Práctica 8. Monitorización de redes.

1. Introducción

La práctica pretende la familiarización con entornos de monitorización y/o gestión de redes. La monitorización de redes, sistemas y servicios es una de las tareas mas importantes de un ingeniero de red. Existe una notable diferencia entre la monitorización y la gestión de redes. Para monitorizar redes existen herramientas de alta calidad, genéricas e incluso con código abierto y con licencias de caracter libre. La gestión de los dispositivos de red es en cambio mucho mas particular, destacándose herramientas específicas de fabricantes, que dan soporte exclusivo a sus dispositivos como CiscoWorks o HPOpenview.

Para redes heterogéneas con distintos fabricantes no existe una herramienta unificada de gestión, y suelen utilizarse los accesos particulares a cada uno de los dispositivos que componen la red. Así, las herramientas mas importantes de gestión son el *ssh*, *telnet* o el propio acceso web cuando el dispositivo no disponga de otra interfaz. Se utilizan además de forma sistemática los lenguajes de scripting para automatizar tareas como backups de configuraciones o cambios masivos de las mismas en los diferentes dispositivos. Entre estos lenguajes se destaca el *expect*, que permite el acceso automático a los sistemas operativos de los dispositivos. No obstante, el protocolo SNMP que se describe en la documentación permite además de la monitorización, la administración de determinados parámetros de dispositivos que el fabricante haya puesto a disposición en el desarrollo de sus MIB (Management Information Base).

De esa forma, el administrador de redes necesitará monitorizar los dispositivos con las herramientas disponibles (algunas presentadas en este curso) para tomar las actuaciones pertinentes sobre los dispositivos identificados, utilizando en ese caso los accesos de gestión que dichos dispositivos ofrezcan, o modificando mediante SNMP los parámetros pertinentes.

Al entorno global que engloba los dos anteriores, se le denomina entorno de administración.

El objetivo del curso es presentar diferentes opciones de administración de red, utilizando como base el protocolo SNMP. Para ello, se presenta la implementación práctica del protocolo SNMP, el entorno web de monitorización de sistemas y de redes "NAGIOS", la herramienta gráfica de monitorización de históricos "MRTG", un entorno de monitorización de históricos mas completo (con monitorización de sistemas, servicios y redes) llamado CACTI y otras herramientas menores de ayuda a la monitorización y gestión.

La red sobre la que se trabajará será la propia red plana del laboratorio. La red de monitorización se reproduce en la figura 1, siendo a nivel IP una red independiente de la de los PC en el rango *10.11.12.0/24*.

Todos los equipos, tanto terminales como dispositivos de red son monitorizables.

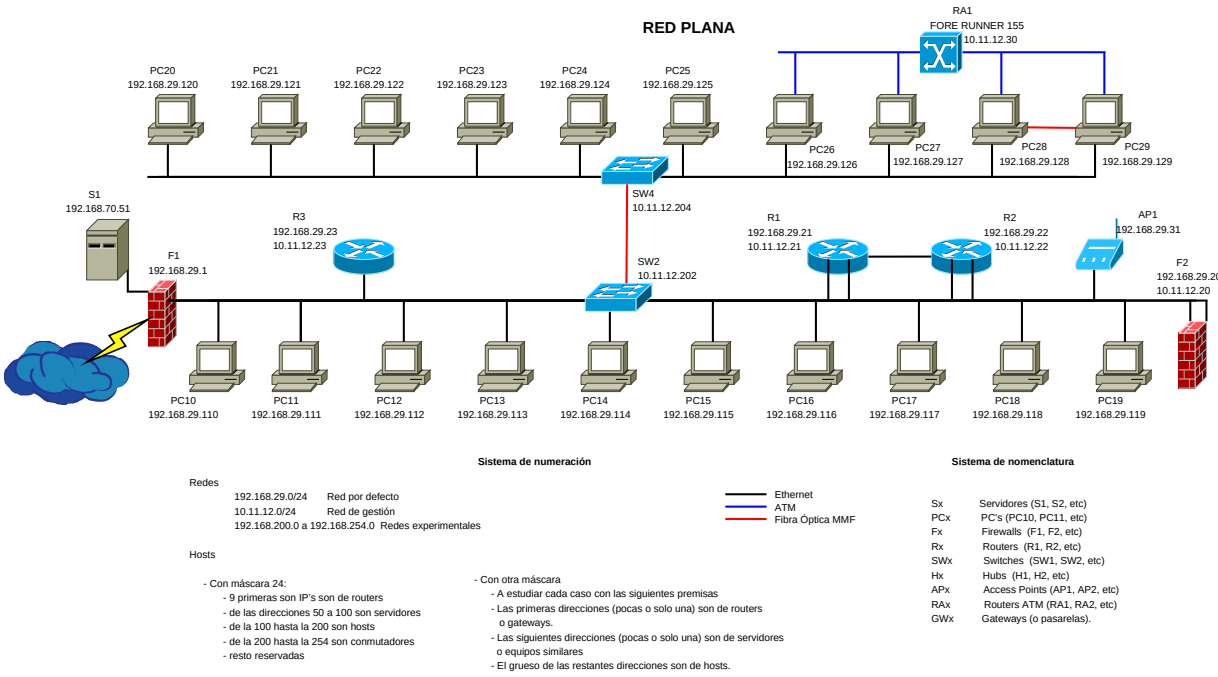

Figura 1: Red de monitorización del laboratorio

2. SNMP

El protocolo simple de gestión de red (Simple Network Management Protocol) trabaja sobre redes IP con transporte no confiable UDP sobre los puertos 161 y 162. Existen 3 versiones del protocolo, aunque se utilizarán sólo las dos primeras.

Básicamente es un protocolo de transmisión de información para la monitorización de dispositivos de red (tanto intermedios como terminales), aunque también permite la gestión y cambio de parámetros. Consta de cuatro partes diferenciadas:

- El sistema de almacenamiento y organización de la información a gestionar, que recibe el nombre de **MIB** (Management Information Base).
- El **agente SNMP**, que consulta/cambia a/en cualquier componente de su equipo la información, la almacena en su MIB respectiva y la sirve al equipo gestor (por ello los agentes son realmene los servidores SNMP).
- El **gestor SNMP**, al que también se llama **NMS** (Network Management System), que consulta/-cambia a/en cualquier agente SNMP la información almacenada en la MIB. Por tanto, los gestores SNMP son clientes.
- El conjunto de **mensajes SNMP**, que dispone de diferentes comandos de consulta/respuesta

La figura 2 muestra una topología general con los elementos del sistema SNMP.

Además de permitir el intercambio de información de gestión y/o monitorización, el protocolo es capaz de enviar avisos llamados *trap* cuando sucede algún evento particular en el agente.

2.1. Agente SNMP

El agente SNMP es un software instalado en los dispositivos gestionados que tiene dos funciones:

- Sirve como servidor de escucha para las peticiones de datos (consulta o modificación) de los clientes (gestores SNMP).
- Sirve como emisor (cliente) cuando se produce algún evento en el dispositivo en el que está hospedado. El agente SNMP envía en ese caso avisos llamados “traps”.

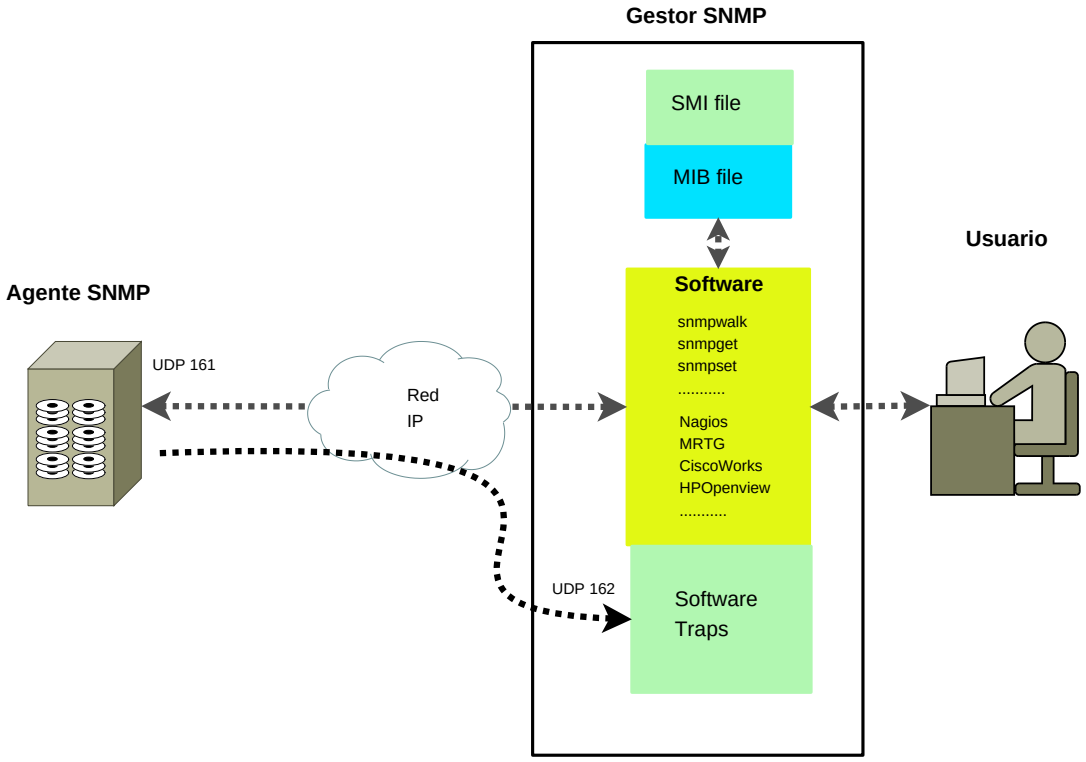

Figura 2: Topología básica SNMP

El software de agente SNMP consulta las variables de los dispositivos que gestiona bajo demanda y las formatea para ser entregadas a través de respuestas SNMP. Cada variable de dispositivo que gestiona es una MIB, y tiene su OID asignado, que es lo que consulta el gestor.

2.2. Información MIB

El proceso de consulta de las MIB requiere de una organización de las mismas. Las MIB están organizadas de forma jerárquica y única, de forma que se pueden consultar las mismas MIB en cualquier agente que cumpla los estándares, y MIB particulares para dispositivos que las implementen. La forma de nombrar a una MIB es numérica, separada por puntos, llamadas **OID** (Object IDentifier). SNMP utiliza estos identificadores de objeto (OID) para identificar las variables de la MIB a recuperar o modificar.

Existen numerosas MIB con sus OID respectivos, organizados en segmentos. Un esquema básico de organización puede verse en la figura 3

Algunos ejemplos:

- MIB generales: .iso.org.dod.internet.mgmt.mib-21 (.1.3.6.1.2.1).

  Es probablemente el MIB mas consultado, siempre con el OID inicial "1.3.6.1.2.1" que representa los objetos habituales en dispositivos IP (.iso.org.dod.internet.mgmt.mib-21). Dentro de ese OID existen las siguientes divisiones por categorías:

MIB .iso.org.dod.internet.mgmt.mib-21 (.1.3.6.1.2.1)

system	1
interfaces	2
at	3
ip	4
icmp	5
tcp	6
udp	7
egp	8
transmission	9
snmp	10
...	...

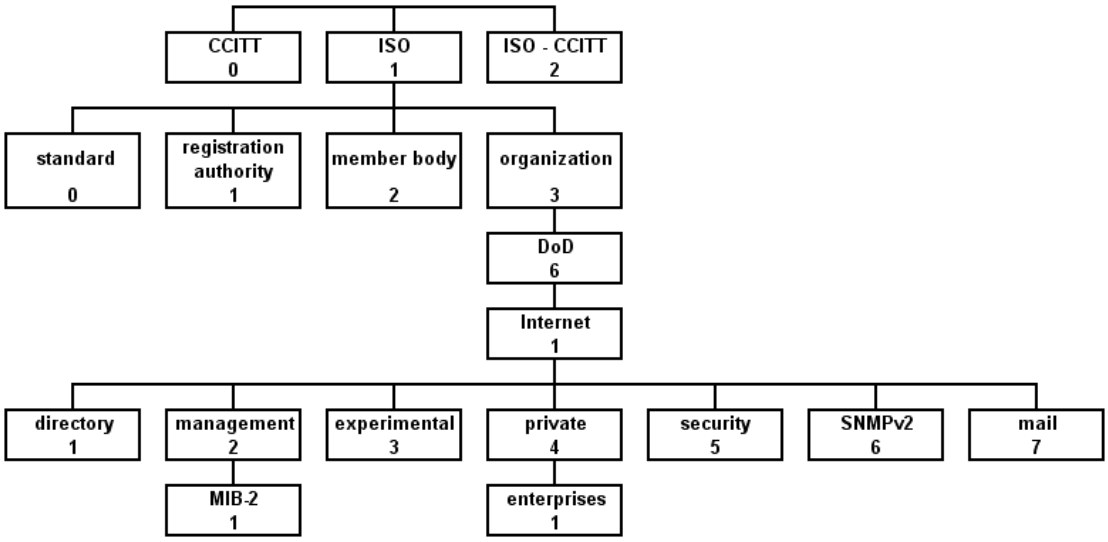

Figura 3: Árbol MIB de Internet

- MIB particular UCDavis: iso.identified-organization.dod.internet.private.enterprise.2021 (.1.3.6.1.4.1.2021)

  Este MIB permite consultar diferentes servicios predefinidos, e incluso generar de forma sencilla OID propios mediante scripting. El MIB suele estar compilado para equipos Linux.

  Algunos OID interesantes son [1]:

  - .1.3.6.1.4.1.2021.2 : resultado de procesos, como el *mountd*, el *ntalkd* o *sendmail*. Se definen en el fichero de configuración del agente.
  - .1.3.6.1.4.1.2021.9 : resultado de consulta a la informació sobre el disco duro. Se define en el fichero de configuración del agente.
  - .1.3.6.1.4.1.2021.10 : estado de carga del procesador. Se define en el fichero de configuración del agente.

  A resaltar, los OID generados de forma particular en el identificador .1.3.6.1.4.1.2021.8, donde la salida es generada por el propio usuario mediante scripting [2].

- Existen innumerables MIB, por lo que se anima al lector a buscar y conocer los que considere oportunos. Hay que tener en cuenta que cada dispositivo dispone de los que el fabricante o desarrollador haya implementado en él.

Como los OID tienen un sistema de numeración relativamente engorroso, existe la posibilidad de utilizar alias para cada número del árbol OID, como ya puede verse en la figura 3, donde el alias está puesto encima del OID numérico. La asociación de alias a los identificadores numéricos aparece en el fichero de mibs del gestor (ver sección 2.7.1, y permite tanto a los usuarios como a las aplicaciones de usuario hacer la consulta a los agentes con identificadores mas amigables. Estos ficheros llevan además información del tipo de datos de cada OID y del modo de acceso (lectura/escritura).

El sistema SNMP permite la creación de tipos de datos personalizados, que puede ser útil cuando la plataforma de gestión y visualización de resultados y monitorización necesita que los resultados que devuelve el agente se presenten en otros formatos. Esto se puede hacer mediante ficheros **SMI** (Structure Management Information), que mapean los tipos de datos personalizados en los diferentes tipos de datos originales. Un ejemplo es crear un tipo de datos llamado "datmio" que incluya un booleano con valores 0 y 10 (en lugar de los valores 1 y 2 que traen los booleanos SNMP por defecto) y un entero de 32 bits. Este tipo de dato será servido por un agente específico, que habrá sido creado también. SMI es un componente utilizado en administración de redes; nombra objetos, define los tipos de datos que se pueden almacenar en un objeto y muestra cómo pueden ser codificados los datos para su transmisión a través de la red.

[1] Nota: los OID disponibles para este MIB pueden consultarse realizando sobre el agente el comando *snmpwalk* como se verá en la sección 2.8.4

[2] Nota: en el propio fichero de configuración de agente en distribuciones Linux (*/etc/snmp/snmpd.conf*) aparece un ejemplo, y se presenta otro en la sección 2.6

Para diferenciar grupos de dispositivos a gestionar, existen las comunidades SNMP. Se distinguen dos tipos de comunidades, las públicas y las privadas. Las comunidades públicas simplementen realizan labores de consulta y transmisión de información, y las privadas pueden además cambiar parámetros. La pertenencia de un dispositivo a una comunidad u otra, tiene que definirse en cada agente, y cada gestor tiene que hacer la consulta o modificación a un dispositivo e indicar la comunidad a la que pertenece.

2.3. Gestor SNMP

Habitualmente llamado Network Management System (NMS) es el encargado de enviar las consultas SNMP, recibir las respuestas y recibir los avisos no solicitados o traps. Suele llevar un software de gestión de la información SNMP, como puede ser Nagios, CACTI o MRTG, además de software actuador como el propio Nagios, CiscoWorks o HPOpenview.

2.4. Mensajes SNMP

En la versión v2c hay 4 tipos de mensajes entre el NMS y el Agente, 2 tipos de mensajes entre el Agente y el NMS, y 2 tipos de mensajes entre un NMS y otro NMS. La figura 4 muestra todos ellos.

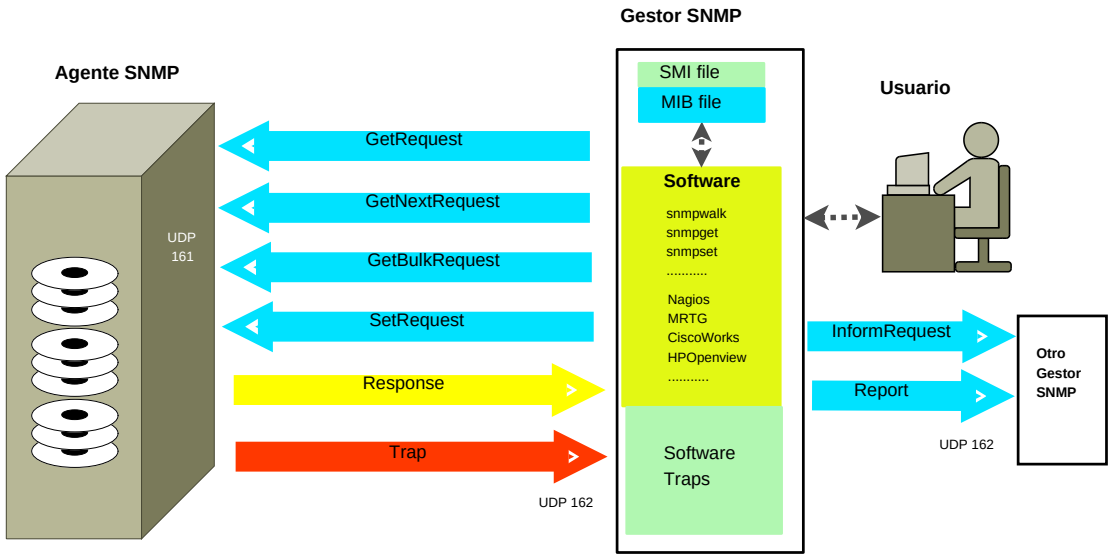

Figura 4: Mensajes SNMP

Los mensajes entre el NMS y el Agente son enviados al puerto UDP 161 del Agente por parte del NMS. La respuesta del Agente es devuelta al puerto de salida del NMS con el que se estableció el Socket, que estará por encima del puerto UDP 1024.

- GetRequest
- GetNextRequest
- SetRequest
- GetBulkRequest

Los mensajes entre el Agente y el NMS son de dos tipos, uno de respuesta a las consultas entre NMS y Agente, y otro generado automáticamente por el Agente ante un evento imprevisto; a este último mensaje se le llama Trap, y es enviado por el agente (utilizando cualquier puerto superior al UDP 1024) al puerto UDP 162 del NMS.

- Response
- Trap

Los mensajes entre NMS tienen como objetivo mantener un sistema de gestión jerárquico. Estos mensajes se envían al puerto UDP 162, al igual que los traps de agentes.

- InformRequest
- Report

2.5. Instalación

Será necesario instalar en los agentes los dos componentes inciales (MIB y agente servidor) y en el equipo gestor además las herramientas de consulta SNMP.

2.5.1. Agentes SNMP

La instalación de los agentes SNMP, es decir, el software servidor de MIB que responde a las peticiones del gestor, depende del equipo que hace de agente.

Linux En la mayoría de las distribuciones Linux, el software de agente está disponible en los repositorios, de esa forma, sólo será necesario descargarlo e instalados con las herramientas disponibles. En particular, para distribuciones Debian se tiene que instalar el paquete *snmpd*

```
alumno@pcX:/home/alumno$ su
Contrasena:
pcX:/home/alumno# apt-get install snmpd
...........
```

Este paquete incluye algunas MIB básicas, suficientes para una topología inicial.

LEAF/LRP Esta es una distribución Linux creada específicamente para convertir hosts en routers (Host Based Routers - HBR). El proceso es parecido al caso anterior, aunque, al tener una limitación de espacio (pues se instala desde un disquete), la instalación se realiza remotamente. En el caso de la topología de la práctica, se dispone de un servidor TFTP en *S1* (*192.168.70.51*), y de un cliente TFTP en la pasarela LEAF (*F1*). Por ello será necesario descargarse los paquetes necesarios desde el servidor TFTP, e instalarlos manualmente. El proceso, siguiendo un orden riguroso sería [3]:

1. Habilitar el servidor TFTP en *S1*

```
root@S1# /usr/sbin/in.tftpd  /home/isauro/tftp_dir
```

2. Descargarse desde *R1* y *R2* los paquetes *libm.lrp*, *libsnmp.lrp*, *netsnmpd.lrp*.

```
RX# tftp 192.168.70.51 -c get libm.lrp
RX# tftp 192.168.70.51 -c get libsnmp.lrp
RX# tftp 192.168.70.51 -c get netsnmpd.lrp
```

3. Instalar los paquetes por riguroso orden

```
RX# apkg -i libm.lrp
libm already installed.
RX# apkg -i libsnmp.lrp
libsnmp installed
RX# apkg -i netsnmpd.lrp
netsnmpd installed
```

OpenWRT Dependiendo de si el dispositivo de red dispone de memoria suficiente o nó, se instalarán unos paquetes u otros

Instalación en un Linksys WRT54GL.

La versión de OpenWRT instalada es la Backfire 10.03. Al no disponerse de suficiente espacio para el software *net-snmp* completo, se ha optado por el paquete *mini-snmpd*. Para ello hay que instalar previamente el módulo para IPv6.

```
F2# opkg -i kmod-ipv6_2.6.32.25-1_brcm47xx.ipk
F2# opkg -i mini-snmpd_1.2b-1_brcm47xx.ipk
```

[3] Nota: en los equipos instalados en el laboratiorio de automática

Instalación en un RB433

Las routerboard Microtik 433 disponen de suficiente memoria para una instalación mas completa. La instalación se ha hecho sobre la versión Kamikaze 8.09.02 y sobre la versión Backfire 10.03 de OpenWRT para AR71XX

Se instala por este orden:

```
F2# opkg -i libelf_0.8.10-1_mips.ipk
F2# opkg -i libnetsnmp_5.1.2-3.1_mips.ipk
F2# opkg -i snmpd_5.1.2-3.1_mips.ipk
```

La configuración se realiza sobre el fichero */etc/config/snmpd*. Cuando se lanza el demonio *snmpd* en el arranque (*/etc/init.d/snmpd start*) lee la configuración de ese fichero y genera el fichero central de configuración en */etc/snmpd/snmpd.conf* que es un enlace hacia */var/run/snmpd.conf*.

2.5.2. Gestor SNMP

GNU/Linux Para la instalación de un gestor SNMP en las distribuciones Linux mas comunes, es decir, un cliente SNMP que consulta a los agentes/servidores SNMP, basta con instalar el paquete "snmp".

Para Debian:

```
alumno@pcX:/home/alumno$ su
Contrasena:
pcX:/home/alumno# apt-get install snmp
...........
```

Windows Existen numerosas plataformas de visualización de MIBs para Windows, la mayoría de pago, aunque algunas también gratuitas pero no de código abierto. Una de las gratuitas mas utilizadas es ManageEngine MIB Browser (http://www.manageengine.com/products/mibbrowser-free-tool/index.html).

Otras plataformas Será necesario ver la documentación para cada caso.

2.6. Configuración de Agentes SNMP

La configuración de un servidor SNMP varía de forma notable para cada dispositivo en particular.

2.6.1. Configuración de dispositivos

Conmutadores Cisco SW3 y SW6 Para habilitar la característica SNMP en la "comunidad" en los equipos Cisco, será necesario conectarse al equipo (SW3, SW6) y activar la funcionalidad SNMP.

Nota: en el desarrollo de la práctica ya está habilitado el servicio SNMP en los conmutadores.

```
PCX:~# telnet 10.11.12.203
Trying  10.11.12.203.
Connected to 10.11.12.203.
Escape character is '^]'.

User Access Verification

Password:
SW3 >enable
Password:
SW3 # configure terminal
SW3 (config) #  snmp-server  community  public
SW3 (config) #  snmp-server host 10.11.12.126 public
SW3 (config) #  snmp-server enable traps
```

La palabra "public" es la comunidad pública SNMP creada para la red del laboratorio.

La descripción completa está en el documento de configuración del IOS de Cisco para este equipo.

Conmutador FORE ATM. ATM1 El conmutador ATM de FORE es también monitorizable en la dirección 10.11.12.30.

Para acceder a él, es necesario que alguno de los equipos que dispongan de tarjeta ATM y que estén conectados a él ejerza como Router o como Bridge. Además será necesario habilitar una ruta en el equipo gestor hacia el router ATM.

Nota: en el desarrollo de la práctica ya está habilitado el servicio SNMP en el router ATM.

```
PCX:/home/alumno# telnet 10.11.12.30
Trying 10.11.12.30...
Connected to 10.11.12.30.
Escape character is '^]'.

S_ForeThought_5.2.0 FCS (1.23449) (le155) (ATM SWITCH)

login: su_rdo
Password:

ATM Management Interface v5.2.0
Copyright (c) 1994-1998 FORE Systems, Inc.

General commands:
  '?' to get list of commands at the current level
  'up' to go up one menu level
  'top' to go to the root menu
  'exit' to leave AMI

Opening a session for "127.0.0.1", please wait...

Connected to "127.0.0.1" (le155).

ATM SWITCH::> conf

ATM SWITCH::configuration> snmp

ATM SWITCH::configuration snmp> community read public

ATM SWITCH::configuration snmp> show
SET operations from network are enabled

ATM SWITCH::configuration snmp> sets ?
usage: sets (enable | disable)

ATM SWITCH::configuration snmp> trap ?
  destinations>      log>

ATM SWITCH::configuration snmp> trap dest ?
  delete             new                show

ATM SWITCH::configuration snmp> exit
```

Dispositivos GNU/Linux El proceso de configuración de los agentes SNMP es muy parecido en todas las distribuciones Linux. El agente mas utilizado es el contenido en la suite "net-snmp".

El fichero de configuración se encuentra en "/etc/snmp/snmpd.conf", donde hay que cambiar/personalizar algunas lineas de inicio, en particular las referidas al control de acceso.

Para versiones Debian 5.0 o anteriores, el control de acceso se realiza en cuatro pasos.

1. Para dar acceso al agente SNMP sólo a equipos de la red local, se pueden crear varios perfiles de acceso, por ejemplo el del equipo local (localhost) y el de la red de gestión (MonitNet).

   ```
   #com2sec paranoid default public
     com2sec local   localhost public
     com2sec MonitNet 10.11.12.0/24 public
   #com2sec readwrite default private
   ```

   El orden de creación de los perfiles de acceso es importante, pues cuando un equipo pertenezca a varios, se aplicarán las normas de acceso del primero al que pertenezca.

2. Se mapean los perfiles de acceso en grupos

   ```
   #               sec.model  sec.name
   group MyROSystem v1        paranoid
   group MyROSystem v2c       paranoid
   group MyROSystem usm       paranoid
   group MyROGroup v1         local
   group MyROGroup v2c        local
   group MyROGroup usm        local
   group MyROGroupNet v1         MonitNet
   ```

```
group MyROGroupNet v2c        MonitNet
group MyROGroupNet usm        MonitNet
group MyRWGroup v1         readwrite
group MyRWGroup v2c        readwrite
group MyRWGroup usm        readwrite
```

3. Se crean diferentes vistas

```
#           incl/excl subtree                          mask
view all    included  .1                               80
view system included  .iso.org.dod.internet.mgmt.mib-2.system
```

4. Finalmente se asocian los diferentes grupos a las vistas creadas

```
#               context sec.model sec.level match  read   write  notif
access MyROSystem ""      any       noauth    exact  system none   none
access MyROGroup ""       any       noauth    exact  all    none   none
access MyROGroupNet ""      any       noauth    exact  all    none   none
access MyRWGroup ""       any       noauth    exact  all    all    none
```

Seguidamente será necesario reiniciar el demonio "snmpd".

```
PCX$ /etc/init.d/snmpd restart
```

En algunas distribuciones (se ha encontrado en distribuciones Debian 5.0 y algunas anteriores), los parámetros de arranque del demonio (Ojo, no tiene relación con los parámetros del fichero de configuración) se leen del fichero "/etc/default/snmpd". Estos parámetros de arranque, limitan el acceso al servicio snmp sólo al localhost (dirección 127.0.0.1). Para poder abrir el servicio a todos los equipos de la forma que se gestiona en el fichero de configuración "/etc/snmp/snmpd.conf" será necesario modificar el fichero "/etc/default/snmpd". Habrá que cambiar la línea

```
SNMPDOPTS='-Lsd -Lf /dev/null -u snmp -I -smux -p /var/run/snmpd.pid 127.0.0.1'
```

por la línea

```
SNMPDOPTS='-Lsd -Lf /dev/null -u snmp -I -smux -p /var/run/snmpd.pid'
```

Para distribuciones Debian 6.0 o posteriores, el formato del fichero de configuración ha cambiado, habiéndose simplificado en general.

Si se quiere permitir el acceso a gestores diferentes de la máquina local, será necesario que el comienzo del fichero de configuración tenga la forma:

```
#agentAddress  udp:127.0.0.1:161
#  Listen for connections on all interfaces (both IPv4 *and* IPv6)
#agentAddress udp:161,udp6:[::1]:161
#  Liste for connections only on IPv4 interfaces
agentAddress udp:161
```

El agente SNMP basado en Linux sólo acepta una línea "agentAddress". Se ha comprobado que poner dos o más líneas "agentAddress" no arranca (o no arranca correctamente) el agente. El control de acceso es ahora mas sencillo:

```
view   systemonly  included   .1.3.6.1.2.1.1
view   systemonly  included   .1.3.6.1.2.1.25.1
                                          #  Full access from the local host
#rocommunity public  localhost
                                          #  Default access to basic system info
# rocommunity public  default    -V systemonly
  rocommunity public default
```

Primero se definen vistas con la directiva view, y después se definen comunidades donde se establece quién puede acceder y qué vista pueden visualizar. La palabra clave "default" indica que puede acceder todo el mundo. El indicador "-V" indica la vista a la que pueden acceder.

Tras modificar el fichero de configuración, será necesario reiniciar el agente:

```
PCX$ /etc/init.d/snmpd restart
```

Dispositivos LRP/LEAF Para configurar la distribución de paquetes snmp en "lrp" es necesario instalar el paquete "snmp.lrp" en el router, o si hay espacio suficiente, el paquete "netsnmp.lrp" , mas actualizado y completo. Tras instalarlo es necesario ponerlo en funcionamiento haciendo simplemente "/etc/init.d/snmpd start".

Es necesario habilitar el acceso en el fichero hosts.allow.

```
.....................
# Allow anything from the local net
ALL: 192.168.29.0/255.255.255.0
...........
```

El fichero de configuración "/etc/snmpd/snmpd.conf" deberá tener unas líneas

```
...........
#com2sec    paranoid   default  _ltd_Private_Community_Name_
com2sec    readonly   default  public
#com2sec    readwrite  default  _rw_Private_Community_Name_
com2sec     readwrite  default private
...........
```

Se arranca de la forma:

```
RX# /etc/init.d/snmpd start
```

Y se comprueba su funcionamiento:

```
RX# ps -ef |grep snmpd
```

Como añadido, existe la posibilidad de capturar paquetes SNMP con el software "wireshark".

Estos equipos serán monitorizables en todas sus interfaces, pero existirá una interfaz virtual en cada uno dedicada a la transmisión de mensajes SNMP. En particular:

- R1: será monitorizable en la dirección IP 10.11.12.21
- R2: será monitorizable en la dirección IP 10.11.12.22
- F2: será monitorizable en la dirección IP 10.11.12.20

Dispositivos con OpenWRT La configuración en estos dispositivos si se ha instalado el paquete "snmpd" en su versión de enlace dinámico se encuentra en el fichero "/etc/config/snmpd". El formato del fichero "/etc/config/snmpd" es relativamente sencillo, dividido en secciones de configuración llamadas "config". El parseo de este fichero hacia el formato correcto en el fichero "/etc/snmpd/snmpd.conf" se realiza en el script "/etc/init.d/snmpd", donde pueden visualizarse las diferentes opciones.

La inserción en OIDs de salidas de scripts concretos del agente en lugar de los que están compilados directamente por los fabricantes o los que están globalmente asignados, se hace en la sección "config exec", de la forma

```
config exec
        option name     arp
        option prog     '/etc/snmp/arp'
        option args     ""
#       option miboid   1.3.6.1.3.19811018.30
```

donde "option prog" indica el script que se quiere ejecutar. En este caso es la visualización del fichero "/proc/net/arp", realizado en el script "/etc/snmp/arp" cuyo contenido podría ser:

```
#! /bin/sh
cat /proc/net/arp
```

El gestor SNMP puede consultar la salida con el OID "extOutput.1" donde el número final indica el orden de las diferentes secciones "config exec".

2.6.2. Extensiones de agente

En la suite *net-snmp* de Linux, se permite utilizar algunos OID para generar y enviar información no existente previamente en una MIB.

Para ello será necesario conocer de antemano la información a transmitir desde el agente al gestor. Como ejemplo sencillo, se servirá desde un equipo Linux información de su versión de Kernel. Para la realización del ejemplo será necesario crear en el agente un script previo que muestre en pantalla la información a transmitir, llamado "/home/alumno/ver_kernel.sh", . Su contenido podría ser

```
#! /bin/sh
   /bin/uname -r
   exit 69
```

Deberá tener permisos de ejecución;

```
PCX$ chmod 755 /home/alumno/ver_kernel.sh
```

En agentes Linux, el fichero de configuración permite la generación de OID a medida con la información recopilada por el script anterior utilizando varios métodos:

- Uno de ellos es utilizando el MIB de UC Davis (1.3.6.1.4.1.2021.8) de la sección de MIB de empresas (1.3.6.1.4.1). Como ejemplo de su uso, se va a utilizar el script anterior que permite visualizar la versión del kernel del agente, aunque su utilidad es manifiestamente superior, destacándose la salida de programas o demonios que estén funcionando en el agente y que quieran ser monitorizados, como por ejemplo el consumo energético, la salida de determinados sensores, o los resultados de consultas como accesos al sistema, compras realizadas o cualquier dato que se pueda programar y presentar de forma textual o numérica.

  Se indicará dentro del fichero de configuración del agente ("/etc/snmp/snmpd.conf") en la sección *Executables/scripts* lo siguiente:

  ```
  exec Ver_Kern /bin/sh /home/alumno/ver_kernel.sh
  ```

  donde "exec" es un parámetro para utilizar el MIB de UC Davis (1.3.6.1.4.1.2021.8), "shelltest" es el nombre arbitrario que se le ha puesto al script, */bin/sh* es el programa o script a ejecutar y */home/alumno/ver_kernel.sh* son los argumentos del programa.

  La línea podría sustituirse por

  ```
  exec Ver_Kern /home/alumno/ver_kernel.sh
  ```

  ya que *snmpd* reconoce los formatos de varios ejecutables, incluidos perl o shell.

  En el dispositivo NMS se realizará la consulta al OID .1.3.6.1.4.1.2021.8, ya sea desde el host local, o desde el gestor.

  ```
  PCX$ snmpwalk -On -v2c -c public localhost .1.3.6.1.4.1.2021.8
  .1.3.6.1.4.1.2021.8.1.1.1 = INTEGER: 1
  .1.3.6.1.4.1.2021.8.1.2.1 = STRING: Ver_Kern
  .1.3.6.1.4.1.2021.8.1.3.1 = STRING: /bin/sh
  .1.3.6.1.4.1.2021.8.1.100.1 = INTEGER: 69
  .1.3.6.1.4.1.2021.8.1.101.1 = STRING: 3.2.9
  .1.3.6.1.4.1.2021.8.1.102.1 = INTEGER: 0
  .1.3.6.1.4.1.2021.8.1.103.1 = STRING:
  ```

  donde puede verse el resultado del script ".1.3.6.1.4.1.2021.8.1.101.1 = STRING: 2.6.26-2-686" que muestra la versión del kernel solicitada. También puede verse ".1.3.6.1.4.1.2021.8.1.100.1 = INTEGER: 69" que muestra el código de salida del script ("exit 69").

  Para salidas de script de mas de una línea, se utiliza el OID .1.3.6.1.4.1.2021.50 o mayores (.1.3.6.1.4.1.2021.51 ..)

- Otro es utilizando el parámetro "extend". Sería similar al anterior, pero el OID donde se presenta el resultado es otro.

  Habrá que añadir al final del fichero de configuración de agente */etc/snmp/snmpd.conf* una línea como:

```
#   extend <script_name>    <script_full_path>  <arguments>
    extend   Ver_Kern   /home/alumno/ver_kernel.sh
```

Para visualizar la información desde un NMS gestor, será necesario hacer la consulta al MIB NET-SNMP-EXTEND-MIB::nsExtendOutputFull, que se corresponde con el OID .1.3.6.1.4.1.8072.1.3.2.3.1.2

```
PCX$ snmpwalk -On -v2c -c public localhost NET-SNMP-EXTEND-MIB::nsExtendOutputFull
.1.3.6.1.4.1.8072.1.3.2.3.1.2.8.86.101.114.95.75.101.114.110 = STRING: 3.2.9
.1.3.6.1.4.1.8072.1.3.2.3.1.2.9.115.104.101.108.108.116.101.115.116 = STRING: 3.2.9
```

o si se quiere incluir toda la información, se hace la consulta al MIB NET-SNMP-EXTEND-MIB::nsExtendObjects, que se corresponde con el OID .1.3.6.1.4.1.8072.1.3.2

```
PCX$ snmpwalk -On -v2c -c public localhost NET-SNMP-EXTEND-MIB::nsExtendObjects
.1.3.6.1.4.1.8072.1.3.2.1.0 = INTEGER: 2
.1.3.6.1.4.1.8072.1.3.2.2.1.2.8.86.101.114.95.75.101.114.110 = STRING: /bin/sh
.1.3.6.1.4.1.8072.1.3.2.2.1.2.9.115.104.101.108.108.116.101.115.116 = STRING: /home/alumno/↩
    scripts/ver_kernel.sh
.1.3.6.1.4.1.8072.1.3.2.2.1.3.8.86.101.114.95.75.101.114.110 = STRING: /home/alumno/scripts/↩
    ver_kernel.sh
.1.3.6.1.4.1.8072.1.3.2.2.1.3.9.115.104.101.108.108.116.101.115.116 = STRING:
.1.3.6.1.4.1.8072.1.3.2.2.1.4.8.86.101.114.95.75.101.114.110 = STRING:
.1.3.6.1.4.1.8072.1.3.2.2.1.4.9.115.104.101.108.108.116.101.115.116 = STRING:
.1.3.6.1.4.1.8072.1.3.2.2.1.5.8.86.101.114.95.75.101.114.110 = INTEGER: 5
.1.3.6.1.4.1.8072.1.3.2.2.1.5.9.115.104.101.108.108.116.101.115.116 = INTEGER: 5
.1.3.6.1.4.1.8072.1.3.2.2.1.6.8.86.101.114.95.75.101.114.110 = INTEGER: exec(1)
.1.3.6.1.4.1.8072.1.3.2.2.1.6.9.115.104.101.108.108.116.101.115.116 = INTEGER: shell(2)
.1.3.6.1.4.1.8072.1.3.2.2.1.7.8.86.101.114.95.75.101.114.110 = INTEGER: run-on-read(1)
.1.3.6.1.4.1.8072.1.3.2.2.1.7.9.115.104.101.108.108.116.101.115.116 = INTEGER: run-on-read(1)
.1.3.6.1.4.1.8072.1.3.2.2.1.20.8.86.101.114.95.75.101.114.110 = INTEGER: permanent(4)
.1.3.6.1.4.1.8072.1.3.2.2.1.20.9.115.104.101.108.108.116.101.115.116 = INTEGER: permanent(4)
.1.3.6.1.4.1.8072.1.3.2.2.1.21.8.86.101.114.95.75.101.114.110 = INTEGER: active(1)
.1.3.6.1.4.1.8072.1.3.2.2.1.21.9.115.104.101.108.108.116.101.115.116 = INTEGER: active(1)
.1.3.6.1.4.1.8072.1.3.2.3.1.1.8.86.101.114.95.75.101.114.110 = STRING: 3.2.9
.1.3.6.1.4.1.8072.1.3.2.3.1.1.9.115.104.101.108.108.116.101.115.116 = STRING: 3.2.9
.1.3.6.1.4.1.8072.1.3.2.3.1.2.8.86.101.114.95.75.101.114.110 = STRING: 3.2.9
.1.3.6.1.4.1.8072.1.3.2.3.1.2.9.115.104.101.108.108.116.101.115.116 = STRING: 3.2.9
.1.3.6.1.4.1.8072.1.3.2.3.1.3.8.86.101.114.95.75.101.114.110 = INTEGER: 1
.1.3.6.1.4.1.8072.1.3.2.3.1.3.9.115.104.101.108.108.116.101.115.116 = INTEGER: 1
.1.3.6.1.4.1.8072.1.3.2.3.1.4.8.86.101.114.95.75.101.114.110 = INTEGER: 69
.1.3.6.1.4.1.8072.1.3.2.3.1.4.9.115.104.101.108.108.116.101.115.116 = INTEGER: 17664
.1.3.6.1.4.1.8072.1.3.2.4.1.2.8.86.101.114.95.75.101.114.110.1 = STRING: 3.2.9
.1.3.6.1.4.1.8072.1.3.2.4.1.2.9.115.104.101.108.108.116.101.115.116.1 = STRING: 3.2.9
```

2.6.3. snmptrap

El programa "snmptrap" de dispositivos con sistema operativo basado en Linux, permite generar traps a voluntad. Ver la página del manual para conocer su sintaxis. Algunos ejemplos de su uso sería:

```
PCX:~$ snmptrap -v 1 -c public 127.0.0.1 .1.3.6.1 localhost 6 17 '' .1.3.6.1 s "Trap numero 1"
PCX:~$ snmptrap -v 2c -c public 192.168.29.128 "" NET-SNMP-EXAMPLES-MIB::↩
    netSnmpExampleHeartbeatNotification SNMPv2-MIB::sysLocation.0 s "Trap de prueba"
PCX:~$ snmptrap -v 2c -c public 192.168.29.129 "" NET-SNMP-EXAMPLES-MIB::↩
    netSnmpExampleHeartbeatNotification netSnmpExampleHeartbeatRate i 123456
```

Es necesario instalar el paquete "snmp-utils" para poder utilizar este comando en las distribuciones OpenWRT y LEAF/Bering

2.7. Configuración de gestor

Habitualmente al gestor se le llama NMS (Network Management System). El NMS no necesita una configuración especial, y debe tener las siguientes características:

- Disponer de herramientas de consulta SNMP. Estas se ven en la sección 2.8. Estas realizarán consultas al puerto UDP 161.
- Tener abierto el puerto UDP 162 para la recepción de traps.
- Disponer de herramientas para el desarrollo de entornos visuales de monitorización, como Nagios, MRTG, o la posibilidad de creación de software propio.

2.7.1. Fichero de MIBs

Para facilitar la gestión de determinados agentes, estos proveen de un fichero con la descripción de los OID que pueden ser consultados. Este fichero es el fichero de MIBS, y el gestor NMS puede tenerlo instalado para facilitar la consulta de dichos agentes y conocer toda la información al respecto. Los fichero de MIBs suelen descargarse de fabricantes o proveedores en general. Los ficheros donde se encuentran las mibs tienen un formato normalizado.

Existen varias formas de indicarle al sistema NMS dónde se encuentran los ficheros de MIBS; algunos de ellos dependen de la versión del sistema operativo.

- Las MIB que se instalan con el sistema se encuentran en los directorios por defecto. Estos directorios pueden visualizarse con el comando:

```
    PCX $ net-snmp-config --default-mibdirs
/home/user/.snmp/mibs:/usr/share/mibs/site:/usr/share/snmp/mibs:/usr/share/mibs/iana:/usr/share↩
    /mibs/ietf:/usr/share/mibs/netsnmp
```

- Una forma de ampliar las MIB existentes es descargarlas e incluirlas en alguno de los directorios resultado del comando anterior.

- Otra forma de ampliar a otras MIB descargadas es indicar en el fichero de configuración "/etc/snmp/snmp.conf" el directorio donde se encuentran dichas MIB, de la forma:

```
#
# As the snmp packages come without MIB files due to license reasons, loading
# of MIBs is disabled by default. If you added the MIBs you can reenable
# loaging them by commenting out the following line.
mibdirs :/home/alumno/mibs
mibs :ALL
```

  Si se quieren meter mas MIBS:

```
mibs :<nombre-mib>:<nombre-mib2>: ...
```

  Si se quieren usar varios directorios:

```
mibdirs :<directorio1>:<directorio2>: ...
```

  ¡Ojo!: es necesario dejar el espacio en blanco entre la declaración (mibdirs o mibs) y los dos puntos ":".

- Una última forma de actualizar el conjunto de MIBs del sistema Linux es utilizando el comando *snmp-mibs-downloader*, que habrá que descargar previamente:

```
    PCX # apt-get install snmp-mibs-downloader
```

  Ese paquete provee el comando *download-mibs* e instala las MIB IETF en el directorio habitual */usr/share/mibs*. Si se quieren actualizar dichas MIB, se puede ejecutar:

```
    PCX # download-mibs
```

Si por alguna razón no se visualizan los nombres de las variables en lugar de su OID, se recomienda primeramente comentar las lineas referidas a las MIB del fichero */etc/snmp/snmp.conf*

2.7.2. Gestión de traps: snmptrapd

Este programa recibe y almacena los distintos traps que envía el agente. Sin parámetros, presenta los traps recibidos en pantalla y los almacena en el syslog. Es necesario ejecutarlo como superusuario.

Dispone de un fichero de configuración en "/etc/snmp/" llamado "snmptrapd.conf". Es necesario insertar en ese fichero información de acceso para que pueda operar. Bastaría una línea como la siguiente:

```
authCommunity log,execute,net public
.......................
```

Un ejemplo de uso:

```
PCX # snmptrapd -o /home/alumno/trapd.log
```

En el ejemplo anterior, almacena los traps recibidos en el fichero "/home/alumno/trapd.log".

El programa "snmptrapd" permite que sea un programa externo el que gestione los traps recibidos. Esto es muy importante cuando se quiere que estos sean gestionados desde una suite de monitorización como Nagios. .

2.8. Herramientas consulta SNMP

Los equipos gestores SNMP disponen de varias herramientas de consulta de MIB de los agentes. Se muestran algunas de ellas.

2.8.1. snmptranslate

Este comando permite la traducción de los OID en sus nombres respectivos.

```
PCX # snmptranslate .1.3.6.1.2.1.2
IF-MIB::interfaces
```

2.8.2. snmpget

Este comando básico permite localizar el valor de un OID de un dispositivo concreto de red. Se utilizará en la comunidad publica.

La sintaxis puede averiguarse ejecutando el comando sin opciones, y no tiene que ser ejecutado con ningún permiso particular:

```
PCX $ snmpget
```

Como ejemplo, se puede localizar el número de interfaces del conmutador SW2:

```
PCX $ snmpget -v 1 -c public 10.11.12.202 interfaces.ifNumber.0
IF-MIB::ifNumber.0 = 51
```

Se puede utilizar el OID en lugar del descriptor, en cuyo caso, la consulta snmpget quedaría

```
PCX $ snmpget -v 1 -c public -On 10.11.12.202 .1.3.6.1.2.1.2.1.0
.1.3.6.1.2.1.2.1.0 = 51
```

Conviene resaltar que hay que añadir el valor "0" al final del descriptor del OID. También hay que resaltar que el número de interfaces devuelto no tiene porqué coincidir con el número de interfaces físicas del dispositivo, y dependerá del fabricante.

2.8.3. snmpset

Esta utilidad sirve para cambiar el contenido de una MIB concreta en un dispositivo. Es necesario resaltar que el dispositivo tiene que tener activa la comunidad privada.

La sintaxis puede averiguarse ejecutando el comando sin opciones, y no tiene que ser ejecutado con ningún permiso particular:

```
1     PCX $ snmpset
```

Como ejemplo, se puede cambiar el contacto del router R1

```
PCX:# snmpset -v 1 -c privada 10.11.12.21 system.sysContact.0 s contacto@prueba.laisa
SNMPv2-MIB::sysContact.0 = STRING: contacto@prueba.laisa
```

2.8.4. snmpwalk

La herramienta de gestión snmpwalk permite conocer un segmento de variables contínuas de un dispositivo. Utiliza la PDU *get-next-request* para solicitar la siguiente variable hasta que se sirva el segmento completo. Puede utilizarse también para la confirmación de la existencia de un agente SNMP en la dirección de host indicada entre sus parámetros.

Básicamente, snmpwalk es un comando que ejecuta un primer "GET-request" sobre un punto de partida en el árbol OID, (p.ej. "system" con un OID ".1.3.6..1.2.1.1") y acto seguido busca hacia abajo todo el resto del árbol mediante paquetes "GET-next-request".

La sintaxis puede averiguarse ejecutando el comando sin opciones, y no tiene que ser ejecutado con ningún permiso particular:

```
PCX $ snmpwalk
```

Básicamente, será necesario indicar la versión SNMP, la comunidad SNMP solicitada y el agente al que se solicita, y opcionalmente el OID. Una utilidad concreta de esta herramienta es la de averiguar que MIB están disponibles en un dispositivo, así, sin poner el OID se recuperan todos los MIB del dispositivo al que se consulta.

Como ejemplo de su uso, se puede consultar todo el segmento "system" del conmutador SW4

```
PCX:# snmpwalk -v 1 -c public 10.11.12.204 system
system.sysDescr.0 = Cisco Internetwork Operating System Software
IOS (tm) C2900XL Software (C2900XL-C3H2S-M), Version 12.0(5)WC3b, RELEASE SOFTWARE (fc1)
Copyright (c) 1986-2002 by cisco Systems, Inc.
Compiled Fri 15-Feb-02 10:14 by antonino
system.sysObjectID.0 = OID: enterprises.9.1.220
system.sysUpTime.0 = Timeticks: (88141574) 10 days, 4:50:15.74
system.sysContact.0 =
system.sysName.0 = SW4
system.sysLocation.0 =
system.sysServices.0 = 2
system.sysORLastChange.0 = Timeticks: (0) 0:00:00.00
```

2.8.5. snmptable

Cuando se sabe que un grupo de variables tienen estructura de tabla (p.ej.: la tabla de direcciones IP).

```
PCX $ snmptable -v2c -c public 10.11.12.204 -Ov ipAddrTable
```

2.8.6. Navegadores OID

Existen diferentes navegadores gratuitos y libres para consultar la estructura OID de forma gráfica y cómoda. Aunque suelen ser herramientas muy simples, dan facilidad al ingeniero de red para saber qué buscar y con qué OID hacerlo.

Habitualmente, el ingeniero o administrador de la red, crea sus propios entornos o realiza importantes adaptaciones de algunos ya creados con herramientas libres (como Nagios) para monitorizar y administrar la red.

Algunos navegadores son:

- mbrowse: se encuentra en "http://sourceforge.net/projects/mbrowse/".
- tkmib: se encuentra en el repositorio de Debian, y está instalado en algunos equipos del laboratorio.

2.9. Ejercicios propuestos

Los ejercicios se ejecutarán desde PC con distribuciones Linux, Debian 5.0 y 6.0.

2.9.1. Consulta de toda la información SNMP disponible del equipo local

Se trata de comprobar la información SNMP disponible. El comando sería el "snmpwalk".

2.9.2. Consulta de toda la información SNMP de otro equipo remoto.

Se trata de poder acceder a un equipo remoto Linux para consultar la información disponible. El equipo remoto será el agente, y el equipo local hará de gestor.

2.9.3. Consulta de información específica en un agente.

Se sugiere encontrar y visualizar las informaciones de:

- las interfaces del equipo agente.
- La utilización (en bytes) de las unidades de almacenamiento locales en los servidores (equipos agente).
- Determinar cuánto tiempo han estado funcionando los equipos agente (p.ej. los PC, RB2, RB3, R1, R2, SW1, SW2 ...

2.9.4. Limitar el acceso a la información proporcionada por un agente Linux

Este ejercicio limitará el acceso a la información SNMP de un agente Linux a una vista de información de sistema para ser visualizada por un equipo remoto, pero el equipo local podrá acceder a toda la información. Es decir, el equipo remoto sólo visualizará la información configurada en una "vista", y el equipo local podrá visualizar toda la información SNMP disponible.

También se impedirá el acceso total a algún equipo específico.

2.9.5. Enviar información de traps por parte de un agente y el almacenamiento de la información por parte del gestor

Este ejercicio generará traps por parte del agente. Primero se generará un trap a medida, y posteriormente se generará un trap de red dando de baja una interfaz.

En el gestor se almacenarán los traps recibidos en un fichero y se visualizará el contenido.

2.9.6. Generar información SNMP a medida

Este ejercicio propone generar información SNMP a medida por parte del agente utilizando UCDavis. Esta información se consultará por parte del gestor.

2.9.7. Cálculo de la utilización de la interface de un router (en porcentaje).

Este ejercicio requiere de un proceso de consulta periódica (cada T segundos), además de un scripting en el equipo gestor NMS.

3. Nagios 3

Nagios es una plataforma que permite monitorizar numerosos recursos y servicios de una red. El funcionamiento de Nagios parte simplemente de tres pasos muy simples: obtener, interpretar y decidir. Para ello, estructuralmente, Nagios se va a componer del siguiente modo:

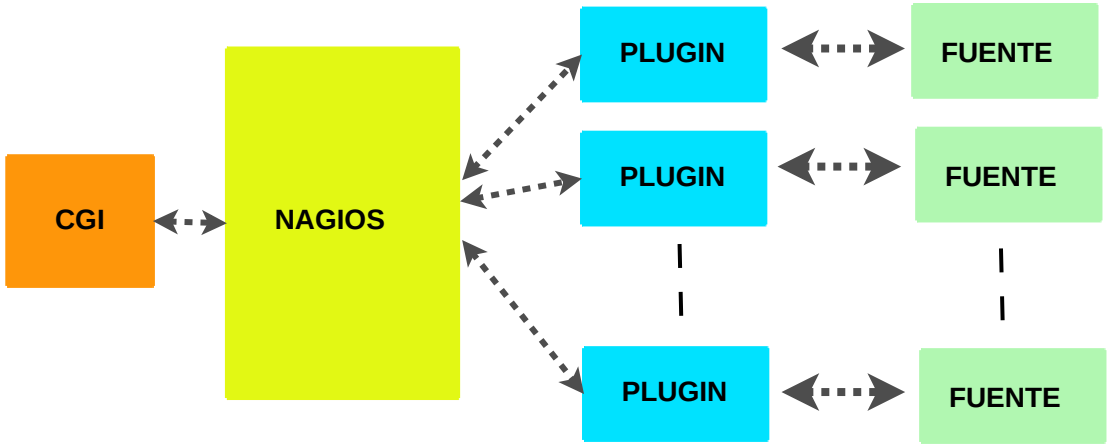

Figura 5: Esquema de funcionamiento de Nagios

Por una parte, los CGIs, son pequeños módulos ejecutables que Nagios va a usar para recopilar información para su funcionamiento (dirigido por el demonio de ejecución Nagios) así como también para la presentación de los datos web. Para ello, los plugins harán de ojos y oídos en las distintas fuentes que se desean monitorizar, de modo que chequearán en las diferentes fuentes si determinados servicios para los que han sido diseñados se encuentren activos, devolviendo a Nagios simplemente un resultado que encuadrará dentro de los siguientes, OK, WARNING, CRITICAL y UNKNOW.

Asimismo, debe facilitársele en la configuración un mapa tanto de los servicios (en caso de implementar alguno a mayores de los básicos) así como de la red, especificando los equipos que la conforman y qué datos se pretender recoger de cada uno, para que Nagios sepa lo que se desea monitorizar en cada caso. Habrá más ficheros configurables, hay opciones de seguridad, y opciones para integrar Nagios con otros programas externos o para manejar de distintos modos las características básicas de Nagios.

Con Nagios, y en conjunción con SNMP, se puede monitorizar desde los servicios software de los nodos hasta parámetros hardware, como carga de CPU o uso de memoria.

3.1. Instalación

En general, todas las distribuciones Linux disponen de un repositorio donde se incluyen los paquetes del sistema Nagios. Será necesario conocer el procedimiento de instalación de paquetes en cada caso. Suele ser recomendable la instalación desde las fuentes del producto, pero ese proceso se deja para usuarios avanzados o profesionales que necesiten tener un control total sobre la versión, parcheado y novedades del producto.

En la mayoría de los casos, los gestores SNMP o en general equipos gestores de sistemas y redes, son equipos basados en Linux, por lo que no se presentarán instalaciones sobre equipos con sistema operativo basado en Windows, dejándose al criterio y trabajo del usuario su instalación e implementación.

3.1.1. Debian Lenny 5, squeeze 6

Aunque se recomienda la instalación desde las fuentes y posterior compilación, se ha optado por la instalación desde los repositorios. Se parte de una instalación de Debian Lenny 5 o Debian Squeeze con apache2 y php5.

1. Descargar e instalar Nagios3.

```
alumno@pcX:/home/alumno$ su
Contrasena:
pcX:/home/alumno# apt-get install nagios3
...........
```

2. Crear el usuario “nagiosadmin”.

```
alumno@pcX:/etc/nagios3$ su
Contrasena:
pcX:/etc/nagios3# htpasswd -c htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
```

3. Habilitar el acceso a través de http.

   Para ello hay que llevar el fichero “/etc/nagios3/apache2.conf” al directorio de sitios disponibles de apache, y activarlo en el de sitos habilitados. Posteriormente hay que reiniciar apache. Opcionalmente se le puede cambiar de nombre.

```
alumno@pcX:/etc/nagios3$ su
Contrasena:
pcX:/etc/nagios3# mv apache2.conf /etc/apache2/sites-available/nagios.conf
pcX:/etc/nagios3# cd /etc/apache2/sites-enabled
pcX:/etc/apache2/sites-enabled# ln -s ../sites-available/nagios nagios.conf
pcX:/etc/apache2/sites-enabled# apache2ctl restart
```

La base del funcionamiento de Nagios es que los hosts aparecen en hostgroups, que los servicios (services) que funcionan en los hosts son comprobados por comandos (commands), y que los servicios pueden ser asociados con hosts o con hostgroups, en cuyo caso, el servicio es comprobado en todos los hosts del grupo.

3.1.2. Raspberry Pi. Debian Squeeze

Al igual que en el caso anterior, se recomienda la instalación desde las fuentes y posterior compilación (ver http://i-security.ro/linux/raspberry-pi-and-nagios-open-source-monitoring/), se ha optado por la instalación desde los repositorios. Se parte de una instalación de Debian Lenny 5 o Debian Squeeze con apache2 y php5.

1. Descargar e instalar Nagios3.

```
alumno@pcX:/home/alumno$ su
Contrasena:
pcX:/home/alumno# apt-get install nagios3
............
```

   En el proceso pide dar de alta la contraseña de “nagiosadmin”.

2. Habilitar el acceso a través de http.

   Ya viene por defecto, ya que existe un enlace entre el fichero “/etc/apache2/conf.d/nagios3.conf“.

```
alumno@PCX:/etc/apache2/conf.d $ ls -l
total 16
...........
lrwxrwxrwx 1 root root   25 Mar 25 01:06 nagios3.conf -> /etc/nagios3/apache2.conf
............
```

   Esto permitiría acceder via navegador desde el directorio raiz al directorio ”nagios3“.

La base del funcionamiento de Nagios es que los hosts aparecen en hostgroups, que los servicios (services) que funcionan en los hosts son comprobados por comandos (commands), y que los servicios pueden ser asociados con hosts o con hostgroups, en cuyo caso, el servicio es comprobado en todos los hosts del grupo.

3.2. Introducción

3.2.1. Estructura de configuración

Nagios se configura mediante ficheros de configuración. El fichero principal de configuración es “/etc/-nagios3/nagios.cfg” donde se define la ubicación de los restantes ficheros de configuración y servicios; los restantes ficheros de configuración se encuentran en el mismo directorio o en “/etc/nagios3/conf.d/” y los

ficheros de objetos están concentrados en el directorio "/etc/nagios3/objects". Los iconos se encuentran en "/usr/share/nagios3/htdocs/images/logos/", donde las extensiones "'.gd2" se utilizan para la representación en el mapa global y las extensiones ".png" se utilizan al lado del nombre del dispositivo en la representación de los servicios.

La configuración de los principales plugins se encuentra en el directorio "/etc/nagios-plugins/config/", y los plugins propiamente dichos se encuentran en "/usr/lib/nagios/plugins/"

Hay que resaltar los siguientes ficheros de configuración en el directorio principal:

- nagios.cfg : fichero principal. Lleva la configuración básica, destacando los ficheros de esquemas, hosts, etc, comenzando todos ellos por "cfg_file". La descripción completa de este fichero puede verse en "http://nagios.sourceforge.net/docs/nagioscore/3/en/configmain.html" para la versión 3.X.
- commands.cfg : define algunos comandos básicos como la notificación por correo. Se suele indicar explícitamente en el fichero "nagios.cfg".
- cgi.cfg: fichero que gestiona la sección web de la plataforma.
- resource.cfg: fichero que contiene algunas macros utilizadas por el programa. En particular, contiene el path a los plugins como la macro $USER1$.

Nagios monitoriza **servicios** ubicados en **hosts**. Para definir los hosts y los servicios utiliza plantillas (*templates.cfg) y otros recursos útiles en la gestión, como la agrupación de hosts (hostgroups.cfg), la definición de contactos (contacts.cfg) o la definición de espacios temporales (timeperiods.cfg).

La estructura de directorios y ficheros de configuración es generada a voluntad por el administrador de nagios, teniendo que ser indicada mediante las clausulas "cfg_file" y "cfg_dir" en el fichero de configuración principal "nagios.cfg". Todos los ficheros indicados en "cfg_file" y contenidos en los directorios indicados en "cfg_dir" que tengan como extensión ".cfg" serán leidos por Nagios. Esto puede provocar ciertas duplicidades en las definiciones si no se tiene bien estructurado el esquema de configuración, lo que dará errores en el reinicio del servicio.

Dependiendo de la distribución utilizada, la instalación de Nagios genera una estructura por defecto en "/etc/nagios3" (ver documentación específica de la distribución), que básicamente suele contener los directorios "conf.d" (conteniendo recursos de configuración) y "objects" (conteniendo las plantillas). El directorio "stylesheets" contiene las css de la web, y no se suele modificar salvo que se quieran cambiar cosas del aspecto. Estos directorios ya contienen ficheros estándar de plantillas o configuración, desde los que partir en el proceso de configuración de la red a monitorizar.

Como ejemplo de estructura, se utilizará la que viene por defecto con los ficheros representados (alguno de los cuales se ha tenido que crear ex-proceso, o se le ha cambiado el nombre). Es bueno recordar que los nombres de los ficheros no son críticos, pues en el fichero de configuración principal "nagios.cfg" se puede indicar que lea todos los ficheros con extensión ".cfg" de un directorio, mediante la directiva "cfg_dir":

```
/etc/
    nagios3/
        <topologia>.cfg
        commands.cfg
        objects/
            hosts_templates.cfg
            services_templates.cfg
            contacts_templates.cfg
        conf.d/
            contacts.cfg
            services.cfg
            hostgroups.cfg
            timeperiods.cfg
            extinfo.cfg
    nagios-plugins/
            config/
                <diferentes_comandos>.cfg
```

Todos estos ficheros están relacionados entre sí, de forma que en la definición de alguno de los elementos de un fichero, se utilizan definiciones de otros ficheros. Por ejemplo, en la definición de las plantillas de hosts ("hosts_templates.cfg"), se utilizan contactos definidos en "contacts.cfg" y periodos de tiempo definidos en "timeperiods.cfg". En los manuales de Nagios se sugiere una estructura de relaciones como la presentada en la figura 6, aunque esta puede modificarse o crearse de otra forma.

Se describen a continuación los ficheros mas importantes, y existen ejemplos y plantillas en el directorio "/usr/share/doc/nagios3/examples":

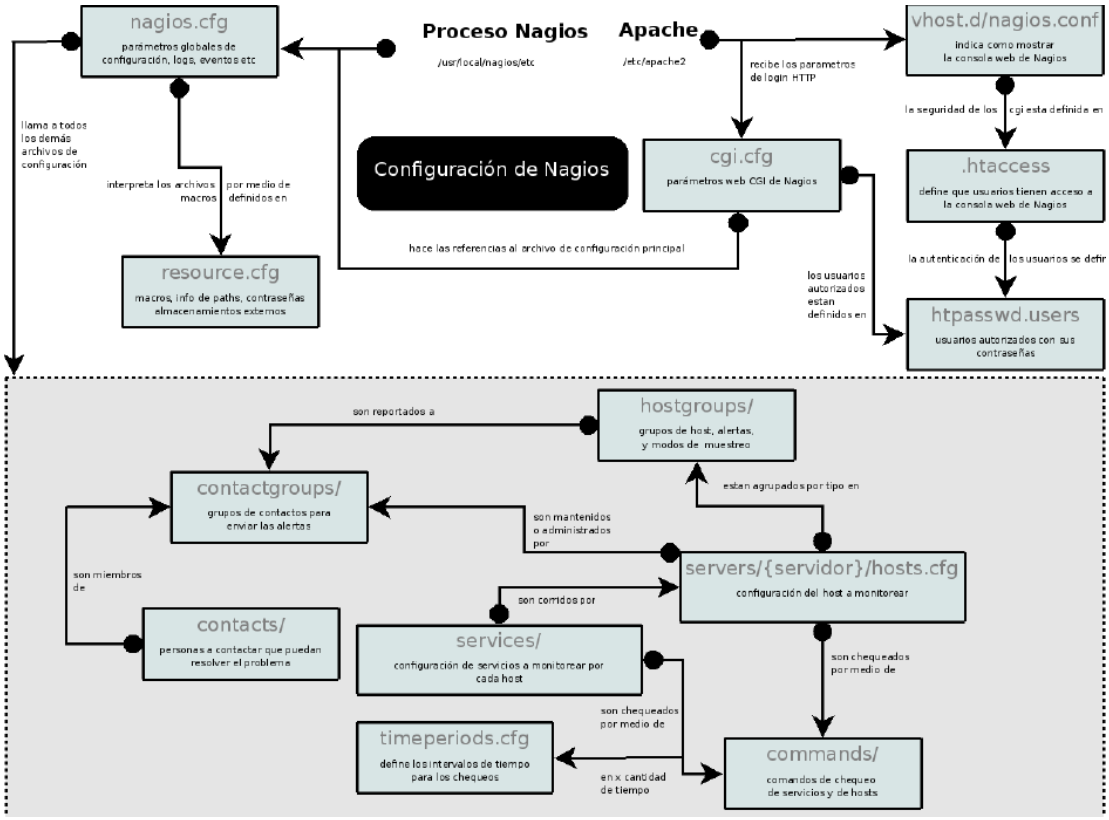

Figura 6: Esquema de configuración de Nagios

- *_templates.cfg : ficheros de plantillas. La configuración de los diferentes elementos de una red en nagios es jerárquica. Se hereda mediante la clausula "use" en los ficheros de configuración de objetos, y se pueden heredar diferentes descripciones de diferentes plantillas. Las descripciones de los objetos en las plantillas son iguales a las de los demás ficheros de configuración, salvo en la clausula "register" que está puesta obligatoriamente a "0" para identificar al objeto como plantilla, en lugar de como objeto a gestionar. Un ejemplo de contenido del fichero "hosts_templates.cfg" sería:

```
# Generic host definition template - This is NOT a real host, just a template!

define host{
    name                    generic-host    ; The name of this host template
    notifications_enabled       1           ; Host notifications are enabled
    event_handler_enabled       1           ; Host event handler is enabled
    flap_detection_enabled      1           ; Flap detection is enabled
    failure_prediction_enabled  1           ; Failure prediction is enabled
    process_perf_data           1           ; Process performance data
    retain_status_information   1           ; Retain status information across program restarts
    retain_nonstatus_information    1           ; Retain non-status information across program ↩
        restarts
    notification_period     24x7        ; Send host notifications at any time
    register            0           ; DONT REGISTER THIS DEFINITION - ITS NOT A REAL HOST, JUST↩
         A TEMPLATE!
}


# Linux host definition template - This is NOT a real host, just a template!

define host{
    name                linux-server    ; The name of this host template
    use             generic-host    ; This template inherits other values from the generic-host↩
         template
    check_period            24x7        ; By default, Linux hosts are checked round the clock
    check_interval          5       ; Actively check the host every 5 minutes
    retry_interval          1       ; Schedule host check retries at 1 minute intervals
    max_check_attempts      10      ; Check each Linux host 10 times (max)
        check_command               check-host-alive ; Default command to check Linux hosts
    notification_period     workhours   ; Linux admins hate to be woken up, so we only notify ↩
        during the day
                            ; Note that the notification_period variable is being overridden ↩
                                from
```

```
                            ; the value that is inherited from the generic-host template!
    notification_interval       120     ; Resend notifications every 2 hours
    notification_options        d,u,r       ; Only send notifications for specific host states
    contact_groups          admins      ; Notifications get sent to the admins by default
    register            0       ; DONT REGISTER THIS DEFINITION - ITS NOT A REAL HOST, JUST A ↩
        TEMPLATE!
    }
.........................................
```

- services.cfg : fichero que indica los servicios a monitorizar y en qué equipos o grupos de equipos se monitorizan. Se propone un ejemplo en la sección 3.2.3. Un ejemplo de contenido de este fichero sería:

```
# check that web services are running
define service {
    hostgroup_name          http-servers
    service_description     HTTP
    check_command           check_http
    use             generic-service
    notification_interval       0 ; set > 0 if you want to be renotified
}

# check that ssh services are running
define service {
    hostgroup_name      ssh-servers
    service_description SSH
    check_command       check_ssh
    use         generic-service
    notification_interval   0 ; set > 0 if you want to be renotified
}

# check that ping-only hosts are up
define service {
    hostgroup_name          ping-servers
    service_description     PING
    check_command           check_ping!100.0,20%!500.0,60%
    use             generic-service
    notification_interval       0 ; set > 0 if you want to be renotified
}
.........................................
```

- hostgroups.cfg : fichero que define grupos de hosts. Los hosts individuales se definen en el fichero de topología, que se presenta en la sección 3.2.2. Un ejemplo de contenido se muestra a continuación:

```
# Some generic hostgroup definitions

# A simple wildcard hostgroup
define hostgroup {
    hostgroup_name  all
    alias       All Servers
    members     *
    }

# A list of your Debian GNU/Linux servers
define hostgroup {
    hostgroup_name      debian-servers
    alias           Debian GNU/Linux Servers
    members         localhost
    }

# A list of your web servers
define hostgroup {
    hostgroup_name  http-servers
    alias           HTTP servers
    members         localhost
        }

# A list of your ssh-accessible servers
define hostgroup {
    hostgroup_name  ssh-servers
    alias           SSH servers
    members         localhost
}

# nagios doesn't like monitoring hosts without services, so this is
# a group for devices that have no other "services" monitorable
# (like routers w/out snmp for example)
define hostgroup {
    hostgroup_name  ping-servers
    alias           Pingable servers
    members         *
}
```

```
.....................................
```

- contacts.cfg : fichero que define los contactos a los que avisar de determinados eventos. Este fichero suele incluir los grupos de contactos. Un ejemplo del contenido sería:

```
# Contact definition to receive alerts.

define contact{
        contact_name                    root
        alias                           Root
        service_notification_period     24x7
        host_notification_period        24x7
        service_notification_options    w,u,c,r
        host_notification_options       d,r
        service_notification_commands   notify-service-by-email
        host_notification_commands      notify-host-by-email
        email                           root@localhost
        }

define contact{
        contact_name                    miguel
        alias                           Miguel
        service_notification_period     24x7
        host_notification_period        24x7
        service_notification_options    w,u,c,r
        host_notification_options       d,r
        service_notification_commands   notify-service-by-email
        host_notification_commands      notify-host-by-email
        email                           mdcacho@gmail.com
        }

# Contact groups.

define contactgroup{
        contactgroup_name       admins
        alias                   Nagios Administrators
        members                 root,miguel
        }
.....................................
```

- timeperiods.cfg : este fichero define los espacios temporales que se utilizarán para la comprobación de servicios y el envío de avisos. Un ejemplo de contenido sería:

```
# timeperiods.cfg

define timeperiod{
        timeperiod_name 24x7
        alias           24 Hours A Day, 7 Days A Week
        sunday          00:00-24:00
        monday          00:00-24:00
        tuesday         00:00-24:00
        wednesday       00:00-24:00
        thursday        00:00-24:00
        friday          00:00-24:00
        saturday        00:00-24:00
        }

# Here is a slightly friendlier period during work hours
define timeperiod{
        timeperiod_name workhours
        alias           Standard Work Hours
        monday          09:00-17:00
        tuesday         09:00-17:00
        wednesday       09:00-17:00
        thursday        09:00-17:00
        friday          09:00-17:00
        }

# The complement of workhours
define timeperiod{
        timeperiod_name nonworkhours
        alias           Non-Work Hours
        sunday          00:00-24:00
        monday          00:00-09:00,17:00-24:00
        tuesday         00:00-09:00,17:00-24:00
        wednesday       00:00-09:00,17:00-24:00
        thursday        00:00-09:00,17:00-24:00
        friday          00:00-09:00,17:00-24:00
        saturday        00:00-24:00
        }
```

```
# This one is a favorite: never :)
define timeperiod{
        timeperiod_name never
        alias           Never
        }
.....................................
```

3.2.2. Topología

La creación de los hosts a monitorizar incluye la propia topología y enlaces entre ellos. Todo ello se puede realizar en un solo fichero de configuración, que deberá ser indicado en el fichero principal mediante la directiva "cfg_file". Los diferentes hosts utilizarán las plantillas definidas anteriormente para simplificar el diseño.

Se creará un fichero llamado "red_laboratorio.cfg" en el directorio principal (/etc/nagios3) que servirá como ejemplo de configuración para la modificación y reestructuración de la red del laboratorio presentada en la figura 1. En el fichero de configuración principal "nagios.cfg" se incluirá la línea siguiente:

```
# Network topology
cfg_file=/etc/nagios3/red_laboratorio.cfg
```

El fichero "red_laboratorio.cfg" tiene una estructura como la que se presenta, con direcciones IP por defecto y resumida para no incluir los 20 equipos de usuario del aula.

```
# Define a host for the local machine

define host{
    use         linux-server    ; Name of host template to use
    ; This host definition will inherit all variables that are defined
    ; in (or inherited by) the linux-server host template definition.
    host_name       localhost
    alias           S1
    icon_image      debian.png
    statusmap_image     debian.gd2
    address         127.0.0.1
    }


define host{
    use                 generic-switch
    host_name       GW1
    alias           GW1
    icon_image      firewall.png
    statusmap_image     firewall.gd2
    address         192.168.1.1
    parents         localhost
    }

define host{
    use                 windows-server
    host_name       PC10
    alias           PC10
    icon_image      pc.png
    statusmap_image     pc.gd2
    address         192.168.1.110
    parents         GW1
    }

define host{
    use                 windows-server
    host_name       PC11
    alias           PC11
    icon_image      pc.png
    statusmap_image     pc.gd2
    address         192.168.1.111
    parents         GW1
    }

define host{
    use                 windows-server
    host_name       PC12
    alias           PC12
    icon_image      pc.png
    statusmap_image     pc.gd2
    address         192.168.1.112
    parents         GW1
    }
.....................................
```

Nagios usa una estructura jerárquica en la que cada objeto hereda las características definidas previamente en las plantillas que está usando mediante la clausula "use". La declaración de estas plantillas se hace en los archivos *_templates.cfg. Allí se puede ver, por ejemplo, el contenido del objeto linux-server que, a su vez, se deriva de otro denominado generic-host. Por defecto, existen 4 plantillas (linux-server, windows-server, generic-printer, generic-switch), aunque se pueden crear tantas como se considere necesario.

Hay que resaltar también la clausula parents mediante la que se especifican las conexiones entre los diferentes hosts. En principio, la única máquina a la que no se le asigna un padre es a la que tiene Nagios instalado. Esa máquina es el núcleo del sistema de monitorización. Una máquina puede tener más de un host como padre. Para ello basta con especificar los nombres de todos los padres que se consideren necesarios separados por comas.

Las clausulas "host_name", "alias" y "address" especifican el nombre del dispositivo, el alias con el que se le conocerá y la dirección IP que tiene configurada. Las clausulas icon_image y statusmap_image no son necesarias mas que para embellecer el esquema; los ficheros de imágenes pueden descargarse de "nagios exchange" un repositorio de utilidades para Nagios que se encuentra en su web, y se almacenan en "/usr/local/nagios3/htdocs/images/logos".

3.2.3. Servicios

El objetivo de esta plataforma es presentar los resultados de operación de determinados servicios de red, desde la disponibilidad de equipos o consumos de capacidad de red, hasta el funcionamiento de servicios como bases de datos, servidores http, sistema de correo, o cualquier otro que se pueda monitorizar. Además de la presentación de los resultados de las consultas sobre el estado de estos servicios, Nagios posibilita unos umbrales de monitorización con tres posibles resultados: Operación normal (OK), Warning, Critical, Unknown.

Los servicios que monitoriza se consultan mediante plugins, ubicados en "/usr/lib/nagios/plugins" y que en general son ejecutables que devuelven un valor resultado de la consulta y unas alertas en función del valor y de los umbrales propuestos. Un ejemplo puede verse a continuación.

```
PCX$ /usr/lib/nagios/plugins/check_ping -H 10.11.12.206 -w 200.0,20% -c 600.0,60% -p 5
```

En este plugin, la opción "-c" indica el valor umbral para considerar un resultado crítico (en el ejemplo sería cuando el round trip time sobrepasa los 600 mseg o hay mas del 60 % de paquetes perdidos) y "-w" para considerarlo en situación de warning. Estas opciones pueden ser diferentes en otros plugins.

Una de las grandes ventajas de Nagios es la posibilidad de desarrollar plugins a medida, con la existencia de librerias y documentación al respecto, además de existir un catálogo muy amplio de los mismos. La documentación de plugins disponibles y para desarrollar plugins propios se encuentra en "http://nagiosplugins.org/".

En función del resultado, en la plataforma Nagios se visualizarán los estados del servicio, y se podrán generar alertas por SMS o correo-e.

3.3. Desarrollo

El entorno nagios se arranca en el sistema ejecutando el comando

```
PCx # /etc/init.d/nagios3  start
```

Si ha habido algún fallo en la configuración mandará un mensaje en el proceso de arranque y abortará el proceso.

La visualización del entorno se realiza entrando desde un navegador web a la dirección localhost o a la dirección IP del servidor Nagios si se accede remotamente, y en el cuadro de dialogo se le dan como usuario "nagiosadmin" y la contraseña antes indicada.

```
http://<direccion_IP_equipo_Nagios>/nagios3
```

La práctica arranca con el esquema de la red plana inicial definida en el fichero "red_plana.cfg" que puede visualizarse en la figura 7.

Se pide implementar todo el esquema utilizando los ficheros propuestos y ampliar la red a monitorizar.

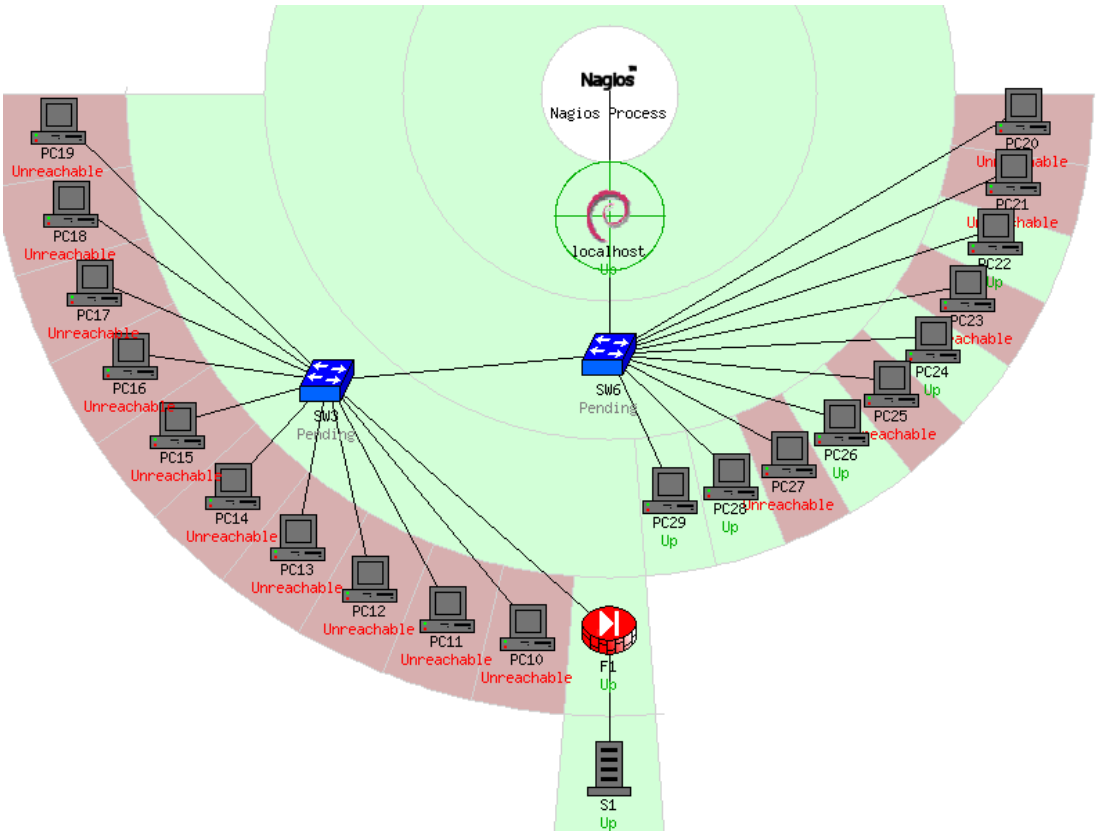

Figura 7: Topología de red plana del laboratorio

3.3.1. Disponibilidad

La monitorización de la disponibilidad se suele hacer mediante el juego de comandos "ping" del protocolo ICMP, y puede configurarse (o incluso cambiarse) en el fichero "services.cfg" (o en otros que nagios cargue de sus directivas "cfg_file" o "cfg_dir". Existe un comando definido en "ping.cfg" llamado "check-host-alive" utilizado por varias plantillas de host. La disponibilidad está por tanto limitada a la capa 3 IP por defecto. Cuando esté definida, todos los dispositivos aparecerán en un estado de disponibilidad Up o Down, apoyados por los colores verde y rojo respectivamente.

La disponibilidad se aplica tanto a hosts como a servicios, y las directivas fundamentales son:

- check_command: comando utilizado para comprobar la disponibilidad. Para hosts suele utilizarse un comando derivado del "ping" definido en "/usr/nagios-plugins/config/ping.cfg" llamado "check-host-alive".
- check_interval: directiva que indica el número de unidades de tiempo entre comprobaciones normales. Una comprobación normal se considera cuando un host (o servicio) está en estado UP, o cuando lleva varios intentos en estado DOWN. Las unidades utilizadas por las dos directivas anteriores están definidas por la directiva "interval_lenght" en el fichero principal "nagios.cfg", donde por defecto está puesta a 60, es decir, un minuto.
- retry_interval: indica el intervalo de chequeo que se realiza cuando el host o servicio pasan al estado de DOWN.
- max_check_attempts: indica el número de veces que se realiza el intervalo de chequeo "retry_interval" cuando un host o servico pasan al estado de DOWN. Despues de ese número de intentos, se chequea cada "check_interval".

La figura 8 muestra el sistema de chequeo de disponibilidad con las directivas comentadas.
El formato de la sección "services" podría ser parecido al siguiente:

```
define service{
```

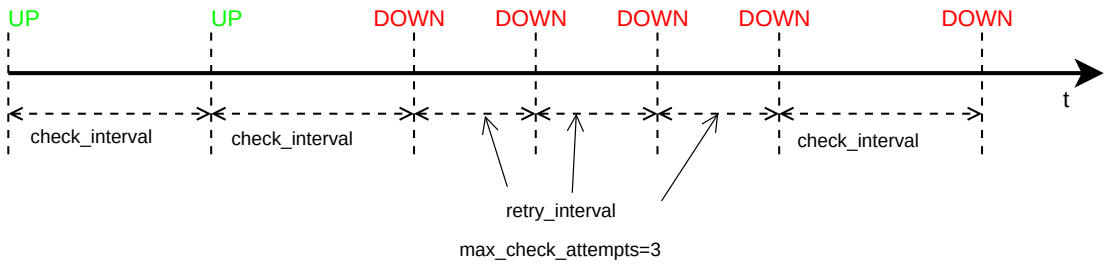

Figura 8: Esquema temporal de parámetros de disponibilidad en Nagios

```
        use             generic-service
        host_name       nagios, GW1, S1, PC10, PC11, Internet
        service_description     PING
        check_command       check_ping!200.0,20%!600.0,60%
        normal_check_interval   5
        retry_check_interval    1
    }
```

La directiva "use" intuye la existencia de una plantilla en el fichero "services_template.cfg" que contiene las especificaciones de "generic-service" y que es utilizada por la definición de este servicio, llamado PING.

La directiva "host_name" informa de los dispositivos a los que se aplica el servicio. En el caso del ejemplo, se aplica a los equipos SW3, SW6, R1, R2, F2, y los equipos de usuario. Puede utilizarse la directiva "hostgroup_name" para indicar grupos completos de dispositivos.

La directiva "check_command" es la que selecciona el comando que realiza el servicio. En este caso es el comando "check_ping!200.0,20 %!600.0,60 %". La definición del comando está en "/etc/nagios-plugins/config/ping.cfg" (o en su defecto en el fichero "commands.cfg") y en él aparece el formato del comando.

La directiva "normal_check_interval" indica que este servicio se ejecuta en condiciones normales cada 5 min, y la directiva "retry_check_interval" indica que en condiciones WARNING o CRITICAL se ejecuta cada 1min.

De forma resumida, en la definición del servicio (*define service*) se especifica el comando *check_ping* definida su sintaxis (*define command*)en *ping.cfg*, donde se utiliza el plugin *check_ping* del directorio de plugins "/usr/lib/nagios/plugins" [4].

```
    # check_ping command definition
    define command {
        command_name        check_ping
        command_line        $USER1$/check_ping -H $HOSTADDRESS$ -w $ARG1$ -c $ARG2$ -p 5
    }
```

$USER1$ es una variable que apunta al directorio donde se encuentran los plugins (/usr/lib/nagios/plugins/) y $HOSTADDRESS$ es la IP del host que se monitoriza. Los argumentos que se invocan separados por signos de exclamación se sustituyen aquí por los $ARG1$ y $ARG2$. Es decir, cuando Nagios lanza esta comprobación para, por ejemplo, S1, está ejecutando lo siguiente:

```
    PCX$ /usr/lib/nagios/plugins/check_ping -H 192.168.10.51 -w 200.0,20% -c 600.0,60% -p 5
```

Básicamente en casi todos los scripts que monitorizan servicios en Nagios, existe un umbral que es considerado como warning y otro como critical (-w y -c respectivamente). Para saber el formato del comando, basta con ejecutarlo con un "-h". Las directivas "normal_check_interval" y "retry_check_interval" indican el periodo de ejecución del comando en estado normal (o Up) y el período de comprobación cuando entra en estado crítico. Puede ponerse otra directiva "max_check_attempts" que limita el número de veces que se hace la comprobación en estado crítico.

3.3.2. SNMP

Existe un plugin llamado *check_snmp* que permite en su sintaxis seleccionar la MIB a consultar sobre el dispositivo seleccionado. Esta solución de monitorización es extremadamente útil para dispositivos con

[4] Nota: no confundir el plugin *check_ping* con el comando del mismo nombre, pues el primero tiene una sintaxis definida por el desarrollador del plugin, y el segundo tiene la sintaxis definida en *ping.cfg*.

agente SNMP, pues permite conocer y generar política de avisos sobre el estado de las interfaces de conmutadores, rutas en los enrutadores, reglas en los cortafuegos, ocupación de disco, de memoria y un sinfín de otras variables.

Como ejemplo, se va a proceder a conocer el tiempo que llevan encendidos los dispositivos gestionados, y que el servicio se ponga en estado warning cuando lleve mas de 5 dias encencido. Para ello, previamente se ha activado el servicio SNMP en los dispositivos a monitorizar, en este caso en GW1 y S1.

La sintaxis básica del plugin *check_snmp* es [5]:

```
check_snmp -H <ip_address> -o <OID> [-w warn_range] [-c crit_range] [-C community] [-P snmp ↩
    version]
```

Será necesario definir el comando, basándose en la sintaxis del plugin, por lo que en el fichero "/etc/nagios-plugins/config/snmp.cfg" habrá que añadir las líneas

```
.....................
define command{
    command_name check_public_oid
    command_line /usr/lib/nagios/plugins/check_snmp -H $HOSTADDRESS$ -C public -o $ARG1$
}
.....................
```

En el fichero "/etc/nagios3/conf.d/hostgroups.cfg" se definen los grupos sobre los que aplicar los servicios. Habrá que añadir entonces:

```
.............................
define hostgroup {
    hostgroup_name      snmp-devices
    alias           Devices with SNMP
    members         GW1,S1
}
.............................
```

El dato "uptime" se ofrece generalmente en la OID 1.3.6.1.2.1.1.3.0, que devuelve timeclicks. Cada uno de esos timeclicks representan 10 milisegundos. Por tanto, el estado de WARNING se debe activar con un valor igual o superior a $5 * 24 * 3600 * 100 = 43200000$.

En el fichero "/etc/nagios3/conf.d/services.cfg" se definen qué servicios aplicar a cada grupo. Habrá que añadir entonces:

```
.................................
# check via snmp uptime
define service{
    use         generic-service ; Inherit values from a template
    hostgroup_name      snmp-devices
    service_description Uptime
    check_command       check_public_oid!1.3.6.1.2.1.1.3.0 -w 43200000
}
.................................
```

Existe la posibilidad de definir un servicio para un sólo host, o un grupo de varios sin que estén en previamente definidos en un grupo. En ese caso sería:

```
.................................
# check via snmp uptime
define service{
    use         generic-service ; Inherit values from a template
    hostname        S1
    service_description Uptime
    check_command       check_snmp!1.3.6.1.2.1.1.3.0
}
.................................
```

En este ejemplo no se han definido valores límite.

También se pueden visualizar valores que se generaron en agentes SNMP de forma personalizada (Ver sección 2.6.2) utilizando el script *check_snmp_extend.sh* y guardandolo en el directorio de plugins de Nagios */usr/local/nagios/libexec*. Habría a su vez que añadir una definición de comando y de servicio:

[5] Nota: la sintaxis completa de la mayoría de los plugins puede verse en http://nagiosplugins.org/man, o ejecutandolo directamente sin parámetros

```
define command{
      command_name  check_snmp_extend
      command_line  $USER1$/check_snmp_extend.sh $HOSTADDRESS$ $ARG1$
}

define service{
      use           generic-service
      host_name         remote.server
      service_description   SomeService status
      check_command     check_snmp_extend!servicename
}
```

3.3.3. Otros servicios

Existen otros plugins ampliamente utilizados para la monitorización de estado de servidores web, de correo, ssh, etc. Será necesario conocer la sintaxis del plugin para poder definir los comandos correctamente. Estos servicios en general no disponen de OID tipo SNMP, por lo que suele ser necesario utilizar otros recursos.

Como ejemplo, por defecto están activados plugins para conocer el estado de servidores HTTP y SSH. Las opciones por defecto de estos plugins cubren una buena parte de los requerimientos, por lo que podrá utilizarse como comando el propio plugin. Estos plugins no utilizan prácticamente opciones, y símplemente comprueban la disponibilidad de los puertos asociados a los servicios (HTTP puerto 80, y SSH puerto 22)

```
.................................
# A list of your Debian GNU/Linux servers
define hostgroup {
    hostgroup_name  debian-servers
    alias           Debian GNU/Linux Servers
    members         localhost
}

# A list of your web servers
define hostgroup {
    hostgroup_name  http-servers
    alias           HTTP servers
    members         localhost,S1
}

# A list of your ssh-accessible servers
define hostgroup {
hostgroup_name  ssh-servers
    alias           SSH servers
    members         localhost,S1
}
.................................
```

```
.................................
# check that web services are running
define service {
    hostgroup_name      http-servers
    service_description HTTP
    check_command       check_http
    use         generic-service
    notification_interval   0 ; set > 0 if you want to be renotified
}

# check that ssh services are running
define service {
    hostgroup_name      ssh-servers
    service_description SSH
    check_command       check_ssh
    use         generic-service
    notification_interval   0 ; set > 0 if you want to be renotified
}
.................................
```

3.4. Sistema de avisos

Los avisos se configuran tanto en los servicios como en los hosts. Estos se generan cuando aparece un estado tipo HARD, o cuando ha pasado un tiempo *notification_interval* desde el paso a estado HARD o desde el último aviso. Los estados tipo HARD ocurren cuando un host o servicio está caido y sigue caido despues de *max_check_attemps* intentos; se entiende por caido un estado de "WARNING" o "CRITICAL".

En la definición del servicio o host, los miembros del grupo nombrado en la clausula *contactgroups* son los que reciben el aviso.

Para que se envie un aviso, se tienen que pasar varios filtros. El primero es que en el fichero principal "nagios.conf" tiene ue estar activa la clausula *enable_ notifications*. El siguiente filtro es que el momento en que se debe producir el aviso no esté dentro del periodo de tiempo incluido en la clausula *scheduled downtime* de la definición del servicio o host. También se comprueba si el servicio o host está en estado de "flapping". Después se comprueban las diferentes clausulas de notificaciones de las definiciones del servicio o host. Seguidamente se comprueban los periodos de tiempo mediante la clausula *notification_ interval*. Finalmente, cada contacto tiene sus propias opciones de notificación en su definición, que tienen que cumplirse antes de enviar el aviso.

Un ejemplo de avisos se puede encontrar en la sección 3.6.4

3.4.1. Aviso por correo-e

Los métodos de aviso pueden ser muy diversos, aunque por defecto se utiliza el correo-e. Para ello tiene que estar configurado un sistema de correo en el servidor. Puede encontrarse una descripción detallada en "http://www.telnetport25.com/2012/02/configuring-e-mail-notifications-in-nagios-core/". Existe también la posibilidad de utilizar un redirector de correo (e-mail forwarder), como el *SSMTP*, partiendo de una cuenta ya existente.

Es recomendable crear una cuenta específica y asociarla a Nagios y a otros sistemas automatizados.

Existe un plugin para utilizar el "gmail" como sistema de avisos. Básicamente es un sistema de redirección de correo utilizando funciones específicas de python. El plugin se ha descargado de *http://exchange.nagios.org/directory/Plugins/Operating-Systems/Linux/Nagios-Alerts-via-gmail-and-python/details*, y se encuentra instalado en "/usr/lib/nagios/plugins/".

El plugin hay que configurarlo para que use la cuenta específica (en este caso "redesdecomputadorasdos@gmail.com", con la clave "provisional"). Se muestran las líneas del script donde hay que realizar la configuración

```
.................
    server.ehlo('redesdecomputadorasdos@gmail.com')
..............
    server.ehlo('redesdecomputadorasdos@gmail.com')  # say hello again
    server.login('redesdecomputadorasdos@gmail.com', 'provisional')

    server.sendmail('redesdecomputadorasdos@gmail.com', Addresses, "Subject: " + subject + '\nTo:' + ↩
        address + '\n\n' + body)

............
```

En el fichero */etc/nagios3/commands.cfg* o en el fichero en el que se encuentren definidos los comandos *notify-host-by-email* y *notify-service-by-email*, estos dos comandos deben incluir el script de python, quedando de la forma:

```
# 'notify-host-by-email' command definition
define command{
        command_name    notify-host-by-email
        command_line /usr/bin/python /usr/lib/nagios/plugins/send_gmail.py -a $CONTACTEMAIL$ -b "↩
            ***** Nagios *****\nnn\nnnNotification Type: $NOTIFICATIONTYPE$\nnnHost: $HOSTNAME$\↩
            nnnState: $HOSTSTATE$\nnnAddress: $HOSTADDRESS$\nnnInfo: $HOSTOUTPUT$\nnnDate/Time: ↩
            $LONGDATETIME$" -s "** Nagios $NOTIFICATIONTYPE$ Host Alert: $HOSTNAME$ is $HOSTSTATE$ ↩
            **"
        }


# 'notify-service-by-email' command definition
define command{
        command_name    notify-service-by-email
        command_line /usr/bin/python /usr/lib/nagios/plugins/send_gmail.py -a $CONTACTEMAIL$ -b "↩
            ***** Nagios *****\nnn\nnnNotification Type: $NOTIFICATIONTYPE$\nnnService: ↩
            $SERVICEDESC$\nnnHost: $HOSTALIAS$\nnnAddress: $HOSTADDRESS$\nnnState: $SERVICESTATE$\↩
            nnnDate/Time: $LONGDATETIME$\nnnAdditional Info: $SERVICEOUTPUT$" -s "** Nagios ↩
            $NOTIFICATIONTYPE$ Service Alert: $HOSTALIAS$
/$SERVICEDESC$ is $SERVICESTATE$ **"
 }
```

3.4.2. Event handlers

Existe la posibilidad de programar los manejadores de eventos o "event handlers". Un event handler es un programa que intenta resolver un problema de forma proactiva, como reiniciar un servicio caido, escribir en una base de datos un evento, etc. Suelen utilizarse en estados tipo SOFT.

3.5. Visualización Web

La visualización de los resultados de la monitorización se hace a través de http, y por tanto se accede a ellos a través de un navegador.

La página principal de nagios presenta enlaces de acceso a los servicios y hosts. Se sugiere visualizar la topología de red configurada en el enlace "map" a la izquierda de la pantalla.

Las pantallas de estátus (si los equipos o servicios están accesibles o nó) se actualizan a baja frecuencia. La gestión de la web viene en el fichero */etc/nagios3/cgi.cfg*, del que cabe resaltar la directiva "refresh_rate" que mide el periodo de refresco en segundos de varios CGI's (status, statusmap, extinfo y outages).

```
.................
refresh_rate=60
.................
```

La visualización del funcionamiento de los servicios y hosts no se hace en tiempo real ya que se requeririan incontables recursos de red en redes grandes. No obstante, algunos equipos o servicios pueden necesitar un conocimiento de su estado con mayor rapidez. Para ello se manejan las siguientes directivas, ya vistas en la sección 3.3.1 y algunas de ellas pueden visualizarse en la figura 8:

- interval_lenght: directiva que se encuentra en */etc/nagios3/nagios.cfg* y que indica la unidad de tiempo mínima en la que se miden los periodos de chequeo.
- check_interval: directiva que indica el número de unidades de tiempo entre comprobaciones normales. Una comprobación normal se considera cuando un host (o servicio) está en estado UP, o cuando lleva varios intentos en estado DOWN. Las unidades utilizadas por las dos directivas anteriores están definidas por la directiva "interval_lenght" en el fichero principal "nagios.cfg", donde por defecto está puesta a 60, es decir, un minuto.
- retry_interval: indica el intervalo de chequeo que se realiza cuando el host o servicio pasan al estado de DOWN.
- max_check_attempts: indica el número de veces que se realiza el intervalo de chequeo "retry_interval" cuando un host o servico pasan al estado de DOWN. Despues de ese número de intentos, se chequea cada "check_interval".

Por tanto, si se dispone a monitorizar con una frecuencia importante un determinado servicio o equipo, habrá que hacer:

1. Utilizar un refresh_rate de pantalla bastante bajo, por ejemplo 10 segundos. Aunque haya equipos sobre los que no sea necesario conocer su estado con rapidez, dominan los otros.

```
.................
refresh_rate=10
.................
```

2. Definir una directiva "interval_lenght" muy baja (se recomienda no bajar de los 10 segundos, para que no haya posibilidad de comprobar disponibilidades con un periodo menor que ese).

```
.................
interval_lenght=10
.................
```

3. En cada definición de servicio o host que tenga requerimientos de casi tiempo real, se le añade la directiva "check_interval" con el menor valor posible. A los equipos que no lo necesiten se dejan valores bastante mas altos (del orden de 20 o 30.

```
.................
check_interval=1
.................
```

4. En cada definición de servicio o host que tenga requerimientos de casi tiempo real, se pone la directiva "max_check_attempts" a un valor bajo (p. ej. 2) para que indique que está caido a los dos errores de conexión consecutivos. Al resto se le deja con valores en torno a 5..

```
.................
max_check_attempts=1
.................
```

Para personalizar imágenes e iconos que se presentan en la topología, se pueden usar iconos que pueden descargarse de "nagios exchange" un repositorio de utilidades para Nagios que se encuentra en su web, y se almacenan en "/usr/local/nagios3/htdocs/images/logos". Las clausulas "icon_image" y "statusmap_image" gestionan el uso de esos iconos. Las extensiones "'.gd2" se utilizan para la representación en el mapa global y las extensiones ".png" se utilizan al lado del nombre del dispositivo en la representación de los servicios.

3.6. Ejercicios propuestos

3.6.1. Crear estructura de directorios y ficheros

La estructura de ficheros y directorios puede crearse de forma personalizada.

Se deberá crear un directorio */home/alumno/nagios/<GRUPO_nombre>*, en el que se almacenarán ficheros y directorios de configuración. En el fichero *nagios.cfg*, las directivas referentes a ficheros (cfg_file) y directorios (cfg_dir) de configuración deberá tener como directorio base el específico de cada usuario (*/etc/nagios3/<GRUPO_nombre>*), a excepción de las mas generales, como *cfg_dir=/etc/nagios-plugins/config* o las referidas a cachés y a procesos.

De esa forma, suponiendo el usuario con nombre "mcacho" perteneciente al grupo G1, el fichero */etc/nagios3/nagios.cfg* tendrá como únicas directivas *cfg_file* y *cfg_dir*:

```
.................
cfg_dir=/home/alumno/nagios/G1_mcacho
.................
cfg_dir=/etc/nagios-plugins/config
.................
```

Todas las modificaciones que se hagan desde ese momento, se harán en ficheros de ese directorio creado.

Para mantener varias configuraciones, se pide que el fichero */etc/nagios3/nagios.cfg* que trabaje cada usuario, sea renombrado al final de la práctica como */etc/nagios3/<GRUPO_nombre>/nagios.cfg.<GRUPO_nombre>*.
Cuando se inicie la práctica, se deberá hacer la copia inversa, es decir, el fichero */etc/nagios3/nagios.cfg.<GRUPO_nombre>* se deberá copiar como */etc/nagios3/nagios.cfg*.

Tras ello, se utilizará la estructura propuesta en la sección 3.2.1.

La configuración mínima que necesita nagios incluye al menos un host, un servicio y un contacto, y dentro del contacto que estén definidos los comandos de notificaciones de servicios y de hosts. Los nombres de estos ficheros pueden cambiar.

- El host deberá definirse en el fichero "red_plana.cfg"
- El contacto deberá definirse en el fichero "contacts.cfg"
- El servicio deberá definirse en el fichero "services.cfg"
- Los comandos de notificaciones de servicios y hosts están ya definidos en el fichero "commands.cfg".

Para la configuración mínima se proponen los siguientes contenidos:

- Fichero "red_plana.cfg":

```
define host{
    host_name       localhost
    alias           Localhost
    address         127.0.0.1
    max_check_attempts  5
    }
```

- Fichero "services.cfg":

```
define service{
    host_name       localhost
    service_description HTTP
    check_command       check_http
    max_check_attempts  5
    }
```

El comando *check_ http* está ubicado en uno de los ficheros del directorio */etc/nagios-plugins/config*

- Fichero "contacts.cfg":

```
define contact{
    contact_name        root
    host_notification_commands  notify-host-by-email
    service_notification_commands   notify-service-by-email
    email           root@localhost
    }
```

- Fichero "commands.cfg":

  Este fichero ya suele venir en la instalación por defecto y se encuentra en */etc/nagios3/commands.cfg*. Bastará con copiarlo al directorio de usuario que se ha creado anteriormente. El contenido de ese fichero puede tener la forma:

```
# 'notify-host-by-email' command definition
define command{
    command_name    notify-host-by-email
    command_line    /usr/bin/printf "%b" "***** Nagios *****\n\nNotification Type: ↩
        $NOTIFICATIONTYPE$\nHost: $HOSTNAME$\nState: $HOSTSTATE$\nAddress: $HOSTADDRESS$\nInfo:↩
         $HOSTOUTPUT$\n\nDate/Time: $LONGDATETIME$\n" | /usr/bin/mail -s "** $NOTIFICATIONTYPE$↩
         Host Alert: $HOSTNAME$ is $HOSTSTATE$ **" $CONTACTEMAIL$
    }

# 'notify-service-by-email' command definition
define command{
    command_name    notify-service-by-email
    command_line    /usr/bin/printf "%b" "***** Nagios *****\n\nNotification Type: ↩
        $NOTIFICATIONTYPE$\n\nService: $SERVICEDESC$\nHost: $HOSTALIAS$\nAddress: $HOSTADDRESS$↩
        \nState: $SERVICESTATE$\n\nDate/Time: $LONGDATETIME$\n\nAdditional Info:\n\↩
        n$SERVICEOUTPUT$" | /usr/bin/mail -s "** $NOTIFICATIONTYPE$ Service Alert: $HOSTALIAS$/↩
        $SERVICEDESC$ is $SERVICESTATE$ **" $CONTACTEMAIL$
    }

define command{
    command_name    process-host-perfdata
    command_line    /usr/bin/printf "%b" "$LASTHOSTCHECK$\t$HOSTNAME$\t$HOSTSTATE$\↩
        t$HOSTATTEMPT$\t$HOSTSTATETYPE$\t$HOSTEXECUTIONTIME$\t$HOSTOUTPUT$\t$HOSTPERFDATA$\n" ↩
        >> /var/lib/nagios3/host-perfdata.out
    }

define command{
    command_name    process-service-perfdata
    command_line    /usr/bin/printf "%b" "$LASTSERVICECHECK$\t$HOSTNAME$\t$SERVICEDESC$\↩
        t$SERVICESTATE$\t$SERVICEATTEMPT$\t$SERVICESTATETYPE$\t$SERVICEEXECUTIONTIME$\↩
        t$SERVICELATENCY$\t$SERVICEOUTPUT$\t$SERVICEPERFDATA$\n" >> /var/lib/nagios3/service-↩
        perfdata.out
    }
```

Como puede verse, los comandos *notify-host-by-email* y *notify-service-by-email* que aparecen en el fichero de definición de contactos *contacts.cfg* están definidos en el fichero *commands.cfg*. El comando *check_ http* está definido en */etc/nagios-plugins/config/http.cfg*.

Todos estos ficheros estarán ubicados en el directorio de usuario creado anteriormente. Ya existe una copia del fichero "commands.cfg" en el directorio principal de nagios. Hay que hacer referencia de todos estos

ficheros (o del directorio donde se ubiquen) dentro del fichero "nagios.cfg" con la directiva "cfg_file" (o con la directiva "cfg_dir"). En el caso de esta práctica no será necesario referenciarlo en */etc/nagios3/nagios.cfg* siempre que se copie al directorio de usuario nagios creado anteriormente.

Tras ello habrá que reiniciar nagios:

```
PCX:# /etc/init.d/nagios3 restart
OK
PCX:#
```

3.6.2. Primer contacto con la disponibilidad

Se sugiere que se creen mas definiciones particularmente de hosts para seguir trabajando la configuración. El objetivo final es crear la topología del laboratorio.

Para comprobar la disponibilidad se añaden en las definiciones de host anteriores la calusula "check_command":

```
define host{
    host_name       F0
    alias           F0
    address         192.168.29.1
    max_check_attempts  2
    check_command       check-host-alive
    check_interval      1
    }
```

Para agilizar el proceso de actualización del estado de disponibilidad, se recomienda ver la sección 3.5 con cautela.

3.6.3. Uso de las directivas "use" y "parents"

Las plantillas pueden crearse en el propio fichero de topología previo *red_plana.cfg* o mas recomendable en un fichero creado ex-proceso, como *templates.cfg*.

Crear una plantilla llamada "PC_laboratorio" y utilizarla en la definición de los equipos de usuario del laboratorio. En dicha plantilla pueden utilizarse directivas "check_command" o "max_check_attempts" o cualquier otra que vaya a ser utilizada de forma bastante común por los PCs del laboratorio.

```
# Generic host definition template - This is NOT a real host, just a template!

define host{
        name                            generic-host    ; The name of this host template
        check_command                   check-host-alive
        max_check_attempts              5
    check_interval          3
        register                        0       ; DONT REGISTER THIS DEFINITION - ITS NOT A REAL HOST↩
            , JUST A TEMPLATE!
        }
```

Hay que recordar que cualquier plantilla tiene que finalizar con la directiva "register" puesta a cero, para indicar que es una plantilla.

3.6.4. Comprobación de disponibilidad de los dispositivos

Se pretende que Nagios monitorice un servicio HTTP de un host servidor S1. Deberá generar un aviso a un usuario "alumno" cuando el servicio pase a un estado de WARNING o CRITICAL. Los estados WARNING o CRITICAL están definidos en el plugin "check_http" (ubicado en "/usr/lib/nagios/plugins/") con un formato parecido al presentado en la sección 3.2.3.

```
    PCX:/usr/lib/nagios/plugins $ ./check_http
check_http: Could not parse arguments
Usage:
 check_http -H <vhost> | -I <IP-address> [-u <uri>] [-p <port>]
       [-w <warn time>] [-c <critical time>] [-t <timeout>] [-L] [-a auth]
       [-b proxy_auth] [-f <ok|warning|critcal|follow|sticky|stickyport>]
       [-e <expect>] [-s string] [-l] [-r <regex> | -R <case-insensitive regex>]
       [-P string] [-m <min_pg_size>:<max_pg_size>] [-4|-6] [-N] [-M <age>]
       [-A string] [-k string] [-S <version>] [--sni] [-C <warn_age>[,<crit_age>]]
       [-T <content-type>] [-j method]
```

Según la documentación del plugin, la respuesta pasa a estado CRITICAL cuando el destino rechaza la conexión HTTP o cuando ha pasado un tiempo *-t timeout* sin respuesta (por defecto son 10 segundos). Si tiene cualquier otro error de conexión pasa al estado UNKNOWN, y si tiene conexión pero la respuesta a la solicitud HTTP es incorrecta, pasa al estado WARNING.

El plugin debe estar incluido en un "comando" que es lo que entiende nagios con directivas como *check_command*. En este caso es el comando *check_http -H '$HOSTADDRESS$' -I '$HOSTADDRESS$'*, que se puede visualizar en */etc/nagios-plugins/config/http.cfg*.

```
# 'check_http' command definition
define command{
        command_name    check_http
        command_line    /usr/lib/nagios/plugins/check_http -H '$HOSTADDRESS$' -I '$HOSTADDRESS$'
        }
```

En ese fichero de comandos se encuentran otros comandos que utilizan el plugin *check_http* con otros argumentos. Si se reproduce el comando con los argumentos necesarios, se puede comprobar la salida del mismo:

```
PCX:/usr/lib/nagios/plugins $ ./check_http -H S1 -I 192.168.29.53
HTTP OK: HTTP/1.1 200 OK - 85072 bytes in 0.657 second response time |time=0.657093s;;;0.000000 size↩
    =85072B;;;0
PCX:/usr/lib/nagios/plugins $
```

```
PCX:/usr/lib/nagios/plugins $ check_http -I 192.168.29.111
Connection refused
HTTP CRITICAL - Unable to open TCP socket
```

Deberán estar definidos como mínimo los siguientes *define*. Estas definiciones pueden estar en un sólo fichero o repartidas en varios, según se ha visto en la estructura de directorios en las secciones 3.2.1 y 3.6.1. Estas definiciones no pueden estar duplicadas.

Básicamente deberá existir el servidor *S1*, que tendrá incluidas o en su directiva global *define host* o en una plantilla mediante la directiva *use* incluida en *define host* lo siguiente:

```
define host{
        use                     generic-host            ; Name of host template to use
        host_name               S1
        parents                 localhost
        address                 192.168.29.53
        }

# Generic host definition template - This is NOT a real host, just a template!

define host{
        name                            generic-host    ; The name of this host template
        notifications_enabled           1       ; Host notifications are enabled
        event_handler_enabled           1       ; Host event handler is enabled
        flap_detection_enabled          1       ; Flap detection is enabled
        failure_prediction_enabled      1       ; Failure prediction is enabled
        process_perf_data               1       ; Process performance data
        retain_status_information       1       ; Retain status information across program restarts
        retain_nonstatus_information    1       ; Retain non-status information across program ↩
            restarts
        check_command                   check-host-alive
        max_check_attempts              10
        notification_interval           0
        notification_period             24x7
        notification_options            d,u,r
        contact_groups                  admins
        register                        0       ; DONT REGISTER THIS DEFINITION - ITS NOT A REAL HOST↩
            , JUST A TEMPLATE!
        }

# check that web services are running
define service {
        host_name                   S1
        service_description             HTTP_S1
        check_command                   check_http
        use                             generic-service
        notification_interval           0 ; set > 0 if you want to be renotified
}

# generic service template definition
define service{
        name                            generic-service ; The 'name' of this service template
```

```
        active_checks_enabled           1       ; Active service checks are enabled
        passive_checks_enabled          1       ; Passive service checks are enabled/accepted
        parallelize_check               1       ; Active service checks should be parallelized (↩
            disabling this can lead to major performance problems)
        obsess_over_service             1       ; We should obsess over this service (if necessary)
        check_freshness                 0       ; Default is to NOT check service 'freshness'
        notifications_enabled           1       ; Service notifications are enabled
        event_handler_enabled           1       ; Service event handler is enabled
        flap_detection_enabled          1       ; Flap detection is enabled
        failure_prediction_enabled      1       ; Failure prediction is enabled
        process_perf_data               1       ; Process performance data
        retain_status_information       1       ; Retain status information across program restarts
        retain_nonstatus_information    1       ; Retain non-status information across program ↩
            restarts
        notification_interval           0       ; Only send notifications on status change by default↩
            .
        is_volatile                     0
        check_period                    24x7
        normal_check_interval           5
        retry_check_interval            1
        max_check_attempts              4
        notification_period             24x7
        notification_options            w,u,c,r
        contact_groups                  admins
        register                        0       ; DONT REGISTER THIS DEFINITION - ITS NOT A REAL ↩
            SERVICE, JUST A TEMPLATE!
        }
```

El comando *check_http* está definido en */etc/nagios-plugins/config/http.cfg*.

En el fichero *contacts.cfg*, deberá añadirse el contacto *alumno* e incluirlo en el grupo de administradores *admin*:

```
define contact{
    contact_name            alumno
    service_notification_period     24x7
    host_notification_period        24x7
    service_notification_options    w,u,c,r
    host_notification_options       d,r
    host_notification_commands  notify-host-by-email
    service_notification_commands   notify-service-by-email
    email               redesdecomputadorasdos@gmail.com
    }
```

```
#
# CONTACT GROUPS
#
define contactgroup{
        contactgroup_name       admins
        alias                   Nagios Administrators
        members                 root,alumno
        }
```

La definición de los *timeperiods* también deberá existir; en este caso será en el fichero *timeperiods.cfg*. En principio no debería modificarse, y es donde están definidos los terminos 24x7.

Reiniciando nagios, y posteriormente desactivando el servidor HTTP en S1, el usuario *alumno* debería recibir un aviso en forma de correo si se cumplen los filtros presentados en la sección 3.4:

1. En el fichero *nagios.conf* tiene ue estar activa la clausula *enable_notifications 1*.
2. El aviso no se produce en el periodo de tiempo incluido en la clausula *scheduled downtime* de la definición (o plantilla) del servicio.
3. El servicio no está en *flapping* si la clausula *enable_flap_detection=1* en el fichero *nagios.cfg* está activa.
4. Se revisan las clausulas de tipo *notification* en el servicio. Si alguna no se cumple no se envía. No hay que confundir estas clausulas con las de tipo *check*, ya que estas últimas están referidas al descubrimiento del evento que generaría el aviso, y las clausulas *notification* generarían el aviso.
5. Finalmente el contacto o usuario tiene que cumplir también con sus clausulas particulares. Algunas de las clausulas

También se comprueba si el servicio o host está en estado de "flapping". Después se comprueban las diferentes clausulas de notificaciones de las definiciones del servicio o host. Seguidamente se comprueban

Cuadro 1: notification

host_notification_options	d,r,f,n	down,recovery,unreachable,none
service_notification_options	w,c,u,r	warning,critical,unreachable,recovery

los periodos de tiempo mediante la clausula *notification_interval*. Finalmente, cada contacto tiene sus propias opciones de notificación en su definición, que tienen que cumplirse antes de enviar el aviso.

El correo en la cuenta *redesdecomputadorasdos@gmail.com* deberá tener un contenido parecido a:

```
***** Nagios *****

Notification Type: PROBLEM
Service: HTTP
Host: S1
Address: 192.168.29.53
State: CRITICAL
Date/Time: Mon Mar 25 16:54:29 CET 2013
Additional Info: Connection refused
```

3.6.5. Crear topología de la red del laboratiorio

Se creará una topología de la red del laboratiorio, incluyendo los equipos de la red plana (192.168.29.0/24) y los dispositivos de red disponibles, entre los que se encuentran los conmutadores SW3 y SW6, los routers R1 y R2, los cortafuegos F1 y F2, los servidores VPN RB2 y RB4, el router ATM RA1 (actualmente en desuso) y los puntos de acceso inalámbricos AP1, AP2, AP3 y AP4.

El fichero donde está creada la topología se llamará "red_plana.cfg" y se almacenará en la raíz de la estructura de directorios creada en el ejercicio anterior. En el fichero "nagios.cfg" se pondrá en la sección inicial con la directiva *cfg_file*.

4. RRDTOOL

RRDTool es una re-implementación de MRTG también escrita por Tobias Oetiker, pero con muchas menos limitaciones y también totalmente openSource. Es una herramienta que permitirá la supervisión de la red, además de la creación de gráficas sobre los diferentes parámetros de la misma, RRDTool puede graficar sobre cualquier parámetro monitorizable, no necesariamente datos referentes a servicios de red, aunque se utilizará en conjunción con SNMP para la obtención de datos y con técnicas de base de datos Round Robin, de ahí su nombre (Round Robin Database Tool).

Este tipo de bases de datos son denominadas también bases de datos circulares, ya que el concepto gráfico sería un círculo, en el que se pintan puntos (los datos a almacenar), sin importar que haya puntos ya pintados en el mismo lugar, esto es, simplemente, en cada región de la base de datos, se apunta al elemento más reciente. De este modo, el tamaño de la base de datos es siempre el mismo, no crecerá con los datos, esto es una de las características fundamentales de este tipo de sistemas, y que ha permitido su expansión al no necesitar grandes capacidades para el almacenamiento de los históricos.

Habrá tres pasos que se han de llevar a cabo para poder utilizar la herramienta: crear los almacenes de información, obtener y guardar los datos en los mismos y finalmente graficar a partir de dichos datos. Para ello, RRDTool pondrá a disposición herramientas para guardar los datos obtenidos en formato RRD, para recuperarlos y, finalmente, para graficar.

En primer lugar, va a ser necesaria la creación de las bases de datos circulares en las que se almacenan los datos. Para ello, habrá que tener en cuenta los parámetros que definen los tipos de datos que se guardan y lo que se va a registrar, ya que luego se deberá graficar conforme a los mismos y sus tipos. En este caso, han sido creadas varias bases de datos, una para cada nodo a monitorizar (ya que no se pueden registrar datos nulos, es conveniente independizar los nodos y sus bases de datos, de modo que se van a poder hacer pruebas aisladas con unos nodos y no con otros al mismo instante), del siguiente modo:

4.1. Desarrollo

Se especifica, en primer lugar, el nombre de la base de datos (una para cada dispositivo a monitorizar), por ejemplo "SW6.rrd", a continuación, con "-s 1" se especifica que la toma de datos se va a llevar a cabo cada segundo, y seguidamente, se han de definir la siguiente ristra de parámetros: [DS:ds- name:DST:dst arguments] [RRA:CF:cf arguments], cada uno de los cuales se explica a continuación:

- DS:ds-name:DST:dst arguments: DS es el acrónimo de Data Source, y será la especificación de las fuentes de datos que van a ser guardadas. Ds-name será el nombre, mientras que los argumentos definidos después han sido:
  - COUNTER: tipo del dato a guardar, en este caso, especialmente diseñado para contadores, de manera que será perfecto para guardar los octetos referentes al tráfico de los routers, se trata de almacenar una cantidad de datos que nunca decrece en la base de datos (a no ser que haya un desbordamiento de la misma).
  - 60 (HEARTHBEAT): definición del máximo tiempo, en segundos, que pasará desde una toma de datos hasta la siguiente.
  - U:U: especifica el rango (máximo y mínimo) de los valores que se pueden tomar, aunque en caso de ser cualquier valor posible, se puede especificar que será desconocido a través de "U" (Unkown).
- RRA:CF:cf argumentos: parámetros de definición del archivo que albergará la base de datos, esto es, especificará cómo van a ser guardados los datos. En este caso, se han definido:
  - AVERAGE: para guardar los datos, RRDTool utilizará una función de consolidación (CF), esto indica que de todos los datos tomados, se guardará la media de los mismos (podrían guardarse los máximos o mínimos).
  - 0.5 (xff): factor que define que parte de los datos se estiman y los que no en cada intervalo, tomará rangos desde 0 hasta 1.
  - 1 (steps): define cuántos datos son necesarios para guardar en cada intervalo.
  - 300 (rows): define cuántos valores van a ser guardados en la base de datos.

Teniendo en cuenta esto, fijándose en cómo se ha hecho la definición de la base de datos, tenemos que van ha ser guardados 300 datos, tomados de uno en uno y en espacios de tiempo de 1 segundo, esto ha sido realizado de este modo para que el espacio medido sean siempre 300s, o lo que es lo mismo, 5 minutos, que ha sido la duración decidida para las pruebas y será el tiempo que se va a registrar (esto es, se le especifica que se van a guardar octetos durante 5 minutos).

El siguiente paso, será la obtención de los datos, para lo cual ha sido necesaria la creación de scripts que recolectasen los datos y los guardasen en la base de datos RRD.

Un ejemplo de uno de los scripts que han sido desarrollados para uno de los nodos es el siguiente:

```
#recolecta datos AP1
#!/bin/sh
while [ 1 ];
do
        time=`perl -e 'print time, "\n" '`
        FOUND_1_in=`snmpget -v 1 10.138.4.1 -c public 1.3.6.1.2.1.2.2.1.10.3 | cut -d : -f 4`
        FOUND_1_out=`snmpget -v 1 10.138.4.1 -c public 1.3.6.1.2.1.2.2.1.16.3 | cut -d : -f 4`
        FOUND_1_in=$(echo $FOUND_1_in)
        FOUND_1_out=$(echo $FOUND_1_out)
        rrdtool update AP1.rrd N:$FOUND_1_in:$FOUND_1_out
        echo 'rrdtool update AP1.rrd '
        echo $time
        sleep 1
done
```

Simplemente, se obtienen los datos a través de SNMP y se guardan en la base de datos Round Robin creada con anterioridad a través del comando de actualización. Sólo se le facilitará el nombre de la base de datos, en este caso "AP1.rrd" seguido de la hora a la que se asociarán los datos a guardar y a continuación, separado por dos puntos, los datos que correspondan a la fuente de datos definida en la creación. En cuanto a la hora a la que se ligan, se dispondrá en formato UNIX, pero para mayor facilidad, se le va a poder pasar una "N", que abrevia a NOW y se trata de la hora actual.

Finalmente, el paso final va a ser la generación de las gráficas. La potencia real de RRDTool radica en la gran versatilidad a la hora de generar gráficas, para ello, habrá que facilitarle una serie de parámetros de creación de las mismas, en las que se va a poder especificar desde el tipo, forma y color de las líneas a utilizar, hasta los intervalos X e Y de las gráficas, como el modo en el que van a ser pintados los resultados, etc. En este caso, han sido creadas del siguiente modo (por ejemplo, para el nodo AP1):

```
rrdtool graph AP1_genera3.png DEF:inoctets1=AP1.rrd:ap1_in:AVERAGE
DEF:outoctets1=AP1.rrd:ap1_out:AVERAGE AREA:1000#FBBEC8
AREA:1000#c7e3ff COMMENT:"
"
AREA:inoctets1#E5E365:" Trafico de entrada del nodo AP1" COMMENT:"
"
AREA:outoctets1#A5D365:" Trafico de salida del nodo AP1"
 -h 300 -w 750 --x-grid
SECOND:300:SECOND:300:SECOND:30:300:%H:%M:%S -s 1246202275 -e
1246202583 -t "Trafico del nodo AP1" -c BACK#fde7eb --font TITLE:14:Arial --
font LEGEND:10:Arial
```

Como se ve, al poder definir un montón de parámetros diferentes, hay un aumento considerable de la complejidad en cuanto al uso de RRDTool se refiere. La estructura básica para la generación de gráficas a través del comando rrdtool graph es la siguiente:

```
rrdtool graph nombre_del_archivo_resultado [opciones ...] definiciones_de_datos ...]
[calculos_sobre_los_datos ...] [definicion_de_variables ...] [elementos_del_grafico ...]
[elementos_a_imprimir ...]
```

Además de las opciones y demás parámetros que se le pasan en el ejemplo, hay muchas más características que pueden serle facilitadas a RRDTool a la hora de generar gráficas, de modo que no van a ser definidas todas ellas, pero como puntos clave, habrá que definir las transformaciones de las fuentes de datos (DS) definidas en la creación de los RRA, de modo que serán presentados de uno u otro modo, esto, se hará además dándole nombres como si de variables se tratase, para luego aplicarle diferentes reglas, según como se desee que vaya a ser pintado:

- DEF:inoctets1=AP1.rrd:ap1_in:AVERAGE: definición de la variable "inoctect1", que será el DS "ap1_in almacenado" en la base de datos rrd "AP1.rrd" y de tipo media, tal y como se vio en su creación. En este punto podrían llevarse transformaciones de diferente índole sobre le valor de la variable.

- AREA:inoctets1#E5E365:"Tráfico de entrada del nodo AP1": pintado de los datos de la variable definida en el gráfico. Para ello, han sido definidos el modo de pintarlo, "AREA" (habrá muchos más, como LINE), la variable a dibujar, el color en el que será pintado en el gráfico, en este caso "#E5E365", y finalmente la leyenda que aparecerá en el gráfico ligada a dicha representación.

- Opciones que afectan al gráfico en general: "-h 300 -w 750", que representará el alto y ancho respectivamente, en píxeles. "–x-grid SECOND:300:SECOND:300:SECOND:30:300: %H: %M: %S" representará la forma en la que se va a pintar el eje X, de modo que también afectará a como estarán pintados los datos en el gráfico final; para ello, habrá que definir, primero, qué se pintará en los siguientes pares de valores "GTM:GST:MTM:MST:LTM:LST": el primer par "GTM:GST" definirá qué se pintará en la base de la red pintada en el gráfico (se pintará en segundos, y con una longitud de 300 segundos), el segundo "MTM:MST" el valor máximo que tomará (idem, en segundos y también 300) y finalmente "LTM:LST", que definirá cómo se van a pintar las etiquetas a lo largo del eje X del gráfico (en este caso, una vez más en segundos y de 30 en 30 segundos); por último se va a definir como se va a pintar cada una de las labels (y su posición, teniendo en cuenta lo que se ponga para representar, es decir, si se pusiese en segundos o bien horas variará el número y lugar de las etiquetas), en este caso "300: %H %M %S", con lo que se van a pintar labels para los 5 minutos, poniendo la hora, los minutos y los segundos.

5. MRTG

El Multi Router Traffic Grapher (MRTG) es una herramienta para monitorizar la carga de tráfico en enlaces de red de forma histórica. Utiliza bases de datos circulares RRD (Round Robin Database). MRTG genera páginas HTML que contienen gráficas en formato PNG, y muestran una representación de forma periódica (por defecto cada 5 min) del tráfico además de diferentes variables de equipos de red (o hosts). La página de la herramienta está en http://www.stat.ee.ethz.ch/mrtg/.

5.1. Instalación

El proceso de instalación es relativamente sencillo en máquinas Linux, pues forma parte del repositorio de la mayoría de las distribuciones. La instalación se realiza con el comando:

```
PCX# apt-get install mrtg
```

5.2. Configuración

La configuración requiere dos pasos sencillos, a saber, la configuración del paquete mrtg mediante la creación del fichero "/etc/mrtg.cfg" y la configuración del servidor web apache para la presentación de los resultados.

5.2.1. Configuración básica de MRTG

El proceso de configuración básica de MRTG consta de dos partes:

1. Creación del fichero de configuración mediante la herramienta "cfgmaker" (Ver el manual de cfgmaker para conocer todas las opciones). La forma básica de uso sería:

   ```
   PCX# cfgmaker  public@10.11.12.203  --output=/etc/mrtg.cfg
   ```

   Esto crea el fichero de configuración para la monitorización de los parámetros de las interfaces del equipo 10.11.12.203 (el Switch SW3) en la comunidad "public" SNMP. El equipo "10.11.12.203" debe tener habilitado el servicio SNMP. Para saberlo, basta con disponer del paquete "net-snmp" (se instala de forma sencilla con "apt-get install snmp") en la estación monitorizadora (estación ejecutando mrtg) y ejecutar

   ```
   PCX# snmpwalk -c  public    10.11.12.203
   ```

   presentandose, en caso de tener habilitado snmp todas las MIB y sus contenidos disponibles.

   El fichero de configuración (/etc/snmp/snmpd.conf) puede (y en muchos casos debe) ser editado para una modificación manual si se pretende gestionar un equipo basado en linux (ver capítulo 2).

   Si se quieren monitorizar varios dispositivos, será necesario ejecutar el comando "cfgmaker" por cada uno de ellos y añadir la salida al fichero "mrtg.cfg". Así, para añadir al SW3 anterior, las gráficas de los equipos F2, R1 y R2 se añadirían de la siguiente forma:

   ```
   PCX# cfgmaker  public@10.11.12.20  >> /etc/mrtg.cfg
   PCX# cfgmaker  public@10.11.12.21  >> /etc/mrtg.cfg
   PCX# cfgmaker  public@10.11.12.22  >> /etc/mrtg.cfg
   ```

2. Creación de un fichero "index.html" que presente las gráficas generadas.

   Por defecto, las páginas e imágenes generadas por mrtg se guardan (en el caso de Debian) en /var/www/mrtg. Se puede generar una página de inico con todas las interfaces monitorizadas ejecutando

   ```
   PCX# indexmaker /etc/mrtg.cfg  --output=/var/www/mrtg/index.html
   ```

3. Habilitar la ejecución periodica mediante “cron”.

   Habitualmente, tras la instalación, se crea la línea, o el fichero en el sistema ”cron”; en el caso particular de Debian 2.6.26-2-686, se crea el fichero “/etc/cron.d/mrtg” con un contenido como el que sigue,

```
*/5 *        * * *   root    if [ -x  /usr/bin/mrtg ] && [ -r  /etc/mrtg.cfg ];
then   /usr/bin/mrtg   /etc/mrtg.cfg >> /var/log/mrtg/mrtg.log 2>&1;
 fi
```

   que viene a significar la ejecución (actualización) del programa /usr/bin/mrtg cada 5 minutos. Esto hace que se disponga de una gráfica actualizada cada 5 minutos.

5.2.2. Configuración básica de Apache

Se trata exclusivamente de habilitar el directorio donde se guardan las imágenes de gráficas “mrtg” para ser visible en web.

Los ficheros de configuración de apache dependen de su versión. En el caso en que estén unificados en uno “httpd.conf”, bastaría con añadir al fichero unas líneas como

```
<Directory /var/www/mrtg>
Options ExecCGI
Options Indexes FollowSymlinks
AuthType Basic
Authname "Directorio Restringido"
AuthUserFile /etc/apache/.htpass_general
Require valid-user
AllowOverride AuthConfig
order deny,allow
allow from all
</Directory>

Alias /mrtg/ /var/www/mrtg/
```

Tras ello hay que reiniciar el servidor web apache de la forma:

```
PCX# apachectl  restart
```

o en el caso de la versión 2 de Apache

```
PCX# apache2ctl  restart
```

que añade además el sistema de seguridad de apache para acceder a páginas restringidas. Esto se consigue con el comando “htpasswd” en las versiones actuales del servidor web. Mediante estas líneas en el fichero “httpd.conf” se están sirviendo las gráficas en la url http://localhost/mrtg/ o de forma remota cambiando localhost por la dirección IP del servidor web.

Con esta configuración, se visualiza solo la actividad de las interfaces de los equipos monitorizados, pero esta utilidad permite la visualización de otras muchas variables de los sistemas, ya sea mediante protocolo de gestión SNMP u otros métodos.

5.3. Opciones avanzadas

MRTG está limitado a la presentación de dos datos por gráfica, pero estos pueden generarse de cualquier fuente. La configuración básica de MRTG presentada en la sección 5.2.1 utiliza SNMP para localizar y parsear los datos de tráfico de las interfaces de los dispositivos a monitorizar, pero estos no son los únicos datos que pueden ser presentados.

MRTG permite la representación gráfica de cualquier par de datos que se generen en dos líneas consecutivas. De esa forma, se pueden generar gráficas para cualquier parámetro, tanto del sistema local donde está instalado MRTG, como de un sistema remoto que le remita los datos.

Se presentan dos ejemplos, uno para el sistema local, y otro para un sistema remoto.

5.3.1. Sistema local

Se presentarán los datos de carga de procesador del sistema local, para lo que se creará un script llamado “carga_cpu_local.sh” que contiene un parseo del comando “uptime”. El comando “uptime” presenta en el

quinto campo la carga media del procesador en los últimos 5 minutos, por lo que simplemente será necesario parsear ese campo. El script quedaría:

```
#!/bin/bash
valor=`uptime |cut -d "," -f 5 |tr -d "." |tr -d " "`
echo "$valor"
echo "0"
```

Para decirle a MRTG de dónde puede capturar los datos a presentar, es necesario editar el fichero "/etc/mrtg.conf" y añadirle las siguientes líneas

```
Target[carga_CPU]: `/home/alumno/scripts/carga_cpu_local.sh`
Options[carga_CPU]: nopercent,gauge
MaxBytes[carga_CPU]: 100
Title[carga_CPU]: Monitorizacion de la carga de CPU del servidor
PageTop[carga_CPU]: <H1>Análisis de carga para servidor CPU</H1>
 <TABLE>
    <TR><TD>System:</TD>     <TD>servidor MRTG</TD></TR>
       <TR><TD>Maintainer:</TD> <TD>alumno</TD></TR>
          <TR><TD>Description:</TD><TD>CPU  </TD></TR>
             <TR><TD>ifName:</TD>     <TD></TD></TR>
                <TR><TD>Max carga:</TD>  <TD>100</TD></TR>
                 </TABLE>
```

Finalmente, será necesario volver a generar el fichero "index.html"

```
PCX# indexmaker /etc/mrtg.cfg  --output=/var/www/mrtg/index.html
```

5.3.2. Sistema remoto

Este ejemplo presenta el estado de memoria de un equipo remoto (por ejemplo PC24). Para ello se hace uso de la utilidad "netcat" presentada en la práctica 2.

En este caso, deberá servirse el dato por parte del sistema remoto, en un puerto libre. Esto se puede hacer directamente con el comando "netcat" o utilizando el fichero "/etc/inetd.conf". El uso de "netcat" para servir datos podría realizarse de la siguiente forma:

```
PCX# usodememoria | nc -l -p 6660
```

Si se desea disponer de un servidor de esos datos de forma contínua, habrá que realizar dos pasos:

1. En el fichero "/etc/inetd.conf" habrá que añadir el servicio de entrega de los datos del script, de la forma:

```
.............
#:OTHER: Other services
nc_monit        stream   tcp     nowait  root    /home/miguel/scripts/nc_monit
nc_monit        dgram    udp     nowait  root    /home/miguel/scripts/nc_monit
.............
```

2. Al final del fichero "/etc/services" deberán ponerse las líneas que relacionan al servicio con un puerto, de la forma:

```
.............
nc_monit    6660/tcp    # monitorizacion tipo MRTG de parametros
nc_monit    6660/udp    # monitorizacion tipo MRTG de parametros
.............
```

Finalmente, para que el equipo MRTG reciba los datos y el programa MRTG los procese para su presentación, deberá procederse como en el ejemplo anterior, es decir, creación de un script (en este caso con el programa "netcat"), inclusión de una nueva gráfica en el fichero "/etc/mrtg.conf" y creación de un nuevo index.html que incluya lo anterior. Por pasos:

Script "mem_PC24.sh" que lee los datos de memoria del equipo remoto (PC24) y los formatea para ser compatibles con MRTG

```
#!/bin/bash
destino=10.11.12.124
valor=`nc $saturno 6660 |sed -n -e '1p' | sed -e "s/ * /|/g"`
echo $valor | cut -d "|" -f 7
echo $valor | cut -d "|" -f 6
```

Inserción de una nueva gráfica en "/etc/mrtg.cfg"

```
Target[mem_PC24]: `/home/alumno/scripts/mem_PC24.sh`
Options[mem_PC24]: nopercent,gauge
MaxBytes[mem_PC24]: 100
Title[mem_PC24]: Monitorizacion de la memoria RAM de PC24
PageTop[mem_PC24]: <H1>Analisis de memoria para PC24</H1>
 <TABLE>
    <TR><TD>System:</TD>      <TD>PC24</TD></TR>
       <TR><TD>Maintainer:</TD> <TD>alumno</TD></TR>
          <TR><TD>Description:</TD><TD>Mem RAM </TD></TR>
             <TR><TD>ifName:</TD>      <TD></TD></TR>
                <TR><TD>Max mem:</TD>  <TD>100</TD></TR>
                 </TABLE>
```

Generación del fichero "index.html"

```
PCX# indexmaker /etc/mrtg.cfg  --output=/var/www/mrtg/index.html
```

Una representación de la salida en un navegador web se presenta en la figura 9

Figura 9: Presentación MRTG

6. CACTI

6.1. Introducción

Cacti es una capa de presentación y gestión de gráficas de monitorización de multitud de parámetros en todos los dispositivos susceptibles de ser monitorizados. Su principio de funcionamiento es lineal y sencillo, dividiendose en tres capas: captura de datos, almacenamiento de datos, presentación de datos. La figura 10 sacada del manual de cacti (ubicado en http://www.cacti.net/downloads/docs/html/) muestra este flujo de actividades

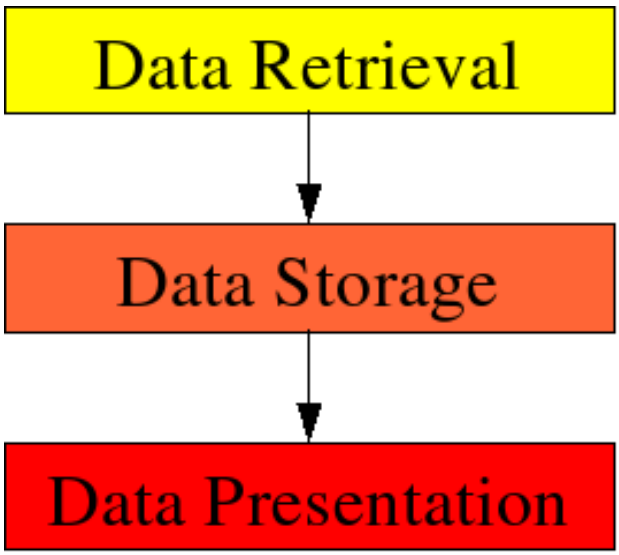

Figura 10: Principio de operación de CACTI

- La captura de datos suele hacerse mediante el protocolo SNMP hacia los diversos agentes instalados en los dispositivos a monitorizar. Existen otros métodos de captura de datos, basados en scripting o en otras herramientas.
- El almacenamiento de datos se hace en la base de datos circular RRDTOOL, que permite un ahorro importante de espacio a la hora de guardar muchos datos de muchos dispositivos a lo largo de mucho tiempo.
- La presentación de los datos se realiza mediante gráficas, con el sistema de generación de gráficas ya incluido en RRDTOOL.

Cacti permite seleccionar qué datos, de qué forma y dónde conseguirlos, y permite seleccionar la forma de presentación gráfica de los mismos. Todo ello en un entorno web.

El proceso de configuración estándar de Cacti se realiza completamene a través de la propia interfaz web del programa, aunque existen scripts que permiten procesos de automatización.

El manual de CACTI se encuentra en http://www.cacti.net/documentation.php.

6.2. Instalación

En las distribuciones Linux mas importantes CACTI forma parte del repositorio, por lo que bastará con la instalación típica. En general se recomienda la descarga y compilación de la última versión y posterior parcheado.

Para la instalación desde el repositorio en una distribución Debian o Ubunto, se hará:

```
SX# apt-get install cacti
```

Cacti necesita de otros paquetes para instalarse, que son instalados automáticamente en el proceso de instalación. No obstante, algunos paquetes son de gran importancia y pueden tener influencia sobre el sistema, como la base de datos mysql, el servidor web apache, el lenguaje php o la propia herramienta de generación de gráficas rrdtool.

Durante el proceso de instalación se preguntará por diversos parámetros de esos paquetes, como la clave de administrador de mysql.

Una vez descargado e instalado el paquete, será necesario acceder vía navegador web a la página "http://localhost/cacti" y responder a las sencillas preguntas que hace el proceso de instalación. Finalizado ese paso, se accederá a cacti con el usuario **admin** y clave **admin**, para proceder a cambiarla en la siguiente pantalla.

Para la instalación en equipos Windows será necesario seguir las instrucciones del manual de CACTI (ubicado en http://www.cacti.net/documentation.php).

6.3. Operación

Existen dos componentes diferenciados para la operación de cacti. Por un lado los parámetros que gestionan el aspecto, forma, color, leyenda y demás variables de las gráficas, y por otro lado los parámetros que gestionan los datos a presentar.

La creación de las gráficas se inicia con la identificación de los dispositivos y los parámetros que se quieren monitorizar.

Los dispositivos se crean pulsando en el menú "devices" y en la parte superior derecha en "Add". Del menú que aparece sólo es obligatorio rellenar los dos primeros campos. El tercer campo "Host template" permite autorellenar algunos parámetros por defecto, seleccionando la plantilla mas adecuada para ese dispositivo, o no rellenar nada seleccionando "none".

Una vez dados los dos o tres primeros campos, se pulsa al final del menú "Save", y el menú anterior se amplía con nuevas opciones, en particular de tipo SNMP. Este nuevo menú permite la creación de gráficas, con algunas ya creadas por defecto, en función de la plantilla de dispositivo seleccionada en el menú anterior.