Universidad Zaragoza
1542

Trabajo Fin de Grado

IMPLEMENTACIÓN DEL GRUPO FILTER DE UNA SONDA RMON CON PYTHON Y LIBPCAP

Autor/es

Jorge Sancho Larraz

Director/es

Álvaro Alesanco Iglesias

Escuela de Ingeniería y Arquitectura

2014

Agradecimientos

En primer lugar quería dar las gracias a ***Álvaro****, ya que sin él este proyecto no habría sido posible, y sobretodo agradecerle la confianza que ha depositado en mí en todo momento.*

A mis padres y mi hermana, quienes realmente han hecho posible que haya llegado hasta aquí. Gracias por estar siempre a mi lado, animándome en los malos momentos y aguantándome en aquellos de mayor estrés.

A todos los compañeros de la universidad por hacer que todas esas largas horas en el aula y en los laboratorios, así como las interminables semanas durante el periodo de exámenes en la sala de estudios hayan transcurrido de la forma más amena posible.

A toda la gente de la AATUZ, con quienes tantas horas he pasado este año en el despacho viendo "frikadas" y bebiendo café entre risas. En especial quiero agradecerles el esfuerzo que realizan para organizar las jornadas NEOCOM y los talleres desinteresadamente para que puedan disfrutarlos todo el mucho.

A mis amigos de Tauste, por el interés y apoyo que me habéis mostrado en todo momento y por todos esos momentos que hemos compartido a lo largo de los años.

A todos los miembros de la banda, por soportar mis repetidas ausencias a los ensayos en los periodos de exámenes.

A la charanga, por todos los buenos momentos que compartimos tanto mientras tocamos como estando de fiesta.

En definitiva, a todas las personas importantes para mí, simplemente gracias.

Implementación del grupo Filter de una sonda RMON con Python y LibPCAP

RESUMEN

Actualmente el protocolo más extendido para gestión de red es SNMP (Simple Network Management Protocol). Este protocolo permite realizar labores de monitorización gracias al grupo RMON (Remote Network Monitoring) de la MIB-2 (Management Information Base II).

Durante este proyecto se ha implementado el grupo Filter de RMON. Para ello, en primer lugar se ha desarrollado un agente SNMP con Python, utilizando la API PySNMP, que implementase la MIB de RMON. Para mejorar la usabilidad del agente, se han desarrollado una serie de ficheros que permiten el control del agente del mismo modo que el resto de servicios del sistema en entornos Linux

A continuación, para dar soporte a la MIB-2, necesaria para el correcto funcionamiento del grupo Filter, se ha realizado un modulo para nuestro agente capaz de redireccionar las peticiones correspondientes a OIDs (Object Identifier) de la MIB-2 hacia un agente NetSNMP que estará corriendo en un puerto privado de la misma máquina.

Para llevar a cabo las funciones especificas del grupo Filter, ha sido necesario realizar otro modulo capaz de realizar eficientemente las labores de filtrado. Para este propósito se han desarrollado las funciones necesarias para generar dinámicamente filtros BPF a partir de los datos introducidos por el gestor en el agente. Mediante la librería LibPCAP se han utilizado dichos filtros para realizar el filtrado de paquetes a nivel de Kernel, lo cual es imprescindible para mantener la eficiencia.

También se ha propuesto una forma de realizar la gestión de las comunidades en los agentes a través de mensajes SNMP. Para ello se ha creado una nueva MIB, denominada communityManagement MIB, y se ha desarrollado un modulo que la implemente en nuestro agente.

Por último se ha desarrollado una interfaz gráfica que facilite la realización de las tareas básicas ofrecidas por la sonda desarrollada de forma rápida e intuitiva.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1 Gestión de red: monitorización y control

La gestión de red consiste en monitorizar y controlar los recursos de una red con el fin de evitar que ésta llegue a funcionar incorrectamente degradando sus prestaciones. Las cinco grandes áreas funcionales de la gestión son: rendimiento, fallos, contabilidad, configuración y seguridad. La monitorización es la parte de la gestión encargada de observar y analizar el estado y el comportamiento tanto de las redes como de los equipos que las componen, permitiendo así detectar anomalías y fallos y tomar las medidas oportunas. El control es la parte encargada de modificar parámetros e iniciar acciones en los equipos, normalmente como respuesta frente a algún evento detectado por la monitorización.

SNMP es actualmente el estándar de facto en la gestión de redes TCP/IP, y puede ser utilizado en un amplio espectro de equipos, tales como end systems, switches, routers y equipamiento de telecomunicaciones entre otros. RMON es un grupo de SNMP que extiende su funcionalidad, incluyendo la gestión de redes de área local así como de los equipos conectados a estas redes. Especial atención dentro de RMON requiere el grupo Filter, el cual dota a SNMP de la funcionalidad necesaria para un filtrado eficiente de los paquetes de red, identificando aquellos que cumplen un patrón introducido por el gestor y así poder monitorizarlos.

A día de hoy, SNMP es un estándar ya maduro, sobre el que existen numerosos

estudios y se encuentra implementado en una ingente cantidad de equipos. Sin embargo, a pesar de existir varias implementaciones gratuitas de la MIB-2, las implementaciones de RMON son escasas y muy costosas, por lo que en este proyecto se propone el desarrollo del grupo Filter de una sonda RMON multiplataforma y basada en software libre que posibilite la monitorización de tráfico específico, definido por el gestor de la red mediante filtros.

1.2 Objetivos

El objetivo principal de este proyecto es la implementación del grupo Filter de una sonda RMON, multiplataforma y basada en software libre. Para considerar el desarrollo como exitoso, además se deberan cumplir los siguientes objetivos:

- Integración de la sonda con el sistema operativo, de forma que pueda manejarse del mismo modo que el resto de los servicios del sistema.
- Además de ser funcional, cumpliendo con el estándar SNMP/RMON, debe ser estable a largo plazo.
- La utilización de la sonda no debe degradar el rendimiento general del sistema operatvo, esto implica la optimización del consumo de recursos del sistema.
- La gestión de comunidades de acceso al sistema debe integrarse dentro de la arquitectura de gestión SNMP.

1.3 Materiales y herramientas utilizadas

En cuanto a recursos físicos únicamente fue necesario un ordenador con un adaptador de red Ethernet y el siguiente software, gratuito en su totalidad.

- **Debian:** Sistema operativo sobre el que se ha realizado el desarrollo. Es una distribución de Linux gratuita, fácil de instalar y cuenta con una ingente cantidad de software. Muestra una de las mejores relaciones entre funcionalidad y recursos empleados.

- **Python:** Lenguaje de programación interpretado, multiparadigma y de código abierto. Se ha utilizado el entorno de desarrollo por defecto, IDLE, por venir ya instalado con Python y por contar con una interfaz muy sencilla de utilizar.

  - **PyASN1:** Librería para trabajar con el lenguaje ASN1 desde Python, es necesaria para poder trabajar con el agente SNMP.
  - **PyCryto:** Librería de seguridad para Python, permite utilizar una gran cantidad de funciones criptográficas, tanto algoritmos criptográficos, AES o DES, como funciones de hash, SHA1 o MD5.
  - **PySNMP:** Librería que cuenta con todo lo necesario para trabajar con SNMP desde Python.
  - **MySQLdb:** Librería que permite establecer una conexión con una base de datos MySQL desde Python.
  - **Signal:** Librería para utilizar señales en Python .
  - **PyLibPCAP:** Librería que permite utilizar todas las funciones de LibPCAP desde Python. LibPCAP es una librería open source de filtrado de paquetes.
  - **WxPython:** Librería para la realización de interfaces graficas en Python

- **NetSNMP:** Agente SNMP gratuito que implementa la mayoría de grupos de la MIB-2

- **MySQL:** Motor de bases de datos gratuito.

- **EMMA:** Gestor de bases de datos MySQL que facilita la realización de consultas SQL y la visualización de los datos.

- **Cacti:** Herramienta para la representación grafica de datos. Permite varios modos de adquisición de datos y permite la representación de históricos gracias a la base de datos que utiliza.

- **Iperf:** Herramienta para la monitorización del ancho de banda disponible que permite la generación de trafico tanto TCP como UDP.
- **VirtualBox:** Entorno de virtualización gratuito.

1.4 Organización de la memoria

La memoria está estructurada de la siguiente manera:

- **Capítulo 1: Introducción.** Es el capítulo actual y contiene una breve descripción del trabajo realizado, así como sus principales objetivos.
- **Capítulo 2: La arquitectura SNMP.** En este capítulo se describen las principales características de la arquitectura SNMP, y las mejoras que le aporta RMON.
- **Capítulo 3: Arquitectura y desarrollo del sistema.** En este capítulo se presenta la arquitectura desarrollada así como los diferentes elementos que se han implementado.
- **Capítulo 4: Banco de pruebas.** En este capítulo se detallan las pruebas a las que ha sido sometido el programa y los resultados obtenidos.
- **Capítulo 5: Conclusiones y líneas futuras.** Este es el último capítulo de la memoria y contiene las conclusiones que se han sacado en este proyecto y las posibles líneas futuras que se podrían seguir.

También se han añadido los siguientes anexos:

- **Anexo A. Acrónimos.**
- **Anexo B. Diagrama de flujo de la interacción agente SNMP - MIB.**
- **Anexo C. Proceso de definición de la MIB.**
- **Anexo D. communityManagement MIB.**

- **Anexo E. Guía para la gestión de las comunidades.**
- **Anexo F. Implementaciones desechadas de la funcionalidad de filtrado.**
- **Anexo G. Guía de manejo de la GUI.**
- **Anexo H. /etc/init.d/rmon**
- **Anexo I. Guía completa de instalación de la sonda en sistemas Debian.**
- **Anexo J. Banco de pruebas.**

Capítulo 2

La arquitectura SNMP

2.1 Introducción a SNMP

SNMP es una arquitectura propuesta por el IETF para la gestión y monitorización de red. Dicha arquitectura se ha convertido en el estándar de facto en redes TCP/IP debido a su simplicidad y potencia. SNMP define tanto el protocolo para el intercambio de información de gestión como el formato para la representación de esa información (SMI) y un marco para organizar sistemas distribuidos en gestores y agentes.

La primera versión del protocolo tenía fallos tanto de seguridad como funcionales, tales como la imposibilidad de pedir grandes cantidades de información en un mismo paquete. En la segunda versión se solucionan algunos problemas funcionales, pero la seguridad que implementaba no fue aceptada por fallos en su definición, así que se propuso a una tercera versión en la que se implementa un sistema de seguridad efectivo, manteniendo el funcionamiento de las versiones anteriores.

En la versión 3 del protocolo se sigue manteniendo el payload (campos de datos) de las versiones anteriores del protocolo, pero se añade un sistema de seguridad basado en usuarios (USM, User-based Security Model), que garantiza la confidencialidad, autenticación e integridad gracias al uso de algoritmos de cifrado (DES-CBC o AES de 128 bits) y de Hash (MD5 o SHA-1). Por su parte el sistema

VACM (Viewbased Access Control Model) gestiona a que partes de la MIB, es decir, a que datos, tiene acceso cada usuario.

La arquitectura de SNMP define dos roles básicos, gestor y agente, que interactuan como se muestra en la figura 2.1:

- **Gestor:** es el encargado de pedir información y modificarla según las necesidades de funcionamiento de la máquina en la que reside el agente.
- **Agente:** programa situado en el equipo que se va a monitorizar, sus misiones son recolectar y guardar información local, así como responder ante peticiones del gestor y enviar información de forma asíncrona cuando sucede algún evento.

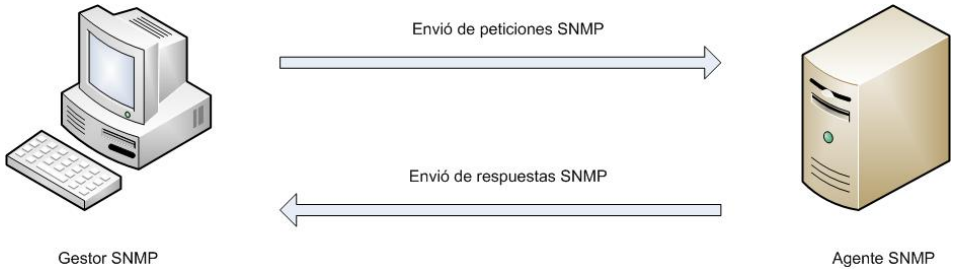

Figura 2.1: Interacción entre gestor y agente.

En SNMP existen varios tipos de mensajes, que pueden clasificarse dependiendo de si es el gestor o el agente el que lo genera. Todos ellos pueden observarse en la figura 2.2.

Mensajes enviados por el gestor al agente:

- **GetRequest:** petición de una o varias variables incluidas en el mensaje de respuesta Response.
- **GetNextRequest:** petición del valor inmediatamente siguiente al indicado en el mensaje, cuya respuesta se incluye también en un mensaje Response.
- **GetBulkRequest:** tipo de petición presente solo a partir de la versión 2. Permite recibir una lista de variables consecutivas para cada variable solicitada. El número de variables consecutivas viene determinado por el valor

max-repetitions. Además permite pedir variables sin repetir, cuyo número viene determinado por el valor non-repeaters. Esto posibilita la petición de una cantidad mayor de información utilizando menor número de mensajes, lo que incrementa la eficiencia.

- **SetRequest:** mensaje que modifica una variable de la base de datos. Si se ha producido algún error durante la realización de la modificación, el agente lo comunicará en el mensaje de respuesta mediante el código de error correspondiente.

Mensajes enviados por el agente al gestor:

- **Response:** mensaje de respuesta para los mensajes enviados por el gestor. En él se incluyen los valores de las variables requeridas, y el código del error en caso de que se produzca alguno.

- **Trap:** mensaje generado y transmitido de forma asíncrona como respuesta a un evento excepcional. En él se incluye el sysUpTime, que es el tiempo que lleva encendido el dispositivo, además de las variables que correspondan al tipo de trap generado.

Mensajes enviados de gestor a gestor:

- **InformationRequest:** mensaje para enviar una alerta de un gestor a otro.

Estos mensajes son codificados con BER (Basic Encoding Rules) y se realizan de forma atómica, es decir, o se completan exitosamente todas las instrucciones de la petición o no se realizará ninguna.

La forma de conocer la información de la que dispone un agente SNMP es a través de las MIBs que implementa. Una MIB es una base de datos modelada en lenguaje SMI (Structure of management Information) y definida por una serie de objetos, el tipo de dato que representa el objeto y un identificador conocido como OID (Object Identifier), el cual está formado por una secuencia de números separados por puntos, representado una estructura jerárquica.

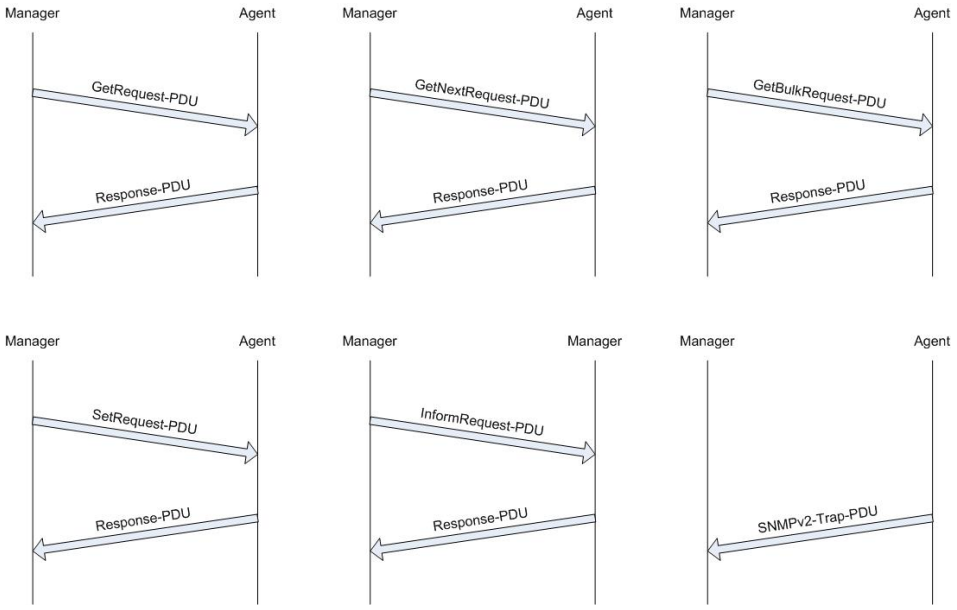

Figura 2.2: Tipos de mensajes definidos en SNMPv2.

Las MIBs se dividen en dos tipos, públicas y privadas. Las públicas están definidas mediante estándares y proporcionan información general del sistema. Las privadas están definidas por los fabricantes y ofrecen información acerca del equipo concreto. La MIB más conocida es la MIB-2, debido a que tiene una amplia información tanto de la red, por ejemplo parámetros estadísticos de tráfico, como de los dispositivos, por ejemplo el uso de CPU o de memoria. La estructura de esta MIB puede apreciarse en la figura 2.3.

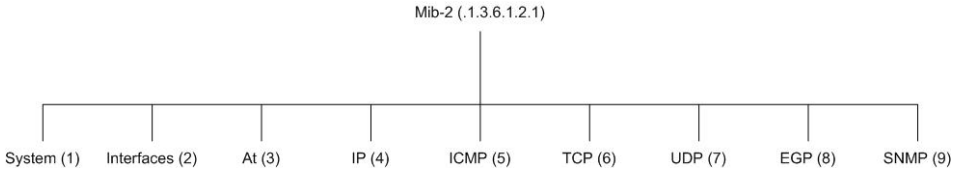

Figura 2.3: Estructura de la MIB-2.

2.2 Introducción a RMON

RMON es un grupo de la MIB-2 formado por una familia de MIBs y extensiones diseñadas por el IETF para dar soporte a la monitorización y análisis de protocolos en redes LAN. La versión original (a veces referida como RMON1, RFC 2819) se

centra en las capas 1 y 2 del modelo OSI en redes ethernet y token ring. Existen numerosas extensiones como RMON2 (RFC 4502) para dar soporte a las capas de red y de aplicación, SMON (RFC 2613) para redes conmutadas, DSMON (RFC 3287) para la monitorización de Servicios diferenciados (Differentiated Services), HCRMON (RFC 3273) para redes de alta capacidad y otras muchas con fines muy diversos.

El caso de mayor éxito ha sido RMON1, ya provee a SNMP de una gran expansión de su funcionalidad, mediante la definición de una MIB, sin realizar ningun cambio en el protocolo. Todo esto ha permitido que RMON haya sido implementado en una gran cantidad de equipos de los principales fabricantes. Dicha MIB está formada por 10 subgrupos, tal y como se muestra en la figura 2.4:

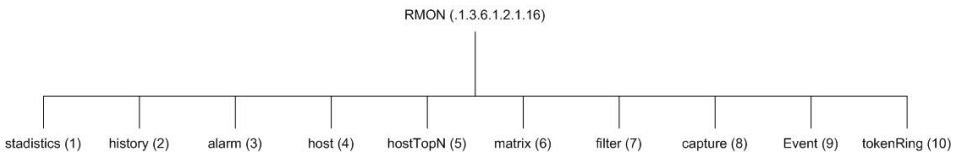

Figura 2.4: Estructura de la MIB de RMON.

- **stadistics:** mantiene estadísticas de bajo nivel sobre la utilización y los errores de cada una de las subredes monitorizadas por el agente.

- **history:** guarda periódicamente muestras estadísticas de la información disponible en el grupo stadistics.

- **alarm:** permite al gestor definir un intervalo de muestreo y un umbral para cualquier contador o entero guardado por la sonda RMON.

- **host:** contiene contadores de varios tipos de trafico con origen o destino a los host conectador a la subred.

- **hostTopN:** contiene almacenadas estadísticas de los host que encabezan una lista basada en algunos parámetros en la tabla de control.

- **matrix:** muestra información acerca de los errores y la utilización en forma matricial, así el gestor puede recibir información acerca de cada pareja de direcciones de red.

- **filter:** permite al gestor observar los paquetes que cumplen un patrón determinado. Los paquetes que cumplen el filtro pueden ser almacenados o simplemente contados.

- **capture:** maneja la forma en que los datos son enviados hasta el gestor.

- **event:** contiene una tabla con todos los eventos generados por la sonda RMON.

- **tokenRing:** mantiene las estadísticas e información de configuración para subredes token ring.

Especial atención merece el grupo Filter, formado por dos tablas, channelTable y filterTable. Cada una de las filas de la tabla channelTable define un canal único. Asociado a cada canal hay una o más filas de la tabla filterTable, las cuales definirán los filtros asociados. Cada vez que un paquete que cumpla los filtros llegue a un interfaz de la sonda se incrementara un contador en la fila correspondiente de la tabla channelTable. También es posible gracias al grupo capture que en el caso de un paquete atraviese el canal exitosamente, dicho paquete sea almacenado en un buffer de la sonda, y quede disponible para que el gestor pueda descargarlo para su posterior análisis.

Capítulo 3

Arquitectura y desarrollo del sistema

3.1 Arquitectura del sistema

Dado que se necesitaba control total sobre el agente, no era suficiente con extender un agente sino que era necesario desarrollar uno propio. Para realizar esta tarea existen varias APIs que evitan la programación de algunos aspectos de bajo nivel, como la codificación/decodificación con BER o conocer exactamente que bits componen cada uno de los campos del paquete. A pesar de existir APIs para varios lenguajes de programación, se decidió realizar el desarrollo en Python por ser un lenguaje de fácil aprendizaje y rápido desarrollo, soportar la programación orientada a objetos y contar con una ingente cantidad de librerías.

La arquitectura que se ha decidido desarrollar es la mostrada en la figura 3.1. En ella se pueden ver todos los módulos desarrollados, que serán explicados posteriormente con mayor grado de detalle.

- **Dispatcher:** Elemento encargado de iniciar el sistema, tras lo cual se ocupa de recibir paquetes SNMP de la red y enviar las respuestas, extraer y analizar los campos de cada paquete, y en función del objeto sobre el que se pregunte delegar en el modulo correspondiente.
- **Security:** Elemento que verifica que una petición tiene los permisos

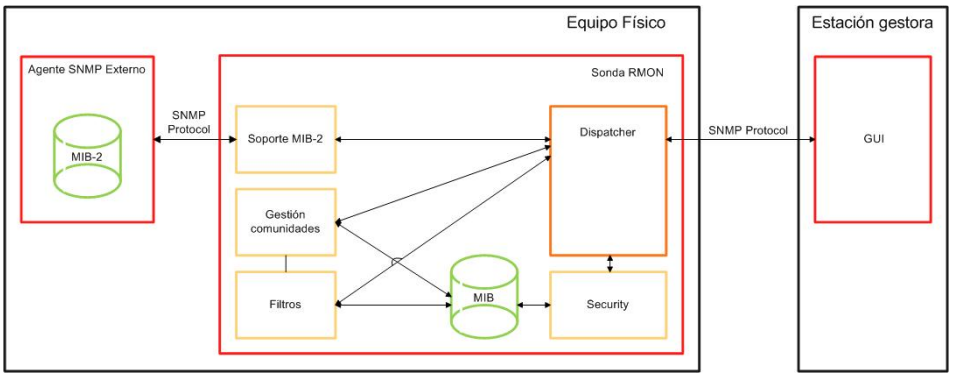

Figura 3.1: Arquitectura del sistema.

necesarios para realizarse.

- **MIB:** Elemento encargado de almacenar la información tanto introducida por el gestor como recogida por el agente. Este elemento almacena la información referente tanto a la gestión de las comunidades como la de RMON.

- **Soporte MIB-2:** Elemento permite utilizar la funcionalidad de un agente externo corriendo en el mismo equipo de forma transparente al usuario, como si se tratase de un único agente.

- **Gestión Comunidades:** Elemento que nos permite añadir y eliminar comunidades SNMP, así como editar los privilegios de aquellas ya existentes, mediante la interacción con una MIB definida para tal fin.

- **Filtros:** Elemento que interactua con la implementación del grupo Filter de RMON realizada en este trabajo, y genera los filtros necesarios.

- **GUI:** Elemento que facilita la interacción con la sonda RMON posibilitando la configuración de la misma de forma gráfica.

3.1.1 Dispatcher

El Dispatcher es el primer elemento en ser lanzado, por lo que es el encargado de iniciar el agente. En primer lugar lee un fichero de configuración, /etc/rmon/rmon.conf, en el cual el usuario puede añadir varios aspectos de la

configuración del agente sin necesidad de modificar el código. Con la información recuperada de dicho fichero el Dispatcher es capaz de establecer esta configuración en el Sistema Operativo.

Una vez el sistema está completamente iniciado, el Dispatcher pasa a su funcionamiento normal quedando encargado de recibir las peticiones SNMP. Para cada una de las peticiones extrae la versión del protocolo utilizada, la comunidad, el tipo de petición, el ID para añadirlo al paquete de respuesta (este campo permite al gestor saber a qué petición corresponde cada respuesta), los campos non-repeaters y max-repetitions si procede, y los varBinds. Un varBind es una asociación entre un OID y un valor. Este valor únicamente es relevante en las peticiones de tipo "Set". Para cada uno de los varBinds, en primer lugar, el dispatcher invoca al módulo de seguridad pasando como parámetros la comunidad, el OID y el tipo de petición. Este módulo devuelve un 1 en caso de tener los permisos adecuados y un 0 en caso contrario.

En caso de no tener permisos el dispatcher ejecutara la función de rollback cuya finalidad se explicara más adelante y se añadirán los campos de error correspondientes al paquete de respuesta. Si los permisos son los necesarios, el dispatcher invocara a una función de backup, que será explicada junto a la de rollback nombrada anteriormente, decidirá a que modulo tiene que invocar (Soporte MIB-2, Gestión comunidades o Filtros) en función del OID, y lo hará pasandole como parámetros el tipo de petición, el OID y el valor (este último únicamente para las peticiones de tipo "Set"). Este módulo devolverá el OID, el valor que se encuentre almacenado en la MIB y un campo status que indicará si todo se ha realizado correctamente y en caso contrario el tipo de error que ha ocurrido.

En el caso de que haya producido algún error, al igual que en el caso de no tener los permisos necesarios, se ejecutara la función de rollback y se añadirán los campos de error correspondientes al paquete de respuesta en función de lo indicado en el campo status. Si todo el proceso se ha completado satisfactoriamente el dispatcher añade los campos OID y valor a un varBind del paquete de respuesta con el formato apropiado y pasa a procesar el siguiente varBind en caso de que exista. Una vez

se han acabado de procesar todos los varBinds se procede a enviar el paquete de respuesta.

La finalidad de las funciones de backup y rollback nombradas anteriormente es cumplir la especificación del estándar que indica que todas las instrucciones dentro de un mismo mensaje deben ser ejecutadas atómicamente, es decir, o se ejecutan todas correctamente o no se ejecutara ninguna. Como existen algunas instrucciones para las cuales el éxito de su ejecución depende de los valores que toman otros campos, que pueden ser modificados en el mismo mensaje, no podremos saber si una instrucción es posible ejecutarla hasta que lo intentemos junto con todas las anteriores. Para solucionar este problema la función de backup es invocada cada vez que llega un paquete de tipo set y se encarga de almacenar el estado de la MIB antes de realizar modificación alguna. La función de rollback es invocada cada vez que se produce algún error mientras se lleva a cabo alguna de las tareas relacionadas con dicho paquete y se encarga de devolver la MIB al estado salvado por la función de backup.

3.1.2 MIB

La MIB es el elemento que almacenará tanto los datos introducidos por el gestor como la información recogida por el agente. Dado que una MIB esta modelada en SMIv2, el cual es un lenguaje abstracto que no define el tipo de almacenamiento sino la estructura de la información, existen diversas posibilidades a la hora de su implementación en un dispositivo físico. Habrá que recurrir a un medio de almacenamiento no volátil de la información debido a que en el caso de reiniciar el agente no se debe perder la información almacenada. Se decidió utilizar una base de datos MySQL por adaptarse perfectamente a las especificaciones del proyecto (gratuita y multiplataforma) y existir una librería para trabajar con ella desde Python (PyMySQL).

La forma de organizar la base de datos se ha basado en la estructura propuesta en el Proyecto Fin de Carrera, “Implementación automática de un agente SNMP a partir de la definición formal de su MIB”[1], para mantener la coherencia con la

forma de trabajo de proyectos anteriores y la compatibilidad con dichos proyectos. Esta estructura se basa en la utilización de tres tipos de tablas, tablas secundarias (ts_), tablas de control (tc_) y tablas de datos (td_). Las tablas secundarias son indexadas por el siguiente fragmento del OID, y contiene el nombre de la siguiente tabla en la que se buscará, permitiendo recorrer de forma jerárquica la estructura que definen los OIDs. Las tablas de control están formadas por metadatos e incluyen toda la información contenida en la MIB. Las tablas de datos son las que contienen los valores de la instancia del objeto, y en caso de representar a una tabla de SNMP los índices de la tabla de datos coincidirán con los de la tabla SNMP.

Esta estructura de tablas se ha utilizado para representar tanto la MIB "*communityManagement*", la cual se explicará en el punto 3.1.5, como la MIB del grupo Filter de RMON.

3.1.3 Security

La forma de gestionar los privilegios en las dos primeras versiones del protocolo es a través de las comunidades. Cada comunidad puede tener una o varias vistas asociadas, donde cada vista es una dupla compuesta por un OID y el nivel de acceso sobre esa parte de la MIB, siendo posibles cuatro niveles de acceso: Sin permisos, Solo lectura, Solo escritura y Lectura-Escritura. La vista tiene efecto tanto sobre el OID indicado como sobre todos sus descendientes.

Para verificar que una petición tiene los privilegios necesarios, se ha implementado un módulo el cual es invocado por el Dispatcher cada vez que recibe una petición. Dicho módulo obtiene como parámetros la comunidad, el OID y el tipo de petición, y recorre todas las vistas asociadas a la comunidad en busca del máximo nivel de acceso correspondiente a ese OID. Se asigna como máximo nivel de acceso el asociado a aquella vista cuyo OID sea el ascendente más cercano al OID de la petición. En caso de no existir ninguna vista aplicable, el nivel de acceso será "*Sinpermisos*". A continuación comprueba que el nivel de acceso obtenido sea mayor o igual al requerido por el tipo de petición, siendo de lectura para todas

la peticiones excepto para "*Set*" en cuyo caso es de escritura. Si se cumple la condición anterior devuelve un 1 y en caso contrario un 0.

3.1.4 Soporte de la MIB-2

Para un correcto funcionamiento del grupo Filter es obligaría la presencia de algunos grupos de la MIB-2, como el grupo "interfaces", el cual asigna un identificador numérico a cada uno de los interfaces de red del equipo. Dichos identificadores son necesarios para seleccionar el interfaz en el que se desea configurar cada uno de los canales (conjunto de filtros).

Para dar soporte a la MIB-2 se ha desarrollado un módulo que es invocado por el Dispathcer cada vez que recibe una petición con un OID correspondiente a dicha MIB. Este módulo recibirá como parámetros el tipo de petición, OID y valor, y con ellos generará una nueva petición SNMP que enviará a un agente NetSNMP (agente SNMP gratuito que implementa la MIB-2 completa) el cual estará instalado y esperando peticiones en uno de los puertos privados del propio equipo. Tras enviar la petición quedara a la espera de la respuesta y una vez la haya recibido la procesará. En este procesado se analizará la posible existencia de errores y se le asignará el valor correspondiente al campo status. A continuación, en caso de no existir errores se extraerán los campos OID y valor. Una vez finalizada su ejecución, este modulo devolverá al Dispatcher los tres campos mencionados anteriormente.

A pesar de existir la opción de implementar los grupos necesarios como un módulo de la sonda, se decidió utilizar este otro método debido a que permitía dar soporte a la MIB-2 completa y a que recoger parte de la información que contiene la MIB-2 requiere de una delicada interacción con el núcleo del sistema que puede afectar seriamente a la estabilidad y al rendimiento.

3.1.5 Gestión de las comunidades

Este módulo es el encargado de interactuar con la base de datos que implementa la MIB "*communityManagement*". Cada vez que llega una petición relacionada

con dicha MIB este módulo es lanzado recibiendo como parámetros el tipo de petición, el OID y el valor. Con estos parámetros recorre la estructura de tablas siguiendo el esquema presentado en el anexo B hasta obtener el valor deseado. Si todo se ha realizado satisfactoriamente, asigna un 0 al campo status y devuelve los valores recogidos de la base de datos al Dispatcher. En caso de haberse producido algún error devuelve status igual a 1 e interrumpe la ejecución.

La MIB communityManagement se ha definido en este proyecto para permitir integrar la gestión de las comunidades de acceso al sistema dentro de la arquitectura SNMP, ya que a pesar de que el uso de comunidades es común a todos los agentes, la forma de gestionarlas (crearlas, modificar las vistas asociad, etc.) no es estándar. En la mayor parte de los agentes esta gestión se realizar bien a través de un fichero de configuración, como es el caso de NetSNMP, o bien mediante una interfaz de comandos, como sucede en una gran cantidad de equipos comerciales.

La estructura de esta MIB puede apreciarse en la figura 3.2, y el proceso de definición de la MIB, la definición formal de la misma y una de uso se encuentran adjuntas en los anexos D, E y F.

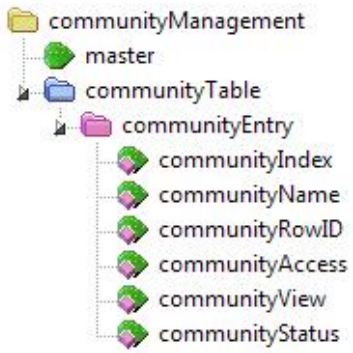

Figura 3.2: Estructura de la MIB communityManagement.

- **master:** Comunidad con permisos de lectura/escritura sobre la tabla commuintyTable.
- **communityIndex:** Índice primario de la tabla. Es la representación ASCII de cada uno de los carácteres que forman el nombre de la comunidad separados por puntos.

- **communityName:** El nombre de la comunidad.
- **communityRowID:** Índice secundario de la tabla. Identificador numérico que permite identificar las diferentes vistas pertenecientes a una misma comunidad.
- **communiteAcces:** Permisos asociados a cada una de las vistas.
- **communityView:** OID raid sobre el que se aplica la vista.
- **communityStatus:** Campo del tipo "EntryStatus"que permite añadir filas a la tabla, borrarlas, modificarlas, activarlas y desactivarlas.

3.1.6 Filtros

Del mismo modo que el módulo "Gestión de las comunidades"se encarga de interactuar con la base de datos que modela la MIB "communityManagement", el módulo "*Filtros*" interactuará con la base de datos que implementa la MIB del grupo Filter de RMON.

Además, deberá realizar las funciones de filtrado que caracterizan a RMON, para lo que se utilizará la librería LibPCAP, la cual permite realizar la captura y filtrado de paquetes de red. Los programas basados en dicha librería se ejecutan en la zona de usuario, pero la captura de paquetes se realiza en la zona del Kernel, por lo que es necesaria la transferencia de datos desde el kernel-space hacia el user-space. Esto conlleva una serie de operaciones muy costosas y que es necesario minimizar para evitar degradar el rendimiento del sistema.

Se han testeado varios esquemas de trabajo pero en este capítulo únicamente se explica el utilizado en la versión definitiva de la sonda. El resto de las implementaciones se encuentran comentadas en el anexo F.

Cuando un channelStatus toma el valor "*valid*", el módulo "*Filtros*" crea un nuevo proceso que quedará al cargo de ese canal. Este proceso correrá un programa que recibe como parámetro el índice del canal en cuestión, y con él recupera de la base de datos toda la información relacionada con dicho canal. A partir de esa información el programa generará un filtro BPF que cumpla con las

especificaciones configuradas por el gestor. Estos filtros BPF se ejecutan a nivel de Kernel y únicamente pasan a nivel de usuario aquellos paquetes que han superado el filtro, como puede apreciarse en la figura 3.3.

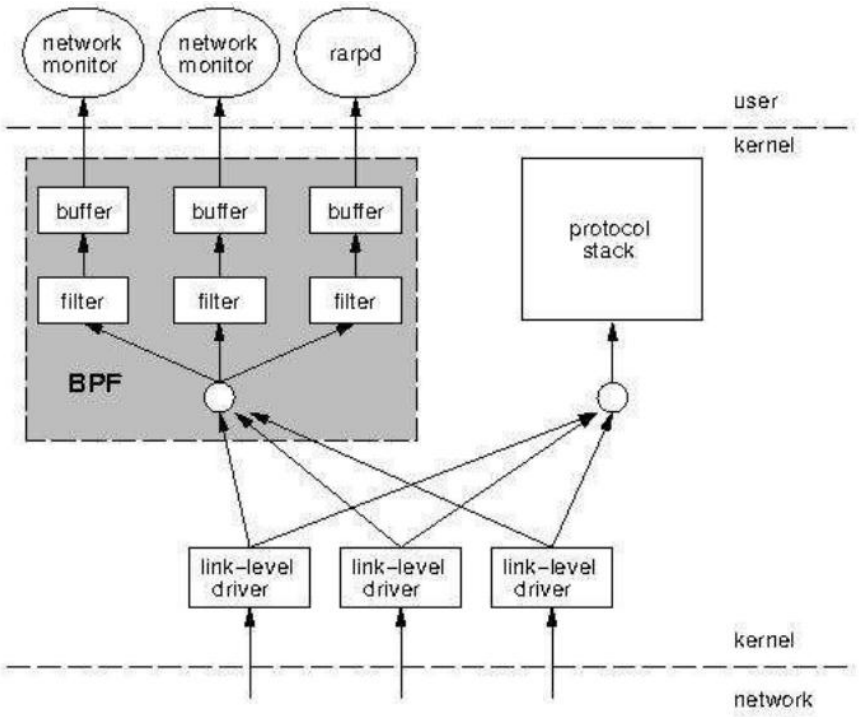

Figura 3.3: Funcionamiento de los filtros BPF.

Para configurar un filtro de este tipo en Kernel se utiliza la librería LibPCAP, y cada vez que un paquete supera el filtro se ejecuta una función de callback. Dado que el proceso del agente es el encargado de crear los procesos de filtrado, estos pueden compartir una zona de memoria que les servirá para comunicarse, por lo que en la función de callback anterior simplemente se incrementará el contador situado en la zona de memoria asignada para ese filtro. El esquema de trabajo es el mostrado en la figura 3.4

La actualización de la base de datos la realizará el proceso del agente, el cual dispone de toda la información necesaria en la zona de memoria compartida. Esta actualización la realizará periódicamente en el Callback de una señal de alarma configurada al inicio el agente. Como la memoria compartida ha de reservarse al inicio de la ejecución, el número de filtros estará limitado por la cantidad de memoria reservada. Además será necesario mantener una lista de los filtros activos

y en qué posición de memoria se encuentran.

3.1.7 Desarrollo de la GUI

Se ha desarrollado una aplicación que mediante una interfaz gráfica facilite el desarrollo de las tareas más habituales realizadas con RMON. Para ello se ha utilizado la librería WxPython, la cual está basada en WxWidgets (una librería multiplataforma C/C++). Esta librería es muy rápida, soporta una gran cantidad de elementos multimedia e interactivos, cuenta con contenedores nativos en todas las plataformas y permite separar completamente el diseño de la interfaz del código Python.

La aplicación permite mantener una lista de conexiones con los equipos gestionados, permitiendo añadir conexiones a la lista así como editar y eliminar las ya existentes. Además cuenta con una opción para seleccionar la conexión sobre la que realizarán las acciones posteriores.

Para realizar las tareas relacionadas con el grupo Filter cuenta con dos apartados, el primero permite definir y eliminar modelos de filtro, y la segunda parte referida al equipo al que estamos conectados que permite ver los filtros que tiene establecidos, eliminarlos o añadir uno nuevo de los modelados previamente.

La apariencia de la aplicación puede observarse en la figura 3.4 y la guía de manejo se encuentra en el anexo G.

3.2 Integración con el sistema operativo

El agente desarrollado es completamente multiplataforma, pero para mejorar la experiencia de usuario y facilitar la interacción con él, se han creado una serie de scripts en Bash para sistemas Debian, que permitan manipular la sonda de la misma manera que el resto de servicios del sistema operativo, utilizando las siguientes instrucciones desde una terminal:

- **/etc/init.d/rmon start** Permite iniciar el agente. En caso de que ya estuviese iniciado unicamente informa de ello.

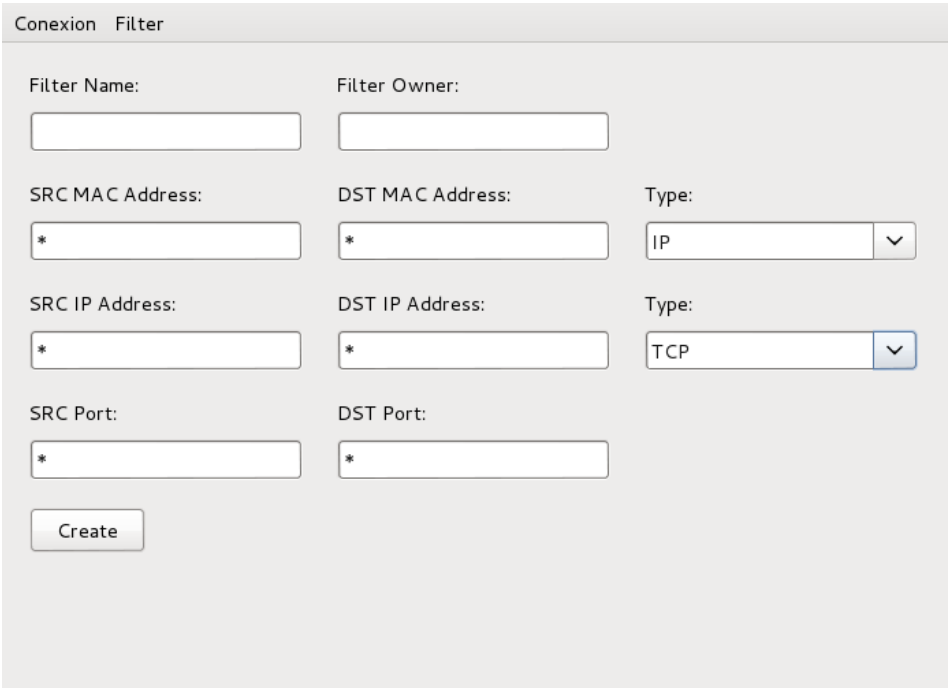

Figura 3.4: Interfaz gráfica de usuario.

- **/etc/init.d/rmon stop** Detiene el agente por completo. En el caso de que quedase algún proceso de una ejecución anterior no finalizada correctamente también lo terminará.

- **/etc/init.d/rmon restart** Concatenación de los dos comandos anteriores, en el orden stop-start.

- **/etc/init.d/rmon status** Informa del estado de la sonda, siendo estados posibles running y stopped.

Además, para que el script pueda utilizarse para lanzar el agente durante el arranque de Debian, éste debe responder al formato de un LSB init script, incluyendo una cabecera en la cual se indica el nombre del script, los servicios que han de iniciarse antes de lanzar el propio script y los que deberán de pararse después, los runlevels para los cuales se iniciara o se detendrá el servicio, y una descripción de la funcionalidad del script.

Este script puede verse en el anexo G.

3.3 Instalación

Se ha creado una maquina virtual utilizando VirtualBox con una sonda funcional ya instalada y configurada. En el caso de decir realizar una nueva instalación se han creado una serie de script para facilitar la instalación en sistemas Debian, y una guía de instalación que puede encontrarse en el anexo I. Es muy importante tener en cuenta que si se decide instalar la sonda en equipos con aplicaciones que utilicen bases de datos MySQL, éstas pueden dejar de funcionar debido a la modificación en la configuración de dicha base de datos.

Capítulo 4

Banco de pruebas

4.1 Perdida de paquetes.

Este parámetro indica el porcentaje de paquetes que no son contabilizados por la herramienta aun habiendo superado los filtros con éxito. Estas pérdidas pueden tener dos orígenes, pequeñas pérdidas debidas a una saturación puntual de los buffers en periodos que el sistema utilice todos los recursos para realizar tareas criticas, y grandes pérdidas debidas a que la velocidad del tráfico es mayor que la que el equipo puede procesar.

Para esta tarea se utilizó la herramienta IPERF trabajando con trafico UDP ya que permite generar tráfico a velocidades determinadas. Para evitar que el trabajo de generar paquetes degradase las prestaciones del programa de filtrado, tanto el cliente como el servidor de IPERF se alojaron en equipos distintos al que corría la sonda, tal y como se aprecia en la figura 4.1.

Para automatizar el proceso de toma de medidas, se preparó un banco de pruebas, cuyo fichero principal puede observarse en el anexo J. Se realizaron medidas para un número máximo de filtros igual a 30, cada uno de los cuales monitorizaba un tipo distinto de tráfico y se encuentran enumerados en el anexo J, y una velocidad de trafico de hasta 100 Mbps generados por IPERF, obteniendo en todos los casos que no se perdía ningún paquete.

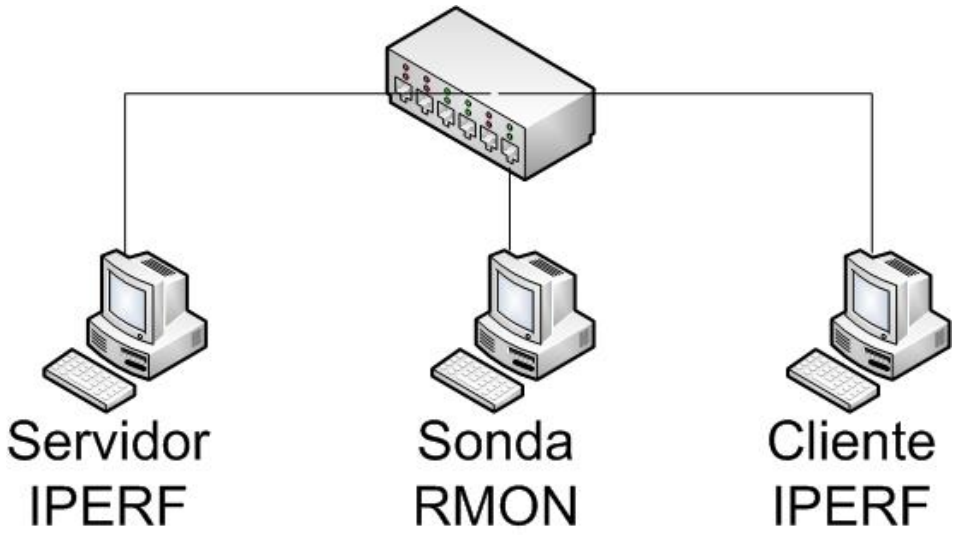

Figura 4.1: Escenario para calcular la perdida de paquetes.

4.2 Funcionamiento y estabilidad.

La estabilidad de la sonda es fundamental para garantizar una correcta y continua monitorización de la red. Para medir la estabilidad de la sonda, se instaló en un router para medir los flujos de paquetes correspondientes a 32 filtros con un tráfico medio de entrada de 300Kbps y de salida de 200Kbps, con picos de hasta 8Mbps. Para poder monitorizar los filtros se utilizo cacti como sistema de monitorización. Cacti es un gestor SNMP basado en un servidor web que puede ofrecer la monitorización de las variables SNMP deseadas de un agente remoto a través de una web. Se configuró cacti para que recogiera los datos de los filtros de la sonda cada 5 minutos y los representase gráficamente, con lo que un fallo de estabilidad podría ser identificado con una discontinuidad en las gráficas.

La sonda permaneció en funcionamiento ininterrumpido durante 4 meses, periodo tras el cual fue apagada por considerarlo suficiente. En la figura 4.2 podemos observar el tráfico ARP existente durante ese periodo.

Para verificar el correcto funcionamiento de la sonda se analizó la coherencia de los patrones obtenidos de la sonda con lo esperado en una red de ese tipo. No se encontró ningún tipo de anomalía, ya que todos los patrones eran coherentes con lo esperado. En las figuras 4.3, 4.4, 4.5 y 4.6 podemos observar algunas de las gráficas generadas por Cacti con los datos recogidos por la sonda durante un día.

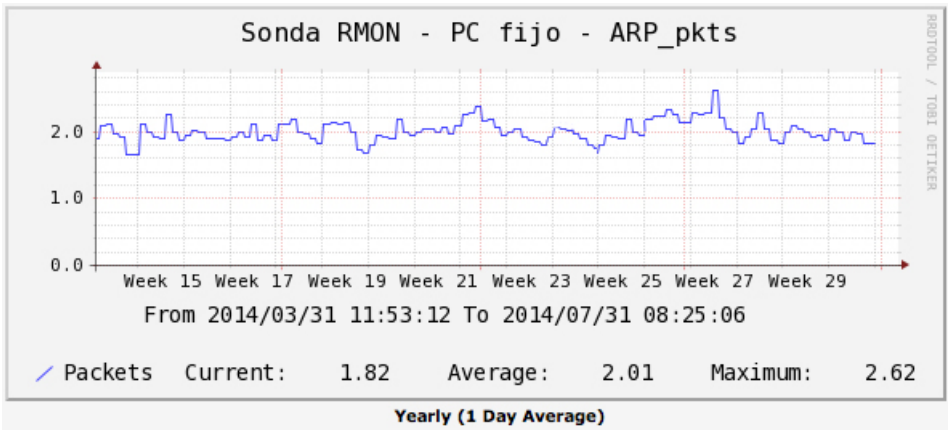

Figura 4.2: Tráfico ARP entre el 31/03/2014 y el 31/07/2014.

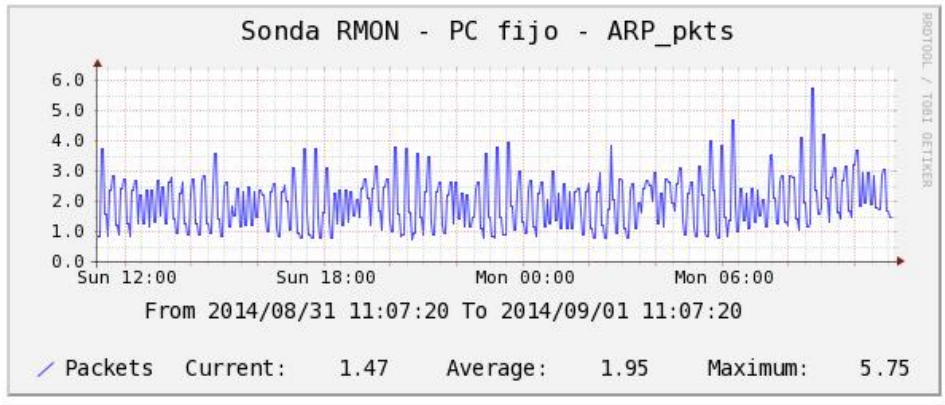

Figura 4.3: Tráfico ARP medido por la sonda.

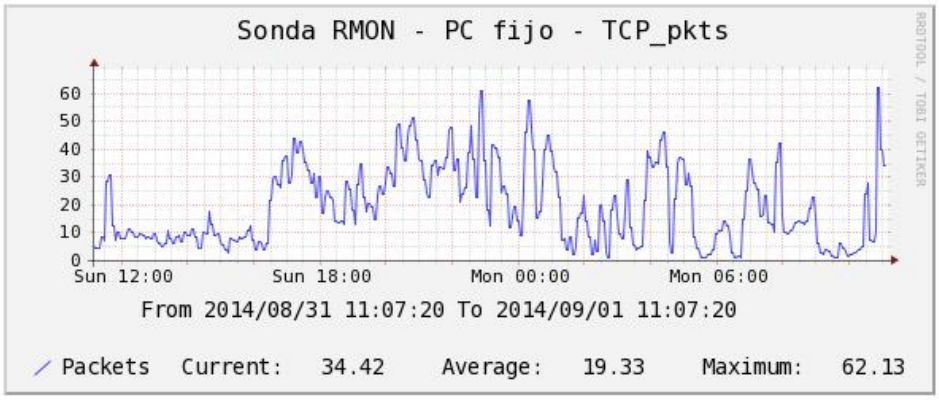

Figura 4.4: Tráfico TCP medido por la sonda.

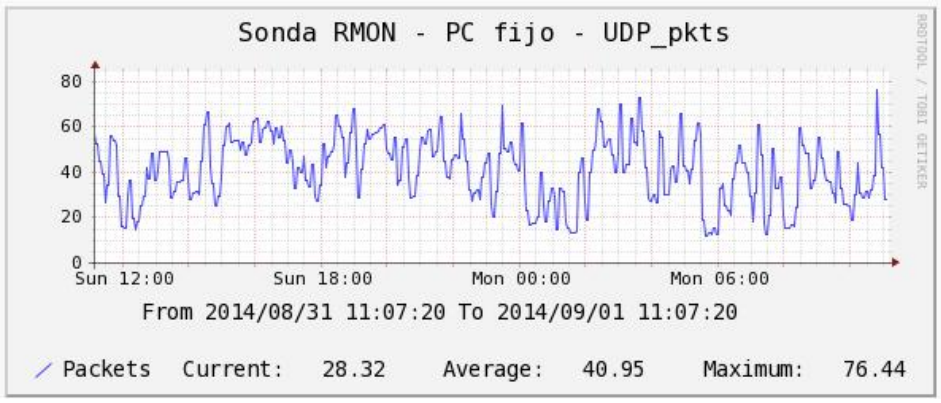

Figura 4.5: Tráfico UDP medido por la sonda.

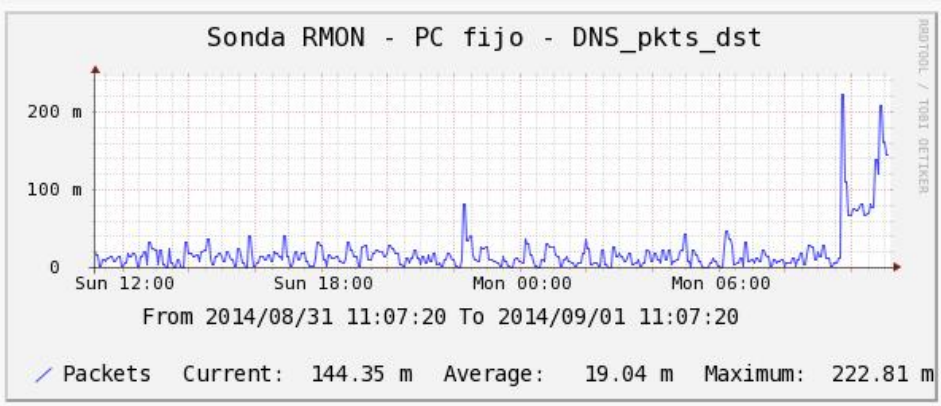

Figura 4.6: Tráfico DNS query medido por la sonda.

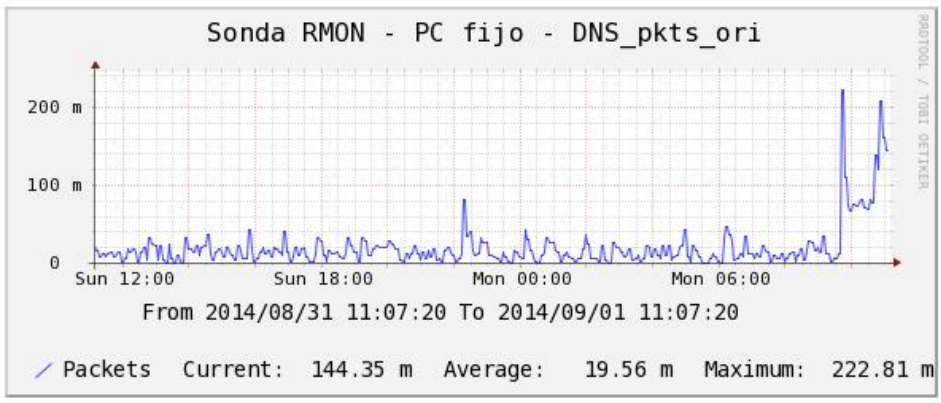

Figura 4.7: Tráfico DNS response medido por la sonda.

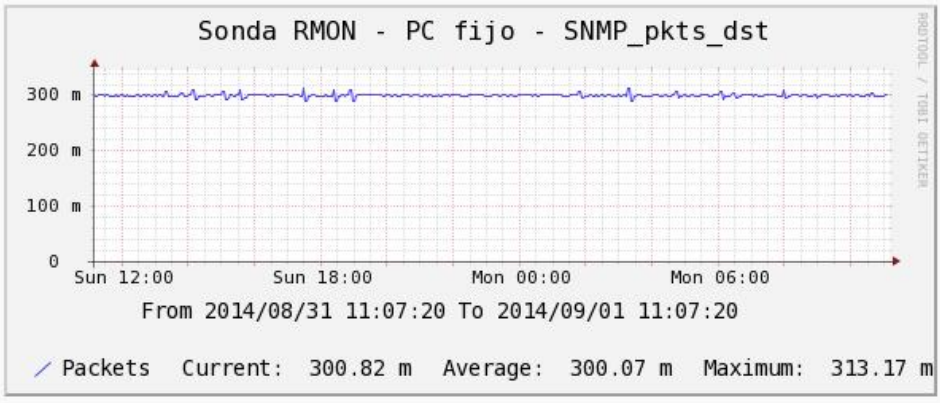

Figura 4.8: Tráfico SNMP request medido por la sonda.

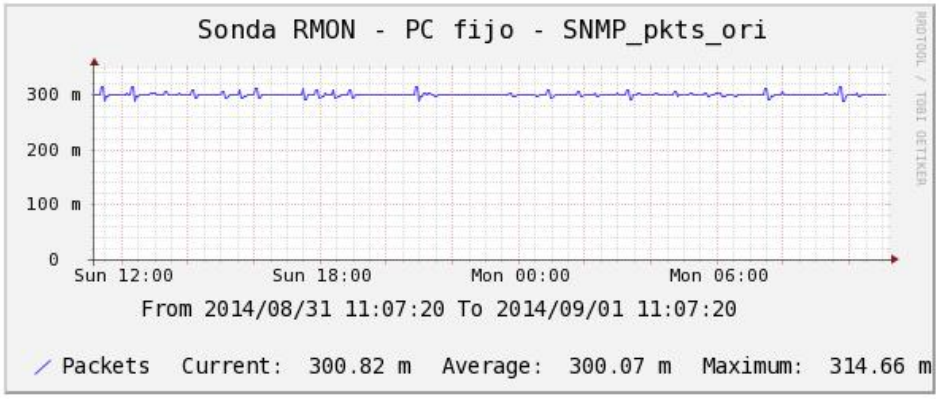

Figura 4.9: Tráfico SNMP response medido por la sonda.

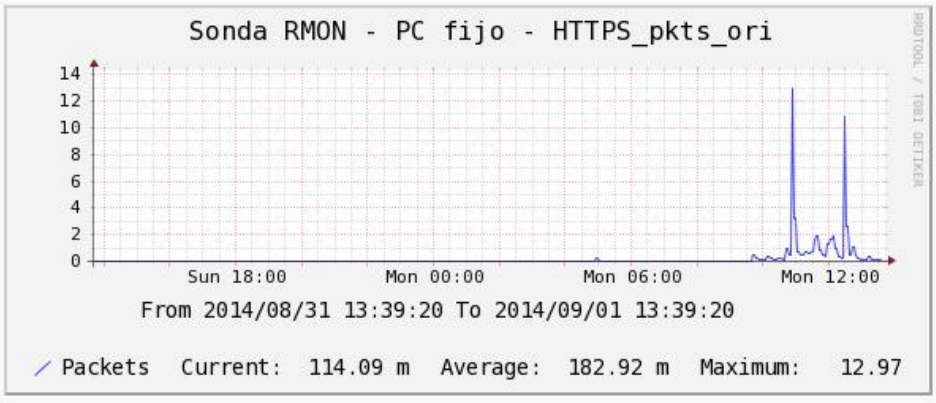

Figura 4.10: Tráfico HTTPS response origen medido por la sonda.

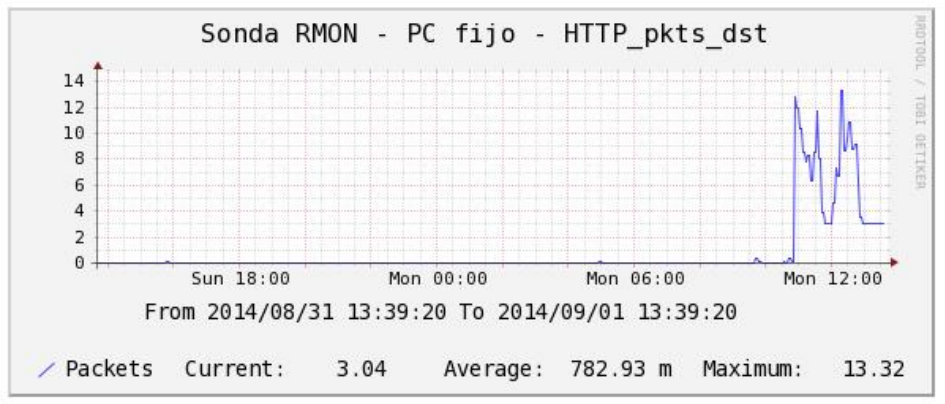

Figura 4.11: Tráfico HTTP request origen medido por la sonda.

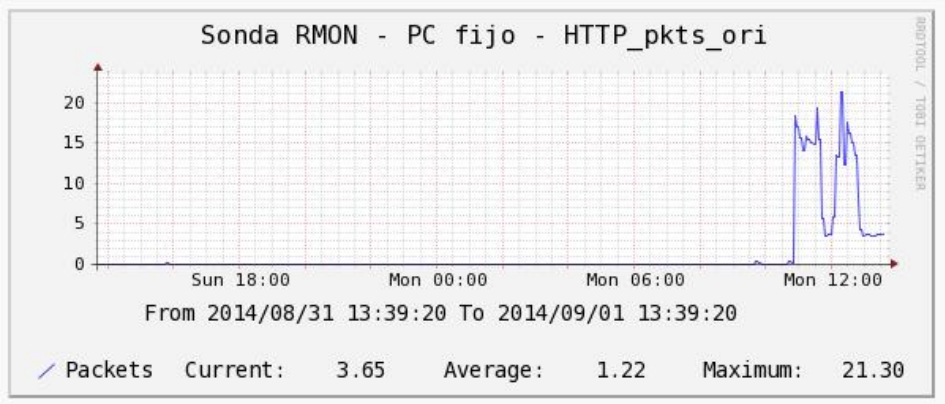

Figura 4.12: Tráfico HTTP response origen medido por la sonda.

Capítulo 5

Conclusiones y líneas futuras

5.1 Conlusiones

En este proyecto se ha desarrollado una sonda RMON que implementa el grupo Filter, multiplataforma y completamente funcional, de forma que ha sido posible integrar el resultado del proyecto en una práctica de la asignatura del grado, "Gestión de Red".

En las diferentes partes del proyecto se han analizado las principales herramientas para realizar cada una de las tareas, y se han seleccionado las opciones más interesantes, siempre tratando de optimizar el rendimiento para obtener un resultado funcional que pudiese ser utilizado incluso en equipos con reducidas prestaciones. Además, se ha desarrollado todo de forma modular para permitir que cualquier parte de este trabajo pudiese ser reaprovechada en futuros proyectos.

El primero de los módulos desarrollados fue el denominado dispatcher, el cual recibe y responde a las peticiones SNMP que llegan al equipo y realiza las acciones adecuadas. El módulo security es utilizado cada vez que se recibe una petición SNMP y verifica que cuenta con los permisos necesarios para ser procesada.

Otro de los módulos desarrollados es el que dota al agente de la funcionalidad necesaria para comunicarse con agentes secundarios. Esto nos permite utilizar un agente externo que implemente la MIB-2, de forma transparente al usuario.

El siguiente módulo es el que permite realizar la gestión de las comunidades

a través de mensajes SNMP. Para este propósito se ha definido formalmente la MIB, se ha dado una estructura apropiada para organizar la base de datos y se ha desarrollado todo lo necesario para interactuar con la base de datos.

El módulo de filtrado, él que da sentido a este proyecto, es capaz de leer la configuración introducida por el gestor en el agente mediante mensajes SNMP, y con esa configuración procesar todos los paquetes de red que llegan a su interfaz (dirigidos a él o no), aplicando todos los filtros introducidos por el gestor y en caso de existir alguno exitoso indicárselo al agente. Este módulo realiza las tareas más críticas de la sonda, por lo que ha sido necesario analizar cada detalle de la implementación para minimizar el consumo de los recursos del sistema.

También se han creado una serie de scripts para integrar la sonda con el sistema operativo posibilitando la interacción con ella del mismo modo que con el resto de los servicios del mismo. Además se ha creado una maquina virtual con una sonda funcional instalada y configurada.

Por último se realizo una aplicación que permite realizar todas las tareas de configuración y la observación de los resultados a través de una interfaz gráfica muy simple e intuitiva.

Respecto a los resultados, las medidas de capacidad de filtrado y de estabilidad indican que la sonda cumple perfectamente con su misión en redes de mediano tamaño con un número estimado de hasta 300 usuarios.

5.2 Líneas de futuro

Aunque todos lo objetivos planteados se han cumplido en el presente TFG, se plantean unas posibles mejoras que aportarían funcionalidades extra a la sonda:

- **Añadir soporte de la versión 3:** Añadir al agente la funcionalidad necesaria para soportar la versión 3 de SNMP, pudiendo reaprovechar todos los módulos desarrollados en este trabajo.

- **Desarrollo de otros grupos de RMON:** Sería muy interesante la programación de grupos tales como,

  - **capture:** Permite configurar que cuando un paquete supere un filtro, además de contarlo, lo almacene en un buffer interno del agente y posteriormente el gestor pueda descargarlo.
  - **alarm:** Permite monitorizar una determinada variable de la MIB de RMON, con una frecuencia de muestre configurable, y disparar un evento en caso de que el valor de esta variable (o su delta) sea mayor o menor que unos determinados umbrales en un periodo de muestreo.
  - **event:** Permite configurar que cuando un evento es disparado se realice una acción localizada en cualquier parte de la MIB e incluso enviar un mensaje de tipo Trap al gestor.

- **Integración con hardware externo:** Con objetivo de mejorar las prestaciones, se podría delegar toda la funcionalidad de filtrado en un dispositivo lógico programable con interfaz Ethernet.

- **Proxy Netconf-RMON:** NetConf es un protocolo para la gestión de red desarrollado y estandarizado por el IETF. Se podría definir de un modulo YANG (equivalentes a las MIB en SNMP) e implementación de un software que haga las veces de proxy permitiendo la explotación de toda la funcionalidad de RMON desde un cliente NetConf.

- **Sonda RMON distribuida:** Estudio de las opciones existentes para que un único agente RMON sea capaz de trabajar con una serie de filtros distribuidos en varios equipos de la topología de la red de forma transparente para el usuario. En este caso podrían estudiarse la posibilidades de cada equipo solo ejecutara unos pocos filtros en lugar de todos cuando se trate de redes no segmentadas. En el caso de redes segmentadas cada equipo podría estar ejecutando el total de los filtros, pero solo lo haría con el trafico que atravesase su segmento. Esto permitiría aprovechar al máximo los recursos disponibles.

Bibliografía

[1] Arribas, J. C. (2012). «Implementación automática de un agente SNMP a partir de la definición formal de su MIB».

[2] Case, J.; Fedor, M.; Schoffstall, M. y Davin, J. (Mayo de 1990). «RFC 1157 - Simple Network Management Protocol (SNMP)». *Informe técnico*, Internet Engineering Task Force (IETF).
`http://tools.ietf.org/html/rfc1157`

[3] McCloghrie, K. y Rose, M. (Mayo de 1990). «RFC 1213 - Management Information Base for Network Management of TCP/IP-based internets: MIB-II». *Informe técnico*, Internet Engineering Task Force (IETF).
`http://tools.ietf.org/html/rfc1213`

[4] Stallings, W. (1996). *SNMP, SNMPv2 and RMON: Practical Network Management*. Addison-Wesley.

[5] Stevens, W. y Rago, S. (2005). *Advanced Programing in the UNIX Environment*. Addison-Wesley.

[6] Waldbusser, S. (Julio del 2002). «RFC 3273 - Remote Network Monitoring Management Information Base for High Capacity Networks». *Informe técnico*, Internet Engineering Task Force (IETF).
`http://tools.ietf.org/html/rfc3273`

[7] —— (Mayo del 2000). «RFC 2819 - Remote Network Monitoring Management Information Base». *Informe técnico*, Internet Engineering Task Force (IETF).
`http://tools.ietf.org/html/rfc2819`

[8] —— (Mayo del 2006). «RFC 4502 - Remote Network Monitoring Management Information Base Version 2». *Informe técnico*, Internet Engineering Task Force (IETF).
`http://tools.ietf.org/html/rfc4502`

[9] Waldbusser, S.; Cole, R.; Kalbfleisch, C. y Romascanu, D. (Agosto del 2003). «RFC 3577 - Introduction to the Remote Monitoring (RMON) Family of MIB Modules». *Informe técnico*, Internet Engineering Task Force (IETF).
`http://tools.ietf.org/html/rfc3577`

[10] Waterman, R.; Lahaye, B.; Romascanu, D. y Waldbusser, S. (Junio del 1999). «RFC 2613 - Remote Network Monitoring MIB Extensions for Switched Networks Version 1.0». *Informe técnico*, Internet Engineering Task Force (IETF).
`http://tools.ietf.org/html/rfc2613`