Anexo A

Acrónimos

- **SNMP:** Simple Network Management Protocol.
- **RFC:** Requests for Comments.
- **RMON:** Remote Network Monitoring.
- **MIB:** Management Information Base.
- **GUI:** Graphical User Interface.
- **ASN:** Abstract Syntax Notation.
- **AES:** Advanced Encryption Standard.
- **DES:** Data Encryption Standard.
- **SHA1:** Secure Hash Algorithm 1.
- **MD5:** Message-Digest Algorithm 5.
- **MySQL:** My Structured Query Language.
- **LibPCAP:** Packet Capture Library.
- **IETF:** Internet Engineering Task Force.
- **SMI:** Structure of Management Information.

- **USM:** User-based Security Model.
- **VACM:** View-based Access Control Model.
- **BER:** Basic Encoding Rules.
- **OID:** Object Identifier.
- **LAN:** Local Area Network.
- **OSI:** Open System Interconnection.
- **SMON:** RMON Extensions for Switched Networks.
- **DSMON:** RMON Extensions for Differentiated Services.
- **HCRMON:** RMON for High Capacity Networks.
- **API:** Application Programming Interface.
- **HTTP:** Hypertext Transfer Protocol.
- **ASCII:** American Standard Code for Information Interchange.
- **BPF:** Berkeley Packet Filter.
- **HTML:** HyperText Markup Language.
- **RTF:** Rich Text Format.
- **STC:** Stylized Text Control.
- **MAC:** Media Access Control.
- **ARP:** Address Resolution Protocol.
- **IP:** Internet Protocol.
- **ICMP:** Internet Control Message Protocol.
- **TCP:** Transmission Control Protocol.
- **UDP:** User Datagram Protocol.

- **USB:** Universal Serial Bus.
- **Mbps:** Megabit por segundo.
- **NetConf:** Network Configuration Protocol.

Anexo B

Diagramas de flujo de la interacción agente SNMP - MIB

Este anexo corresponde con el anexo E del proyecto "Implementación automática de un agente SNMP a partir de la definición formal de su MIB"[1] y ha sido añadido aquí para facilitar al lector la comprensión de la forma de realizar la búsqueda en la base de datos.

En este anexo se muestran los diagramas de flujos creados para la elaboración de los algoritmos pertinentes a la interacción del Agente SNMP con la MIB según el tipo de mensaje PDU que se puede recibir del gestor.

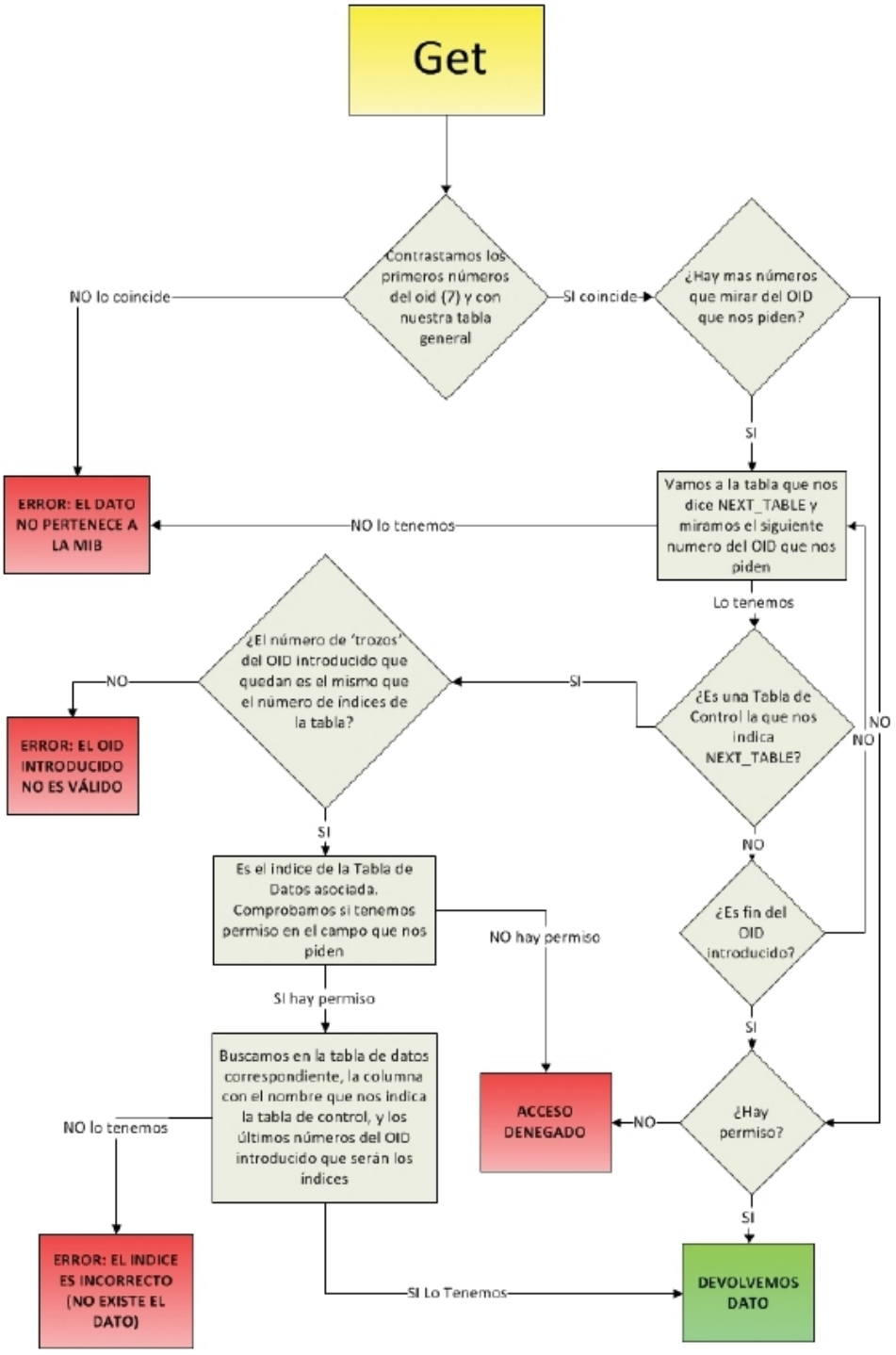

Figura B.1: Workflow Get.

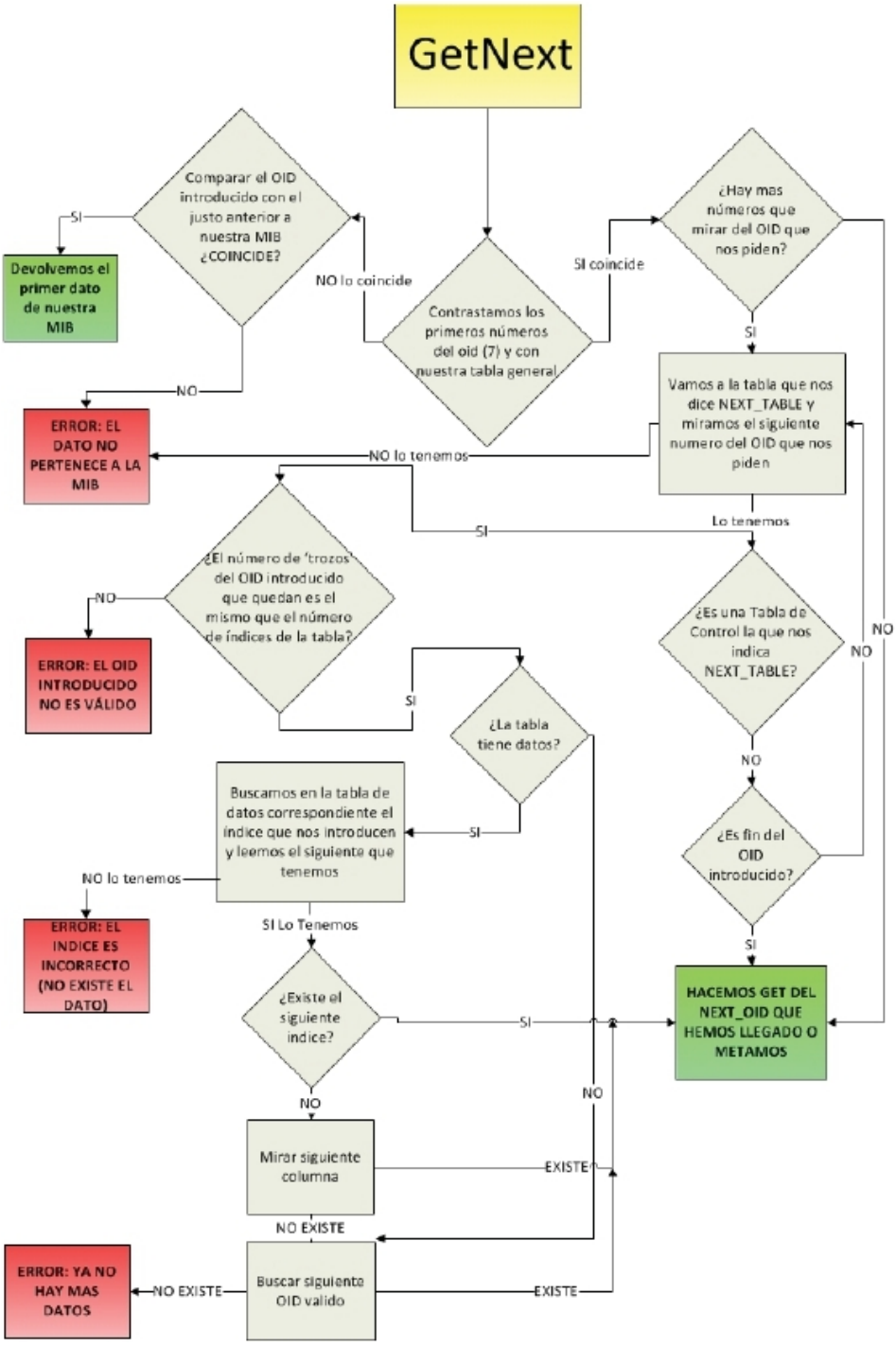

Figura B.2: Workflow GetNext.

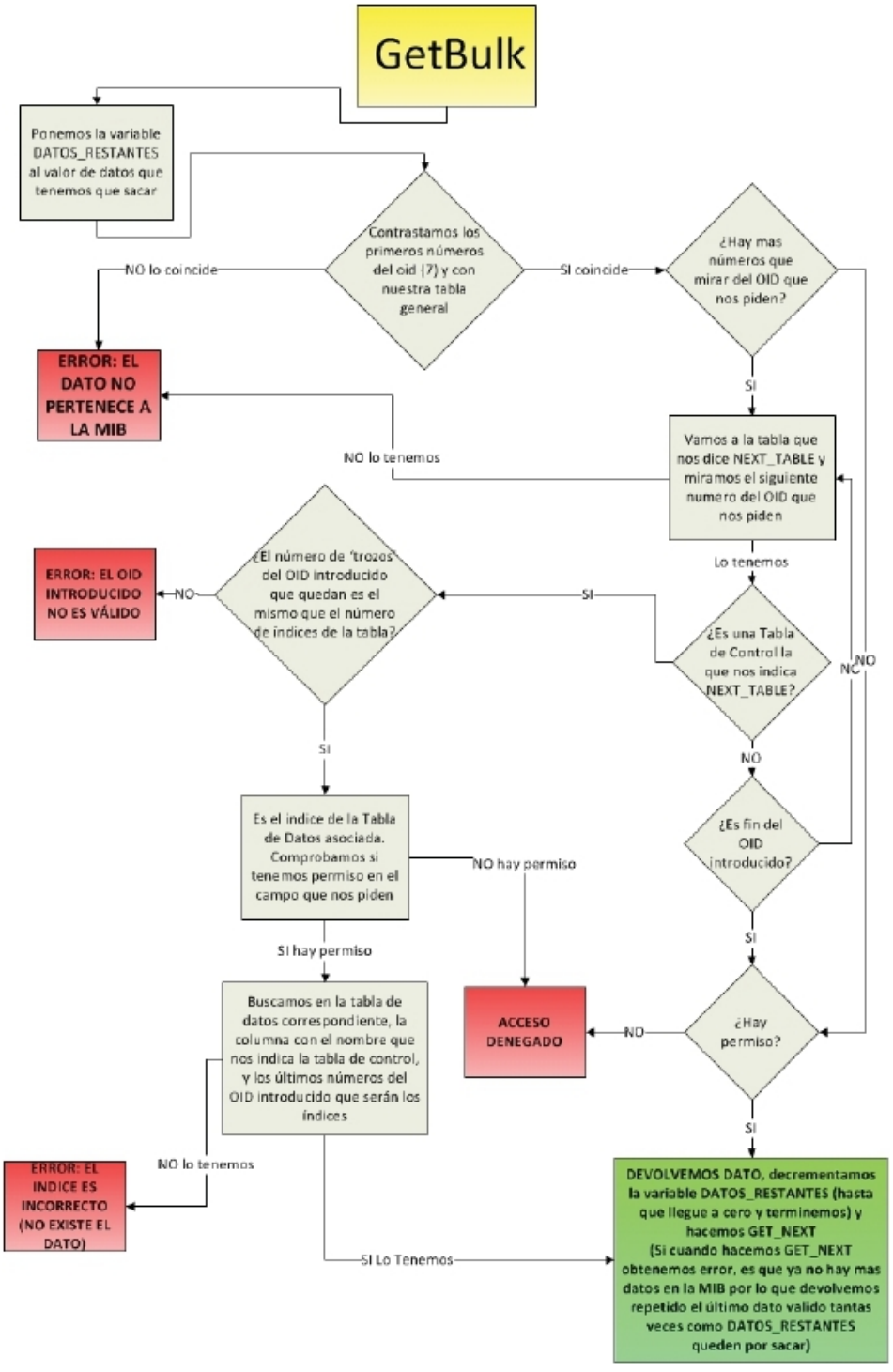

Figura B.3: Workflow GetBulk.

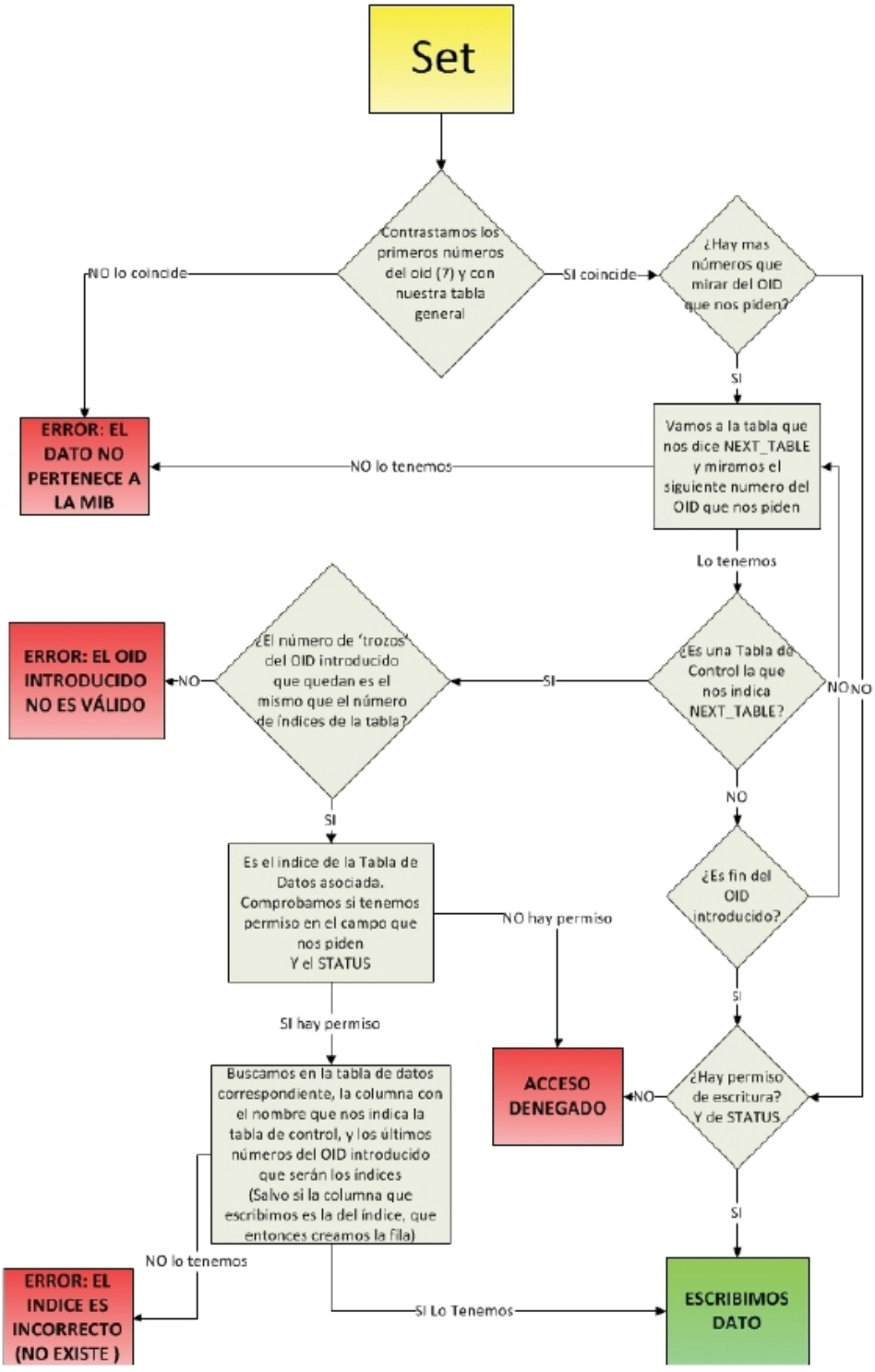

Figura B.4: Workflow Set.

Anexo C

Proceso de definición de la MIB

Para realizar la definición formal de la MIB es esencial tener en cuenta la funcionalidad que se espera que aporte al agente, así como las especificaciones del estándar SNMP, ya que es imprescindible cumplir con todas ellas. Dado que esta MIB debería permitir almacenar las comunidades y las vistas asociadas a ellas, será necesaria una tabla cuyas columnas sean el nombre de la comunidad, el nivel de acceso, el OID sobre el que se tiene acceso y un campo para manejar las filas. A priori parece razonable que esta tabla estuviese indexada por el propio nombre de la comunidad.

Según el estándar de SNMP un String no puede indexar una tabla, por lo que se decidió utilizar la representación ASCII de cada uno de los caracteres que componen el String separados por puntos, lo cual si que esta encaja dentro del estandar. Esto se puede hacer de dos formas y deberá aparecer reflejado en la MIB cuál es el método empleado:

- **Longitud implícita:** no se indica el número de caracteres que componen el índice. Ej: "*dave*" = 'd'.'a'.'v'.'e'
- **Longitud explícita:** debe añadirse el número de caracteres que componen el índice delante de éste. Este método es obligatorio si no es el último índice. Ej: "*dave*" = 4.'d'.'a'.'v'.'e'

Para facilitar la visualización del contenido de la tabla, además del campo correspondiente al índice, se añadirá otro campo que será rellenado

automáticamente por el agente cuando se cree una nueva fila, el cual representara en forma de string el nombre de la comunidad.

Dado que a una misma comunidad se le pueden asignar varias vistas no será suficiente con indexar la tabla de esta manera sino que se deberá añadir un nuevo campo, el cual será un identificador numérico que actuará como índice secundario permitiendo identificar cada una de las vistas asociadas a una misma comunidad.

Por otro lado, al utilizar SNMP para gestionar esta tabla, será necesaria la existencia de una comunidad que cuente con permisos de lectura y escritura sobre dicha tabla. Esta comunidad no puede formar parte de la tabla, ya que en ese caso no habría forma de evitar que fuese borrada. Finalmente se opto por que la comunidad de gestión no perteneciera a la tabla, y definirla como un escalar, los cuales no pueden ser borrados.

Anexo D

communityManagement MIB

ZNMRG-COMMUNITY-MANAGEMENT-MIB DEFINITIONS ::= BEGIN

IMPORTS

enterprises, Integer32, OBJECT-TYPE, MODULE-IDENTITY, OBJECT-IDENTITY, NOTIFICATION-TYPE

FROM SNMPv2-SMI

DateAndTime, DisplayString, RowStatus, TEXTUAL-CONVENTION

FROM SNMPv2-TC;

EntryStatus

FRON RMON-MIB;

– 1.3.6.1.4.1.28308.1.1.1

znmrgGlobalModule MODULE-IDENTITY

LAST-UPDATED ”201408211611Z- August 21, 2014 at 16:11 GMT

ORGANIZATION

“Organization.”

CONTACT-INFO

“Contact-info.”

DESCRIPTION

“Description.”

::= znmrgModules 1

– – Node definitions –

– 1.3.6.1.4.1.28308

znmrgRoot OBJECT-IDENTITY

STATUS current

DESCRIPTION

"The root of the OID sub-tree assigned to Company by the Internet Assigned Numbers Authority (IANA)"

::= enterprises 28308

– 1.3.6.1.4.1.28308.1

znmrgReg OBJECT-IDENTITY

STATUS current

DESCRIPTION

"Sub-tree for registrations"

::= znmrgRoot 1

– 1.3.6.1.4.1.28308.1.1

znmrgModules OBJECT-IDENTITY

STATUS current

DESCRIPTION

"Sub-tree to register the values assigned to modules with the MODULE-IDENTITY construct"

::= znmrgReg 1

– 1.3.6.1.4.1.28308.2

znmrgGeneric OBJECT-IDENTITY

STATUS current

DESCRIPTION

"Sub-tree for common object and event definitions"

::= znmrgRoot 2

– 1.3.6.1.4.1.28308.3

znmrgProducts OBJECT-IDENTITY

STATUS current

DESCRIPTION

"Sub-tree for specific object and event definitions"

::= znmrgRoot 3

– 1.3.6.1.4.1.28308.3.1

communityManagement OBJECT IDENTIFIER ::= znmrgProducts 1

– 1.3.6.1.4.1.28308.3.1.1

master OBJECT-TYPE

SYNTAX DisplayString

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"This is the community used to manage the CommunityManagementTable."

::= communityManagement 1

– 1.3.6.1.4.1.28308.3.1.2

communityTable OBJECT-TYPE

SYNTAX SEQUENCE OF communityEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"This is the table that store all commuintyEntry."

::= communityManagement 2

– 1.3.6.1.4.1.28308.3.1.2.1

communityEntry OBJECT-TYPE

SYNTAX CommunityEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"This Each entry will store one community view and its permissions."

INDEX communityIndex, communityRowID

::= communityTable 1

CommunityEntry ::=

SEQUENCE

communityIndex

OBJECT IDENTIFIER,

communityName

DisplayString,

communityRowID

Integer32 (1..255),

communityAccess

Integer32 (0..3),

communityView

OBJECT IDENTIFIER,

communityStatus

EntryStatus

– 1.3.6.1.4.1.28308.3.1.2.1.1

communityIndex OBJECT-TYPE

SYNTAX OBJECT IDENTIFIER

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This is the ascII encoding of each character in the communityName."

::= communityEntry 1

– 1.3.6.1.4.1.28308.3.1.2.1.2

communityName OBJECT-TYPE

SYNTAX DisplayString

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This is the name of the community."

::= communityEntry 2

– 1.3.6.1.4.1.28308.3.1.2.1.3

communityRowID OBJECT-TYPE

SYNTAX Integer32 (0..255)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This field indexes each fo the views in the same community."

::= communityEntry 3

– 1.3.6.1.4.1.28308.3.1.2.1.4

communityAccess OBJECT-TYPE

SYNTAX Integer32 (0..3)

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"This is the access level the community has for this view:

0) No permissions

1) Read only

2) Write only

3) Read and Write”

::= communityEntry 4

– 1.3.6.1.4.1.28308.3.1.2.1.5

communityView OBJECT-TYPE

SYNTAX OBJECT IDENTIFIER

MAX-ACCESS read-write

STATUS current

DESCRIPTION

”This This community will have the permissions specified in this OID and all its descendants.”

::= communityEntry 5

– 1.3.6.1.4.1.28308.3.1.2.1.6

communityStatus OBJECT-TYPE

SYNTAX EntryStatus

MAX-ACCESS read-write

STATUS current

DESCRIPTION

”This column is used to manage each row of this table.”

::= communityEntry 6

– 1.3.6.1.4.1.28308.4

znmrgCaps OBJECT-IDENTITY

STATUS current

DESCRIPTION

”Sub-tree for agent profiles”

::= znmrgRoot 4

– 1.3.6.1.4.1.28308.5

znmrgReqs OBJECT-IDENTITY

STATUS current

DESCRIPTION

"Sub-tree for management application requirements"

::= znmrgRoot 5

– 1.3.6.1.4.1.28308.6

znmrgExpr OBJECT-IDENTITY

STATUS current

DESCRIPTION

"Sub-tree for experimental definitions"

::= znmrgRoot 6

END

Anexo E

Guía gestión de las comunidades

Como ya se ha explicado, la forma de controlar los permisos en SNMP es a través de comunidades, pero no existe una forma estándar para gestionarlas y esto queda a elección de cada agente, bien sea mediante un fichero de texto, una página web o línea de comandos.

En este trabajo se ha decidido implementar todo lo relativo a la gestión de las comunidades a través del envío de mensajes SNMP, definiendo para ello una nueva MIB, que se adjunta al final del documento y tiene una estructura parecida a la siguiente:

+ enterprise
 |——— + Zaragoza Network Management Research Group
  |——— + master
  |——— + communityTable
   |——— + communityEntry
    |——— + communityIndex
    |——— + communityName
    |——— + communityRowId
    |——— + communityAccess
    |——— + communityView
    |——— + communityStatus

A continuación se explicará el funcionamiento y los procedimientos empleados

para añadir y eliminar comunidades así como para modificar los permisos de las ya existentes.

En primer lugar, debido a que la gestión de las comunidades se va a realizar a través de mensajes SNMP será necesaria la existencia de una comunidad con permisos para realizar dicha gestión. Esta comunidad será el valor del campo master y por defecto es .ªdmin". Para modificarla basta con enviar un mensaje set como el siguiente:

snmpset -v [versión] -c [comunidad de gestión actual] [host del agente] .1.3.6.1.4.1.28308.1 s [comunidad de gestión nueva]

El siguiente paso es crear comunidades con diferentes permisos de lectura y/o escritura en las diferentes MIBs soportadas por el agente. Para ello habrá que crear nuevas entradas en la communityTable. Esto se hace con el mismo sistema de creación de filas implementado por RMON, mediante un campo del tipo EntryStatus.

Cada fila se indexa mediante los campos communityIndex y communityRowId. El communityIndex identifica a la comunidad y el communityRowId a cada una de las entradas correspondientes a dicha comunidad, ya que cada comunidad puede tener diferentes permisos de lectura y escritura sobre cada parte de la MIB.

Dado que se desea utilizar como índice de la fila un string, habrá que utilizar la representación ASCII de cada uno de los caracteres que forman el string. Esto puede hacerse de forma muy sencilla gracias a NetSNMP y para crear una entrada en la tabla basta con:

snmpset -v [versión] -c [comunidad de gestión] [host del agente] 1.3.6.1.4.1.28308.2.1.6.\'[comunidad]\'.[fila] i 2

A continuación se definirá una vista para esta entrada. La vista es los OID a los que va a afectar esta entrada, y serán todos los OIDs que cuelguen del valor introducido en el campo communityView. Se añadirá una vista a la entrada que se ha creado con anterioridad del siguiente modo:

snmpset -v [versión] -c [comunidad de gestión] [host del agente] 1.3.6.1.4.1.28308.2.1.5.\'[comunidad]\'.[fila] s [OID padre]

NOTA: El campo "OID padre"deberá tener una longitud mínima de dos (Ej:

.1.3) ya que si intentamos introducir el valor "1" obtendremos como resultado un error.

Una de las últimas cosas por hacer es determinar los permisos que tendrá la vista que hemos creado. Estos pueden ser:

0: no tiene permisos

1: read only

2: write only

3 read and write

Para añadir permisos a una entrada ya existente:

snmpset -v [versión] -c [comunidad de gestión] [host del agente] 1.3.6.1.4.1.28308.2.1.4.\'[comunidad]\'.[fila] i [tipo de permiso]

Y por ultimo solo resta pasar la entrada a valid:

snmpset -v [versión] -c [comunidad de gestión] [host del agente] 1.3.6.1.4.1.28308.2.1.6.\'[comunidad]\'.[fila] i 1

En el caso de que una comunidad cuente con varias entradas que afecten a un mismo OID se tomara siempre la más cercana y la más restrictiva. Por ejemplo si existen dos entradas una con vista 1.3.6.1.2.1 y access 1 y otra con 1.3.6.1.2.1.16 con access 3 se podrá leer todos los grupos de la MIB2 pero escribir únicamente en los correspondientes a RMON. Por otra parte si existen dos entradas con la vista 1.3.6.1.2.1, una con access 1 y otra con access 3 únicamente se podrá leer los grupos de la MIB2 y no escribir nada en ellos.

Dado que se está utilizando el mismo mecanismo de gestión de filas que RMON para borrar una entrada de la tabla será suficiente con:

snmpset -v [versión] -c [comunidad de gestión] [host del agente] 1.3.6.1.4.1.28308.2.1.6.\'[comunidad]\'.[fila] i 4

En el caso de querer modificar una entrada ya creada, primero será necesario pasarla a UnderCreation:

snmpset -v [versión] -c [comunidad de gestión] [host del agente] 1.3.6.1.4.1.28308.2.1.6.\'[comunidad]\'.[fila] i 3

y a continuación realizar las modificaciones requeridas.

Anexo F

Implementaciones desechadas de la funcionalidad de filtrado

F.1 Primera implementación

En el momento del inicio del agente se crea un proceso TCPDUMP el cual guardará en un fichero todos los paquetes que llegan a su interfaz. Para cada uno de los canales configurados, cuando su channelStatus toma el valor valid, se crea un proceso independiente del propio agente. Dicho proceso recibirá como único parámetro el índice de la fila cuyo channeStatus se ha modificado, con el cual recuperará toda la información relativa a ese filtro de la base de datos y lo aplicará continuamente a todos los paquetes nuevos en el fichero. En caso de encontrar alguna coincidencia actualizará la base de datos.

Para evitar que este fichero crezca de forma indefinida se configura TCPDUMP con un tamaño máximo para el fichero, alcanzado el cual se creará un nuevo fichero con nombres correlativos. Para eliminar los ficheros que ya han sido procesados, en el momento del inicio del agente se crea otro proceso que a modo de garbage collector (recolector de basura) comprobara si el fichero más antiguo está siendo accedido por algún proceso, y en caso contrario lo eliminara.

Este modelo de trabajo queda ilustrado en las figuras F.1 y F.2.

Este método degradaba significativamente las prestaciones ya que todos los

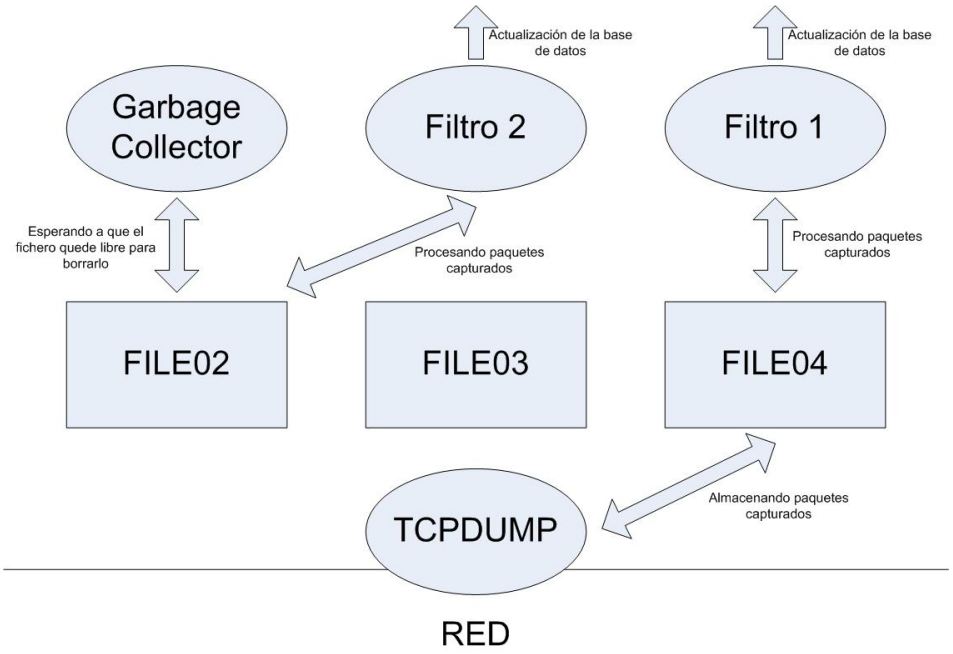

Figura F.1: Primera implementación del módulo de filtrado 1.

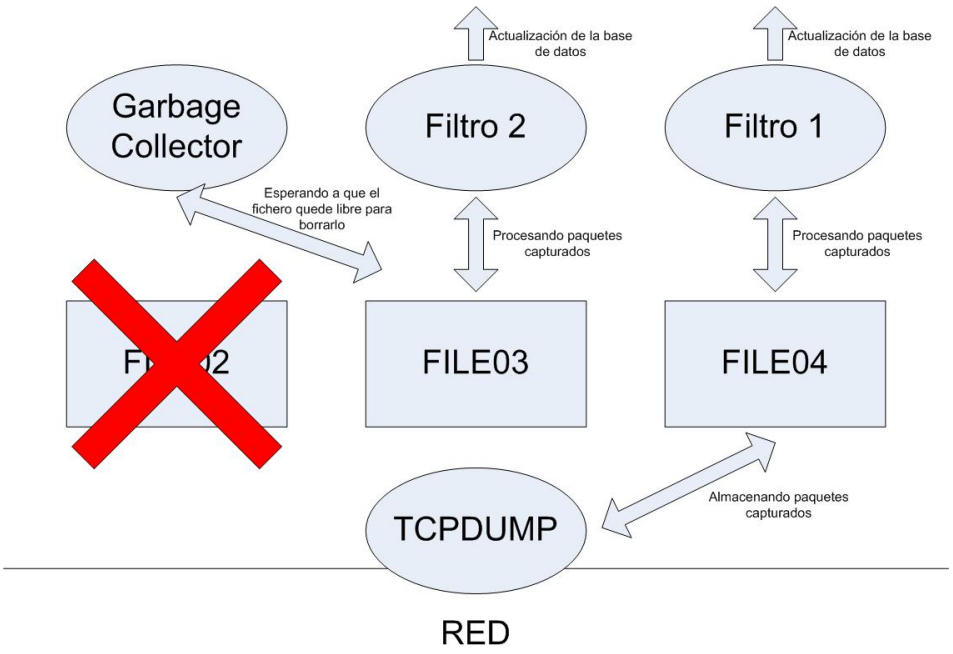

Figura F.2: Primera implementación del módulo de filtrado 2.

paquetes tenían que traspasar la frontera entre espacio de Kernel y espacio de Usuario. Además el mismo proceso encarado de realizar el filtrado debía actualizar la base de datos.

F.2 Segunda implementación

Al inicio del agente se crea un segundo proceso que ejecutará un programa principal que funciona a modo de command receiver. Ambos procesos estarán comunicados mediante una tubería, y cuando un channelStatus pase a valid, el agente, a través de la tubería, indicará al otro proceso el índice del filtro siendo éste quien creará el proceso de filtrado.

Este programa principal utilizará el índice para recuperar toda la información relativa al filtro de la base de datos, y a partir de ella generar el filtro BPF. El programa de filtrado configurará este filtro en el Kernel gracias a la librería Scapy (basada en LibPCAP), y en caso de que algún paquete supere el filtro se ejecutará una función del callback.

El programa principal crea una tubería con cada uno de los programa de filtrado que lanza, por lo que la función de callback simplemente deberá escribir en la tubería para indicar que se ha superado ese filtro, y el programa principal actualizará la base de datos.

El esquema completo de trabajo queda ilustrado en la figura F.3.

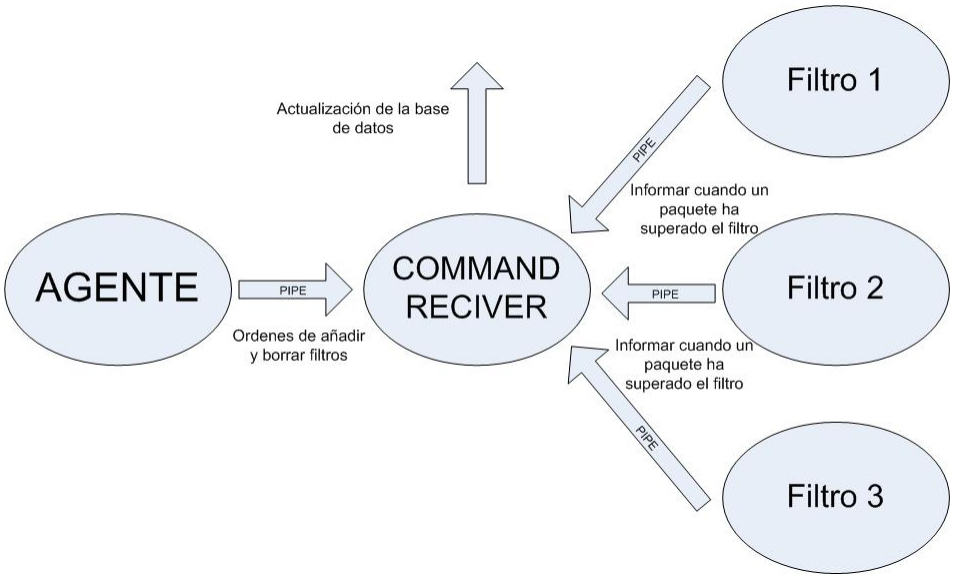

Figura F.3: Segunda implementación del módulo de filtrado.

Este método mejoro significativamente las prestaciones, ya que se redujo significativamente la cantidad de información que tenía que atravesar la frontera

entre el kernel-space y el user-space y a que la actualización de la base de datos ya no era responsabilidad de los programas de filtrado. Sin embargo el uso de tuberías seguía limitando las prestaciones.

Anexo G

Guía de manejo de la GUI

Hay que destacar que el objetivo de esta GUI es facilitar la realización de las tareas básicas, y no explotar todo el potencial que ofrece RMON.

El primer paso a realizar durante la primera utilización de este software es la creación de una nueva conexión. Para ello hay que ir al menú "Conexion"→"Add", donde aparecerá una pantalla como la de la figura G.1, en la cual se presentan los campos Name, Address, Community y Version. El campo Name únicamente sirve para identificar la conexión, el campo Address es la dirección IP del agente, el campo Community es la comunidad utilizada para realizar las peticiones, por lo que es muy recomendable que tenga permisos de lectura y escritura, y el campo Version es la versión del protocolo y se presentan las opciones v1, v2c y v3. Una vez están completados todos los campos hay que pulsar el botón "Add"para crear la conexión.

El menú "Conexion"presenta otras opciones tales como "Edit", Figura G.2, la cual muestra una lista desplegable con todas las conexiones existentes, y al seleccionar de ellas una carga los datos relativos a dicha conexión (Address, Community y Version) y permite modificarlos. Para guardar las modificaciones realizadas simplemente hay que pulsar el botón "Save".

La opción "Delete", Figura G.3, del mismo menú muestra una lista de todas las conexiones existentes y al igual que "*Edit*", al seleccionar una de ellas muestra sus datos, pero en este caso no permite realizar ninguna modificación en ninguno de

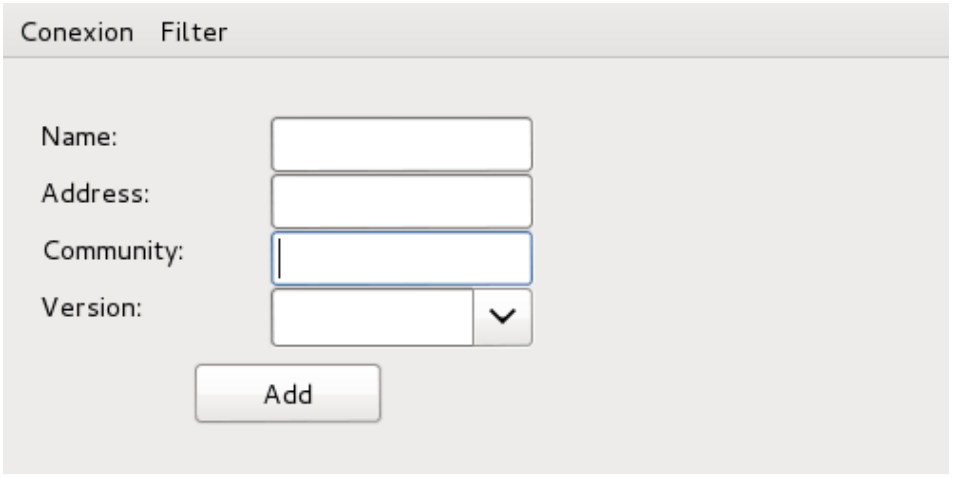

Figura G.1: Menu Conexion→Add.

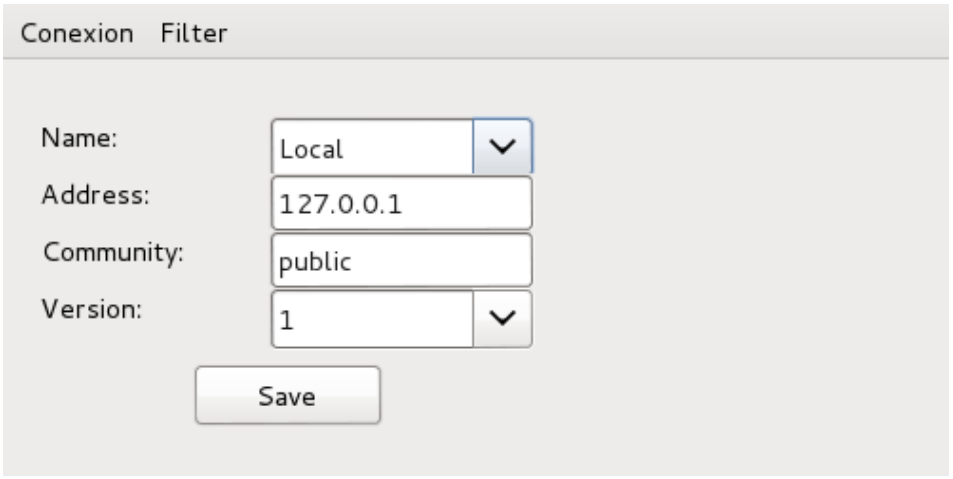

Figura G.2: Menu Conexion→Edit.

los campos, únicamente presenta la posibilidad de utilizar el botón "Delete"para borrar la conexión que se encuentre seleccionada en ese momento.

La última opción que ofrece el menú "Conexion", es "Select", Figura G.4. Esta opción es muy importante ya que permite seleccionar el equipo sobre el que se realizarán todas las acciones posteriores, por ello es imprescindible seleccionar una conexión antes de realizar cualquiera de las acciones que ofrecen los menús "Filter"→"Addz "Filter"→"Show". Al acceder a este menú se muestra una lista con todas las conexiones existentes y simplemente habrá que seleccionar la de interés y pulsar el botón "Select".

En el menú "Filter"la primera opción que se presenta es "Create", Figuras

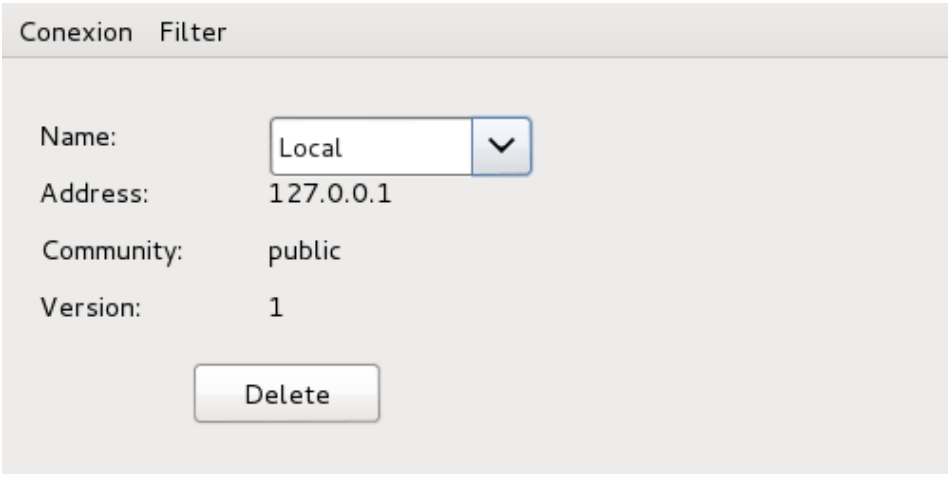

Figura G.3: Menu Conexion→Delete.

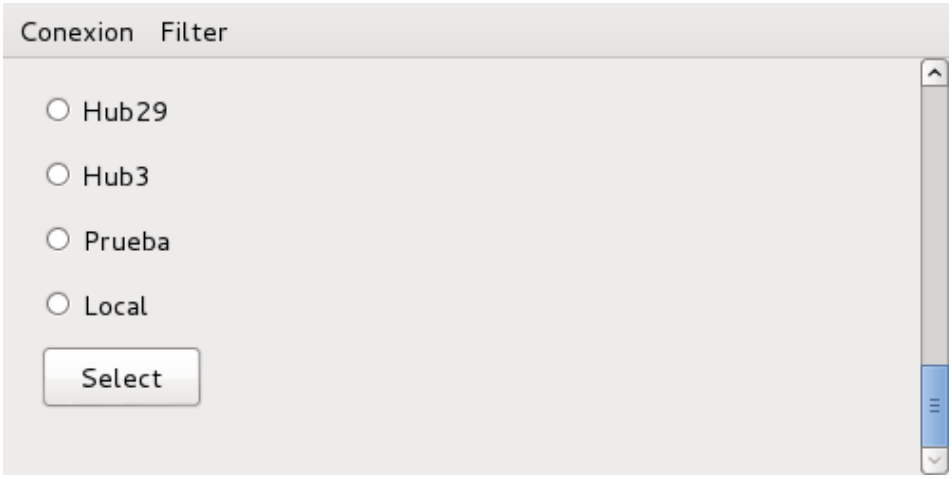

Figura G.4: Menu Conexion→Select.

G.5 y G.6, la cual sirve para definir nuevos filtros. Para definir filtros, los campos "*FilterName*" y "*FilterOwner*" son obligatorios, mientras que en el resto de los campos aunque no pueden dejarse en blanco puede utilizarse el carácter "*"para indicar que cualquier valor de ese campo cumplirá el filtro. Dependiendo de los valores con que se rellenen los campos tipo, se mostraran unos campos u otros en función del protocolo superior seleccionado. Las direcciones MAC han de introducirse con formato xx:xx:xx:xx:xx:xx y las direcciones IP con formato x.x.x.x.

En la opción "Delete"del menú "Filter", Figura G.7, se puede seleccionar uno de los filtros existentes de una lista, se mostrara todo la información relativa a ese

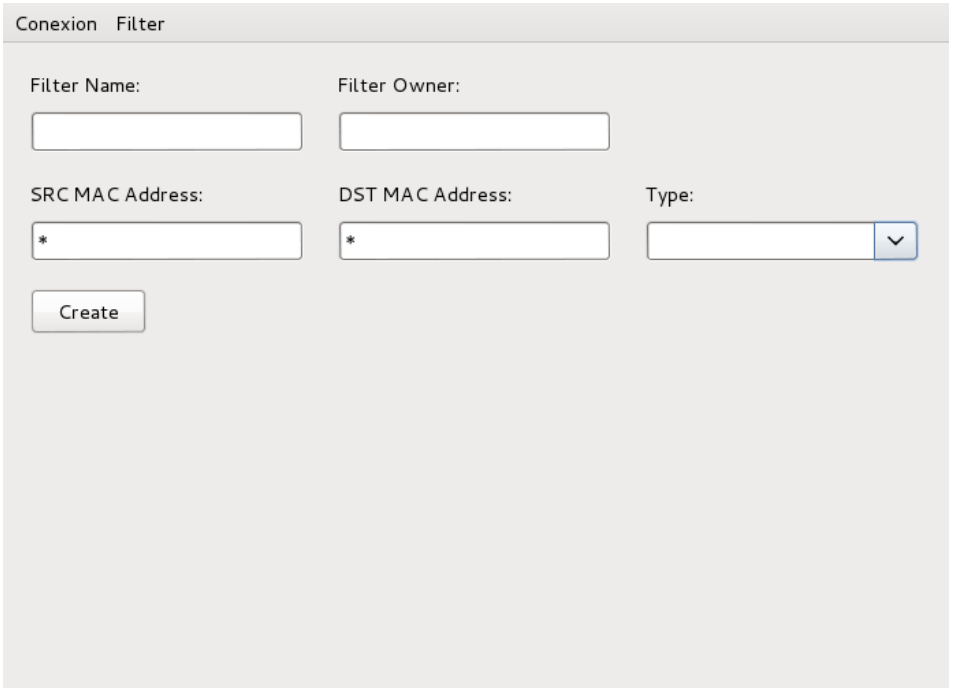

Figura G.5: Menu Filter→Create 1.

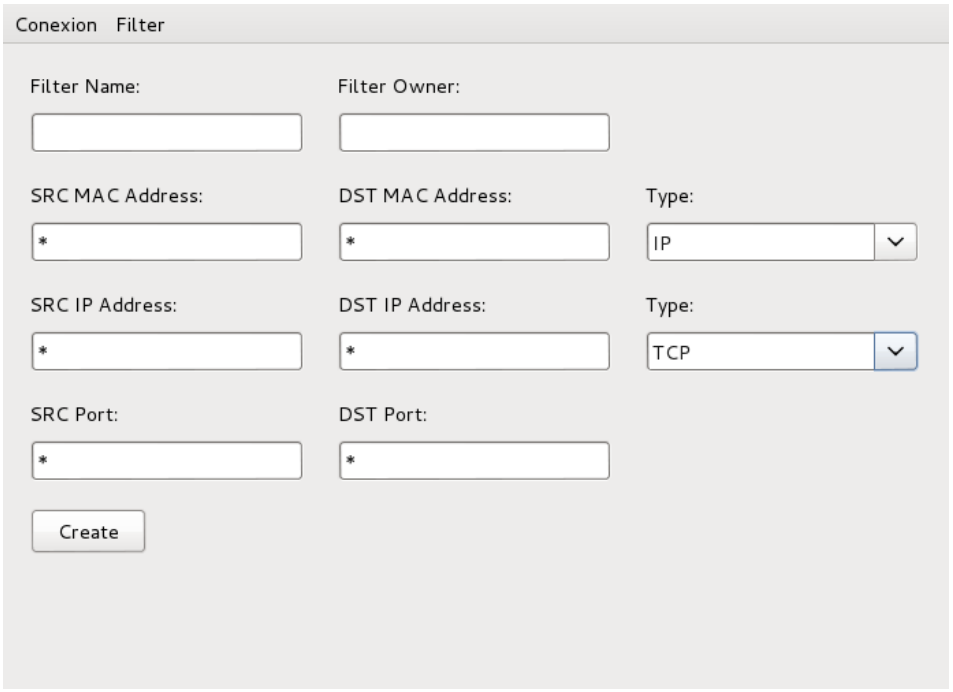

Figura G.6: Menu Filter→Create 2.

filtro, y en caso de pulsar el botón "*Delete*" éste será elimindado.

Como ya se ha comentado anteriormente, para poder utilizar la opción "*Add*", Figura G.8, es necesario haber seleccionado una conexión previamente. Al abrir

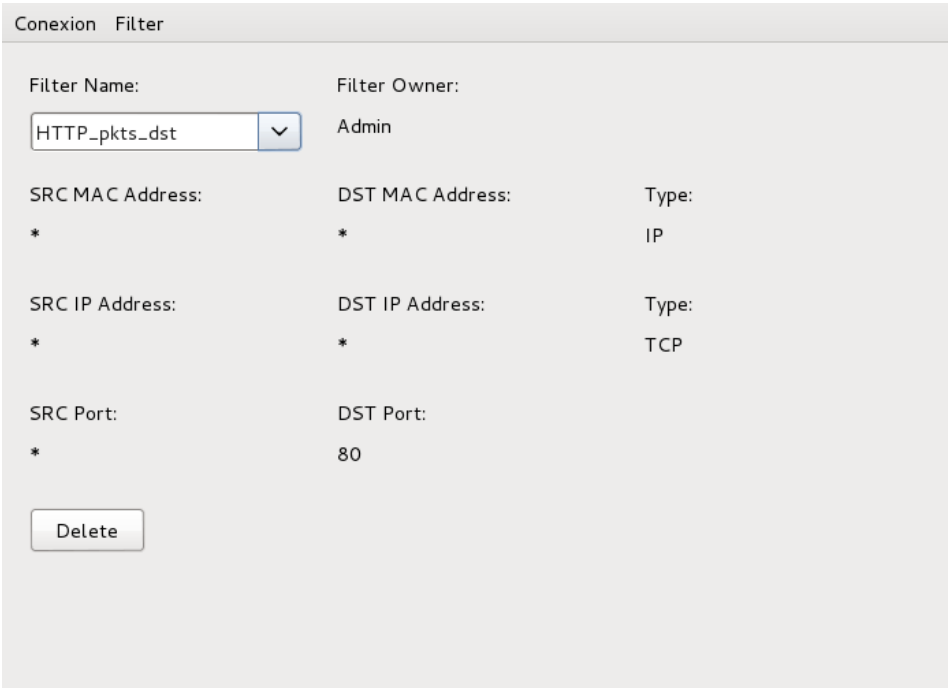

Figura G.7: Menu Filter→Delete.

la opción "Add"se mostrara una lista con checkboxes con todos los filtros que se hayan definido previamente y una lista con todos los interfaces disponibles en el equipo seleccionado. Habrá que seleccionar los filtros que se desean aplicar a dicho equipo, en que interfaz y pulsar el botón "Add".

Por último la opción "Show", Figura G.9, la cual también requiere que haya una conexión seleccionada, muestra una lista con todos los filtros activos en el equipo seleccionado, con su índice, nombre y número de coincidencias. Para eliminar alguno de estos filtros se marcará el checkbox que se encuentra delante de cada índice y se pulsará el botón "Delete".

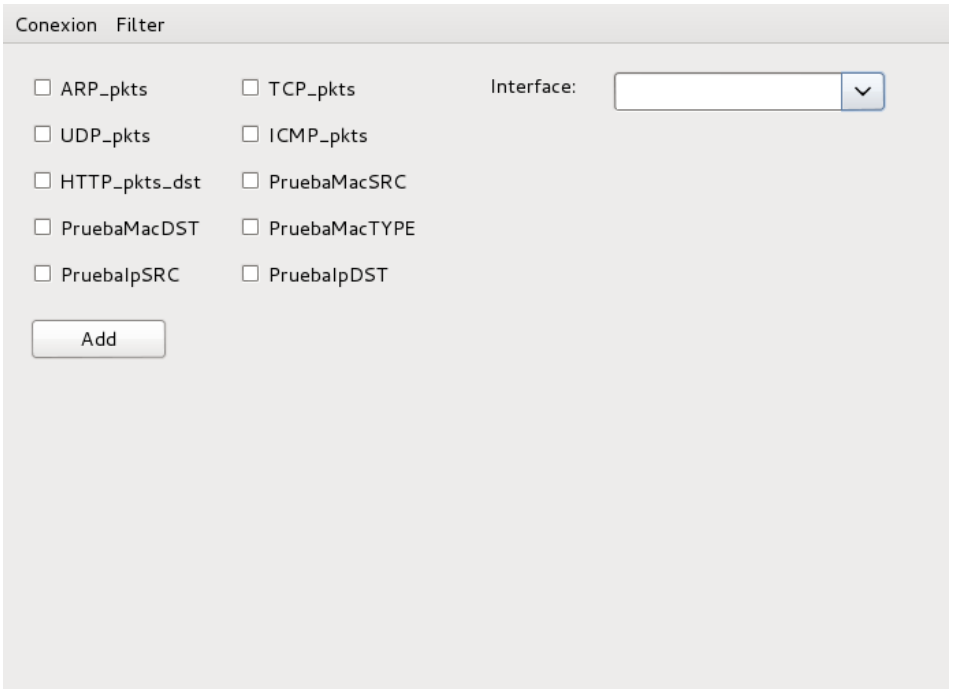

Figura G.8: Menu Filter→Add.

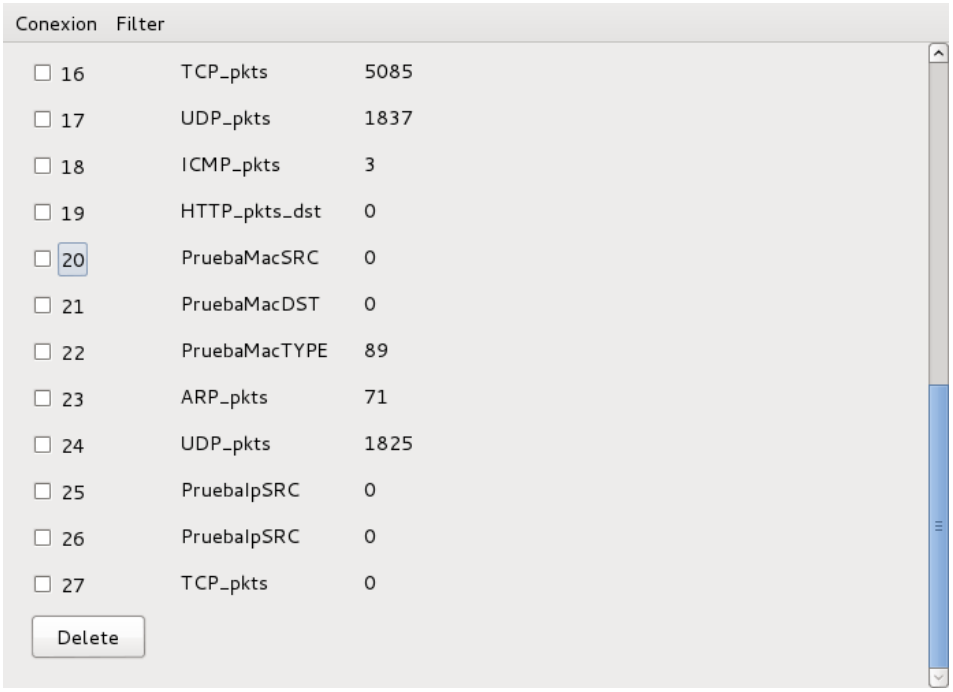

Figura G.9: Menu Filter→Show.

Anexo H

/etc/init.d/rmon

```
### BEGIN INIT INFO
# Provides: rmon
# Required-Start: $all
# Required-Stop: $all
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: Start rmon daemon at boot time
# Description: Enable service provided by rmon daemon
### END INIT INFO

if [ "$1"== "start"]; then
    status=$(ps aux | grep /etc/rmon/ | grep -v grep | awk '{print $2}')
    if [ "$status"! = ""]; then
        echo "rmon already started"
    else
        echo "starting rmon service..."
        python /etc/rmon/start.py &> /dev/null &
    fi
elif [ "$1"== "stop"]; then
    echo "stopping rmon service..."
    filtros=$(ps aux | grep /etc/rmon/ | grep -v grep | awk '{print $2}')
```

```
        IFS=$' '
        for i in $filtros; do a=$i; done
        IFS=$'\n'
        for PID in $a; do kill -9 $PID; done
        unset IFS
    elif [ "$1"== "restart"]; then
        echo "restarting rmon service..."
        filtros=$(ps aux | grep /etc/rmon/ | grep -v grep | awk '{print $2}')
        IFS=$' '
        for i in $filtros; do a=$i; done
        IFS=$'\n'
        for PID in $a; do kill -9 $PID; done
        unset IFS
        python /etc/rmon/start.py &> /dev/null &
    elif [ "$1"== "status"]; then
        status=$(ps aux | grep /etc/rmon/ | grep -v grep | awk '{print $2}')
        if [ "$status"! = ""]; then
            echo "running..."
        else
            echo "stopped"
        fi
    else
        echo "Usage /etc/init.d/rmon start—stop—restart—status"
    fi
```

Anexo I

Guía completa de instalación de la sonda en sistemas Debian

Todos los pasos indicados en esta guía han sido verificados en una maquina virtual generada con VirtualBox, dedicándole al menos 512 MB de memoria RAM y 1.5 GB en un disco duro virtual en formato ".vdi". El sistema operativo utilizado ha sido debían 7.4.0 en su versión para sistemas i386.

Asumiendo que se parte de una instalación limpia de debían, lo primero que habrá que hacer es conseguir los ficheros del programa. Éstos pueden conseguirse de la red del siguiente modo:

- Obtener permisos de súper usuario utilizando el comando *"su"* e introduciendo la contraseña elegida durante la instalación para el usuario root en el momento que se nos requiera.
- Descargar los ficheros utilizando el comando wget y los copiarlos al directorio de trabajo:
  - wget -r http://631500.comlu.com
  - cp -r ./631500.comlu.com/RMON .
  - rm -r 631500.comlu.com
- Moverse al directorio apropiado

  - cd RMON/Configuracion

- Dar permisos de ejecución al fichero de instalación

  - chmod 0777 Configuracion.sh

- Modificar el fichero inittab para permitir el login automático cuando se encienda la máquina:

  - Buscamos la línea *1:2345:respawn:/sbin/getty -autologin USER -noclear 38400 tty1* y modificamos USER por el nombre de usuario creado durante la instalación de debían

- Modificar fichero Configuracion.sh para modificar la contraseña de la base de datos

  - Buscar la línea *mysql -h localhost -u root -p –password=PASS ¡./mysql_config* y cambiar PASS por la contraseña de la base de datos y que se introducirá durante la instalación de la misma.

- Ejecutar el fichero de instalación

  - ./Configuracion.sh

- configuración de la red y del fichero /etc/rmon/rmon.conf.

Por defecto, cuando se encienda la maquina le agente ya se estará ejecutando.Para interactuar con él se pueden utilizar los siguientes comandos:

- /etc/init.d/rmon start
- /etc/init.d/rmon stop
- /etc/init.d/rmon restart
- /etc/init.d/rmon status

Anexo J

Banco de pruebas

```
import os
import subprocess
import sys
import time
for b in range(1,31):
    for n in [5, 10, 15, 20, 25, 30]:
        w = open('resultados.txt', 'a')
        subprocess.call(["python", "borrar_filtros.py"])
        subprocess.call(["python", "rmon_filter_monitoring.py", str(n)])
        aux = subprocess.check_output(["iperf", "-c", "192.168.1.3", "-p",
"5760", "-t", "60", "-u", "-b", str(b) + "m"])
        aux = aux.split(" ")
        sent = int(aux[len(aux)-2])
        time.sleep(10)
        aux = subprocess.check_output(["snmpget", "-v", "1", "-c",
"public", "192.168.1.3", "1.3.6.1.2.1.16.7.2.1.9.1"])
        aux = aux.split(" ")
  recived = int(aux[len(aux)-1])
        loss = round((float(sent - recived) / float(sent)),4) * 100
        w.write(str(loss)+"\n")
```

```
        w.close()
    w = open('resultados.txt', 'a')
    w.write("\n")
    w.close()
```

Donde *rmon_filter_monitoring* es una función creada para cargar en la sonda los filtros correspondientes a trafico ARP, TCP, UDP, ICMP y HTTP, HTTPS, DNS, FTP, SSH, SMTP, POP3, NTP, NETBIOS137, NETBIOS138, NETBIOS139, IMAP, SNMP, RDESKTOP origen y destino. La función *borrar_filtros* sirve para borrar todos estos filtros.