



**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 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	29
4.1. Análisis de tráfico con Wireshark	29
4.1.1. snmpget	29
4.1.2. snmpgetnext	31
4.1.3. snmpwalk	32
4.1.4. snmpset	33
5. Implementación de un modelo de administración de red de SNMP	36
5.1. Pantalla de inicio	36
5.2. Agregar agente	39
5.3. Eliminar agente	41
5.4. Estado del dispositivo	41
5.5. Gráficas de dispositivos	41

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	24
3.17. Octetos en Windows	25
3.18. Dirección Mac de Enp0s3	25
3.19. Tabla de direcciones Mac en Windows	26
3.20. Octetos por interfaz.	26
3.21. Direcciones IP de cada interfaz.	26
3.22. Octetos por interfaz.	26
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.	27
3.27. Entradas en ipRouteTable en Windows.	27
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
4.1. Comando snmpget.	29
4.2. Captura snmpget.	30
4.3. Solicitud snmpget.	30
4.4. Respuesta snmpget.	31
4.5. Comando snmpgetnext.	31
4.6. Captura snmpgetnext.	31
4.7. Solicitud snmpgetnext.	32
4.8. Respuesta snmpgetnext.	32
4.9. Comando snmpwalk.	33
4.10. Capturas snmpwalk.	33
4.11. Comando snmpset.	34
4.12. Captura snmpset.	34
4.13. Solicitud snmpset.	34
4.14. Respuesta snmpset.	35
5.1. Método getHostInfo.	37
5.2. Método getHostInfo, líneas importantes.	37
5.3. Método getAgentInfo.	38
5.4. Método getAgentInfo, líneas importantes.	38
5.5. Método ping.	39
5.6. Método consultaSNMP.	39
5.7. Método addClient.	40
5.8. Pantalla desplegada para añadir un cliente nuevo.	40
5.9. Método makeform.	40
5.10. Pantalla desplegada para añadir un cliente nuevo.	41
5.11. Alerta desplegada al crearse un nuevo agente.	41

Índice de cuadros

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.

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.

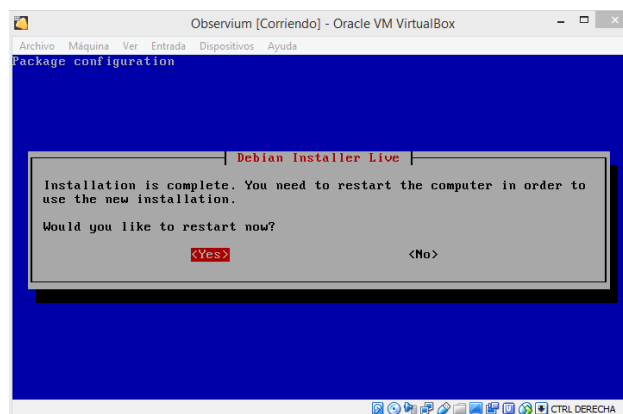


Figura 2.1: Finalización de la instalación del S.O .

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.

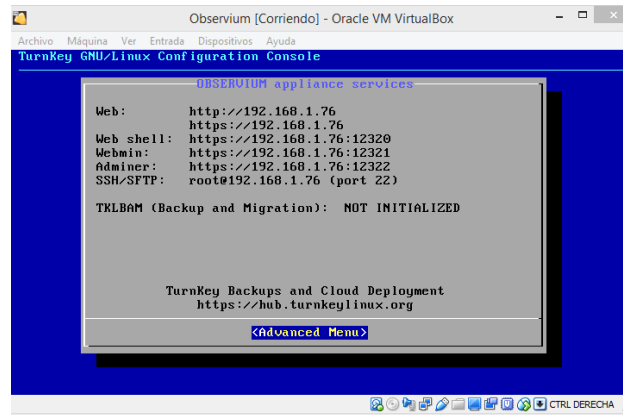


Figura 2.2: Asignación de dirección IP.

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).

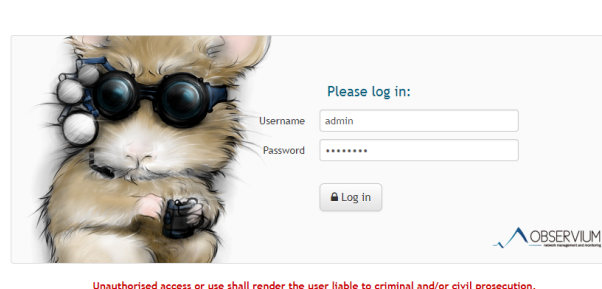


Figura 2.3: Pág. de Observium

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.

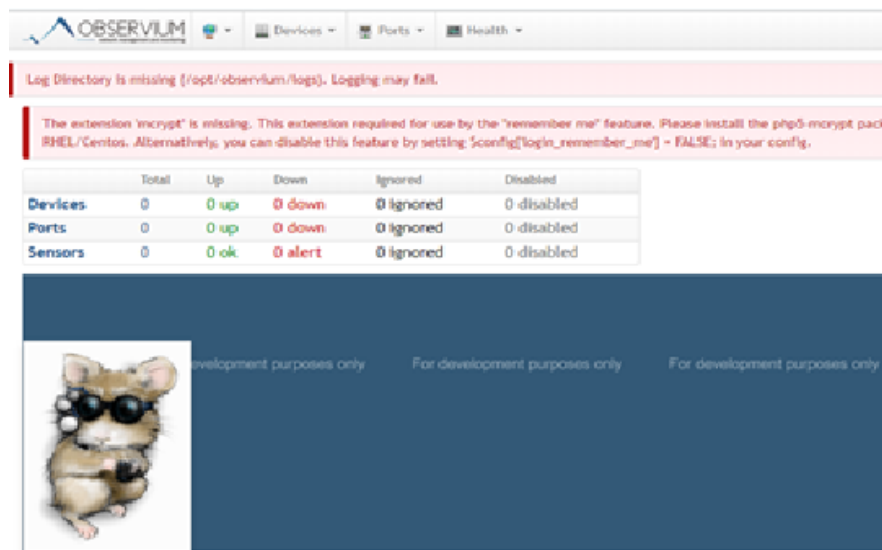


Figura 2.4: Conectividad

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 direccion ip de nuestros sistemas operativos y el nombre de la comunidad que le asignamos a cada S.O. (figura 2.5).

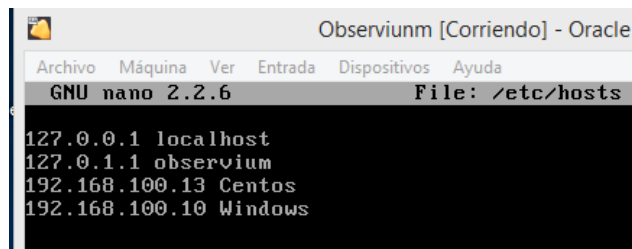


Figura 2.5: Agentes que contiene Observium

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 deseabamos 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 deseabamos 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 (contact 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

The location of the system: Laboratorio progra 1

Finished Output: syslocation "Laboratorio progra 1"

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 know, 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 all]:
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.

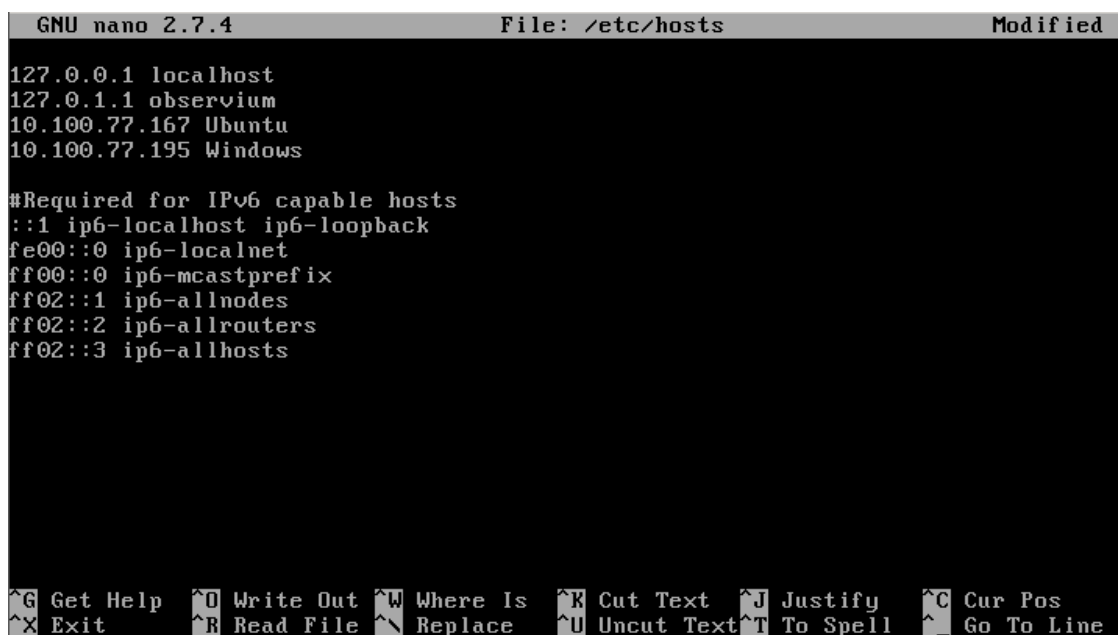
```
GNU nano 2.5.3 File: /etc/snmp/snmpd.conf
#####
# snmpd.conf
# - created by the snmpconf configuration program
#
#####
# SECTION: Access Control Setup
#
# This section defines who is allowed to talk to your running
# snmp agent.
#
# rwcommunity: a SNMPv1/SNMPv2c read-write access community name
# 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.



```

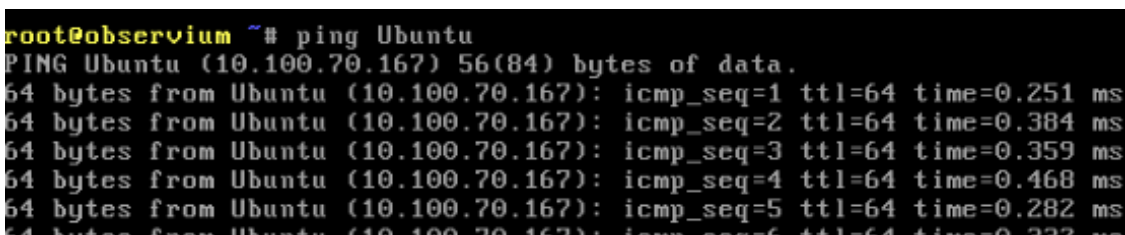
GNU nano 2.7.4      File: /etc/hosts      Modified
127.0.0.1 localhost
127.0.1.1 observium
10.100.77.167 Ubuntu
10.100.77.195 Windows

#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  ^R 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 5.5



```

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=0.337 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 OBSERVUM web interface. The 'Basic Configuration' panel on the left includes fields for Hostname (Ubuntu), Skip PING (checked), Protocol Version (v2c), Transport (UDP), Port (161), Timeout (1), Retries (5), and Ignore existing RRDs (checked). The 'Authentication Configuration' panel on the right shows the SNMP Community (comunidadMarcela). A search bar and navigation icons are at the top. A 'Add device' button is at the bottom of the Basic Configuration panel.

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.

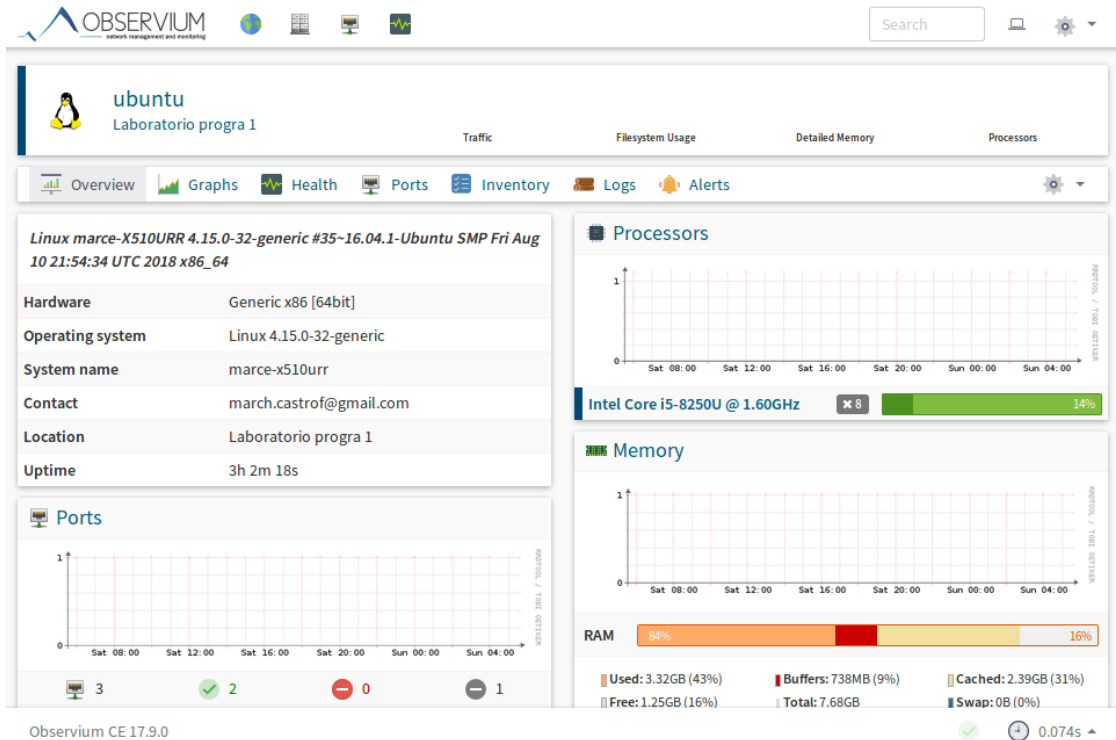


Figura 2.12: Información del agente.

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.

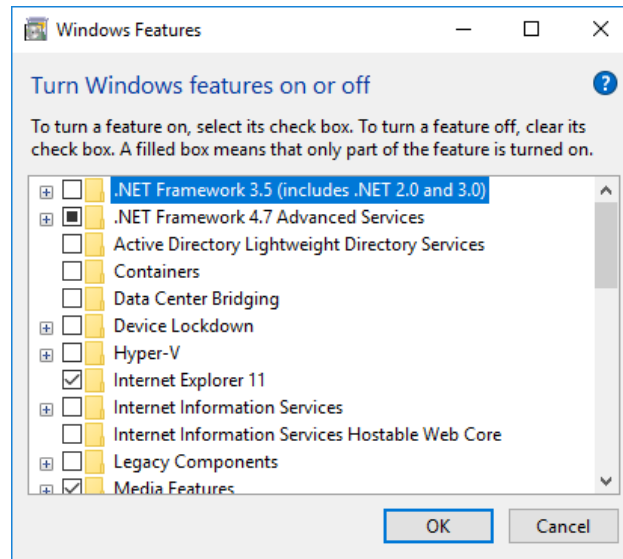


Figura 2.13: Características de Windows.

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.

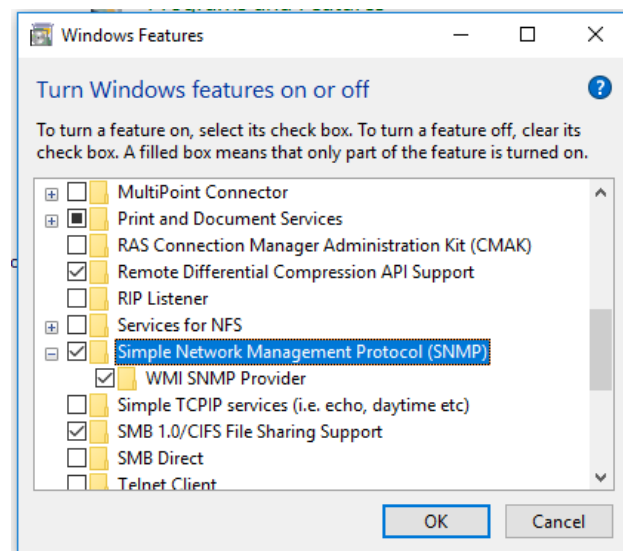


Figura 2.14: Protocolo SNMP.

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:

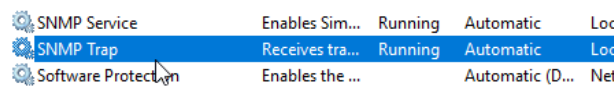


Figura 2.15: Captura SNMP.

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.

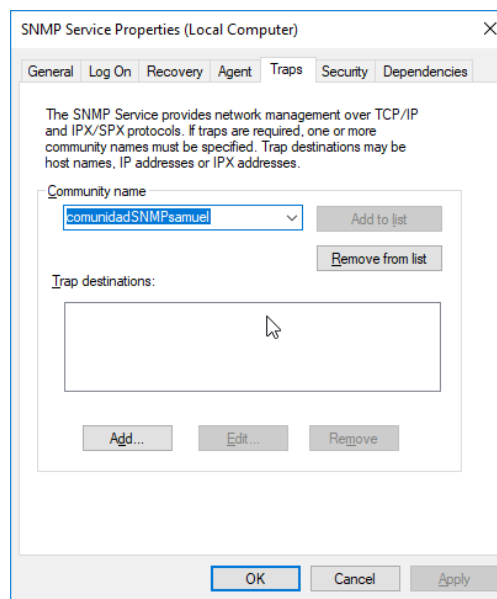


Figura 2.16: Comunidad SNMP.

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**.

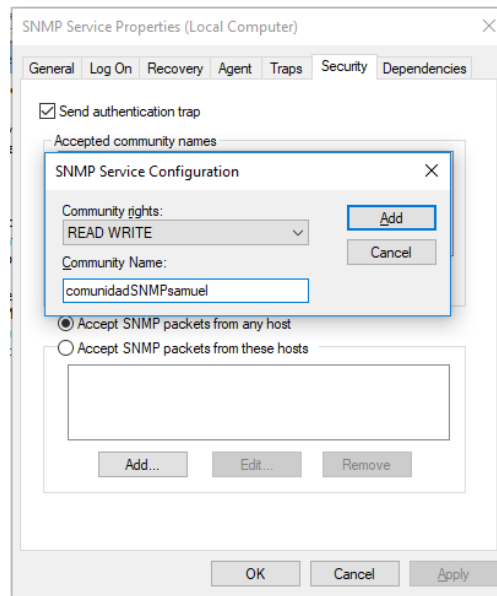


Figura 2.17: Permisos de la comunidad.

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.

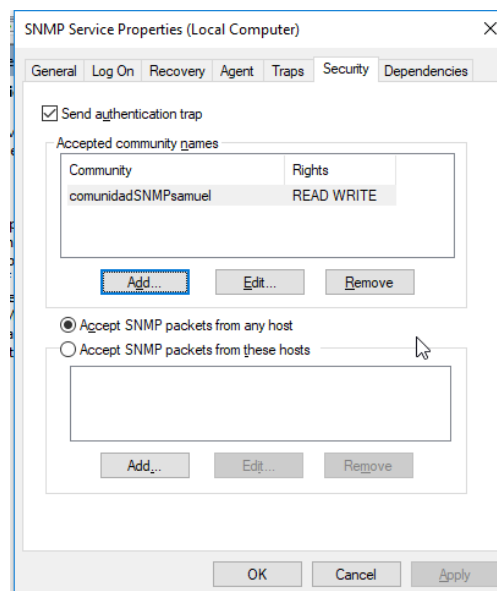


Figura 2.18: Servicio SNMP.

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:

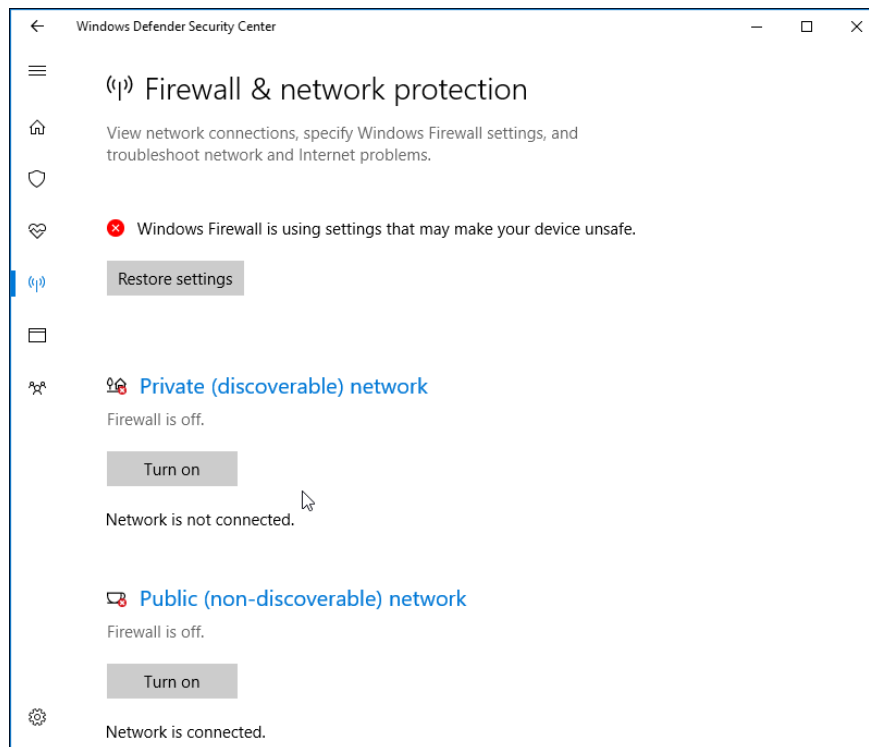


Figura 2.19: Firewall de Windows.

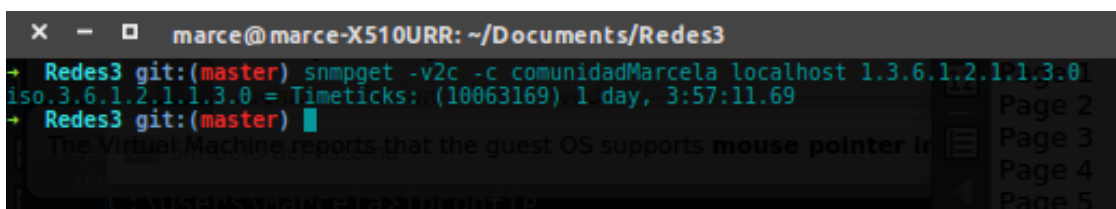
En este capítulo, se observan las diferentes pantallas que responden a las consultas realizadas a la MIB de Linux y de Windows y que de igual manera, muestra el segundo punto de la práctica que se refiere a la utilización del comando `snmpget` y algunos otros.

3.1. Cuestionario

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

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

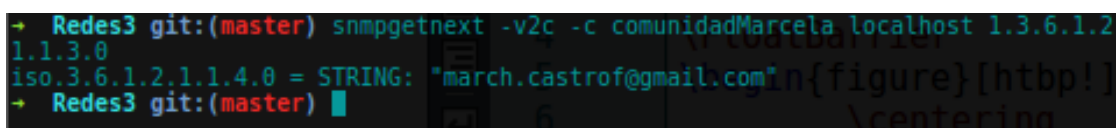
Comando: `snmpget` (figura 3.1)



```
marce@marce-X510URR: ~/Documents/Redes3
→ Redes3 git:(master) snmpget -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.1.3.0
iso.3.6.1.2.1.1.3.0 = Timeticks: (10063169) 1 day, 3:57:11.69
→ Redes3 git:(master) █
```

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

Comando: `snmpgetnext` (figura 3.2)



```
marce@marce-X510URR: ~/Documents/Redes3
→ 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: 'march.castrof@gmail.com'
→ Redes3 git:(master) █
```

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
+ 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"
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: (8798361) 1 day, 0:26:23.61
iso.3.6.1.2.1.1.4.0 = STRING: "march.castrof@gmail.com"
iso.3.6.1.2.1.1.5.0 = STRING: "marce-X510URR"
iso.3.6.1.2.1.1.6.0 = STRING: "\"Laboratorio progra 1\"
iso.3.6.1.2.1.1.8.0 = Timeticks: (65) 0:00:00.65
```

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:

```
Administrador: Símbolo del sistema

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) * snmpwalk -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"
iso.3.6.1.2.1.2.2.1.2.2 = STRING: "Intel Corporation Device 24fd"
+ Redes3 git:(master) * snmpgetnext -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.2.2.1.2.
l
iso.3.6.1.2.1.2.2.1.2.2 = STRING: "Intel Corporation Device 24fd"
+ Redes3 git:(master) * snmpgetnext -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.2.2.1.2.
l
```

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 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
```

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) * 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) * 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 corresponder 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) * 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) * 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) * 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) * 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 Tunneling 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

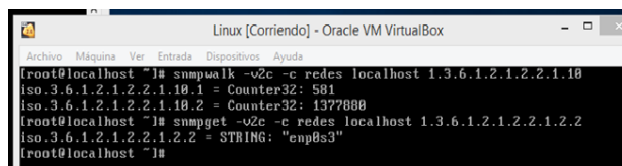
C:\Windows\system32>snmpwalk -v2c -c comunidad3 192.168.1.66 1.3.6.1.2.1.2.2.1.5
IF-MIB::ifSpeed.1 = Gauge32: 1073741824
IF-MIB::ifSpeed.2 = Gauge32: 0
IF-MIB::ifSpeed.3 = Gauge32: 0
IF-MIB::ifSpeed.4 = Gauge32: 0
IF-MIB::ifSpeed.5 = Gauge32: 0
IF-MIB::ifSpeed.6 = Gauge32: 1000000000
IF-MIB::ifSpeed.7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

3.13



```
Linux [Corriendo] - Oracle VM VirtualBox
[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: 1377888
[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.

3.14

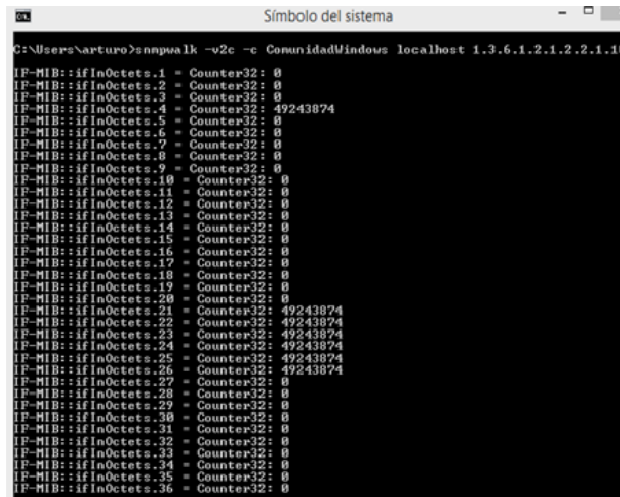


Figura 3.14: tabla de las interfaces en Windows con sus Octetos

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:

3.15

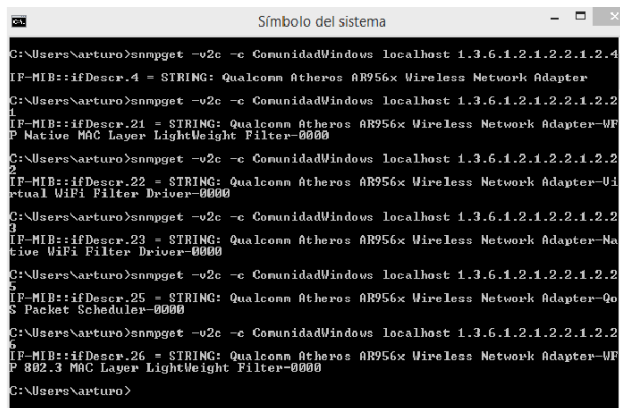


Figura 3.15: Interfaces

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**.

3.16

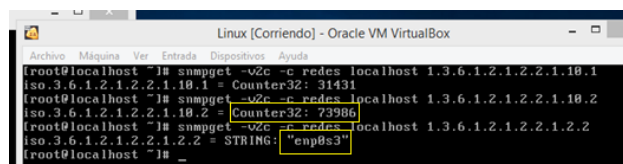


Figura 3.16: Octetos en enp0s3

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

3.17

```
C:\Users\arturo>snmpwalk -v2c -c ComunidadWindows localhost 1.3.6.1.2.1.2.1.10
ifMIB::ifInOctets.1 = Counter32: 0
ifMIB::ifInOctets.2 = Counter32: 0
ifMIB::ifInOctets.3 = Counter32: 0
ifMIB::ifInOctets.4 = Counter32: 49243874
ifMIB::ifInOctets.5 = Counter32: 0
ifMIB::ifInOctets.6 = Counter32: 0
ifMIB::ifInOctets.7 = Counter32: 0
ifMIB::ifInOctets.8 = Counter32: 0
ifMIB::ifInOctets.9 = Counter32: 0
ifMIB::ifInOctets.10 = Counter32: 0
ifMIB::ifInOctets.11 = Counter32: 0
ifMIB::ifInOctets.12 = Counter32: 0
ifMIB::ifInOctets.13 = Counter32: 0
ifMIB::ifInOctets.14 = Counter32: 0
ifMIB::ifInOctets.15 = Counter32: 0
ifMIB::ifInOctets.16 = Counter32: 0
ifMIB::ifInOctets.17 = Counter32: 0
ifMIB::ifInOctets.18 = Counter32: 0
ifMIB::ifInOctets.19 = Counter32: 0
ifMIB::ifInOctets.20 = Counter32: 0
ifMIB::ifInOctets.21 = Counter32: 49243874
ifMIB::ifInOctets.22 = Counter32: 49243874
ifMIB::ifInOctets.23 = Counter32: 49243874
ifMIB::ifInOctets.24 = Counter32: 49243874
ifMIB::ifInOctets.25 = Counter32: 49243874
ifMIB::ifInOctets.26 = Counter32: 49243874
ifMIB::ifInOctets.27 = Counter32: 0
ifMIB::ifInOctets.28 = Counter32: 0
ifMIB::ifInOctets.29 = Counter32: 0
ifMIB::ifInOctets.30 = Counter32: 0
ifMIB::ifInOctets.31 = Counter32: 0
ifMIB::ifInOctets.32 = Counter32: 0
ifMIB::ifInOctets.33 = Counter32: 0
ifMIB::ifInOctets.34 = Counter32: 0
ifMIB::ifInOctets.35 = Counter32: 0
ifMIB::ifInOctets.36 = Counter32: 0
```

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**.

3.18

```
(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 = 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**.

3.19

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@ess:~/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@ess:~/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) * 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) * 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) * 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) * snmpwalk -v2c -c comunidadMarcela localhost 1.3.6.1.2.1.6
iso.3.6.1.2.1.6.1.0 = INTEGER: 1
iso.3.6.1.2.1.6.2.0 = INTEGER: 200
iso.3.6.1.2.1.6.3.0 = INTEGER: 120000
iso.3.6.1.2.1.6.4.0 = INTEGER: -1
iso.3.6.1.2.1.6.5.0 = Counter32: 106258
iso.3.6.1.2.1.6.6.0 = Counter32: 84705
iso.3.6.1.2.1.6.7.0 = Counter32: 38
iso.3.6.1.2.1.6.8.0 = Counter32: 16159
iso.3.6.1.2.1.6.9.0 = Gauge32: 12
iso.3.6.1.2.1.6.10.0 = Counter32: 2179165
iso.3.6.1.2.1.6.11.0 = Counter32: 2167242
iso.3.6.1.2.1.6.12.0 = Counter32: 5918
```

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

11. ¿Cuántos mensajes EGP ha recibido el agente?
12. Indica el Sistema Operativo que maneja el agente.
13. Modifica el estatus administrativo (a down) de la interfaz que ha recibido más octetos.
14. Genera una alerta para avisar cuando se reinicie el agente.
15. Dibuja la MIB del agente.

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-kali1-amd64 #1 SMP Debian 4.17.8-
1kali1 (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.

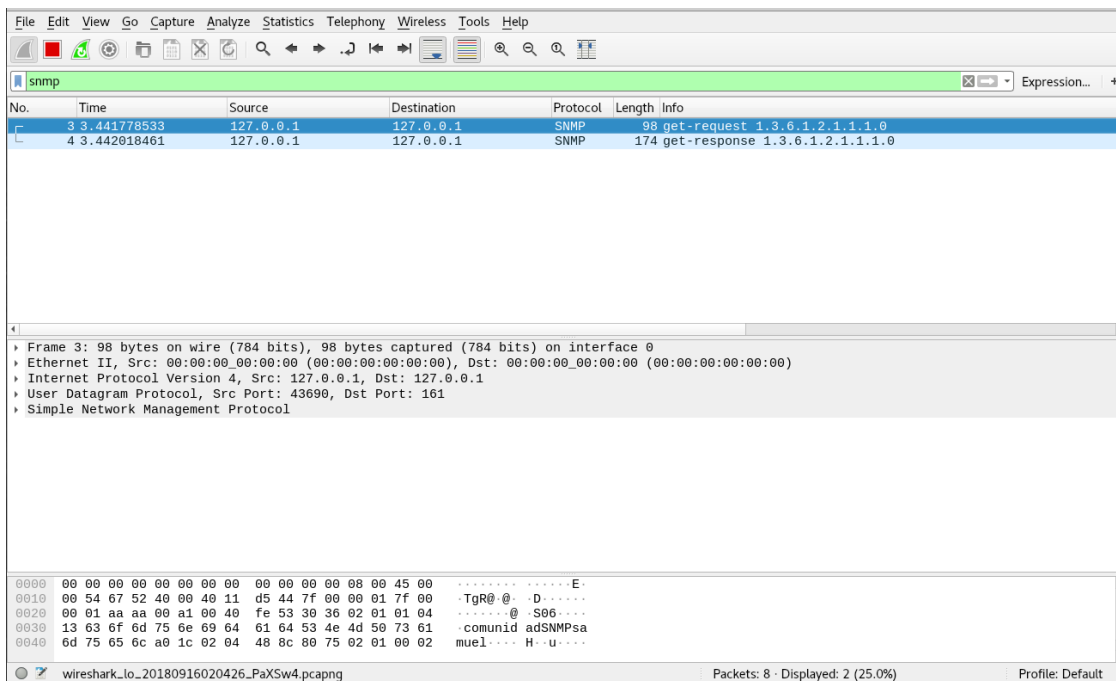


Figura 4.2: Captura snmpget.

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.

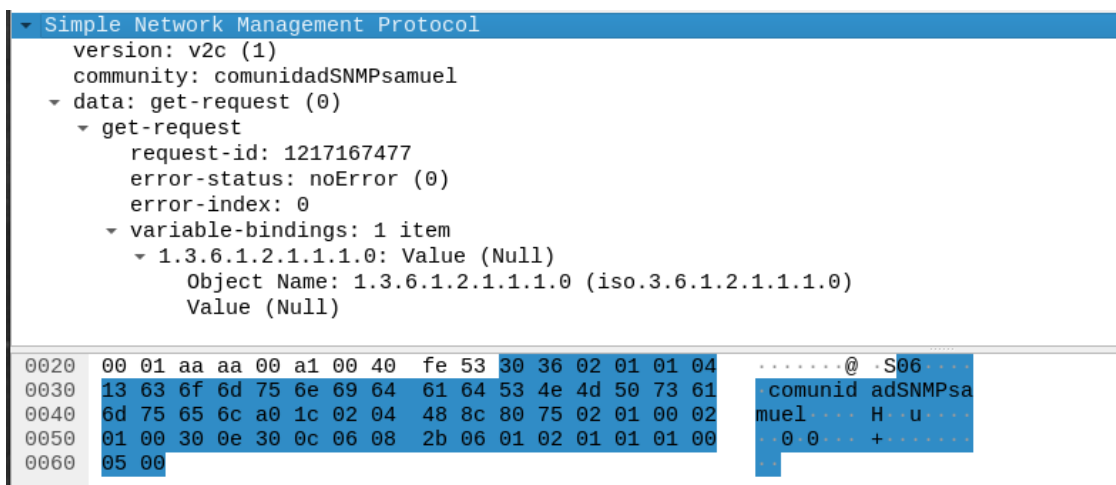


Figura 4.3: Solicitud snmpget.

```

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
        1.3.6.1.2.1.1.1.0: 4c696e757820737320342e31372e302d6b616c69312d616d...
          Object Name: 1.3.6.1.2.1.1.1.0 (iso.3.6.1.2.1.1.1.0)
          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 dadSNMPs
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 .....0Y0W +
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: 73616d75656c2e61736d406f75746c6f66b2e636f6d
          Object Name: 1.3.6.1.2.1.1.4.0 (iso.3.6.1.2.1.1.4.0)
          Value (OctetString): 73616d75656c2e61736d406f75746c6f66b2e636f6d
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 comunidadSNMPsamuel localhost 1.3.6.1.2.1.1
iso.3.6.1.2.1.1.1.0 = STRING: "Linux ss 4.17.0-kali1-amd64 #1 SMP Debian 4.17.8-1kali1 (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 1.3.6.1.2.1.1.0
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	0.001136492	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**.

snmp						
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.

```

4
▶ Frame 3: 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: 37454, Dst Port: 161
▼ Simple Network Management Protocol
  version: v2c (1)
  community: comunidadSNMPsamuel
  data: set-request (3)
    ▼ set-request
      request-id: 1978625716
      error-status: noError (0)
      error-index: 0
      ▼ 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 92 4e 00 a1 00 5c fe 6f 30 52 02 01 01 04	...N... \o0R...
0030	13 63 6f 6d 75 6e 69 64 61 64 53 4e 4d 50 73 61	comunid adSNMPsa
0040	6d 75 65 6c a3 38 02 04 75 ef 6e b4 02 01 00 02	muel 8 u n...
0050	01 00 30 2a 30 28 06 08 2b 06 01 02 01 01 04 00	0*0(... +.....
0060	04 1c 61 73 61 6e 74 69 61 67 6f 6d 31 34 30 31	asanti agom1401

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\..oR..
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	..0*0(.. +.....
0060	04 1c 61 73 61 6e 74 69 61 67 6f 6d 31 34 30 31	..asanti agom1401

Figura 4.14: Respuesta snmpset.

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 dispositivos.
- 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, 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.

A continuación explicaré el código más importante de esta sección.

Primero se ejecuta el método llamado `getHostInfo` como se muestra en la figura [5.1](#)

```

292 def getHostInfo():
293     print 'holaaaaa'
294     global agentCount
295     global canvasFrame
296     ip_comunnity = [] # va con doble m no doble n
297     try:
298         if canvasFrame == None:
299             canvasFrame = Frame(photoCanvas, width=1838, height=880)
300             photoCanvas.create_window(0, 0, window=canvasFrame, anchor='nw')
301
302             add = Tkinter.Button(canvasFrame, text="Agregar 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 #Aquí esta mi contador
322                 if palabras[3].endswith('\n'):
323                     palabras[3] = palabras[3][:-1]
324                     ip_comunnity.append({'ip' : str(palabras[0]), 'port' : str(palabras[2]), 'community' : str(palabras[3])})
325                 else:
326                     ip_comunnity.append({'ip' : str(palabras[0]), 'port' : str(palabras[2]), 'community' : str(palabras[3])})
327
328             getAgentInfo(ip_comunnity)
329             file.close()
330             photoScroll = Scrollbar(photoFrame, orient=VERTICAL)
331             photoScroll.config(command=photoCanvas.yview)
332             photoCanvas.config(yscrollcommand=photoScroll.set)
333             photoScroll.grid(row=0, column=1, sticky="ns")
334
335             hsbar = Scrollbar(photoFrame, orient=HORIZONTAL, command=photoCanvas.xview)
336             photoCanvas.config(xscrollcommand=hsbar.set)
337             hsbar.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 5.1: Método getHostInfo.

Dentro del cuál se observan las siguientes líneas de la figura 5.2 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 #Aquí esta mi contador
    if palabras[3].endswith('\n'):
        palabras[3] = palabras[3][:-1]
        ip_comunnity.append({'ip' : str(palabras[0]), 'port' : str(palabras[2]), 'community' : str(palabras[3])})
    else:
        ip_comunnity.append({'ip' : str(palabras[0]), 'port' : str(palabras[2]), 'community' : str(palabras[3])})

getAgentInfo(ip_comunnity)
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 5.2: 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 5.3.

```

432
433 def getAgentInfo(ip_community):
434     print 'getAgentInfo'
435     status_array = []
436     interface_name_status= []
437     global agentCount
438     r = 0
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('\n * * * * * \nEl status es: ' + status_received)
445             agents = consultaSNMP(computer['community'], computer['port'], computer['ip'], '1.3.6.1.2.1.1.0')
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'], '1.3.6.1.2.1.2.2.1.2.'+str(i))
454                 status_inter = consultaSNMP(computer['community'], computer['port'], computer['ip'], '1.3.6.1.2.1.2.2.1.8.'+str(i))
455
456                 if name_interfaces[1] == '0':
457                     interface_name = name_interfaces[3:].decode('hex')
458                     Label(canvasFrame, text=interface_name, width=100, fg='black').grid(row=ro, column=3, sticky="nsew")
459                 else:
460                     Label(canvasFrame, text=name_interfaces, width=100, 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             Tkinter.Button(canvasFrame, text="Gráficas", width=10, command= lambda name = computer['community'], computer['port'], computer['ip'], '1.3.6.1.2.1.2.2.1.2.'+str(i)): MostrarEstado(name)).grid(row=r, column=6, sticky="nsew")
474             Tkinter.Button(canvasFrame, text="Estado", width=10, command= lambda name = computer['ip']: MostrarEstado(name)).grid(row=r, column=6, sticky="nsew")
475             Tkinter.Button(canvasFrame, text="Eliminar", width=10, command= lambda name = computer['ip']: eliminarAgente(name)).grid(row=r, column=7, sticky="nsew")
476             r = r + int(interfaces)
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=ro, column=3, sticky="nsew")
481             Tkinter.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 5.3: Método `getAgentInfo`.

Al cual, nuevamente mostraremos solo las líneas principales en la siguiente figura 5.4 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'], '1.3.6.1.2.1.1.0')
        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'], '1.3.6.1.2.1.2.2.1.2.'+str(i))
            status_inter = consultaSNMP(computer['community'], computer['port'], computer['ip'], '1.3.6.1.2.1.2.2.1.8.'+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 5.4: Método `getAgentInfo`, líneas importantes.

Por último, es importante mostrar 2 métodos igualmente importantes, uno es el método ping mostrado en la figura 5.5 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 out
                stderr=DEVNULL
            )
            is_up = 'Activa'
            alive_hosts.append(ip)
        except subprocess.CalledProcessError:
            is_up = 'Inactiva'
    print is_up
    return is_up
```

Figura 5.5: Método ping.

Por otro lado, el otro método a mostrar es el de la figura 5.6 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 5.6: Método consultaSNMP.

5.2. Agregar agente

Por otro lado, para agregar un nuevo agente, es necesario presionar sobre el botón en la parte superior que es quien nos manda al método mostrado en la figura 5.7, el cual se encarga de abrir una pequeña ventana como la de la figura 5.8, y en este, se manda a llamar a otros 2 métodos mostrados a continuación.

```
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)))
    b1 = Button(root, text='Agregar agente',command=(lambda e=ents: fetch(e)))
    b1.pack(side=LEFT, padx=5, pady=5)
```

Figura 5.7: Método addClient.

Figura 5.8: Pantalla desplegada para añadir un cliente nuevo.

Por un lado se llama al método al método mostrado en la figura 5.9, 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 5.9: Método makeform.

Y por otro lado se llama al método fetch mostrado en la figura 5.10, 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 5.11 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 + ' '
    print('%s: "%s" ' % (field, text))
file = open("hosts.txt", "a+")
file.write(concatenation + '\n')
file.close()
```

Figura 5.10: Pantalla desplegada para añadir un cliente nuevo.

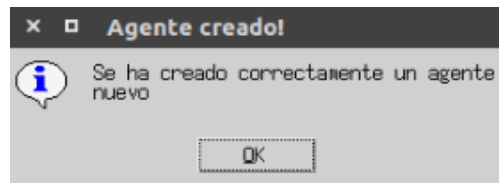


Figura 5.11: Alerta desplegada al crearse un nuevo agente.

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

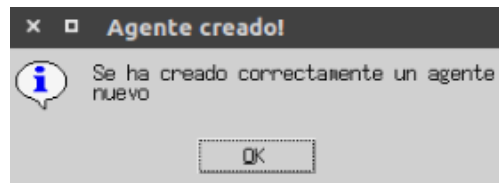


Figura 5.12: Alerta desplegada al crearse un nuevo agente.

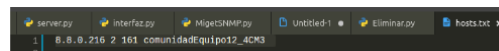


Figura 5.13: Agente agregado en archivo hosts.

Es importante recalcar que en la última línea del método `getHostInfo` mostrado en la imagen 5.2, 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

5.4. Estado del dispositivo

5.5. Gráficas de dispositivos

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].