1 SMP Debian 4.17.8-1kalil (2018-07-24) x86_64"
iso.3.6.1.2.1.1.2.0 = OID: iso.3.6.1.4.1.8072.3.2.10
iso.3.6.1.2.1.1.3.0 = Timeticks: (8334) 0:01:23.34
iso.3.6.1.2.1.1.4.0 = STRING: "samuel.asm@outlook.com"
iso.3.6.1.2.1.1.5.0 = STRING: "ss"
iso.3.6.1.2.1.1.6.0 = STRING: "LabProg1"
iso.3.6.1.2.1.1.8.0 = Timeticks: (18) 0:00:00.18
iso.3.6.1.2.1.1.9.1.2.1 = OID: iso.3.6.1.6.3.11.3.1.1
iso.3.6.1.2.1.1.9.1.2.2 = OID: iso.3.6.1.6.3.15.2.1.1
iso.3.6.1.2.1.1.9.1.2.3 = OID: iso.3.6.1.6.3.10.3.1.1
iso.3.6.1.2.1.1.9.1.2.4 = OID: iso.3.6.1.6.3.1
iso.3.6.1.2.1.1.9.1.2.5 = OID: iso.3.6.1.6.3.16.2.2.1
iso.3.6.1.2.1.1.9.1.2.6 = OID: iso.3.6.1.2.1.49
iso.3.6.1.2.1.1.9.1.2.7 = OID: iso.3.6.1.2.1.4
iso.3.6.1.2.1.1.9.1.2.8 = OID: iso.3.6.1.2.1.50
iso.3.6.1.2.1.1.9.1.2.9 = OID: iso.3.6.1.6.3.13.3.1.3
iso.3.6.1.2.1.1.9.1.2.10 = OID: iso.3.6.1.2.1.92
iso.3.6.1.2.1.1.9.1.3.1 = STRING: "The MIB for Message Processing and Dispatching."
iso.3.6.1.2.1.1.9.1.3.2 = STRING: "The management information definitions for the SNMP User-based Security Model."
iso.3.6.1.2.1.1.9.1.3.3 = STRING: "The SNMP Management Architecture MIB."
iso.3.6.1.2.1.1.9.1.3.4 = STRING: "The MIB module for SNMPv2 entities"
iso.3.6.1.2.1.1.9.1.3.5 = STRING: "View-based Access Control Model for SNMP."
iso.3.6.1.2.1.1.9.1.3.6 = STRING: "The MIB module for managing TCP implementations"
iso.3.6.1.2.1.1.9.1.3.7 = STRING: "The MIB module for managing IP and ICMP implementations"
iso.3.6.1.2.1.1.9.1.3.8 = STRING: "The MIB module for managing UDP implementations"
iso.3.6.1.2.1.1.9.1.3.9 = STRING: "The MIB modules for managing SNMP Notification, plus filtering."
iso.3.6.1.2.1.1.9.1.3.10 = STRING: "The MIB module for logging SNMP Notifications."
iso.3.6.1.2.1.1.9.1.4.1 = Timeticks: (18) 0:00:00.18
iso.3.6.1.2.1.1.9.1.4.2 = Timeticks: (18) 0:00:00.18
iso.3.6.1.2.1.1.9.1.4.3 = Timeticks: (18) 0:00:00.18
iso.3.6.1.2.1.1.9.1.4.4 = Timeticks: (18) 0:00:00.18
iso.3.6.1.2.1.1.9.1.4.5 = Timeticks: (18) 0:00:00.18
iso.3.6.1.2.1.1.9.1.4.6 = Timeticks: (18) 0:00:00.18
iso.3.6.1.2.1.1.9.1.4.7 = Timeticks: (18) 0:00:00.18
iso.3.6.1.2.1.1.9.1.4.8 = Timeticks: (18) 0:00:00.18
iso.3.6.1.2.1.1.9.1.4.9 = Timeticks: (18) 0:00:00.18
iso.3.6.1.2.1.1.9.1.4.10 = Timeticks: (18) 0:00:00.18
iso.3.6.1.2.1.2.1.0 = INTEGER: 2
iso.3.6.1.2.1.2.2.1.1.1 = INTEGER: 1
iso.3.6.1.2.1.2.2.1.1.2 = INTEGER: 2
iso.3.6.1.2.1.2.2.1.2.1 = STRING: "lo"
iso.3.6.1.2.1.2.2.1.2.2 = STRING: "Intel Corporation 82545EM Gigabit Ethernet Controller (Copper)"
iso.3.6.1.2.1.2.2.1.3.1 = INTEGER: 24
iso.3.6.1.2.1.2.2.1.3.2 = INTEGER: 6
iso.3.6.1.2.1.2.2.1.4.1 = INTEGER: 65536
iso.3.6.1.2.1.2.2.1.4.2 = INTEGER: 1500
iso.3.6.1.2.1.2.2.1.5.1 = Gauge32: 1000000
iso.3.6.1.2.1.2.2.1.5.2 = Gauge32: 1000000000
iso.3.6.1.2.1.2.2.1.6.1 = ""
iso.3.6.1.2.1.2.2.1.6.2 = Hex-STRING: 00 0C 29 FF 59 0C
iso.3.6.1.2.1.2.2.1.7.1 = INTEGER: 1
iso.3.6.1.2.1.2.2.1.7.2 = INTEGER: 1
iso.3.6.1.2.1.2.2.1.8.1 = INTEGER: 1
iso.3.6.1.2.1.2.2.1.8.2 = INTEGER: 1
iso.3.6.1.2.1.2.2.1.9.1 = Timeticks: (0) 0:00:00.00
iso.3.6.1.2.1.2.2.1.9.2 = Timeticks: (0) 0:00:00.00
iso.3.6.1.2.1.2.2.1.10.1 = Counter32: 104918
iso.3.6.1.2.1.2.2.1.10.2 = Counter32: 1004370
iso.3.6.1.2.1.2.2.1.11.1 = Counter32: 2090

```

Figura 3.39: Dibujo de la MiB.

CAPÍTULO 4

Análisis de tráfico

En este capítulo se realizan capturas de tráfico para los comandos básicos de SNMP. Tanto las capturas como el análisis se realiza utilizando la herramienta Wireshark en Linux.

4.1. Análisis de tráfico con Wireshark

4.1.1. snmpget

En la figura 4.1 podemos observar la ejecución del comando **snmpget** consultando al objeto **system**.

```
root@ss:~/redes3/rrdtool2# snmpget -v2c -c comunidadSNMPsamuel localhost 1.3.6.1
.2.1.1.1.0
iso.3.6.1.2.1.1.1.0 = STRING: "Linux ss 4.17.0-kalil-amd64 #1 SMP Debian 4.17.8-
1kalil (2018-07-24) x86_64"
```

Figura 4.1: Comando snmpget.

Con ello, se capturan dos paquetes: uno de solicitud y otro de respuesta mostrados en la figura 4.2.

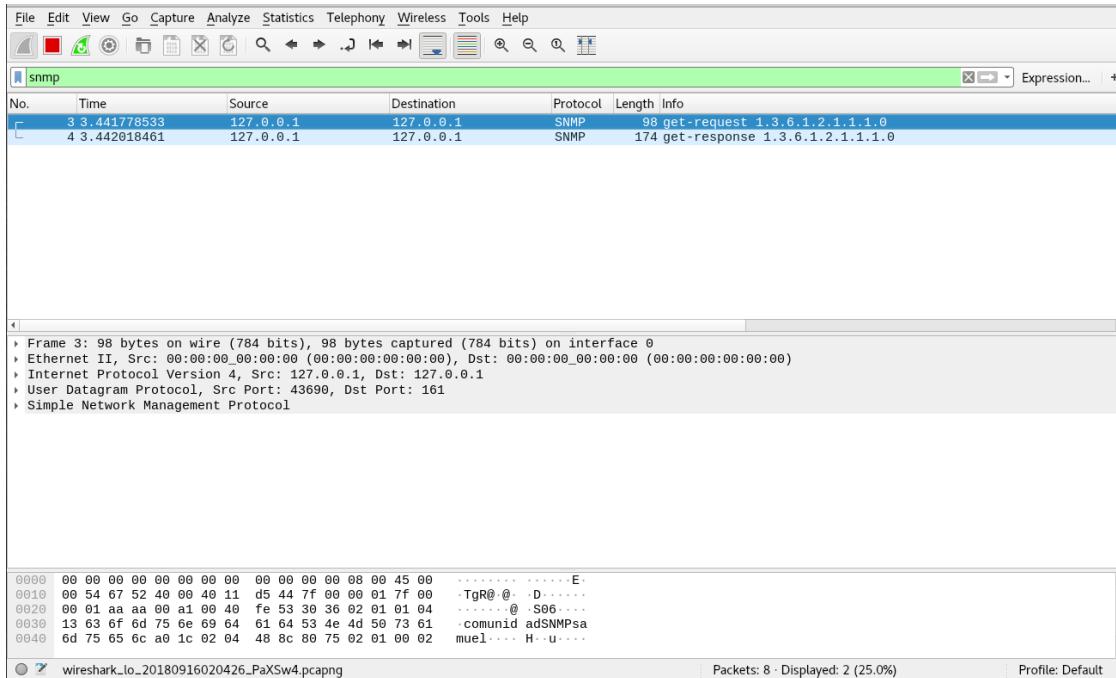


Figura 4.2: Captura snmpget.

Finalmente, en las figuras 4.3 y 4.4 vemos la estructura de los paquetes de solicitud y respuesta respectivamente. Cabe comentar que la información transmitida con la ejecución del comando **snmpget** viaja en claro, es decir, no está cifrada.

Por un lado, observamos que en la solicitud se envía el OID, mientras que en la respuesta se envía como valor la cadena que corresponde al objeto solicitado.

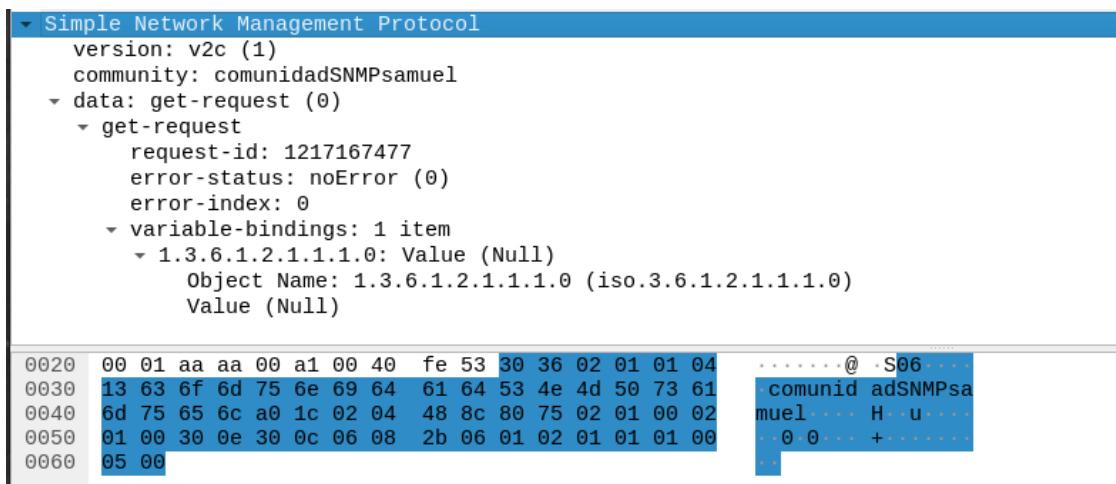


Figura 4.3: Solicitud snmpget.

```

> User Datagram Protocol, Src Port: 161, Dst Port: 43690
- Simple Network Management Protocol
  version: v2c (1)
  community: comunidadSNMPsamuel
  - data: get-response (2)
    - get-response
      request-id: 1217167477
      error-status: noError (0)
      error-index: 0
    - variable-bindings: 1 item
      - 1.3.6.1.2.1.1.0: 4c696e757820737320342e31372e302d6b616c69312d616d...
        Object Name: 1.3.6.1.2.1.1.0 (iso.3.6.1.2.1.1.0)
        Value (OctetString): 4c696e757820737320342e31372e302d6b616c69312d616d...
          Variable-binding-string: Linux ss 4.17.0-kali1-amd64 #1 SMP Debian 4.17.8-1kali1 (2018-07-24) x86_64

0020 00 01 00 a1 aa aa 00 8c fe 9f 30 81 81 02 01 01 ..... . 0
0030 04 13 63 6f 6d 75 6e 69 64 61 64 53 4e 4d 50 73 comuni dadasSNPs
0040 61 6d 75 65 6c a2 67 02 04 48 8c 80 75 02 01 00 amuel g H u
0050 02 01 00 30 59 30 57 06 08 2b 06 01 02 01 01 01 . 0YOW +.
0060 00 04 4b 4c 69 6e 75 78 20 73 73 20 34 2e 31 37 KLinux ss 4.17

```

Figura 4.4: Respuesta snmpget.

4.1.2. snmpgetnext

En el caso del comando **snmpgetnext**, podemos observar en la figura 4.5, que se realizó la solicitud del objeto **sysUpTime** con OID 1.3.6.1.2.1.1.3.0. No obstante, el funcionamiento de **snmpgetnext** consiste en regresar el objeto siguiente. En este caso, devuelve el OID 1.3.6.1.2.1.1.4.0 **sysContact**.

```

root@ss:~/redes3/rrdtool2# snmpgetnext -v2c -c comunidadSNMPsamuel localhost 1.3
.6.1.2.1.1.3.0
iso.3.6.1.2.1.1.4.0 = STRING: "samuel.asm@outlook.com"

```

Figura 4.5: Comando snmpgetnext.

Al observar la captura de paquetes en la figura 4.6, vemos que aunque el comando ejecutado es **snmpgetnext**, se realizan peticiones y respuestas de tipo **snmpget**.

No.	Time	Source	Destination	Protocol	Length	Info
	1 0.000000000	127.0.0.1	127.0.0.1	SNMP	98	get-next-request 1.3.6.1.2.1.1.3.0
	2 0.000212561	127.0.0.1	127.0.0.1	SNMP	120	get-response 1.3.6.1.2.1.1.4.0

Figura 4.6: Captura snmpgetnext.

Finalmente, en las figuras 4.7 y 4.8 vemos que los detalles de los paquetes transmitidos son de solicitud (**get-next-request**) y respuesta (**get-response**).

```

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 48983, Dst Port: 161
-> Simple Network Management Protocol
    version: v2c (1)
    community: comunidadSNMPsamuel
-> data: get-next-request (1)
    -> get-next-request
        request-id: 1702832104
        error-status: noError (0)
        error-index: 0
    -> variable-bindings: 1 item
        -> 1.3.6.1.2.1.1.3.0: Value (Null)
            Object Name: 1.3.6.1.2.1.1.3.0 (iso.3.6.1.2.1.1.3.0)
            Value (Null)

0020 00 01 bf 57 00 a1 00 40 fe 53 30 36 02 01 01 04 ...W...@ .S06...
0030 13 63 6f 6d 75 6e 69 64 61 64 53 4e 4d 50 73 61 .comunid adSNMPsa
0040 6d 75 65 6c a1 1c 02 04 65 7f 27 e8 02 01 00 02 muel.... e.'.....
0050 01 00 30 0e 30 0c 06 08 2b 06 01 02 01 01 03 00 ...0.0| +.....
0060 05 00

Simple Network Management Protocol (snmp), 56 bytes

```

Figura 4.7: Solicitud snmpgetnext.

```

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 161, Dst Port: 48983
-> Simple Network Management Protocol
    version: v2c (1)
    community: comunidadSNMPsamuel
-> data: get-response (2)
    -> get-response
        request-id: 1702832104
        error-status: noError (0)
        error-index: 0
    -> variable-bindings: 1 item
        -> 1.3.6.1.2.1.1.4.0: 73616d75656c2e61736d406f75746c6f6b2e636f6d
            Object Name: 1.3.6.1.2.1.1.4.0 (iso.3.6.1.2.1.1.4.0)
            Value (OctetString): 73616d75656c2e61736d406f75746c6f6b2e636f6d

0020 00 01 00 a1 bf 57 00 56 fe 69 30 4c 02 01 01 04 ....W.V.i0L...
0030 13 63 6f 6d 75 6e 69 64 61 64 53 4e 4d 50 73 61 .comunid adSNMPsa
0040 6d 75 65 6c a2 32 02 04 65 7f 27 e8 02 01 00 02 muel 2.. e.'.....
0050 01 00 30 24 30 22 06 08 2b 06 01 02 01 01 04 00 ...0$0" .. +.....
0060 04 16 73 61 6d 75 65 6c 2e 61 73 6d 40 6f 75 74 ..samuel .asm@out


```

Figura 4.8: Respuesta snmpgetnext.

4.1.3. snmpwalk

El comando **snmpwalk** realiza una serie de peticiones **snmpgetnext** automáticamente y se detiene cuando devuelve resultados que no están más dentro del rango del OID que se ingresó originalmente.

Como se muestra en la figura 4.9, observamos que se realizó la solicitud del OID 1.3.6.1.2.1.1. Con lo cual, se obtuvieron todos los OIDs sucesivos.

```
root@ss:~/redes3/rrdtool2# snmpwalk -v2c -c comunidadesSNMpsamuel localhost 1.3.6.1.2.1.1
iso.3.6.1.2.1.1.1.0 = STRING: "Linux ss 4.17.0-kalil1-amd64 #1 SMP Debian 4.17.8-1kalil (2018-07-24) x86_64"
iso.3.6.1.2.1.1.2.0 = OID: iso.3.6.1.4.1.8072.3.2.10
iso.3.6.1.2.1.1.3.0 = Timeticks: (134001) 0:22:20.01
iso.3.6.1.2.1.1.4.0 = STRING: "samuel.asm@outlook.com"
iso.3.6.1.2.1.1.5.0 = STRING: "ss"
iso.3.6.1.2.1.1.6.0 = STRING: "LabProg1"
iso.3.6.1.2.1.1.8.0 = Timeticks: (30) 0:00:00.30
iso.3.6.1.2.1.1.9.1.2.1 = OID: iso.3.6.1.6.3.11.3.1.1
iso.3.6.1.2.1.1.9.1.2.2 = OID: iso.3.6.1.6.3.15.2.1.1
iso.3.6.1.2.1.1.9.1.2.3 = OID: iso.3.6.1.6.3.10.3.1.1
iso.3.6.1.2.1.1.9.1.2.4 = OID: iso.3.6.1.6.3.1
iso.3.6.1.2.1.1.9.1.2.5 = OID: iso.3.6.1.6.3.16.2.2.1
iso.3.6.1.2.1.1.9.1.2.6 = OID: iso.3.6.1.2.1.49
iso.3.6.1.2.1.1.9.1.2.7 = OID: iso.3.6.1.2.1.4
iso.3.6.1.2.1.1.9.1.2.8 = OID: iso.3.6.1.2.1.50
iso.3.6.1.2.1.1.9.1.2.9 = OID: iso.3.6.1.6.3.13.3.1.3
iso.3.6.1.2.1.1.9.1.2.10 = OID: iso.3.6.1.2.1.92
iso.3.6.1.2.1.1.9.1.3.1 = STRING: "The MIB for Message Processing and Dispatching."
iso.3.6.1.2.1.1.9.1.3.2 = STRING: "The management information definitions for the SNMP User-based Security Model."
iso.3.6.1.2.1.1.9.1.3.3 = STRING: "The SNMP Management Architecture MIB."
iso.3.6.1.2.1.1.9.1.3.4 = STRING: "The MIB module for SNMPv2 entities"
iso.3.6.1.2.1.1.9.1.3.5 = STRING: "View-based Access Control Model for SNMP."
iso.3.6.1.2.1.1.9.1.3.6 = STRING: "The MIB module for managing TCP implementations"
iso.3.6.1.2.1.1.9.1.3.7 = STRING: "The MIB module for managing IP and ICMP implementations"
iso.3.6.1.2.1.1.9.1.3.8 = STRING: "The MIB module for managing UDP implementations"
iso.3.6.1.2.1.1.9.1.3.9 = STRING: "The MIB modules for managing SNMP Notification, plus filtering."
iso.3.6.1.2.1.1.9.1.3.10 = STRING: "The MIB module for logging SNMP Notifications."
iso.3.6.1.2.1.1.9.1.4.1 = Timeticks: (30) 0:00:00.30
iso.3.6.1.2.1.1.9.1.4.2 = Timeticks: (30) 0:00:00.30
iso.3.6.1.2.1.1.9.1.4.3 = Timeticks: (30) 0:00:00.30
iso.3.6.1.2.1.1.9.1.4.4 = Timeticks: (30) 0:00:00.30
iso.3.6.1.2.1.1.9.1.4.5 = Timeticks: (30) 0:00:00.30
iso.3.6.1.2.1.1.9.1.4.6 = Timeticks: (30) 0:00:00.30
iso.3.6.1.2.1.1.9.1.4.7 = Timeticks: (30) 0:00:00.30
iso.3.6.1.2.1.1.9.1.4.8 = Timeticks: (30) 0:00:00.30
iso.3.6.1.2.1.1.9.1.4.9 = Timeticks: (30) 0:00:00.30
iso.3.6.1.2.1.1.9.1.4.10 = Timeticks: (30) 0:00:00.30
```

Figura 4.9: Comando snmpwalk.

Respecto a los paquetes transmitidos, podemos ver en la figura 4.10 que consisten en una serie de solicitudes (**get-next-request**) y respuestas (**get-response**).

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	SNMP	96	get-next-request 1.3.6.1.2.1.1
2	0.000212878	127.0.0.1	127.0.0.1	SNMP	174	get-response 1.3.6.1.2.1.1.1.0
3	0.000313822	127.0.0.1	127.0.0.1	SNMP	98	get-next-request 1.3.6.1.2.1.1.1.0
4	0.000383393	127.0.0.1	127.0.0.1	SNMP	108	get-response 1.3.6.1.2.1.1.2.0
5	0.000419529	127.0.0.1	127.0.0.1	SNMP	98	get-next-request 1.3.6.1.2.1.1.2.0
6	0.000476050	127.0.0.1	127.0.0.1	SNMP	101	get-response 1.3.6.1.2.1.1.3.0
7	0.000511885	127.0.0.1	127.0.0.1	SNMP	98	get-next-request 1.3.6.1.2.1.1.3.0
8	0.000566754	127.0.0.1	127.0.0.1	SNMP	120	get-response 1.3.6.1.2.1.1.4.0
9	0.000597974	127.0.0.1	127.0.0.1	SNMP	98	get-next-request 1.3.6.1.2.1.1.4.0
10	0.000654815	127.0.0.1	127.0.0.1	SNMP	100	get-response 1.3.6.1.2.1.1.5.0
11	0.000685376	127.0.0.1	127.0.0.1	SNMP	98	get-next-request 1.3.6.1.2.1.1.5.0
12	0.000787680	127.0.0.1	127.0.0.1	SNMP	106	get-response 1.3.6.1.2.1.1.6.0
13	0.000831730	127.0.0.1	127.0.0.1	SNMP	98	get-next-request 1.3.6.1.2.1.1.6.0
14	0.000893891	127.0.0.1	127.0.0.1	SNMP	99	get-response 1.3.6.1.2.1.1.8.0
15	0.000926189	127.0.0.1	127.0.0.1	SNMP	98	get-next-request 1.3.6.1.2.1.1.8.0
16	0.001093344	127.0.0.1	127.0.0.1	SNMP	109	get-response 1.3.6.1.2.1.1.9.1.2.1
17	0.001136402	127.0.0.1	127.0.0.1	SNMP	100	get-next-request 1.3.6.1.2.1.1.9.1.2.1
18	0.001199348	127.0.0.1	127.0.0.1	SNMP	109	get-response 1.3.6.1.2.1.1.9.1.2.2
19	0.001232402	127.0.0.1	127.0.0.1	SNMP	100	get-next-request 1.3.6.1.2.1.1.9.1.2.2
20	0.001287019	127.0.0.1	127.0.0.1	SNMP	109	get-response 1.3.6.1.2.1.1.9.1.2.3

Figura 4.10: Capturas snmpwalk.

4.1.4. snmpset

Como lo muestra la figura 4.11, se realizó la modificación del objeto sysContact. A pesar de que en este caso se obtuvo un error. Podemos analizar el tráfico red generado en la figura 4.12 y consiguientes.

```
root@ss:~/redes3/rrdtool2# snmpset -v2c -c comunidadSNMPsamuel localhost 1.3.6.1.2.1.1.4.0 s asantiagom1401@alumno.ipn.mx
Error in packet.
Reason: notWritable (That object does not support modification)
Failed object: iso.3.6.1.2.1.1.4.0
```

Figura 4.11: Comando snmpset.

Vemos que se realiza un petición de tipo **set-request** y se obtiene una respuesta de tipo **get-response**.

No.	Time	Source	Destination	Protocol	Length	Info
3	10.597025472	127.0.0.1	127.0.0.1	SNMP	126	set-request 1.3.6.1.2.1.1.4.0
4	10.597213822	127.0.0.1	127.0.0.1	SNMP	126	get-response 1.3.6.1.2.1.1.4.0

Figura 4.12: Captura snmpset.

En los detalles de la solicitud de la figura 4.13, observamos que se envía un valor, el cual es la cadena que solicitamos modificar.

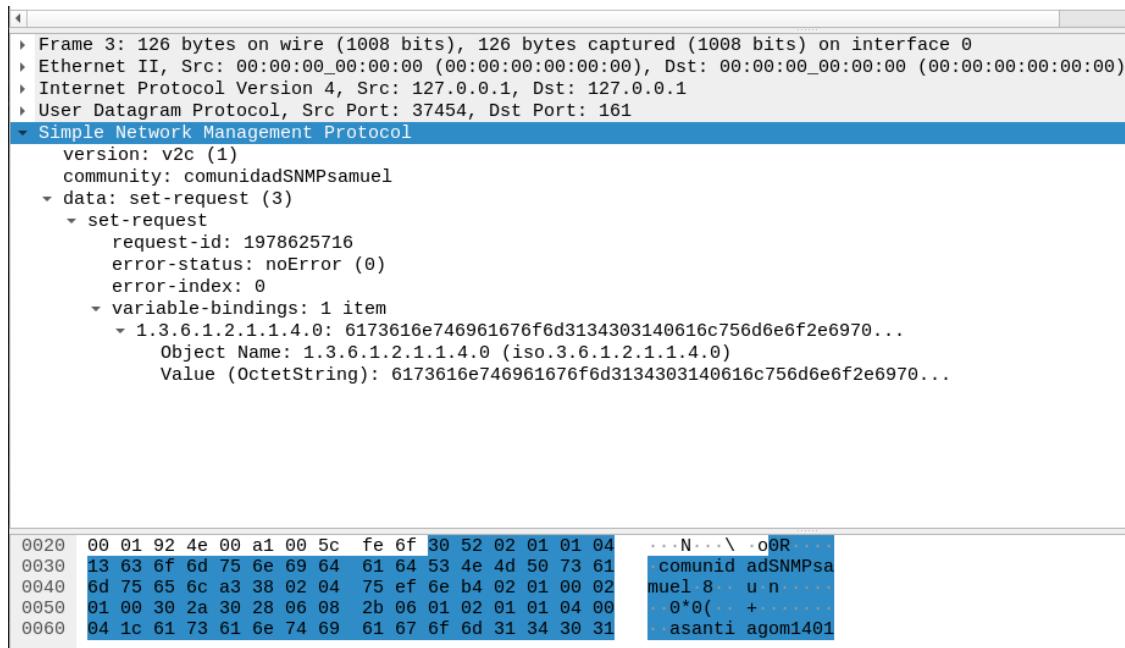


Figura 4.13: Solicitud snmpset.

```

Frame 4: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits) on interface 0
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
User Datagram Protocol, Src Port: 161, Dst Port: 37454
Simple Network Management Protocol
  version: v2c (1)
  community: comunidadSNMPsamuel
  ▾ data: get-response (2)
    ▾ get-response
      request-id: 1978625716
      error-status: notWritable (17)
      error-index: 1
    ▾ variable-bindings: 1 item
      ▾ 1.3.6.1.2.1.1.4.0: 6173616e746961676f6d3134303140616c756d6e6f2e6970...
        Object Name: 1.3.6.1.2.1.1.4.0 (iso.3.6.1.2.1.1.4.0)
        Value (OctetString): 6173616e746961676f6d3134303140616c756d6e6f2e6970...

```

0020	00 01 00 a1 92 4e 00 5c fe 6f 30 52 02 01 01 04N.\ .oDR ..
0030	13 63 6f 6d 75 6e 69 64 61 64 53 4e 4d 50 73 61	.comunid adSNMPsa
0040	6d 75 65 6c a2 38 02 04 75 ef 6e b4 02 01 11 02	muel 8 u n
0050	01 01 30 2a 30 28 06 08 2b 06 01 02 01 01 04 00	..00(+
0060	04 1c 61 73 61 6e 74 69 61 67 6f 6d 31 34 30 31	asanti agom1401

Figura 4.14: Respuesta snmpset.

CAPÍTULO 5

Implementación de un modelo de administración de red de SNMP

En esta sección, se mostrarán tanto el código más significativo como la ejecución del mismo a fin de mostrar el funcionamiento del gestor elaborado por el equipo con el cuál se dio solución al problema planteado en clase.

5.1. Pantalla de inicio

Para la pantalla de inicio, se solicitaba que al ingresar, el usuario pudiera observar un pequeño resumen de los dispositivos que estaban siendo monitoreados en ese momento, dicho resumen debía contener los siguientes puntos:

- Número de dispositivos monitorizados.
- Status de conexión de cada dispositivo.
- El número de interfaces de red que estaban disponibles de cada dispositivo.
- Status de cada interfaz de dichos dispositivos.

De esta forma, la pantalla inicial de nuestro administrador es como la siguiente (figura 5.1), en la que se separa con pequeños titulares cada uno de los puntos solicitados y en donde se observan los datos de cada dispositivo que estaba siendo monitoreado en ese momento.

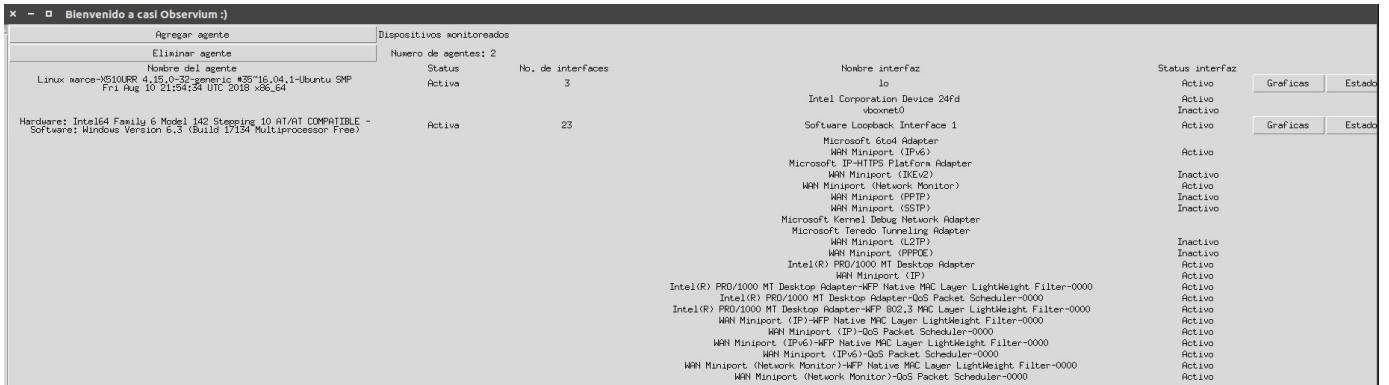


Figura 5.1: Pantalla principal.

A continuación explicaré el código más importante de esta sección.

Primero se ejecuta el método llamado getHostInfo como se muestra en la figura 5.2

```

292     def getHostInfo():
293         print 'holaaaaaa'
294         global agentCount
295         global canvasFrame
296         ip_community = [] # va con doble m no doble n
297         try:
298             if canvasFrame == None:
299                 canvasFrame = Frame(photoCanvas, width=1838, height=800)
300                 photoCanvas.create_window(0, 0, window=canvasFrame, anchor='nw')
301
302                 add = Tkinter.Button(canvasFrame, text ="Aregar agente", width=25, command = addClient).grid(row=0, column=0, sticky="nsew")
303                 delete = Tkinter.Button(canvasFrame, text="Eliminar agente", width=25, command=deleteClient).grid(row=1, column=0,sticky="nsew")
304                 #agentInfo = Tkinter.Button(canvasFrame, text ="Informacion de agente",width=25, command = deleteClient).grid(row=2, column=0, sticky="nsew")
305
306                 #getHostInfo()
307
308                 Label(canvasFrame, text="Dispositivos monitoreados", width=25, fg="black").grid(row=0, column=1, sticky="nsew")
309
310                 title = ['Nombre del agente', 'Status', 'No. de interfaces', 'Nombre interfaz', 'Status interfaz']
311                 col = 0
312                 row = 3
313                 for c in title:
314                     Label(canvasFrame, text=c, width=20, fg="black").grid(row=row, column=col, sticky="nsew")
315                     col = col + 1
316
317                 file = open("hosts.txt", "r")
318                 agentCount = 0
319                 for linea in file.readlines():
320                     palabras = linea.split(" ")
321                     agentCount = agentCount + 1 #Aqui esta mi contador
322                     if palabras[3].endswith('\n'):
323                         palabras[3] = palabras[3][:-1]
324                         ip_community.append({'ip' : str(palabras[0]), 'port' : str(palabras[2]), 'community' : str(palabras[3])})
325                     else:
326                         ip_community.append({'ip' : str(palabras[0]), 'port' : str(palabras[2]), 'community' : str(palabras[3])})
327
328                 getAgentInfo(ip_community)
329                 file.close()
330                 photoScroll = Scrollbar(photoFrame, orient=VERTICAL)
331                 photoScroll.config(command=photoCanvas.yview)
332                 photoCanvas.config(scrollcommand=photoScroll.set)
333                 photoScroll.grid(row=0, column=1, sticky="ns")
334
335                 hscroll = Scrollbar(photoFrame, orient=HORIZONTAL, command=photoCanvas.xview)
336                 photoCanvas.config(xscrollcommand=hscroll.set)
337                 hscroll.grid(row=1, column=5, sticky="ew")
338
339                 canvasFrame.bind("<Configure>", update_scrollregion)
340                 top.after(30000, getHostInfo)
341             except Exception as error:
342                 #pass
343                 print error

```

Figura 5.2: Método getHostInfo.

Dentro del cuál se observan las siguientes líneas de la figura 5.3 en las que se abre nuestro archivo de hosts.txt en el cual se almacena la información de cada agente. Se consulta dicho archivo, se obtiene la información línea por línea de cada agente y se almacenan esos datos en un arreglo que será consultado posteriormente.

```

file = open("hosts.txt", "r")
agentCount = 0
for linea in file.readlines():
    palabras = linea.split(" ")
    agentCount = agentCount + 1 #Aqui esta mi contador
    if palabras[3].endswith('\n'):
        palabras[3] = palabras[3][:-1]
        ip_community.append({'ip' : str(palabras[0]), 'port' : str(palabras[2]), 'community' : str(palabras[3])})
    else:
        ip_community.append({'ip' : str(palabras[0]), 'port' : str(palabras[2]), 'community' : str(palabras[3])})

getAgentInfo(ip_community)
file.close()
photoScroll = Scrollbar(photoFrame, orient=VERTICAL)
photoScroll.config(command=photoCanvas.yview)
photoCanvas.config(yscrollcommand=photoScroll.set)
photoScroll.grid(row=0, column=1, sticky="ns")

hsbar = Scrollbar(photoFrame, orient=HORIZONTAL, command=photoCanvas.xview)
photoCanvas.config(xscrollcommand=hsbar.set)
hsbar.grid(row=1, column=5, sticky="ew")

canvasFrame.bind("<Configure>", update_scrollregion)
top.after(30000,getHostInfo)

```

Figura 5.3: Método getHostInfo, líneas importantes.

Como se puede observar en la figura anterior, en unas líneas más abajo se manda a llamar a nuestro siguiente método a mostrar, el método getAgentInfo mostrado a continuación en la figura 5.4.

```

433     def getAgentInfo(ip_community):
434         print 'getAgentInfo'
435         status_array = []
436         interface_name_status= []
437         global agentCount
438         r = 6
439         ro = 6
440         for computer in ip_community:
441             status_received = ping(computer['ip']) #ip_for['ip']
442             status_array.append(status_received)
443             if status_received == 'Activa':
444                 print('* * * * * \nEl status es: ' + status_received)
445                 agents = consultaSNMP(computer['community'], computer['port'], computer['ip'], '1.3.6.1.2.1.1.1.0')
446                 print agents
447                 if not agents:
448                     continue
449
450                 interfaces = consultaSNMP(computer['community'], computer['port'], computer['ip'], '1.3.6.1.2.1.2.1.0')
451                 print interfaces
452                 for i in range(1,int(interfaces)+1):
453                     name_interfaces = consultaSNMP(computer['community'], computer['port'], computer['ip'], '1.3.6.1.2.1.2.1.2.'+str(i))
454                     status_inter = consultaSNMP(computer['community'], computer['port'], computer['ip'], '1.3.6.1.2.1.2.1.8.'+str(i))
455
456                     if name_interfaces[1] == '0':
457                         interface_name = name_interfaces[3:3].decode('hex')
458                         Label(canvasFrame, text=interface_name, width=10, fg="black").grid(row=ro, column=3, sticky="nsew")
459                     else:
460                         Label(canvasFrame, text=name_interfaces, width=10, fg="black").grid(row=ro, column=3, sticky="nsew")
461
462                     if int(status_inter) == 1:
463                         Label(canvasFrame, text='Activo', width=20, fg="black").grid(row=ro, column=4, sticky="nsew")
464                     elif int(status_inter) == 2:
465                         Label(canvasFrame, text='Inactivo', width=20, fg="black").grid(row=ro, column=4, sticky="nsew")
466                     elif int(status_inter) == 3:
467                         Label(canvasFrame, text='Testing', width=20, fg="black").grid(row=ro, column=4, sticky="nsew")
468                     ro = ro + 1
469
470                 agentes.nombre = Label(canvasFrame, text=agents, width=70, fg="black").grid(row=r, column=0, sticky="nsew")
471                 Label(canvasFrame, text=status_received, width=10, fg="black").grid(row=r, column=1, sticky="nsew")
472                 Label(canvasFrame, text=interfaces, width=10, fg="black").grid(row=r, column=2, sticky="nsew")
473                 Tinker.Button(canvasFrame, text = "Graficas",width=10, command= lambda name = [computer['community'], computer['port'], computer['ip']], '1.3.6.1.2.1.1.0': MostrarEstado(name)).grid(row=r, column=6, sticky="nsew")
474                 Tinker.Button(canvasFrame, text = "Estado",width=10, command = lambda name = computer['ip']: MostrarEstado(name)).grid(row=r, column=6, sticky="nsew")
475                 Tinker.Button(canvasFrame, text = "Eliminar",width=10, command = lambda name = computer['ip'] : eliminarAgente(name)).grid(row=r, column=7, sticky="nsew")
476                 r = r + 1
477             else:
478                 Label(canvasFrame, text = computer['ip'], width=70, fg="red").grid(row=r, column=0, sticky="nsew")
479                 Label(canvasFrame, text=status_received, width=10, fg="red").grid(row=r, column=1, sticky="nsew")
480                 Label(canvasFrame, text= "Informacion no disponible", width=50, fg="red").grid(row=r, column=3, sticky="nsew")
481                 Tinker.Button(canvasFrame, text = "Eliminar",width=10, command = lambda name = computer['ip'] : eliminarAgente(name)).grid(row=r, column=7, sticky="nsew")
482                 r = r + 1
483                 ro = ro + 1
484

```

Figura 5.4: Método getAgentInfo.

Al cual, nuevamente mostraremos solo las líneas principales en la siguiente figura 5.5 en donde se realiza una actividad muy importante, se envían los datos de cada uno de los agentes con un OID en específico para obtener su información y posteriormente plasmarla en sus respectivos labels.

```

for computer in ip_community:
    status_received = ping(computer['ip']) #ip_for['ip']
    status_array.append(status_received)
    if status_received == 'Activa':
        print('\n*****\nEl status es: ' + status_received)
        agents = consultaSNMP(computer['community'], computer['port'], computer['ip'], '1.3.6.1.2.1.1.1.0')
        print agents
        if not agents:
            continue

    interfaces = consultaSNMP(computer['community'], computer['port'], computer['ip'], '1.3.6.1.2.1.2.1.0')
    print interfaces
    for i in range(1,int(interfaces)+1):
        name_interfaces = consultaSNMP(computer['community'], computer['port'], computer['ip'], '1.3.6.1.2.1.2.1.2.'+str(i))
        status_inter = consultaSNMP(computer['community'], computer['port'], computer['ip'], '1.3.6.1.2.1.2.1.8.'+str(i))

        if name_interfaces[1] == '0':
            interface_name = name_interfaces[3:].decode('hex')
            Label(canvasFrame, text=interface_name, width=100, fg='black').grid(row=ro, column=3, sticky="nsew")
        else:
            Label(canvasFrame, text=name_interfaces, width=100, fg='black').grid(row=ro, column=3, sticky="nsew")

        if int(status_inter) == 1:
            Label(canvasFrame, text='Activo', width=20, fg='black').grid(row=ro, column=4, sticky="nsew")
        elif int(status_inter) == 2:
            Label(canvasFrame, text='Inactivo', width=20, fg='black').grid(row=ro, column=4, sticky="nsew")
        elif int(status_inter) == 3:
            Label(canvasFrame, text='Testing', width=20, fg='black').grid(row=ro, column=4, sticky="nsew")
        ro = ro + 1

```

Figura 5.5: Método getAgentInfo, líneas importantes.

Por último, es importante mostrar 2 métodos igualmente importantes, uno es el método ping mostrado en la figura 5.6 con el cual se verifica si la ip está activa o no y en caso de no estarlo, no buscar su información en la MIB.

```

def ping(ip):
    # response = os.system("ping -c 1 -q" + ip)
    with open(os.devnull, 'w') as DEVNULL:
        try:
            subprocess.check_call(
                ['ping', '-c', '1', ip],
                stdout=DEVNULL, # suppress output
                stderr=DEVNULL
            )
            is_up = 'Activa'
            alive_hosts.append(ip)
        except subprocess.CalledProcessError:
            is_up = 'Inactiva'
    print is_up
    return is_up

```

Figura 5.6: Método ping.

Por otro lado, el otro método a mostrar es el de la figura 5.7 mediante el cual, se ejecuta la instrucción snmpget y se obtiene la información solicitada.

```
def consultaSNMP(comunidad, port, host, oid):
    global resultado_final
    # print comunidad,port,host,oid
    try:
        errorIndication, errorStatus, errorIndex, varBinds = next(
            getCmd(SnmpEngine(),
                   CommunityData(comunidad),
                   UdpTransportTarget((host, int(port)), timeout=0.25, retries=0),
                   ContextData(),
                   ObjectType(ObjectIdentity(oid))))
    if errorIndication:
        print(errorIndication), comunidad, host, oid
        return None
    elif errorStatus:
        print('%s at %s' % (errorStatus.prettyPrint(), errorIndex and varBinds[int(errorIndex) - 1][0] or '?'))
        return None
    else:
        for varBind in varBinds:
            varB = (' = '.join([x.prettyPrint() for x in varBind]))
            resultado = varB.split()
            concat = []
            valid = False

            for palabra in resultado:
                if palabra == '=':
                    valid = True
                    continue

                if valid:
                    concat.append(palabra)
            resultado_final = ''
            for palabra in concat:
                if palabra == '=' or palabra == 'SMP':
                    resultado_final = resultado_final + ' ' + palabra + '\n'
                else:
                    resultado_final = resultado_final + ' ' + palabra
    return resultado_final
except Exception as error:
    print error
```

Figura 5.7: Método consultaSNMP.

5.2. Agregar agente

Por otro lado, para agregar un nuevo agente, es necesario presionar sobre el botón en la parte superior [5.8](#), que es quien nos manda al método mostrado en la figura [5.9](#), el cual se encarga de abrir una pequeña ventana como la de la figura [5.10](#), y en este, se manda a llamar a otros 2 métodos mostrados a continuación.

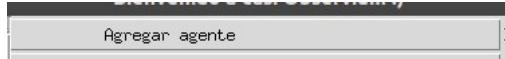


Figura 5.8: Botón agregar agente.

```
def addClient(): #Abre un recuadro a partir del recuadro principal y muestra
    #llamar a la funcion fetch
    root = Tkinter.Toplevel(canvasFrame)
    root.title("Agregar un nuevo agente")
    ents = makeform(root, fields)
    root.bind('<Return>', (lambda event, e=ents: fetch(e)))
    bl = Button(root, text='Agregar agente', command=(lambda e=ents: fetch(e)))
    bl.pack(side=LEFT, padx=5, pady=5)
```

Figura 5.9: Método addClient.

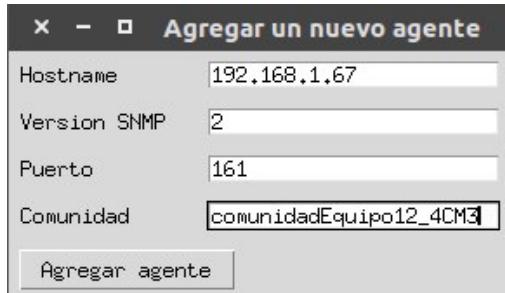


Figura 5.10: Pantalla desplegada para añadir un cliente nuevo.

Por un lado se llama al método mostrado en la figura [5.11](#), el cual únicamente se encarga de posicionar adecuadamente los labels dentro de la ventana.

```
def makeform(root, fields): #Acomoda label en ventana de
    entries = []
    for field in fields:
        row = Frame(root)
        lab = Label(row, width=15, text=field, anchor='w')
        ent = Entry(row)
        row.pack(side=TOP, fill=X, padx=5, pady=5)
        lab.pack(side=LEFT)
        ent.pack(side=RIGHT, expand=YES, fill=X)
        entries.append((field, ent))
    return entries
```

Figura 5.11: Método makeform.

Y por otro lado se llama al método fetch mostrado en la figura [5.12](#), el cual obtiene todos los valores que el usuario ha agregado en los labels para registrar un agente nuevo y los agrega en una nueva línea en el

archivo hosts.txt, en caso de que el archivo no exista, se crea uno nuevo. Y de igual manera, nos muestra una pequeña alerta como la de la figura 5.13 para dar a conocer que nuestro agente ha sido creado correctamente.

```
def fetch(entries): #Recorre todos los datos que agregue y me los imprime en
    showinfo('Agente creado!', 'Se ha creado correctamente un agente nuevo')
    concatenation = ''
    for entry in entries:
        field = entry[0]
        text = entry[1].get()
        concatenation = concatenation + text + '\n'
        print('%s: %s' % (field, text))
    file = open("hosts.txt","a+")
    file.write(concatenation + '\n')
    file.close()
```

Figura 5.12: Pantalla desplegada para añadir un cliente nuevo.

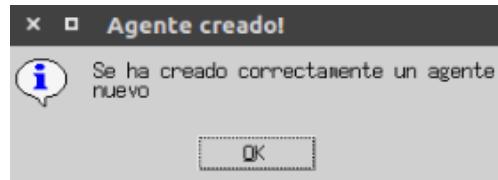


Figura 5.13: Alerta desplegada al crearse un nuevo agente.

Y se puede observar tanto en la pantalla principal(figura 5.14) como en nuestro archivo de hosts (figura 5.15) que nuestro nuevo agente ha sido agregado correctamente.

Bienvenido a casi Observium :)		Dispositivos monitoreados			Status interfaz		
		Nombre del agente	Status	No. de interfaces	Nombre interfaz	Activo	
Linux march->S510UR 4.16.0-32-generic #37~16.04.1-Ubuntu SMP	Fri Aug 10 21:54:34 UTC 2018 x86_64		Activa	3	Intel Corporation Device 24fd	Activo	
Hardware: Intel(R) Family 6 Model 142 Stepping 10 AT/AT COMPATIBLE -	Software: Windows Version 6.3 (Build 17134 Multiprocessor Free)		Activa	23	vboxnet0	Inactivo	
Linux march-Lenovo-150-70 4.15.0-34-generic #37~16.04.1-Ubuntu SMP	Tue Aug 28 10:44:06 UTC 2018 x86_64		Activa	3	Software Loopback Interface 1	Activo	
					Microsoft Endpoint Adapter	Inactivo	
					WAN Miniport (IKEv2)	Activo	
					WAN Miniport (Network Monitor)	Inactivo	
					WAN Miniport (PPTP)	Activo	
					WAN Miniport (SST)	Inactivo	
					Microsoft Kernel Tunnelling Adapter	Activo	
					Microsoft Teredo Tunnelling Adapter	Inactivo	
					WAN Miniport (L2TP)	Activo	
					WAN Miniport (PPPOE)	Inactivo	
					Intel(R) PRO/1000 MT Desktop Adapter	Activo	
					WAN Miniport (IP)	Activo	
					Intel(R) PRO/1000 MT Desktop Adapter-WFP Native MAC Layer LightWeight Filter-0000	Activo	
					Intel(R) PRO/1000 MT Desktop Adapter-QoS Packet Scheduler-0000	Activo	
					Intel(R) PRO/1000 MT Desktop Adapter-QoS MAC Layer LightWeight Filter-0000	Activo	
					WAN Miniport (IP)-WFP Native MAC Layer LightWeight Filter-0000	Activo	
					WAN Miniport (IP)-QoS Packet Scheduler-0000	Activo	
					WAN Miniport (IPv6)-WFP Native MAC Layer LightWeight Filter-0000	Activo	
					WAN Miniport (IPv6)-QoS Packet Scheduler-0000	Activo	
					WAN Miniport (Network Monitor)-WFP Native MAC Layer LightWeight Filter-0000	Activo	
					WAN Miniport (Network Monitor)-QoS Packet Scheduler-0000	Activo	
					Realtek Semiconductor Co., Ltd. RTL8111/8168/8411 PCI Express Gigabit Ethernet Controller	Activo	
					Realtek Semiconductor Co., Ltd. RTL8723BE PCIe Wireless Network Adapter	Inactivo	

Figura 5.14: Pantalla de inicio que muestra nuevo agente agregado.

```
1 192.168.1.68 2 161 comunidadMarcela
2 192.168.1.64 2 161 comunidad3
3 192.168.1.67 2 161 comunidadEquipo12_4CM3
```

Figura 5.15: Agente agregado en archivo hosts.

Es importante recalcar que en la última línea del método getHostInfo mostrado en la imagen 5.3, se

realizar un autollamado al mismo método cada 3 segundos, lo cual permite que al añadir un nuevo agente, solo se debe esperar 30 segundos para que este aparezca en la pantalla de inicio.

5.3. Eliminar agente

Así como se pueden agregar agentes nuevos, también pueden ser eliminados, es por esto que para eliminar un agente, se implementó una funcionalidad muy sencilla. Como se observa en la figura 5.16, tenemos nuestra pantalla principal y a lado de cada agente se observa 3 botones que corresponden a la visualización de las gráficas, de la información de cada agente y a la eliminación del mismo. La figura 5.17 muestra más de cerca los botones mencionados anteriormente.

Agregar agente		Dispositivos monitoreados											
Eliminar agente		Número de agentes: 3			Nombre interfaz			Status interfaz			Graficas Estado Eliminar		
		Status	No. de interfaces		lo	Intel Corporation Device 24Fd	Activo	Graficas	Estado	Eliminar			
Nombre del agente		Activo	3		vboxnet0	Software Loopback Interface 1	Activo	Graficas	Estado	Eliminar			
Nombre del agente		Activo	23			Microsoft Loopback Adapter	Inactivo	Graficas	Estado	Eliminar			
Nombre del agente		Activo	3			WAN Miniport (IPV6)	Activo	Graficas	Estado	Eliminar			
Nombre del agente		Activo	3			Microsoft Native MAC Layer Adapter	Activo	Graficas	Estado	Eliminar			
Nombre del agente		Activo	3			WAN Miniport (Ether2)	Inactivo	Graficas	Estado	Eliminar			
Nombre del agente		Activo	3			WAN Monitor (Network Monitor)	Activo	Graficas	Estado	Eliminar			
Nombre del agente		Activo	3			WAN Miniport (PPPoE)	Inactivo	Graficas	Estado	Eliminar			
Nombre del agente		Activo	3			WAN Miniport (SSTP)	Inactivo	Graficas	Estado	Eliminar			
Nombre del agente		Activo	3			Microsoft Teredo Tunneling Adapter	Inactivo	Graficas	Estado	Eliminar			
Nombre del agente		Activo	3			Microsoft Teredo Tunneling Adapter	Activo	Graficas	Estado	Eliminar			
Nombre del agente		Activo	3			WAN Miniport (L2TP)	Inactivo	Graficas	Estado	Eliminar			
Nombre del agente		Activo	3			WAN Miniport (PPPOE)	Activo	Graficas	Estado	Eliminar			
Nombre del agente		Activo	3			Intel(R) PRO/1000 MT Desktop Adapter	Activo	Graficas	Estado	Eliminar			
Nombre del agente		Activo	3			Intel(R) PRO/1000 MT Desktop Adapter-WFP Native MAC Layer LightWeight Filter-0000	Activo	Graficas	Estado	Eliminar			
Nombre del agente		Activo	3			Intel(R) PRO/1000 MT Desktop Adapter-WFP Native MAC Layer LightWeight Filter-0000	Activo	Graficas	Estado	Eliminar			
Nombre del agente		Activo	3			Intel(R) PRO/1000 MT Desktop Adapter-WFP 802.3 MAC Layer LightWeight Filter-0000	Activo	Graficas	Estado	Eliminar			
Nombre del agente		Activo	3			WAN Miniport (IP)-WFP Native MAC Layer LightWeight Filter-0000	Activo	Graficas	Estado	Eliminar			
Nombre del agente		Activo	3			WAN Miniport (IPv6)-WFP Native MAC Layer LightWeight Filter-0000	Activo	Graficas	Estado	Eliminar			
Nombre del agente		Activo	3			WAN Miniport (Network Monitor)-WFP Native MAC Layer LightWeight Filter-0000	Activo	Graficas	Estado	Eliminar			
Nombre del agente		Activo	3			WAN Miniport (Network Monitor)-QoS Packet Scheduler-0000	Activo	Graficas	Estado	Eliminar			
Nombre del agente		Activo	3			Realtek Semiconductor Co., Ltd. RTL8111/8168/8411 PCI Express Gigabit Ethernet Controller	Activo	Graficas	Estado	Eliminar			
Nombre del agente		Activo	3			Realtek Semiconductor Co., Ltd. RTL8723BE PCIe Wireless Network Adapter	Inactivo	Graficas	Estado	Eliminar			

Figura 5.16: Pantalla de inicio que muestra botón para eliminar un agente.

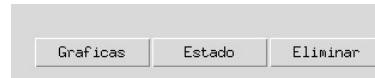


Figura 5.17: Botones de pantalla de principal.

El código para la funcionalidad de esta parte es el mostrado en la figura 5.18, en el cual al presionar dicho botón, se habla al método `eliminarAgente` enviando como parámetro la ip del agente a eliminar, lo que realiza este método es, nuevamente recorrer nuestro archivo hosts e ir guardando en un arreglo todo lo que lee de dicho archivo, en caso de que encuentre la ip que recibió como parámetro, no guarda esa línea de información y posteriormente sobreescribe el archivo con toda la información del arreglo. En otras palabras, sobreescribe el archivo sin la línea del agente que se eliminó.

```
def eliminarAgente(ip):
    global agentes_nombre
    global canvasFrame
    print 'ip a eliminar',ip
    ip_Guardar = ''
    try:
        file = open("hosts.txt", "r")
        for linea in file.readlines():
            #print 'Entre for'
            palabras = linea.split(" ")
            #print palabras[0]
            if ip != palabras[0]:
                ip_Guardar = ip_Guardar + linea
            else:
                pass
        file.close()
        file = open("hosts.txt", "w")
        file.write(ip_Guardar)
        file.close()
        #photoCanvas.delete(agentes_nombre)
        #photoCanvas.update()
        #agentes.pack_forget()
        canvasFrame.destroy()
        canvasFrame = None
        main()
        showinfo('Agente eliminado!', 'Se ha eliminado correctamente el agente')
    except Exception as error:
        print error
```

Figura 5.18: Método eliminarAgente.

Después de que se ha ejecutado este método, aparece una pequeña alerta como la de la figura 5.19 donde se muestra que nuestro agente ha sido eliminado y se llama al método main (figura 5.20) el cual como se observa, llama al método getHostInfo y se repite todo el procedimiento explicado en la sección "Pantalla de inicio".

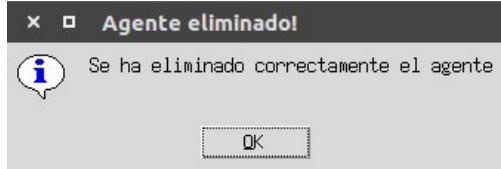


Figura 5.19: Alerta de agente eliminado.

```
def main():
    print 'main'
    top.after(0, getHostInfo)
    top.mainloop()
```

Figura 5.20: Método main.

Nuevamente se recarga la pantalla principal para observar que la información de nuestro agente ya no se plasma en esta (figura 5.21) ni tampoco en nuestro archivo de hosts (figura 5.22).

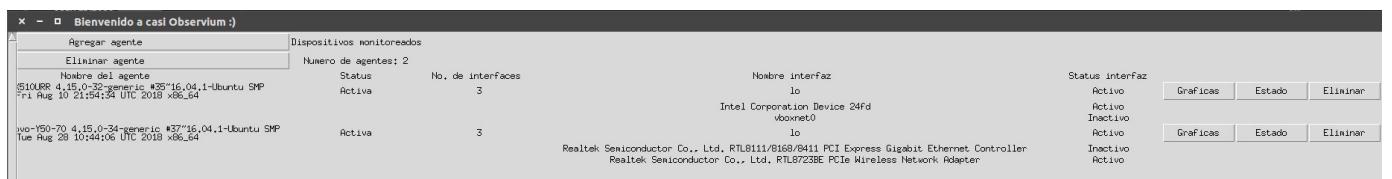


Figura 5.21: Pantalla principal sin el agente eliminado.

```

Untitled-1 • server.py hosts.txt x
1 192.168.1.68 2 161 comunidadMarcela
2 192.168.1.67 2 161 comunidadEquipo12_4CM3
3

```

Figura 5.22: Archivo hosts sin el agente eliminado.

5.4. Estado del dispositivo

5.5. Gráficas de dispositivos

La etapa de graficación se comprende de dos pasos: el primero consiste en la captura de datos y actualización de la base de datos. El segundo paso consiste en la construcción de las gráficas en el momento en el que el usuario seleccione una de las gráficas a desplegar. Ambos pasos se llevan a cabo con apoyo de la biblioteca rrdtool de Python.

El paso de adquisición de datos y actualización de la base de datos se inicia a la par del inicio de ejecución del programa mediante la función `inicia_captura()` mostrada en la figura 5.23, perteneciente al archivo `server.py`. Esta función lee los datos de cada host del archivo `hosts.txt`, verifica por cada host si se encuentra activo y en ese caso inicia un hilo que ejecuta la función `actualizar()` del archivo `actualizarRRD.py` mostrada en la figura 5.24.

```

491 def inicia_capturas():
492     print "Iniciando..."
493     hilos = []
494     with open("hosts.txt", 'r') as hosts:
495         for i in hosts.readlines():
496             datos = i.split(' ')
497             for j in alive_hosts:
498                 if j == datos[0] and datos[0] != "127.0.0.1": # Si esta vivo inicia un hilo con sus actualizaciones
499                     thread = threading.Thread(target=actualizar, args=(
500                         'Actualizando ' + datos[0] + ' ' + datos[3][:-1], datos[3][:-1], datos[0], datos[2],
501                         datos[0] + '-net',))
502                     thread.daemon = True
503                     hilos.append(thread)
504                     break
505     for h in hilos:
506         h.start()

```

Figura 5.23: Inicio de captura de datos.

La función `actualizar()` verifica que existe una base de datos RRD asociada al host, en caso de no existir la crea. Despues, inicia por cada host activo la adquisición de datos. Estos datos son solicitados mediante la función `consultaSNMP()` del archivo `getSNMP.py` recibiendo como parámetro el OID correspondiente. Esto se muestra en la figura 5.26. Las consultas SNMP se llevan a cabo cada segundo.

```

14 def actualizar(cadena,comunidad,host,puerto,rrd):
15     total_input_traffic = 0
16     total_output_traffic = 0
17
18     #Verifica que exista una rrd asociada al host, en caso de no existir crea una rrd nueva
19     archivo_rrd = Path(rrd+".rrd")
20     if archivo_rrd.is_file() == False:
21         crearRRD.crear(rrd+".rrd")
22         print "rrd creada"
23     else:
24         print "Abriendo rrd..."
25
26     #Inicia proceso de adquisicion de datos
27     while 1:
28         print cadena
29         #octetos de entrada de la interfaz eth
30         total_input_traffic = int(
31             consultaSNMP(comunidad,host,puerto,
32                         '1.3.6.1.2.1.2.2.1.10.2'))
33         #octetos de salida de la interfaz eth
34         total_output_traffic = int(
35             consultaSNMP(comunidad,host,puerto,
36                         '1.3.6.1.2.1.2.2.1.16.2'))
37         #-----

```

Figura 5.24: Actualización de la base de datos RRD del host.

Una vez que se tiene el valor de la solicitud SNMP, los datos son concatenados de tal forma que la base de datos RRD los acepte. Esto se muestra en la figura 5.26. Cabe mencionar que existe una base RRD por cada host activo.

```

70     valor = "N:" + str(total_input_traffic) + ":" + str(total_output_traffic) + ":" + str(total_tcp_established) + ":" + str(input_tcp_segs) +
71
72     print valor
73     rrddtool.update(rrd+'.rrd', valor) # actualizamos el archivo previamente creado en rrd1.py
74     rrddtool.dump(rrd+'.rrd',rrd+'.xml') # ver el contenido de la bd opcional
75     time.sleep(1)
76
77     if ret:
78         print rrddtool.error()
79         time.sleep(300)

```

Figura 5.25: Actualización de la base de datos RRD del host.

```

21 def consultaSNMP(comunidad,host,puerto,oid):
22     """
23         Recibe el nombre de la comunidad, el host o ip y el OID a consultar
24     """
25     #resultado = ''
26     errorIndication, errorStatus, errorIndex, varBinds = next(
27         getCmd(SnmpEngine(),
28                 CommunityData(comunidad),
29                 UdpTransportTarget((host, puerto)),
30                 ContextData(),
31                 ObjectType(ObjectIdentity(oid))))
32

```

Figura 5.26: Consulta SNMP.

Una vez que los hilos de actualización de los hosts han sido iniciados el usuario interactúa con la interfaz gráfica de la herramienta. De esta forma, al hacer clic sobre el botón **Gráficas**, el programa ejecutará la función **graphics()** del archivo **server.py**, la cual se muestra en la figura 5.27.

```

86 def graphics(parametros): #Abre un recuadro a partir del recuadro principal y muestra su propio boton para
87     #llamar a la funcion fetch
88     #graphic_window = Tkinter.Toplevel(top)
89     #graphic_window.title("Graficos")
90     #print parametros
91     a = VentanaGraficas(top,parametros[2])
92     #print comunidad, puerto, ip, oid

```

Figura 5.27: Llamada a la pantalla de selección de gráficas.

Esta función instanciará a la clase **VentanaGraficas** del archivo **elegirGraficas.py**. Esta clase se muestra en la figura 5.28. Su constructor recibe como parámetros el host (IP/Hostname) y el objeto asociado a la pantalla padres, es decir, la pantalla principal. Posteriormente, despliega los botones relacionados al tipo de gráfica a mostrar. El usuario, al seleccionar un tipo de gráfica de dicho host provocará que se ejecute el método **inicia_ventana_grafica()**, el cual recibe como parámetro un identificador de gráfica.

```

9 class VentanaGraficas(object):
10     def __init__(self,root,host):
11         self.root = root
12         self.window = None
13
14         self.host = host
15         self.rrd_name = host+"-net.rrd"
16         #self.rrd_name = "net3.rrd" # Nombre de la rrd del host
17
18         #self.b = tk.Button(self.root, text="Graficas de la interfaz", command=self.create_window)
19         #self.b.pack()
20         self.create_window()
21
22     def create_window(self):
23         self.window = tk.Toplevel(self.root)
24         self.window.title("Graficos")
25         tk.Button(self.window, text='Trafico de la interfaz', command= lambda: self.inicia_ventana_grafica(1)).grid(row=1, column=0)
26         tk.Button(self.window, text='Conexiones TCP establecidas', command= lambda: self.inicia_ventana_grafica(2)).grid(row=2, column=0)
27         tk.Button(self.window, text='Segmentos TCP', command= lambda: self.inicia_ventana_grafica(3)).grid(row=3, column=0)
28         tk.Button(self.window, text='Estadísticas ICMP', command= lambda: self.inicia_ventana_grafica(4)).grid(row=4, column=0)
29         tk.Button(self.window, text='Respuestas SNMP', command= lambda: self.inicia_ventana_grafica(5)).grid(row=5, column=0)
30
31     def inicia_ventana_grafica(self,id_grafica):
32         a = Grafica(self.root,self.rrd_name,self.host+"-"+str(id_grafica)+".png",id_grafica)
33

```

Figura 5.28: Clase para la selección de gráficas.

El método **inicia_ventana_grafica()** instanciará a la clase **Grafica** del mismo archivo, como se observa en la figura 5.29. La función de esta clase es iniciar un hilo en su constructor encargado de iniciar el proceso de graficación mediante el método **graficar()** del archivo **graficarRRD.py**, mostrado en la figura 5.30. Una vez iniciado el proceso de graficación desplegará la ventana que refrescará cada segundo la imagen a la gráfica del host correspondiente.

```

34 class Grafica(object):
35     def __init__(self, root, rrd_name, imagen, id_grafica):
36
37         self.root = root
38         self.rrd_name = rrd_name
39         self.imagen = imagen
40         self.id_grafica = id_grafica
41
42         self.window1 = None
43         self.img = None
44         self.display = None
45
46         #Debe ser un hilo por cada grafica iniciada
47         t2 = threading.Thread(target=graficar, args=( "Graficando...",self.rrd_name,self.imagen,self.id_grafica,))
48         t2.daemon = True
49         t2.start()
50
51         self.window1 = tk.Toplevel(self.root)
52
53         if id_grafica == 1:
54             self.window1.title("Trafico de la interfaz")
55         elif id_grafica == 2:
56             self.window1.title("Conexiones TCP establecidas")
57         elif id_grafica == 3:
58             self.window1.title("Segmentos TCP")
59         elif id_grafica == 4:
60             self.window1.title("Estadisticas ICMP")
61         elif id_grafica == 5:
62             self.window1.title("Respuestas SNMP")
63         else:
64             self.window1.title("Graficos")
65
66
67         self.img = ImageTk.PhotoImage(Image.open(self.imagen))
68         self.display = tk.Label(self.window1, image=self.img)
69         self.display.pack(side = "bottom", fill = "both", expand = "yes")
70         self.actualiza_imagen()
71
72     def actualiza_imagen(self):
73         self.img = ImageTk.PhotoImage(Image.open(self.imagen))
74         self.display.config(image=self.img)
75         self.window1.after(1000, self.actualiza_imagen)
76         print "Actualizacion img..."
77

```

Figura 5.29: Clase para la graficación.

Finalmente, observamos en la figura 5.30 que la función `graficar()` recibe como parámetro el identificador de gráfica. Esto le sirve al hilo para saber qué gráfica actualizar. La actualización se lleva a cabo con rrdtool cada segundo.

```

22 def graficar(cadena,rrd,image_name,id_grafica):
23
24     if id_grafica == 1:
25         while 1:
26             print cadena
27             ret = rrdtool.graph( image_name,
28                                 "--start",tiempo_graficacion,
29                                 "--end","N",
30                                 "--vertical-label=Bytes/s",
31                                 "DEF:inoctets="+rrd+":inoctets:AVERAGE",
32                                 "DEF:outoctets="+rrd+":outoctets:AVERAGE",
33                                 "AREA:inoctets#00FF00:In traffic",
34                                 "LINE1:outoctets#0000FF:Out traffic\r")
35
36             time.sleep(5)
37     elif id_grafica == 2:
38         while 1:
39             print cadena
40             ret = rrdtool.graph( image_name,
41                                 "--start",tiempo_graficacion,
42                                 "--end","N",
43                                 "--vertical-label=Numero de conexiones",
44                                 "DEF:establishedtcpconn="+rrd+":establishedtcpconn:AVERAGE",
45                                 "LINE1:establishedtcpconn#0000FF:Conexiones TCP establecidas\r")
46
47             time.sleep(5)
48     elif id_grafica == 3:
49         while 1:
50             print cadena
51             ret = rrdtool.graph( image_name,
52                                 "--start",tiempo_graficacion,
53                                 "--end","N",
54                                 "--vertical-label=TCP Segs/s",
55                                 "DEF:intcpsegs="+rrd+":intcpsegs:AVERAGE",
56                                 "DEF:outtcpsegs="+rrd+":outtcpsegs:AVERAGE",
57                                 "AREA:intcpsegs#00FF00:In TCP traffic",
58                                 "LINE1:outtcpsegs#0000FF:Out TCP traffic\r")
59
60

```

Figura 5.30: Graficación con rrdtool.

CAPÍTULO 6

Conclusiones

6.0.1. Castro Flores Marcela

Creo que el desarrollo de esta práctica fue un poco difícil para mi en primera instancia porque tuve que familiarizarme rápidamente con el lenguaje de programación Python mismo que si bien no es muy difícil, presenta ciertos cambios al momento de manejar ciertas funciones. Por otro lado, nunca antes había utilizado el entorno gráfico de Python, es por ello que me tomó más tiempo del normal realizar ciertas partes de las secciones que me tocaron. Creo que la utilización de sistemas más complejos, pero semejantes al que se realizó como práctica resultan ser bastante útiles para reportar y mejorar las áreas en las que los dispositivos que crean diferentes topologías de red llegan a tener pequeños problemas que alentan las peticiones que los clientes realizan.

6.0.2. Sánchez Cruz Rosa María

La implemetación de la practica fué útil para determinar cómo funciona un agente, un gestor y las diferencias entre ambos. Un agente lo podemos determinar como aquel equipo dentro de una red que se tiene la necesidad de saber algunas características, por ejemplo conexión, rendimiento, Información. El sistema gestor nos ayuda a recibir e interpretar los datos que recibimos del agente, este sistema gestor lo podemos asociar como un administrador el cual puede adquirir la información de los equipos agentes, con el objetivo principal de tener un control de los servicios o como prevención de incidentes. La forma que captura la información o el medio por el cual se recibe la información es a través del protocolo SNMP que por medio de OIDs que son valores pre establecidos y estructurados es posible obtener una respuesta con la información que se ha pedido. Es necesario añadir la dirección ip y una comunidad con la cual el agente esta asociado. Con respecto a la práctica se desarrolló un sistema gestor el cual puede administrar equipos con las funcionalidades de agregar y eliminar agentes con la característica de que cada agente tiene asociado cinco capturas SNMP que se realizan de manera concurrente y los resultados son graficados y presentados por medio de la librería RRDTool. Es importante la aplicación desarrollada por el hecho de aprender a diferenciar agentes y gestores. La característica más importante a considerar es el medio de comunicación y saber interpretar la información que podemos obtener con SNMP ya que es la base del desarrollo de prácticas complejas y un sin número de aplicaciones.

6.0.3. Santiago Mancera Arturo Samuel

La realización de la presente práctica fue de gran utilidad para comprender el funcionamiento de algunas herramientas comerciales como Nagios o Cacti. Realizar desde cero un programa con funciones similares ayuda a comprender la complejidad de su estructura. La parte más importante, personalmente, fue la de la adquisición de información mediante SNMP y su posterior procesamiento para la construcción de gráficas. Es importante mencionar que se comprendió la utilidad del uso de bases de datos de tipo *Round Robin* en la persistencia de datos correspondientes al monitoreo. Por otro lado, el uso de Python facilita, en gran medida, el desarrollo de la herramienta en los aspectos de funcionalidad e interfaz con el usuario.

Referencias y bibliografías

- [1] CARLOS VIALFA, *Protocolo SNMP*. (2017). Disponible en: <https://es.ccm.net/contents/280-protocolo-snmp> [Consultado el 26 Sept. 2018].
- [2] DAVID GUERRERO, *SNMP: Administración y Mantenimiento de Redes con Linux*. (1998). Disponible en: <http://redesdecomputadores.umh.es/aplicacion/snmp.htm> [Consultado el 26 Sept. 2018].