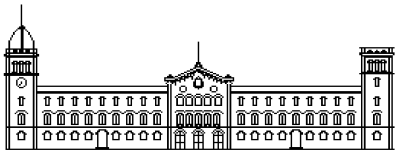


Proyecto final de carrera

INGENIERIA TÉNICA EN INFORMÁTICA DE SISTEMAS

Facultad de Matemáticas
Universidad de Barcelona

IMPLEMENTACIÓN DE UN SISTEMA DE MONITORIZACIÓN PARA EMPRESAS

Manuel Mata García

Director: Sergio Escalera
Realizado en: Departamento de Matemáticas
Aplicada y Análisis. UB
Barcelona, 14 de junio de 2012

INDICE

1. Capítulo 1: contexto y motivación

Idea inicial

El proyecto nace de la necesidad de tener una red remotamente vigilada las 24 horas del día, o un intervalo de horas determinado, con la necesidad de saber cuándo ocurre algo anómalo en ella y poder actuar cuando sea necesario.

En un primer momento se pensó en monitorizar únicamente equipos de comunicaciones Cisco para progresivamente ir introduciendo diferentes entornos y tecnologías. Finalmente, debido al crecimiento del proyecto, se han introducido los siguientes entornos:

- Windows implementados en entornos virtual mayoritariamente y físico.
- Cisco, principalmente switches.
- Solaris.
- AIX.
- Linux.

Se debe plasmar en cada entorno y en cada equipo, los aspectos más importantes de cada uno de ellos, asegurándonos de que tenemos un control total sobre cualquier aspecto que pueda influir en el mal funcionamiento de los equipos.

Para ello se plantean las siguientes cuestiones:

- ¿Qué tipo de equipo se requiere para desarrollar el proyecto?
- ¿Qué sistema operativo sería más útil?
- ¿Qué aplicaciones deberemos tener instaladas?
- ¿Qué aplicación gestionará la monitorización?
- ¿Cómo podemos conectar el servidor remoto con una red local?
- ¿Cómo se podrían enviar notificaciones vía SMS o e-mail, como mínimo?
- ¿Cómo sacaremos la información necesaria de los equipos monitorizados?
- ¿En qué lenguaje podríamos realizar los scripts de consulta?
- ¿Qué deberíamos de hacer para que fuese una interface “amigable” para un usuario que no conozca el entorno?

Se analizarán posibles opciones que se tiene en cada punto, detallando el motivo por el cual se antepone una a otra, con una opinión, que aunque no sea la única, recoja todas las necesidades iniciales del proyecto.

Una de las principales características que tendrá el proyecto, cuando esté finalizado, es la pro-actividad, es decir, con un sistema bien implementado, nos podremos adelantar a posibles incidencias que se vayan a dar en la red, sistemas que vayan a dejar de funcionar, equipos mal dimensionados a nivel de memoria, lentitud en la red, etc.

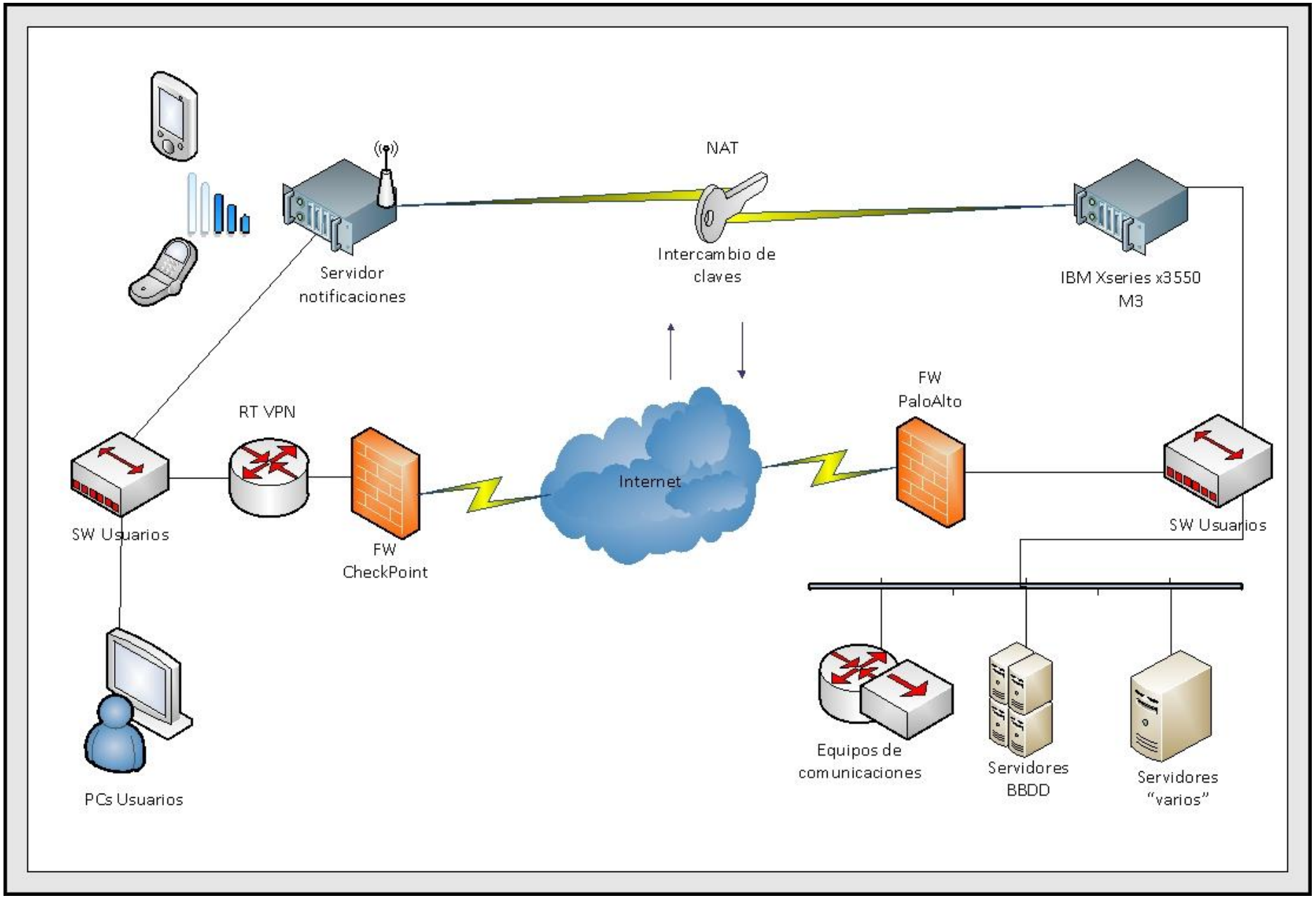
El motivo principal por el cual surge la idea es la necesidad, como ya he indicado anteriormente, de una vigilancia personalizada en un margen de horas determinado. No se busca saber cuándo ha ocurrido algo, sino cuándo va a ocurrir; no se quiere solucionar un fallo de un sistema cuando se tenga que usar, sino tener la posibilidad de saber cuándo ha fallado para arreglarlo antes de que se convierta en un problema.

La idea final es tener una imagen autoinstalable, que ya incluya todas estas aplicaciones y scripts, para que se pueda instalar en unos pocos minutos en un equipo sin necesidad de realizar todos los pasos previos de configuración. Una vez hecho esto, simplemente configurando una IP localmente en el equipo y una vía de salida para el servidor que envía las notificaciones, el sistema quedará solo a la espera de introducir los servicios y servidores para empezar a ser funcional, acción que podrá ser realizada por un administrador o por un usuario de la empresa.

Destacar que el proyecto debe de tener la capacidad de mantener de manera estable las comprobaciones de una red de más de 500 equipos, con lo que puede suponer más de 5000 servicios monitorizados y controlándose simultáneamente.

A fin de dar respuesta a las preguntas detalladas anteriormente, y a otras que surgirán a medida que avance el proyecto, a continuación se discuten cada una de ellas.

Antes de ver las cuestiones previas, se mostrará un diagrama lógico de red para ver la estructura en la cual se ha implantado el proyecto a nivel funcional:


NAT
Servidor
notificaciones
Intercambio de
claves
IBM Xseries x3550
M3
RT VPN
FW
PaloAlto
Internet
SW Usuarios
FW
CheckPoint
SW Usuarios
PCs Usuarios
Equipos de
comunicaciones
Servidores
BBDD
Servidores
"varios"

1.1 ¿Qué tipo de equipo se requiere para desarrollar el proyecto?

El sistemas elegido para instalar todas las aplicaciones ha sido un IBM X Series 3550 M3.

Como posibilidades se tenían o un servidor físico o implementarlo en un servidor virtual. A nivel de aplicación, en los dos casos, se debería obtener las mismas funcionalidades.

A priori, tenemos que:

- La virtual tiene ventajas a la hora de poder redimensionar la memoria asignada sobre la física, con unos simples clicks podemos asignar/quitar recursos, dependiendo de lo que necesite a medida que el nivel de elementos de red crezca.
- La virtual puede resultar mucho más barata que la física.
- Los backups pueden resultar más fáciles y cómodos de gestionar en un equipo virtual.
- En rendimiento quizás gane la física sobre la virtual, además siempre es mejor tener una máquina dedicada con sus propios recursos.

Con estos detalles, entre otros, se ha decido realizar el proyecto sobre un equipo físico. Los motivos por los cuales se ha elegido un servidor físico dedicado y no uno virtual han sido:

- Un servidor virtual hubiese aportado mayor ahorro económico y mayor rapidez a la hora de configurar y manejar, pero el rendimiento hubiese sido un poco inferior, por lo tanto, y debido a la idea final del proyecto se tiende a algo más potente por encima de todo lo demás.

- La idea inicial es que la imagen no se pueda instalar en cualquier tipo de equipos, únicamente en los que cumplan una serie de requisitos. Sacando la imagen tal y como se ha obtenido solo se puede instalar en este equipo IBM.

El hardware consta de:

Imagen de un servidor IBM X3550 M3.

- Procesador quad core Intel Xenon 5500 series.
- 8 GB de memoria RAM.
- 2 fuentes de alimentación: redundancia eléctrica para prevenir posibles caídas de la red.
- 4 ventiladores.
- 2 discos de 500gb configurados en raid mirroring (RAID 1) ampliables hasta seis: nos permitirá tener una réplica exacta de un disco en otro por si se corrompieran los datos de uno de ellos.

1.2 ¿Qué sistema operativo sería más útil?

Se han barajado dos opciones de SO, como son Linux y Windows, en ambos casos todo el software utilizado estaba disponible. Tenemos que:

- Las herramientas principales e imprescindibles funcionan en las dos plataformas.
- En Linux no necesitamos licencia dependiendo de la distribución.
- Linux se puede configurar para que sólo invierta recursos en lo que realmente necesitamos, quitando todos los componentes en la instalación que no creamos oportuno tener.
- No es imprescindible una interface gráfica para realizar ninguna de las acciones que se requieren.

La implementación se ha realizado sobre Linux. La opción que prefería era CentOS, debido a que ya había trabajado en esta distribución y que revisando, todos los paquetes necesarios estaban disponibles. Al final se ha instalado Red Hat EL6, debido a que se disponía de licencias y que es idéntico en funcionamiento a CentOS.

Linux, a priori, gestionará mejor los procesos y no invertirá recursos en la interface gráfica, la cual no necesitamos para nada. Además nos interesa tener un SO multiusuario que nos gestione múltiples sesiones de terminal (Windows) o ssh (Linux), esto lo conseguimos con este último; en Windows necesitaríamos una versión server.

Además, con la idea final que es tener una imagen que se pueda instalar en diferentes equipos nos será mucho más útil Linux por el tema de las licencias.

1.3 ¿Qué aplicaciones deberemos tener instaladas?

Para realizar las funciones básicas necesitamos:

1- Un intérprete de perl: La mayoría de scripts realizados y modificados se ejecutan en lenguaje perl. Aquí nos encontramos otra ventaja de Linux, que en sus versiones más usadas ya tiene instalado este intérprete.

```
[root@app-miquell ~]# ls -la /usr/bin/perl
-rwxr-xr-x 2 root root 13512 Oct  4  2011 /usr/bin/perl
[root@app-miquell ~]#
```

Ruta de instalación del binario de perl

Y si miramos los procesos activos de perl, podemos ver cómo se están ejecutando los scripts en background. Cada vez que se consulta vemos peticiones nuevas, esto indica que funciona correctamente ya que se está gestionando la cola de procesos.

```
[root@app-miquell ~]# ps -ef | grep perl
nagios    5792  5777  0 17:38 ?        00:00:00 /usr/bin/perl -w /usr/lib64/nagios/plugins/check_oracle_health --mode=tablespace-usage --connect=dbsnmp/dbsnmp@FRP.WORLD --tablesp
ace=SYSAUX --warning=90 --critical=95
nagios    5814  5794  0 17:38 ?        00:00:00 /usr/bin/perl -w /usr/lib64/nagios/plugins/check_oracle_health --mode=tablespace-usage --connect=dbsnmp/dbsnmp@FRP.WORLD --tablesp
ace=PSAPSR3 --warning=92 --critical=97
nagios    5848  5829  0 17:38 ?        00:00:00 /usr/bin/perl -w /usr/lib64/nagios/plugins/check_oracle_health --mode=tablespace-usage --connect=dbsnmp/dbsnmp@FRP.WORLD --tablesp
ace=PSAPSR3701 --warning=90 --critical=95
nagios    5874  5853  0 17:38 ?        00:00:00 /usr/bin/perl -w /usr/lib64/nagios/plugins/check_oracle_health --mode=tablespace-usage --connect=dbsnmp/dbsnmp@FRP.WORLD --tablesp
ace=SYSTEM --warning=90 --critical=95
nagios    5886  5875  0 17:38 ?        00:00:00 /usr/bin/perl -w /usr/lib64/nagios/plugins/check_oracle_health --mode=tablespace-usage --connect=dbsnmp/dbsnmp@FRP.WORLD --tablesp
ace=PSAPUNDO --warning=90 --critical=95
nagios    5922  5914  0 17:38 ?        00:00:00 /usr/bin/perl -w /usr/lib64/nagios/plugins/check_oracle_health --mode=tablespace-usage --connect=dbsnmp/dbsnmp@FRP.WORLD --tablesp
ace=PSAPSR3USR --warning=90 --critical=95
nagios    5958  5950  0 17:38 ?        00:00:00 /usr/bin/perl -w /usr/lib64/nagios/plugins/check_oracle_health --mode=tablespace-usage --connect=dbsnmp/dbsnmp@FRP.WORLD --tablesp
ace=PSAPTEMP --warning=90 --critical=95
root      6558  5990  0 17:38 pts/0    00:00:00 grep perl
[root@app-miquell ~]#
```

Scripts perl en ejecución

2- Intérprete de Shell script: Algún script está escrito en Shell script, para ejecutarlo en Linux nos basta con el bash.

```
[root@app-miquell ~]# ls -la /bin/bash
-rwxr-xr-x 1 root root 939824 Jan 27  2011 /bin/bash
[root@app-miquell ~]#
```

Ruta instalación binario de bash

3- Software que gestione las colas de peticiones a los equipos remotos: En este apartado se ha decidido instalar Nagios.
Simplemente necesitamos que genere una cola de peticiones cíclica con los scripts que preguntan a los host/servicios en remoto. El funcionamiento interno se analizará en apartados posteriores. Ya de paso, Nagios nos servirá como una interface para que se controle visualmente el estado de toda la red implantada, cosa que a nivel de usuario es muy útil y le da un valor añadido al software.

Aquí se puede observar la ruta de instalación del software con todos los ficheros de configuración, que también se verán en apartados posteriores.

```
[root@app-miquell nagios]# pwd
/etc/nagios
[root@app-miquell nagios]# ls -la
total 144
drwxr-xr-x    4 root   root    4096 May 14 13:31 .
drwxr-xr-x. 120 root   root   12288 May 17 15:50 ..
-rw-rw-r--    1 nagios apache 12223 May 14 13:31 cgi.cfg
-rw-r--r--    1 root   root   15860 Nov 26  2010 command-plugins.cfg
-rw-r--r--    1 root   root     237 May 14 13:30 htpasswd.users
drwxrwxr-x    2 nagios apache  4096 Dec  6 19:03 import
-rw-r--r--    1 apache nagios 45504 Dec  3 20:20 nagios.cfg
-rw-r--r--    1 apache apache  4625 Dec  3 20:47 ndo2db.cfg
-rw-r--r--    1 apache apache  5138 Dec  3 20:16 ndomod.cfg
-rw-r--r--    1 root   root    7207 Nov 11  2010 nrpe.cfg
-rw-r--r--    1 root   root    5326 Nov 12  2010 nsca.cfg
drwxrwxr-x    5 nagios apache  4096 May 17 14:37 objects
-rw-rw----    1 nagios nagios  1340 Nov 26  2010 resource.cfg
-rw-r--r--    1 root   root    1628 Nov 12  2010 send_nsca.cfg
[root@app-miquell nagios]#
```

Ruta instalación Nagios

4- Cliente de ssh: Necesitamos hacer llamadas al sistema remoto que gestionará el envío de notificaciones (sms y e-mail), éste recibirá peticiones a través de ssh para ejecutar los comandos.

```
/usr/bin/ssh 10.0.7.195 '/usr/bin/mailx -r "Alert SCC <alert2@mss.scc.com>" -s "MAG: $HOSTNAME$
```

Ejemplo de notificación utilizando conexión ssh.

Con estos cuatro puntos, a nivel de aplicaciones, se cubrirían las necesidades primarias para que se pudiese gestionar una red.

Luego hay que sumar las aplicaciones que se han introducido secundarias, para que resulte más fácil su manejo, pero que en ningún caso son imprescindibles, puesto que sólo con estas cuatro citadas el proyecto sería 100% funcional.

1.4 ¿Qué aplicación gestionará la monitorización?

Esta cuestión se ha repasado en el apartado anterior, la solución ha sido nagios.

¿Por qué motivo?

1. A nivel de aplicación: Necesitamos que se gestionen las colas, es decir, se tendrá un conjunto de scripts que se deberán ir lanzando constantemente contra equipos remotos. Esto es lo que nos proporciona esta aplicación, le podemos dar un tiempo de ejecución cada *X* minutos, que hará que se vayan repitiendo sin necesidad de gestionar manualmente las peticiones y que dependerá, en cada caso, de la criticidad de lo que se quiera comprobar. Por ejemplo:

    - Comprobar si un equipo está funcionando se hará cada 5 minutos (crítico).
    - Comprobar si una '/' se llena se hará cada 15 minutos (criticidad media).
    - Comprobar si un servicio de antivirus está activo se hará cada 1 h (no crítico).

2. A nivel de usuario: Tiene una interface de usuario muy útil y visual, con la que a partir de colores, podemos ver el estado de la red y ver los cambios de estado que se producen en ésta cada vez que un scripts de los comentados anteriormente se ejecuta.
   Además incorpora múltiples consultas:

    - Tiempos en correcto funcionamiento por equipo/servicio.
    - Estadísticas de rendimiento.
    - Visión global de equipos/servicios con problemas y correctos on time.
    - Resúmenes de alertas más comunes en periodos de tiempo especificados.

1.5 ¿Cómo podemos conectar el servidor remoto con una red local?

El servidor de envío de notificaciones mediante SMS, e-mails y llamadas, siempre va a estar físicamente en un lugar del mundo fijo, por lo que necesitaremos una comunicación contra él.

La comunicación entre redes que no son de la misma propiedad, es decir, no tienen el mismo direccionamiento debido a que pertenecen a organizaciones diferentes, se puede realizar mediante un NAT.

¿Qué es una NAT?

Un NAT (*Network Adress Translation*) es un mecanismo de enrutamiento de IPs entre redes con direccionamientos incompatibles. Tenemos la red cliente (donde está el equipo IBM) y la red servidor (donde está el equipo que envía las notificaciones) y debemos transformar/traducir una IP de la red cliente en la de la red servidor o viceversa.

El funcionamiento es muy sencillo:

1. La IP interna que tiene el equipo IBM, en este caso, es la 10.100.33.110.
2. La IP interna del equipo que envía las notificaciones, en este caso, es la 10.3.0.40.
3. Son redes incompatibles y situadas en LANs diferentes.
4. Para ellos realizaremos el NAT, en este caso, para que la IP 10.3.0.40 sea visible desde la red en la que se encuentra el 10.100.33.110 utilizando la IP 10.0.7.195.
5. Cuando desde la 10.100.33.110 ataquemos a la “nateada” 10.0.7.195, ésta la transformará en mi red en la 10.3.0.40.

1.6 ¿Cómo se podrían enviar notificaciones vía SMS o e-mail, como mínimo?

Una vez se dispone del NAT operativo, y teniendo en cuenta que todo equipo en un red debe tener usuario y contraseña, nos queda que se "entiendan" entre ellos.

Para ello necesitan tener una relación de confianza entre los equipos, en este caso entre el servidor IBM y el generador de notificaciones. Con esta relación conseguiremos que cuando uno haga una petición al otro, el equipo destino tenga una clave única del equipo origen, el cual al tenerla en su configuración hace que no necesiten usuario ni password para conectarse entre ellos. Los detalles se indicarán en el siguiente capítulo.

Aplicado al caso anterior tenemos que, en orden:

1. El servidor 10.100.33.110 quiere enviar una notificación al 10.3.0.40 (recordamos que son redes diferentes).
2. El servidor 10.100.33.110 debe atacar al NAT, así que hará una petición ssh al NAT 10.0.7.195.
3. A través de esta última petición nos encontramos conectados al 10.3.0.40, saltándonos la seguridad del usuario y password de este último, todo es debido a la relación de confianza de sus claves.

Una vez tenemos la conexión entre ambos extremos, tenemos que:

El servidor que envía las notificaciones es un Linux con un sistema Red Hat, al igual que el equipo IBM, se ha optado por conectar un hardware por el puerto serie que procese todas las peticiones de notificación de eventos.

Dentro de las notificaciones tenemos de dos tipos:

1. SMS y llamadas: Para ello se ha utilizado un Centurion. Este equipo consta de un slot para colocar una tarjeta SIM de cualquier compañía de móvil, y el cual se puede conectar mediante un cable serie a un servidor para interactuar con él. De esta manera se le envía una orden a través del sistema operativo, que enlazará con el puerto serie y enviará la orden al Centurion para enviar SMS o realizar llamadas dependiendo de lo que nos interese en cada caso.

2. E-mail: Para enviar e-mails hacia fuera utilizamos MailX, un software ya instalado en el sistema operativo que no requiere de ningún hardware independiente. Este programa se ejecuta desde la línea de comandos y únicamente requiere de un servidor SMTP configurado.

Hardware que compone el Centurion.

1.7 ¿Cómo sacaremos la información necesaria de los equipos monitorizados?

Se han estudiado los siguientes protocolos antes de implementar los que mejor convenían para cada caso.

- ***NRPE (Nagios Remote Plugin Executor)***

Es un daemon que ejecuta peticiones a equipos remotos. El paquete consta de dos archivos:

NRPE: Es el proceso que se ejecuta en el equipo remoto. Consta de un fichero de configuración en el que se indica el equipo/os cliente que están permitidos a realizar cualquier petición de estado y de un conjunto de scripts que se ejecutarán en la máquina localmente en caso de que la petición sea válida, una vez hecho esto devuelve el resultado a check_nrpe que está en la máquina que ha realizado la petición.

check_nrpe: Es el plugin que se ejecuta en el equipo que realiza constantemente las peticiones remotas y es el encargado de contactar con el proceso NRPE remoto. Este plugin se ejecuta localmente, llama al remoto y espera a que este le devuelva la información solicitada.

- ***SNMP (Simple Network Management Protocol)***

Es un protocolo que facilita el intercambio de información entre elementos de red. Existen 3 versiones (V1, V2 y V3). Su funcionamiento se basa únicamente en la configuración en el equipo remoto al cual se le quiere enviar la petición, consiste en un fichero en el cual se configuran las IPs de equipos, los cuales están permitidos a hacer la petición. Además de las IPs de los equipos permitidos, se debe configurar un string por el cual también se realizarán las peticiones llamado “community”.

- ***NsClient***

Es un servicio que únicamente puede funcionar en Windows. Su funcionamiento es prácticamente igual que el anterior, con un listado de equipos permitidos para realizar las peticiones. El hecho de que solo se pueda implementar en una plataforma, limita mucho su uso.

- ***WMI (Windows Management Instrumentation)***

WMI es un producto de Microsoft y solo se puede implementar en sistemas operativos Windows. Desde Windows 2000 está pre instalado en el equipo y en las versiones anteriores hay que descargarlo. Se basa en la implementación de un agente interno que es llamado proveedor, el cual hace las consultas al hardware y al software sobre su estado. Estos datos son almacenados en un repositorio llamado CIM, luego el agente gestor de la red recolecta estos datos del repositorio CIM.

Una vez estudiado los diferentes casos se han implementado los siguientes para el proyecto:

- ***SNMP:*** Se ha optado por este método debido a que su implementación se puede realizar tanto en Windows como en Linux, cosa que es imposible con NsClient y WMI, que a priori son los dos que nos pueden proporcionar lo mismo. Además las peticiones que se lanzan contra el equipo no genera un tráfico en él que cause problemas a nivel de carga. Una vez configurados en unos pocos, se ve que si un equipo está muy cargado, lo primero que empieza a descartar es el tráfico SNMP, haciendo así que priorice los procesos importantes de sistema por encima de las peticiones de este protocolo. El protocolo NsClient se ha descartado debido a que es menos seguro ya que no incluye el string con el cual se le debe preguntar a la red, no incluye mejoras con respecto a éste y a nivel de configuración para ver el contenido de la máquina remota es mucho más sencillo para el usuario. El protocolo WMI se ha descartado principalmente por no tener soporte en Unix, como ya se ha comentado anteriormente.

- ***NRPE:*** Una vez realizadas pruebas se ha optado por este protocolo debido a que es muy útil, sobretodo, para equipos Unix. Simplemente se debe editar un fichero de configuración (nrpe.cfg) en el cual se añaden los equipos permitidos para realizar las peticiones por seguridad y se configura la ruta donde están los checks que se ejecutan localmente y envían la respuesta al servidor centralizado que ha ejecutado la petición. Los detalles de su funcionamiento se verán en apartados posteriores.

1.8 ¿En qué lenguaje podríamos realizar los scripts de consulta?

En el siguiente apartado se expondrán los lenguajes que se han planteado para realizar los scripts que ejecutará la aplicación continuamente:

1. Perl: Se trata de un lenguaje de programación sencillo con características de C y shell. Gestiona automáticamente la memoria y conoce el tipo y requerimientos de almacenamiento de cada objeto en el programa; reserva y libera espacio para ellos según sea necesario. Se ha considerado como una de las opciones debido a su sencillez ante la necesidad de tener que realizar scripts pequeños con pocas rutinas.

2. Shell script: Es otro sencillo lenguaje de programación muy eficiente a la hora de realizar pequeños programas como los que se necesitan para este proyecto. Viene de "shell" que es el intérprete de comandos del sistema. Estos programas no requieren ser compilados ya que la Shell los interpreta línea a línea.

Una vez expuestos los dos, para el proyecto se ha usado mayoritariamente perl y en algunos casos Shell script.

La mayoría de scripts ya existentes y que se han aprovechado para dar funcionalidad a la aplicación están realizados en perl y también, los que se han requerido (scripts para sistemas Cisco que se detallarán en apartados posteriores) se han realizado en este lenguaje.

1.9 ¿Qué deberíamos de hacer para que fuese una interface "amigable" para un usuario que no conozca el entorno?

Nagios es un software que en su versión inicial viene para ser configurado sobre ficheros de texto plano. Es un método que no es muy manejable para que un usuario configure equipos para insertar en la monitorización. Para solucionar este problema se ha introducido una utilidad llamada NagiosQL.

NagiosQL

¿Qué ventajas aporta NagiosQL?

A fin de hacer más fácil la inserción de nuevos equipos y servicios, esta aplicación añade una interface web para facilitar todo el proceso de alta de equipos.

Únicamente con nagios tenemos la configuración en los siguientes ficheros:

```
[root@app-miquell /]# cd /etc/nagios/objects/
[root@app-miquell objects]# ls -la
total 660
drwxrwxr-x 5 nagios apache   4096 May 18 14:53 .
drwxr-xr-x 4 root   root     4096 May 17 18:19 ..
drwxrwxr-x 4 nagios apache 544768 May 18 14:53 backup
-rw-r--r-- 1 apache apache  18550 May 18 14:53 commands.cfg
-rw-r--r-- 1 apache apache   1444 May 18 14:53 contactgroups.cfg
-rw-r--r-- 1 apache apache   2425 May 18 14:53 contacts.cfg
-rw-r--r-- 1 apache apache    917 May 18 14:53 contacttemplates.cfg
-rw-r--r-- 1 apache apache    662 May 18 14:53 hostdependencies.cfg
-rw-r--r-- 1 apache apache    664 May 18 14:53 hostescalations.cfg
-rw-r--r-- 1 apache apache    682 May 18 14:53 hostextinfo.cfg
-rw-r--r-- 1 apache apache   9630 May 18 14:53 hostgroups.cfg
drwxrwxr-x 2 apache apache  20480 May 18 14:53 hosts
-rw-r--r-- 1 apache apache   5866 May 18 14:53 hosttemplates.cfg
-rw-r--r-- 1 apache apache    668 May 18 14:53 servicedependencies.cfg
-rw-r--r-- 1 apache apache    670 May 18 14:53 serviceescalations.cfg
-rw-r--r-- 1 apache apache    688 May 18 14:53 serviceextinfo.cfg
-rw-r--r-- 1 apache apache   1961 May 18 14:53 servicegroups.cfg
drwxrwxr-x 2 apache apache   4096 May 18 14:53 services
-rw-r--r-- 1 apache apache   3124 May 18 14:53 servicetemplates.cfg
-rw-r--r-- 1 apache apache    258 Nov 16  2011 templates.cfg
-rw-r--r-- 1 apache apache   2007 May 18 14:53 timeperiods.cfg
[root@app-miquell objects]#
```

Despliegue contenido ruta de nagios

Una estructura que no es nada cómoda para un usuario, dentro de cada fichero tenemos que ir configurando IPs, nombres de host, servicios e ir uniendo los unos con los otros.

Instalando NagiosQL pasamos de tener únicamente esto a tener una interface gráfica muy sencilla que permite a nivel visual introducir datos y que estos se añadan a los ficheros de la imagen anterior haciendo las relaciones internas automáticamente.

Con esto pasamos de un fichero configurable manualmente:

```
define host {
        host_name                       tin01
        alias                           tin01
        address                         10.255.255.50
        hostgroups                      Domain Controler y DNS
        use                             NotificacionVariable
        register                        1
}
```

Fichero host sin NagiosQL instalado

A esto otro, idéntico, pero que se ha configurado a través de la interface gráfica:

```
[root@app-miquell hosts]# more tin01.cfg
###############################################################################
#
# Host configuration file
#
# Created by: Nagios QL Version 3.1.1
# Date:       2012-05-18 14:53:05
# Version:    Nagios 3.x config file
#
# --- DO NOT EDIT THIS FILE BY HAND ---
# Nagios QL will overwite all manual settings during the next update
#
###############################################################################

define host {
        host_name                       tin01
        alias                           tin01
        address                         10.255.255.50
        hostgroups                      Domain Controler y DNS
        use                             NotificacionVariable
        register                        1
}
```

Fichero host con NagiosQL instalado

Como se puede ver el resultado es idéntico, sólo cambia la cabecera que indica con el software que se ha generado. Realizar el alta de equipos y servicios a través de la interface web impedirá que se nos olviden datos importantes a rellenar, dupliquemos sistemas, etc.

NagiosQL no impide que se pueda configurar a través de los ficheros de texto plano, ya que se podrá hacer de los dos modos.

La interface y su configuración se verán en capítulos posteriores.

Otra de las mejoras gráficas a nivel de usuario ha sido la introducción de PNP4Nagios y DocuWiki:

PNP4Nagios

Es una sencilla aplicación que recolecta datos y los transforma en gráficos. Esta aplicación soluciona la necesidad de disponer de un histórico de cada servicio monitorizado, con ella se podrá ver la evolución, estudiar dimensionados de discos, consumos de memoria etc., para poder tener un control sobre el rendimiento y dimensionado de los equipos, entre otros.

Su funcionamiento es muy sencillo, simplemente se encarga de pintar en un gráfico un dato que sacamos a través de los scripts realizados continuamente. Si cada *X* minutos se ejecuta el script, cada *X* minutos se actualiza el dato del gráfico.

DocuWiki

Software que funciona como una mini base de datos, en ficheros de texto plano, para anotar indicaciones, procedimientos o cualquier tipo de nota adicional de ayuda en cada equipo o servicio que se esté monitorizando.

Internamente crea un fichero con el nombre del sistema al que pertenece y posteriormente, desde Nagios apuntaremos a la ruta específica para que con un simple “click” en el servicio o servidor deseado, podamos ver el contenido de este fichero de texto.

2. DISEÑO E IMPLEMENTACION

Se analizarán todos los puntos indicados en el capítulo anterior, detallando la instalación y estructura de lo que se considere más relevante para entender cómo funciona internamente la aplicación.

Al igual que en el apartado anterior se partirá de las mimas dudas una vez ya planteada su solución, con lo que nos queda:

2.1 Estudio del equipo elegido

2.2 Instalación del SO y estructura interna

2.3 Análisis de las aplicaciones necesarias

2.4 Estructura interna del software de monitorización

2.5 Estudio del NAT

2.6 Configuración del intercambio de claves y comando que se intercambian

2.7 Configuración de los protocolos de monitorización en cada entorno utilizado

2.8 Ejemplos de scripts realizados y cómo se ejecutan internamente

2.9 Análisis de las aplicaciones secundarias instaladas para interface de usuario

2.10 Mejoras introducidas

2.1 Estudio del equipo elegido

El equipo utilizado ha sido un IBM X Series 3550 M3, el cual a la hora de la implementación nos ha aportado lo siguiente:

2.1.1 HDD

En el desarrollo se han utilizado dos discos de 500 GB Sata para poder utilizar RAID 1 (mirror), esto permitirá tener siempre una duplicidad de datos, para que en caso de que un disco se estropee poder tener redundancia de datos. En caso de que se estropeara o se extrajera uno de los discos, se encendería un led del equipo de un módulo/pantalla extraíble que nos indicaría el problema. Para solventar el problema bastaría con introducir de nuevo un disco con las mismas características que el extraído/dañado para que la controladora de Raid lo sincronizara con el que estuviese intacto.

Posibilidad de ampliar hasta 6 discos por posibles cambios que se quieran realizar del tipo de Raid, duplicidades etc.

2.1.2 Unidades de lectura

Dispone de lector de CD/DVD necesarios, ya que en este caso se ha utilizado una versión de clonezilla, software con el cual se realiza la instalación de la imagen.

Dispone de varias unidades de USB, en este caso la imagen se instala mediante un pen drive que contiene los datos que previamente se han extraído con el "clonezilla" hacia una unidad externa USB.

2.1.3 Tarjetas de red

Dispone de dos tarjetas de red para poder configurar cada una con un direccionamiento IP distinto, preferiblemente, y así disponer de dos posibles entradas a la máquina en caso de que una se estropeara o de que una de las redes se cayera.

2.1.4 Conexiones VGA

Dispone de conexión a pantalla para poder configurar localmente la máquina en caso de no querer realizar por red.

2.1.5 Características de procesador y CPU

El equipo tiene un Quad Core Intel Xenon 5500 series, ideal para poder procesar múltiples peticiones en paralelo. Hay que tener en cuenta que se estarán ejecutando, en este proyecto, entorno a los 5000 procesos cíclicamente y necesitamos que la cola de procesos funcione adecuadamente y no los encole.

2.1.6 Memoria RAM

Dispone de 16 slots para posibles ampliaciones de memoria RAM, en este caso se han instalado 8 GB de los cuales están consumidos continuamente una media de 3 GB a 3,5 GB.

2.1.7 Fuentes de alimentación

Dispone de redundancia en la alimentación, es decir, tiene dos fuentes por si cayera una red eléctrica o se estropeara un módulo poder seguir funcionando correctamente.

2.2 Instalación del SO y estructura interna

A nivel de SO se ha instalado un Red Hat de 64 bits:

```
[root@app-miquell ~]# uname -a
Linux app-miquell 2.6.32-220.el6.x86_64 #1 SMP Wed Nov 9 08:03:13 EST 2011 x86_64 x86_64 x86_64 GNU/Linux
[root@app-miquell ~]#
```

Versión Red Hat

La partición del sistema consta de 50gb (partición /), ésta no crece a no ser que manualmente se almacene algo.

```
[root@app-miquell ~]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/vg_appxxx-lv_root
                       50G  7.6G   40G  17% /
tmpfs                 3.9G  116K  3.9G   1% /dev/shm
/dev/sda2             485M   77M  383M  17% /boot
/dev/sda1             200M  256K  200M   1% /boot/efi
/dev/mapper/vg_appxxx-LogVol02
                      244G   16G  216G   7% /var
/dev/mapper/vg_appxxx-LogVol03
                       49G  7.4G   39G  17% /var/lib
/dev/mapper/vg_appxxx-LogVol04
                      107G  5.1G   97G   5% /var/log
[root@app-miquell ~]#
```

Tabla file systems

No se ha usado la configuración/instalación por defecto en ninguna de las aplicaciones, todo log de aplicación se ha redirigido a /var/log, esto se ha hecho así debido a que es un equipo que está constantemente registrando logs y no se puede permitir que se llene la '/' y deje de funcionar. Con esto se consigue que el equipo funcione sin un mantenimiento constante, todo evento se registrará en

el directorio correspondiente dentro de /var/log/ permitiendo que pueda crecer sin control, en caso de que se llegase a llenar no afectaría al funcionamiento de la aplicación/es.

Directorio del /var/log en el que se almacenan los eventos de las aplicaciones:

```
[root@app-miquel1 ~]# cd /var/log/
[root@app-miquel1 log]# ls -la
total 826668
drwxr-xr-x. 18 root   root        4096 May 20 03:13 .
drwxr-xr-x. 26 root   root        4096 Mar 27 16:04 ..
-rw-------.  1 root   root        9576 Dec  1 12:50 anaconda.ifcfg.log
-rw-------.  1 root   root       29012 Dec  1 12:50 anaconda.log
-rw-------.  1 root   root       96350 Dec  1 12:50 anaconda.program.log
-rw-------.  1 root   root      269408 Dec  1 12:50 anaconda.storage.log
-rw-------.  1 root   root      100095 Dec  1 12:50 anaconda.syslog
-rw-------.  1 root   root       32376 Dec  1 12:50 anaconda.xlog
-rw-------.  1 root   root       97837 Dec  1 12:50 anaconda.yum.log
drwxr-x---.  2 root   root        4096 Dec  9 23:53 audit
-rw-r--r--   1 root   root        3312 Dec 17 18:11 boot.log
-rw-r--r--.  1 root   root        2965 Dec  1 13:15 boot.log-20111204
-rw-------   1 root   utmp        9600 May 21 21:47 btmp
-rw-------   1 root   utmp        3456 Apr 20 14:28 btmp-20120501
drwxr-xr-x.  2 root   root        4096 Dec  1 13:15 ConsoleKit
-rw-------   1 root   root      511297 May 25 10:35 cron
-rw-------   1 root   root      614718 Apr 29 03:42 cron-20120429
-rw-------   1 root   root      611355 May  6 03:17 cron-20120506
-rw-------   1 root   root      639920 May 13 03:37 cron-20120513
-rw-------   1 root   root      673131 May 20 03:11 cron-20120520
drwxr-xr-x.  2 lp     sys         4096 Oct 18  2011 cups
-rw-r--r--   1 root   root       59102 Dec 17 18:10 dmesg
-rw-r--r--   1 root   root       62735 Dec 16 12:13 dmesg.old
-rw-------   1 root   root           0 Jan  1 03:07 dracut.log
-rw-r--r--.  1 root   root      334072 Dec  6 17:18 dracut.log-20120101
drwxr-xr-x.  2 root   root        4096 Aug 26  2011 gdm
drwx------.  2 root   root        4096 May 20 03:13 httpd
-rw-r--r--.  1 root   root      146584 May 25 10:35 lastlog
drwx------.  2 root   root       16384 Dec  1 12:23 lost+found
-rw-------   1 root   root           0 May 20 03:13 maillog
-rw-------   1 root   root           0 Apr 22 03:08 maillog-20120429
-rw-------   1 root   root           0 Apr 29 03:42 maillog-20120506
-rw-------   1 root   root           0 May  6 03:17 maillog-20120513
-rw-------   1 root   root           0 May 13 03:37 maillog-20120520
-rw-r--r--.  1 root   root           0 Dec  1 13:01 mcelog
-rw-------   1 root   root    66467780 May 25 10:35 messages
-rw-------   1 root   root    88226050 Apr 29 03:42 messages-20120429
-rw-------   1 root   root    85198652 May  6 03:17 messages-20120506
-rw-------   1 root   root    73770278 May 13 03:37 messages-20120513
-rw-------   1 root   root    80102301 May 20 03:13 messages-20120520
-rw-rw----.  1 mysql  root        1569 Apr 24 12:12 mysqld.log
drwxr-xr-x.  7 nagios nagios      4096 May 25 10:35 nagios
drwxr-xr-x.  2 ntp    ntp         4096 May 13  2010 ntpstats
drwx------   2 nx     root        4096 May 24 03:24 nx
-rw-r--r--.  1 root   root          89 Mar 27 10:31 pm-powersave.log
drwx------.  2 root   root        4096 Mar  5  2010 ppp
drwxr-xr-x.  2 root   root        4096 Oct 20  2011 prelink
-rw-r--r--   1 root   root         502 May  9 14:04 prueba.tmp
drwxr-xr-x  37 root   root        4096 May 17 13:52 remotos
drwxr-xr-x.  2 root   root        4096 May 24 03:24 rhsm
drwxr-xr-x.  2 root   root        4096 May 25 00:00 sa
drwx------.  3 root   root        4096 Oct 26  2011 samba
-rw-------   1 root   root    66942754 May 25 10:35 secure
-rw-------   1 root   root   143240690 Apr 29 03:42 secure-20120429
-rw-------   1 root   root   112453811 May  6 03:16 secure-20120506
-rw-------   1 root   root    68085694 May 13 03:37 secure-20120513
-rw-------   1 root   root    57013588 May 20 03:13 secure-20120520
drwxr-xr-x.  2 root   root        4096 Oct 29  2011 spice-vdagentd
-rw-------   1 root   root         205 May 23 12:39 spooler
-rw-------   1 root   root           0 Apr 22 03:08 spooler-20120429
```

Al sistemas se le ha configurado una IP para poder tener gestión sobre el equipo, desde un dispositivo de esta red este equipo será accesible a través de la 10.100.33.110, pero remotamente existe otro NAT (no el del capítulo anterior) para acceder, el cual es 10.0.7.219.

```
[root@app-miquell ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr E4:1F:13:BF:E1:14
          inet addr:10.100.33.110  Bcast:10.100.33.255  Mask:255.255.255.0
          inet6 addr: fe80::e61f:13ff:febf:e114/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1727075765 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1746591482 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:228290546908 (212.6 GiB)  TX bytes:260537923197 (242.6 GiB)
          Interrupt:28 Memory:92000000-92012800
```

Configuración interface de red activa

El siguiente paso sería configurar otra IP para la segunda tarjeta de red y así tener redundada la administración en caso de caída de una de las redes o de que se estropeara una de las tarjetas.

Hay que tener en cuenta que lo recomendable es configurar cada tarjeta de red con una IP de redes diferentes (en caso de que sea posible por configuración de red), para así prevenir la pérdida de señal por caída total de una red.

El uso de memoria está correctamente dimensionado para la aplicación, podemos demostrarlo en la siguiente imagen:

```
[root@app-miquel1 ~]# top
top - 10:41:31 up 162 days, 16:02,  2 users,  load average: 5.75, 3.71, 3.53
Tasks: 283 total,   2 running, 281 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.7%us,  0.7%sy,  0.0%ni, 74.1%id, 24.4%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   8015140k total,  2769272k used,  5245868k free,   106032k buffers
Swap: 10256376k total,       72k used, 10256304k free,  1448860k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1340 nagios    20   0  239m  33m 2584 R  7.8  0.4 425:53.28 nagios
 1202 root      20   0 15224 1456  968 R  0.3  0.0   0:00.04 top
 1430 root      20   0     0    0    0 S  0.3  0.0  87:40.78 kondemand/5
22149 root      20   0     0    0    0 D  0.3  0.0   8:05.48 flush-253:4
    1 root      20   0 19400 1572 1260 S  0.0  0.0  97:22.56 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.00 kthreadd
```

Memoria y procesos activos en sistema

Se puede observar a la hora de hacer el “top” a la máquina que:

- RAM: de los 8 GB asignados el consumo en la captura es inferior a 3 GB.

- El “load average” o “nivel de carga” de la máquina en intervalos de tiempo de 1 minuto, 5 minutos y 15 minutos es correcto, teniendo en cuenta todos los procesos que está ejecutando internamente.

- El uso de CPU es muy bajo, se puede observar que el total de CPU libre es del 75% aproximadamente.

- SWAP: al sistema se le han asignado 10 GB de swap por posibles problemas puntuales o a largo plazo con la memoria RAM. Como se puede observar en la imagen, el uso de swap es mínimo, unos 72 kb, con casi su totalidad libres.

- El número de procesos en ejecución es adecuado, teniendo en cuenta el background de la aplicación, en la captura se observa que está gestionando unos 283 procesos, teniendo 2 en ejecución y los demás dormidos a la espera de alguna señal y que no existen procesos que se hayan quedado zombies por algún problema de la aplicación.
  Entre estos procesos, se puede ver arrancado y consumiendo CPU el nagios, con un 7,8% de uso de CPU en el momento de la captura y como procesos que más recursos está consumiendo.

2.3 Análisis de las aplicaciones necesarias

En este punto se ampliarán los 4 puntos comentados en el capítulo 1.

2.3.1 Intérprete de perl

Casi todos los scripts realizados, modificados y aplicados directamente para su uso utilizan perl para su ejecución.

A continuación se muestra la lista de los implicados en los equipos cisco, por poner un ejemplo:

```
[root@app-miquell plugins]# ls -la check_CISCO_*
-rwxr-xr-x 1 root root 3880 May 24 10:04 check_CISCO_CPU.pl
-rwxr-xr-x 1 root root 4075 May 24 11:01 check_CISCO_Fuentes.pl
-rwxr-xr-x 1 root root 2505 May  2 11:49 check_CISCO_interface.pl
-rwxr-xr-x 1 root root 4026 May 23 22:34 check_CISCO_Memoria.pl
-rwxr-xr-x 1 root root 2942 May 24 09:17 check_CISCO_Modulos.pl
-rwxr-xr-x 1 root root 3066 May 24 09:01 check_CISCO_Ventilador.pl
```

Perl scripts de cisco

Aparte de los mostrados anteriormente existen otros que controlan otros aspectos en los equipos. El encargado de ir ejecutando la cola en orden es Nagios, se puede ver en cada momento la ejecución de todo script de perl que se está ejecutando si miramos los procesos asociados a este lenguaje.

Se van a poner unos ejemplos de consultas realizadas a intervalos de 2 o 3 segundos:

```
[root@app-miquell plugins]# ps -ef | grep perl
nagios    9045  9044  1 23:39 ?       00:00:00 /usr/bin/perl /usr/lib64/nagios/plugins/check_snmp_storage.pl -H 10.100.2.201 -C Miquel -m ^Virtual -w 85 -c 95 -f
root      9086 19546  0 23:39 pts/1   00:00:00 grep perl
```

```
[root@app-miquell plugins]# ps -ef | grep perl
nagios    8981 24610  8 23:39 ?        00:00:00 /usr/bin/perl /usr/local/pnp4nagios/libexec/process_perfdata.pl
nagios    8982 24610  7 23:39 ?        00:00:00 /usr/bin/perl /usr/local/pnp4nagios/libexec/process_perfdata.pl
root      8984 19546  0 23:39 pts/1    00:00:00 grep perl
```

```
[root@app-miquell plugins]# ps -ef | grep perl
root      8987 19546  0 23:39 pts/1    00:00:00 grep perl
```

Capturas de procesos perl en ejecución

Como se aprecia en las capturas anteriores, los procesos que requieren perl se van procesando en un orden, los procesos van entrando y ejecutándose en ese orden. En la última captura se aprecia como

en ocasiones, puede haber momentos en los cuales no lleguen procesos a ejecutarse que requieran perl.

2.3.2 Intérprete de Shell script

En algunos casos se ha requerido de un script que requería ser ejecutado con Shell scripting.

```
[root@app-miquell plugins]# ls -la *.sh
-rwxr-xr-x 1 root root   299 Apr 13 11:16 a.sh
-rwxr-xr-x 1 root root  1369 Jan 11 16:34 cacti_cpu.sh
-rwxr-xr-x 1 root root  4548 Jan 25 11:16 cacti_mem_individual.sh
-rwxr-xr-x 1 root root  1644 Jan 11 16:34 cacti_zigbee.sh
-rwxr-xr-x 1 root root   701 Jun 29  2011 check_block_sessions.sh
-rwxr-xr-x 1 root root  2607 Jan 11 16:34 check_hp_fans.sh
-rwxr-xr-x 1 root root  4845 Jan 11 16:34 check_hp_insight_log.sh
-rwxr-xr-x 1 root root  3936 Jan 11 16:34 check_hp_logdrv.sh
-rwxr-xr-x 1 root root  2259 Jan 11 16:34 check_hp_mem.sh
-rwxr-xr-x 1 root root  2569 Jan 11 16:34 check_hp_phydrv.sh
-rwxr-xr-x 1 root root  3613 Jan 11 16:34 check_hp_pwr.sh
-rwxr-xr-x 1 root root  4321 Jan 11 16:34 check_hp_temp.sh
-rw-r-xr-x 1 root root  6080 Jan 11 16:34 check_ironport.sh
-rwxr-xr-x 1 root root   890 Jun 22  2011 check_netbackup_errors.sh
-rwxr-xr-x 1 root root  5339 Jan 11 16:34 check_RAID.sh
-rwxr-xr-x 1 root root  7444 Feb 24  2011 check_sap3_multi.sh
-rwxr-xr-x 1 root root  3067 Apr 24 13:26 check_sap3.sh
```

Shell scripts

2.3.3 Nagios

Encargado principalmente del gestor de colas de los scripts que miran el estado de los diferentes equipos y servicios monitorizados, su segunda función, en orden de importancia, sería la visualización a nivel gráfico del estado de la red.

Permite la visualización de todo servicio en colores, dependiendo del nivel de alerta que haya generado, por defecto en su configuración tenemos que:

- Verde: Estado OK, no ha ocurrido nada en el servidor/servicio en la última comprobación del estado.
- Amarillo: Estado Warning, en la última comprobación se ha registrado algún evento que requiere nuestra atención.
- Rojo: Estado Crítico, en la última comprobación de estado ha ocurrido un error que se debe solucionar con urgencia.

Además existe un cuarto estado:

- Naranja: Estado Desconocido, existe algún problema no definido en algún servicio.

Este estado es el menos representado, poniendo un ejemplo, se daría un estado desconocido si se extrajese una unidad de disco duro de algún Windows (que no fuese la unidad C:). Esta acción generaría un estado desconocido de la unidad removida.

2.3.4 Cliente de ssh

Necesitamos un cliente de ssh con el cual se puedan abrir conexiones por el puerto 22 tcp hacia un equipo remoto que acepte peticiones por este puerto.

Principalmente se necesita para enviar las notificaciones cada vez que ocurre algún evento en el sistema de monitorización.

Se analizará algunas de las peticiones que se realizan para enviarlas por SMS, e-mail y llamadas, hay que tener en cuenta que dependiendo de si se trata de un equipo o un servicio, la petición ssh es diferente, ya que se necesitan parámetros diferentes de envío.

- SMS:
  - Host: /usr/bin/ssh 10.0.7.195 "/bin/echo 'UPF: $HOSTNAME$ with IP=$HOSTADDRESS$ was $NOTIFICATIONTYPE$ at `date`' | /usr/local/bin/gnokii --sendsms $CONTACTPAGER$"
  - Servicio: /usr/bin/ssh 10.0.7.195 "/bin/echo 'XXX: $SERVICEDESC$ in $HOSTNAME$ with IP=$HOSTADDRESS$ was $SERVICESTATE$ at `date`' | /usr/local/bin/gnokii --sendsms $CONTACTPAGER$"

- E-mail:
  - Host: /usr/bin/ssh 10.0.7.195 '/usr/bin/mailx -r "Alert SCC <alert2@mss.scc.com>" -s "MAG: $HOSTNAME$ $HOSTADDRESS$ Estado del host: $HOSTSTATE$" $CONTACTEMAIL$ </dev/null'
  - Servicio: /usr/bin/ssh 10.0.7.195 '/usr/bin/mailx -r "Alert SCC <alert2@mss.scc.com>" -s "MAG: $HOSTNAME$ $HOSTADDRESS$ Servicio $SERVICEDESC$: $SERVICESTATE$ - $SERVICEOUTPUT$" $CONTACTEMAIL$ </dev/null'

- Llamada:
  - Host y servicio: /usr/bin/ssh 10.0.7.195 "/bin/echo 'MAG: $HOSTNAME$ with IP=$HOSTADDRESS$ was $NOTIFICATIONTYPE$ at `date`' | /usr/local/bin/gnokii --dialvoice $CONTACTPAGER$"

Todas las variables entre '$' son las que nagios cogerá automáticamente y enviará para que el la notificación se vean claramente los datos. En apartados posteriores se verán ejemplos.

2.4 Estructura interna del software de monitorización

En este apartado se verá cómo funciona internamente el software Nagios y cómo se relaciona con las constantes peticiones que se realizan a los equipos.

Recordamos que nagios, en su versión inicial funciona con ficheros de texto plano. Se verán algunos ejemplos de cómo se introducen nuevos datos.

La estructura interna de dichos ficheros es la siguiente:

```
[root@app-miquell objects]# pwd
/etc/nagios/objects
[root@app-miquell objects]# ls -la
total 664
drwxrwxr-x 5 nagios apache   4096 May 24 11:58 .
drwxr-xr-x 4 root   root     4096 May 17 18:19 ..
drwxrwxr-x 4 nagios apache 548864 May 24 11:58 backup
-rw-r--r-- 1 apache apache  18550 May 24 11:58 commands.cfg
-rw-r--r-- 1 apache apache   1444 May 24 11:58 contactgroups.cfg
-rw-r--r-- 1 apache apache   2425 May 24 11:58 contacts.cfg
-rw-r--r-- 1 apache apache    917 May 24 11:58 contacttemplates.cfg
-rw-r--r-- 1 apache apache    662 May 24 11:58 hostdependencies.cfg
-rw-r--r-- 1 apache apache    664 May 24 11:58 hostescalations.cfg
-rw-r--r-- 1 apache apache    682 May 24 11:58 hostextinfo.cfg
-rw-r--r-- 1 apache apache   9630 May 24 11:58 hostgroups.cfg
drwxrwxr-x 2 apache apache  20480 May 24 11:58 hosts
-rw-r--r-- 1 apache apache   5866 May 24 11:58 hosttemplates.cfg
-rw-r--r-- 1 apache apache    668 May 24 11:58 servicedependencies.cfg
-rw-r--r-- 1 apache apache    670 May 24 11:58 serviceescalations.cfg
-rw-r--r-- 1 apache apache    688 May 24 11:58 serviceextinfo.cfg
-rw-r--r-- 1 apache apache   1961 May 24 11:58 servicegroups.cfg
drwxrwxr-x 2 apache apache   4096 May 24 11:58 services
-rw-r--r-- 1 apache apache   3124 May 24 11:58 servicetemplates.cfg
-rw-r--r-- 1 apache apache    258 Nov 16  2011 templates.cfg
-rw-r--r-- 1 apache apache   2007 May 24 11:58 timeperiods.cfg
[root@app-miquell objects]#
```

Estructura directorio nagios

En la imagen anterior aparecen todos los ficheros que se necesitan para realizar una configuración. Vamos a ver algunos casos, aunque recordamos que la forma más fácil de realizarlo y rápida será desde la interface web que se verá en el manual y que nos aporta NagiosQL.

2.4.1 Equipos monitorizados

En el siguiente directorio se exponen todos los ficheros de los equipos monitorizados de la A a la Z.

```
[root@app-miquel1 hosts]# pwd
/etc/nagios/objects/hosts
[root@app-miquel1 hosts]# ls -la
total 1736
drwxrwxr-x 2 apache apache 20480 May 24 11:58 .
drwxrwxr-x 5 nagios apache  4096 May 24 23:01 ..
-rw-r--r-- 1 apache apache   912 May 24 07:59 10.100.7.230.cfg
-rw-r--r-- 1 apache apache   860 May 24 11:58 aluminio.cfg
-rw-r--r-- 1 apache apache   913 May 24 11:58 Americium01.cfg
-rw-r--r-- 1 apache apache   913 May 24 11:58 Americium02.cfg
… (Existen más intermedios)
-rw-r--r-- 1 apache apache   907 May 24 11:58 zirconium101.cfg
-rw-r--r-- 1 apache apache   907 May 24 11:58 zirconium102.cfg
-rw-r--r-- 1 apache apache  1109 May 24 11:58 zirconium103.cfg
-rw-r--r-- 1 apache apache   907 May 24 11:58 zirconium104.cfg
```

Y un ejemplo de uno de ellos sería:

```
define host {
        host_name                       tin01
        alias                           tin01
        address                         10.255.255.50
        hostgroups                      Domain Controler y DNS
        use                             NotificacionVariable
        register                        1
}
```

Ejemplo configuración de un equipo

Como se puede observar contiene los datos básicos para la creación de un equipo, sin ningún servicio, concretamente:

- Host_name: Nombre del equipo.
- Alias: descripción (si la hubiese).
- Address: dirección IP y nombre resuelto por DNS.
- Hostgroups: Grupo al que pertenece para que se vea agrupado (Windows, AIX, DNS, etc.).
- Use: Aquí se especifica una variable que será la que contenga los contactos a los cuales se les enviará un evento en caso de que algo suceda en la red.
- Register: Indica si el equipo está activo o no (estado binario).

2.4.2 Servicios monitorizados

Para los servicios tenemos la siguiente configuración:

```
[root@app-miquel1 services]# pwd
/etc/nagios/objects/services
[root@app-miquel1 services]# ls -la
total 648
drwxrwxr-x 2 apache apache  4096 May 24 11:58 .
drwxrwxr-x 5 nagios apache  4096 May 24 23:01 ..
-rw-r--r-- 1 apache apache  1652 May 24 11:58 check_aix.cfg
-rw-r--r-- 1 apache apache  1682 May 24 11:58 check_bloqueos.cfg
-rw-r--r-- 1 apache apache 18459 May 24 11:58 check_cisco.cfg
… (Existen más intermedios)
-rw-r--r-- 1 apache apache  3534 May 24 11:58 check_tcp.cfg
-rw-r--r-- 1 apache apache  1483 May 24 11:58 check_unix.cfg
-rw-r--r-- 1 apache apache   869 May 24 11:58 check_vrrp2.cfg
-rw-r--r-- 1 apache apache   869 May 24 11:58 check_vrrp.cfg
-rw-r--r-- 1 apache apache   859 May 24 11:58 check_web.cfg
```

Y si miramos dentro del fichero de configuración:

```
define service {
        host_name                       CORE_6500_Torrejon,CORE6500-VM_MA_CORE_1…       (todos)
        service_description             Estado CPU
        use                             local-service
        check_command                   check_Cisco_CPU!Miquel!80!90
        register                        1
}

define service {
        host_name                       CORE_6500_Torrejon,CORE6500-VM_MA_CORE_1…       (todos)
        service_description             Estado fuentes alimentacion
        use                             local-service
        check_command                   check_Cisco_Fuentes!Miquel!1!2
        max_check_attempts              3
        check_interval                  15
        retry_interval                  2
        register                        1
}
```

En estos ejemplos se puede observar la consulta SNMP que se le lanza a cada uno de los equipos que se reflejan en “host_name” (deben de ir todos los equipos cisco en este caso).
Tenemos:

- Host_name: Equipos a los cuales corresponde el servicio (desde 1 a todos).
- Service_Descrption: Nombre que aparecerá para el servicio.
- Use: Aquí se especifica una variable que será la que contenga los contactos a los cuales se les enviará un evento en caso de que algo suceda en la red.
- Check_command: Es la sentencia que especifica los parámetro que se pasarán al hacer la consulta SNMP. Se verá en el siguiente punto un ejemplo.
- Max_check_attempts: Especifica el número de veces que se comprobará el servicio antes de enviar una notificación en caso de ser necesario.
- Check_interval: Marca el tiempo en minutos entre cada comprobación del servicio.

2.4.3 Comandos

En este paso se definirá qué tipo de consulta se realiza (SNMP o NRPE). Para el caso anterior de los equipos Cisco vemos que necesitamos tres parámetros (ARG1, ARG2 y ARG3).

check_command check_Cisco_CPU!**Miquel!80!90**

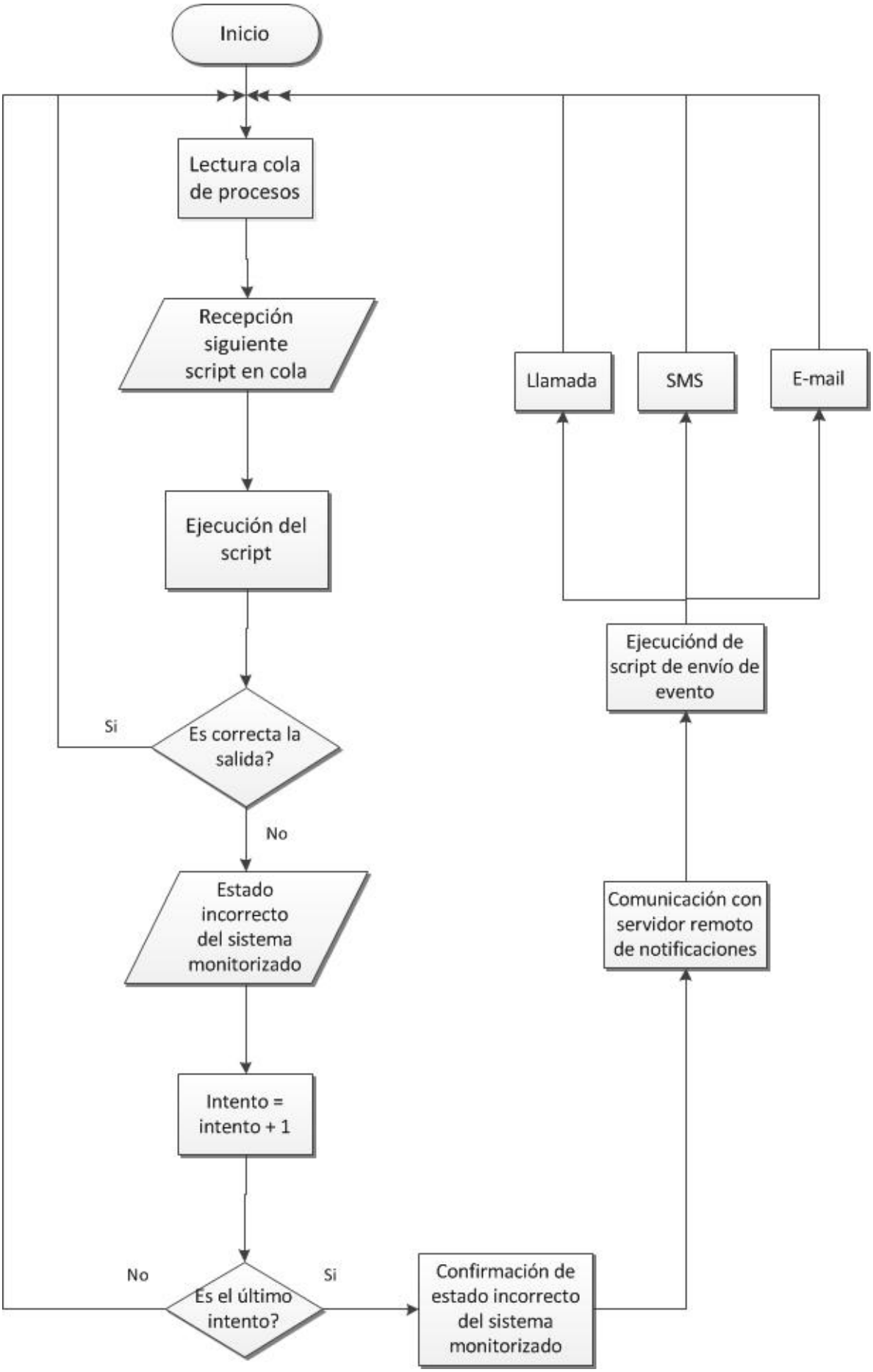
Donde se pueden ver los 3 parámetros requeridos, en orden:

- C (ARG1): Comunidad SNMP, en este caso Miquel.
- w (ARG2): Valor de umbral de “Warning”, en este caso 80.
- c (ARG3): Valor de umbral de “Critical”, en este caso 90.

```
[root@app-miquel1 objects]# pwd
/etc/nagios/objects
[root@app-miquel1 objects]# vi commands.cfg

define command {
        command_name            check_Cisco_CPU
        command_line            $USER1$/check_CISCO_CPU.pl -H $HOSTADDRESS$ -C $ARG1$ -w
$ARG2$ -c $ARG3$
        register                1
}

define command {
        command_name            check_Cisco_Fuentes
        command_line            $USER1$/check_CISCO_Fuentes.pl -H $HOSTADDRESS$ -C $ARG1$ -
w $ARG2$ -c $ARG3$
        register                1
}
```

Los tres ficheros anteriores son de los más importantes en la configuración que tiene Nagios, aun así con estos solo no funcionaría, pero ya podemos tener una idea de cómo funcionarían las peticiones.

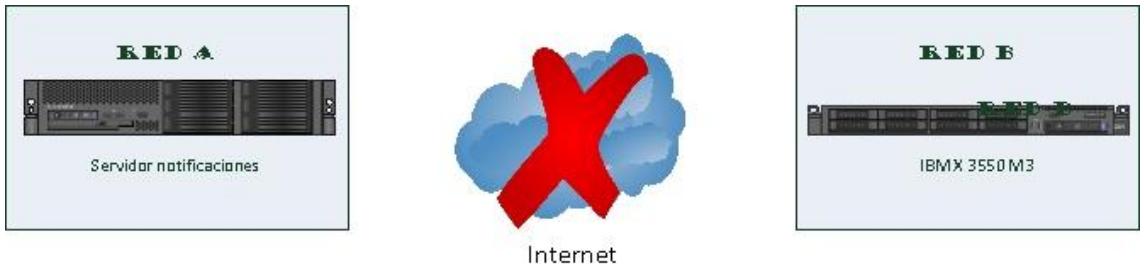
Para entender el ciclo de cada una de las peticiones SNMP o NRPE que compone cada servicio se ha realizado el siguiente diagrama de flujo.


Inicio
Lectura cola de procesos
Recepción siguiente script en cola
Ejecución del script
Es correcta la salida?
Si
No
Estado incorrecto del sistema monitorizado
Intento = intento + 1
Es el último intento?
No
Si
Confirmación de estado incorrecto del sistema monitorizado
Comunicación con servidor remoto de notificaciones
Ejecuciónd de script de envío de evento
Llamada
SMS
E-mail

2.5 Estudio del NAT

En este apartado se analizará el funcionamiento interno del NAT.

El problema que se plantea es la comunicación entre dos IPs de diferente red que no tienen direccionamiento público y por lo tanto, a priori la comunicación directa a través de internet no funciona.



Para ello debemos hacer que una IP de nuestra red la transforme en la de la otra red internamente.

Tenemos:

- Red B: Direccionamiento privado 10.100.33.110.
- Red A: Direccionamiento privado 10.3.0.40.

En la red A tenemos un router VPN, que es un terminador de túneles en el cual debemos configurar el NAT.

Debido al problema con el que nos encontramos necesitamos dos NATs:

1. Necesitamos acceder desde nuestra red A al equipo de la red B para administrar por ssh, interface web, etc.
2. Necesitamos acceder desde el equipo en concreto de la red B al equipo de la red A.

Vemos las configuraciones:

Caso 1: NAT con origen red A y destino red B a la IP 10.100.33.110, accedemos a través de la 10.0.7.219.

```
RT_Vpn01#
RT_Vpn01#sh run | i 10.100.33.110
ip nat outside source static 10.100.33.110 10.0.7.219
RT_Vpn01#
```

Configuración NAT en router

Prueba OK: Sabemos que hay una aplicación apache escuchando por el puerto 80, vamos a ver como atacando a la IP de NAT accedemos.

```
[root@App-scc ~]# telnet 10.0.7.219 80
Trying 10.0.7.219...
Connected to 10.0.7.219 (10.0.7.219).
Escape character is '^]'.
```

Prueba telnet

Caso 2: NAT con origen red B y destino red A a la IP 10.3.0.40, accedemos a través de la 10.0.7.195.

```
RT_Vpn01#
RT_Vpn01#sh run | i 10.0.7.195
ip nat inside source static 10.3.0.40 10.0.7.195 route-map
RT_Vpn01#
```

Configuración NAT en router

Prueba OK: Sabemos que el servidor de notificaciones tiene el puerto 22 (ssh) escuchando, probaremos que realmente atacando al NAT accederemos.

```
[root@app-miquell ~]# telnet 10.0.7.195 22
Trying 10.0.7.195...
Connected to 10.0.7.195 (10.0.7.195).
Escape character is '^]'.
SSH-2.0-OpenSSH_4.3
```

Prueba telnet

Ahora ya tenemos un NAT para cada uso que le queremos dar, el caso número 1 es sólo a modo de administración, el importante sería el caso número 2 que es el que se usa para cuando la aplicación quiere enviar notificaciones.

2.6 Configuración del intercambio de claves y comando que se intercambian

Como ya se ha comentado en el capítulo anterior, es el proceso encargado de que dos equipos de igual o diferente red puedan acceder el uno al otro sin necesidad de introducir las credenciales de acceso.

Se estudiará cómo se ha realizado paso a paso y cómo podemos comprobar que el acceso funciona correctamente.

Configuración paso a paso

1. Entramos al servidor origen de las notificaciones.

```
[root@app-miquell /]# uname -a
Linux app-miquell 2.6.32-220.el6.x86_64 #1 SMP Wed Nov 9 08:03:13 EST 2011 x86_64 x86_64 x86_64 GNU/Linux
```

2. Accedemos al fichero que contiene la clave pública del servidor para copiarla. El fichero que la contiene en esta distribución se llama ID_DSA.pub.

```
[root@app-miquell /]# ls -la /var/log/nagios/.ssh/id_dsa.pub
-rw-r--r-- 1 nagios nagios 608 Mar 27 15:51 /var/log/nagios/.ssh/id_dsa.pub
```

3. Copiamos el texto que haya en su interior.

4. Abrimos una sesión ssh contra en servidor destino (servidor de notificaciones llamado *Kripto*).

```
[root@kripto ~]# uname -a
Linux kripto 2.6.18-238.1.1.el5PAE #1 SMP Tue Jan 4 13:53:16 EST 2011 i686 i686 i386 GNU/Linux
```

5. Accedemos con el usuario nagios, ya que el equipo origen de la petición lo realizará con este usuario.

```
[root@kripto ~]# su nagios
[nagios@kripto root]$
```

6. Vamos a /home/nagios/.ssh/

7. Editamos el fichero authorized_keys2.

```
[nagios@kripto .ssh]$ pwd
/home/nagios/.ssh
[nagios@kripto .ssh]$ vi authorized_keys2
```

8. Copiamos lo obtenido en el punto 3 en este fichero, quedándonos de la siguiente manera:

ssh-dss
AAAAB3NzaC1kc3MAAACBAPhQ11QINVO4ex+2OKqveStdTJ3e1J27b9MyblKclo18uYIa/QyMnIj
MJxroDMYVLY9hcAg1nsJyP+5fqnQr1/8kdXrteWPNN261/T7urTVtDr2CxsQWJnPuuw5qpZOV88

VhihnmZcCiWRwHXUaNR/kKcFfmqjd4vVPCLXHsmi3/AAAAFQDHmyBAx8+Rher8nfgOoNhb2sfJ
2QAAAIBPtiF4EpV8+1V61l4FTpaQwJD7yB1nvYP5On5vKWkam8heJW+/5Ofy5Wq228dD9HqfjfIe
5awXVvQd47o3nivHMdFkWypZwwlAAqOWgVLcWOfQqOnJFa3F+ilgsBjuCKhgQVDdv3j+RM5FA
epVsp0Rk1+iGGcv+ofRVYCmtYMhwgAAAIEA5yhlcY+k9X6AFm3v3y9Vh6uLVzS64EMLPGus201u
nRhj5NIxvUB90ckhhzS6WsD6Q2EeOG307+ynXJ4mGvA2Z0y9KJzqb5VEO9k3Wx5mThLNHeV+1s
RsLfX48lZpMNMdsYAB2LtJtjiJkeaRjwlgOj4xyV5VwBPKZOD/r63otVc= nagios@app-miquel1

En el último punto se observa la clave pública del servidor origen introducida en el servidor destino, con el usuario que realizará la petición y el nombre de host. De esta manera la comunicación es unidireccional, ya que sólo un host conoce a otro.

Para ver que funciona correctamente se realizarán unas pruebas:

Prueba número 1 – Funcionamiento correcto

1. Entramos en el equipo origen de la petición.

```
[root@app-miquel1 ~]# uname -a
Linux app-miquel1 2.6.32-220.el6.x86_64 #1 SMP Wed Nov 9 08:03:13 EST 2011 x86_64 x86_64 x86_64 GNU/Linux
```

2. Accedemos con usuario nagios.

```
[root@app-miquel1 ~]# su nagios
bash-4.1$
```

3. Recordamos el NAT que es la 10.0.7.195.

4. Atacamos por ssh a esta IP.

```
bash-4.1$ ssh 10.0.7.195
Last login: Fri May 25 07:51:14 2012 from 10.0.7.219
[nagios@kripto ~]$ _
```

5. Comunicación establecida con usuario nagios en servidor destino! Como se observa no se ha solicitado ni usuario ni password y ya nos encontramos dentro de Kripto.

Prueba número 2 – Funcionamiento incorrecto

1. Entramos en el equipo origen de la petición.

```
[root@app-miquell ~]# uname -a
Linux app-miquell 2.6.32-220.el6.x86_64 #1 SMP Wed Nov 9 08:03:13 EST 2011 x86_64 x86_64 x86_64 GNU/Linux
```

2. No accedemos al usuario nagios, es decir, nos quedamos con el usuario de root o cualquier otro (en este caso estamos como root).

3. Intentamos atacar a la IP de NAT por ssh.

4. Error! Nos pide el password.

```
[root@app-miquell ~]# ssh 10.0.7.195
root@10.0.7.195's password:
```

Este caso ya no serviría para nada, recordamos que la aplicación necesita una comunicación sin trabas entre los dos equipos, ya que está continuamente a la espera de enviar las notificaciones. Si este último caso se diese, el servidor enviaría la primera petición y se quedaría esperando (eternamente o hasta hacer un time out) a que alguien introdujese la contraseña y así repetidamente.

Prueba 3 – Funcionamiento de una notificación

1. Entramos en el equipo origen de la petición.

```
[nagios@kripto ~]$ uname -a
Linux kripto 2.6.18-238.1.1.el5PAE #1 SMP Tue Jan 4 13:53:16 EST 2011 i686 i686 i386 GNU/Linux
```

2. Accedemos con usuario nagios.

```
[root@app-miquell ~]# su nagios
bash-4.1$
```

3. Se va a simular una de las notificaciones.

```
bash-4.1$ /usr/bin/ssh 10.0.7.195 '/usr/bin/mailx -r "Alerta!" -s "Simulacion e-mail OK" manolo.mg@gmail.com </dev/null'
Null message body; hope that's ok
```

4. La conexión ssh contra el NAT se ha realizado correctamente y el intercambio de claves ha funcionado.

Captura de gmail de prueba

2.7 Configuración de los protocolos de monitorización en cada entorno utilizado

En este punto se estudiará cómo se deben configurar los protocolos seleccionados para el proyecto para poder aceptar peticiones de un equipo remoto.

2.7.1 Configuración de protocolo SNMP

El protocolo SNMP, a modo gráfico, funcionaría de la siguiente forma:

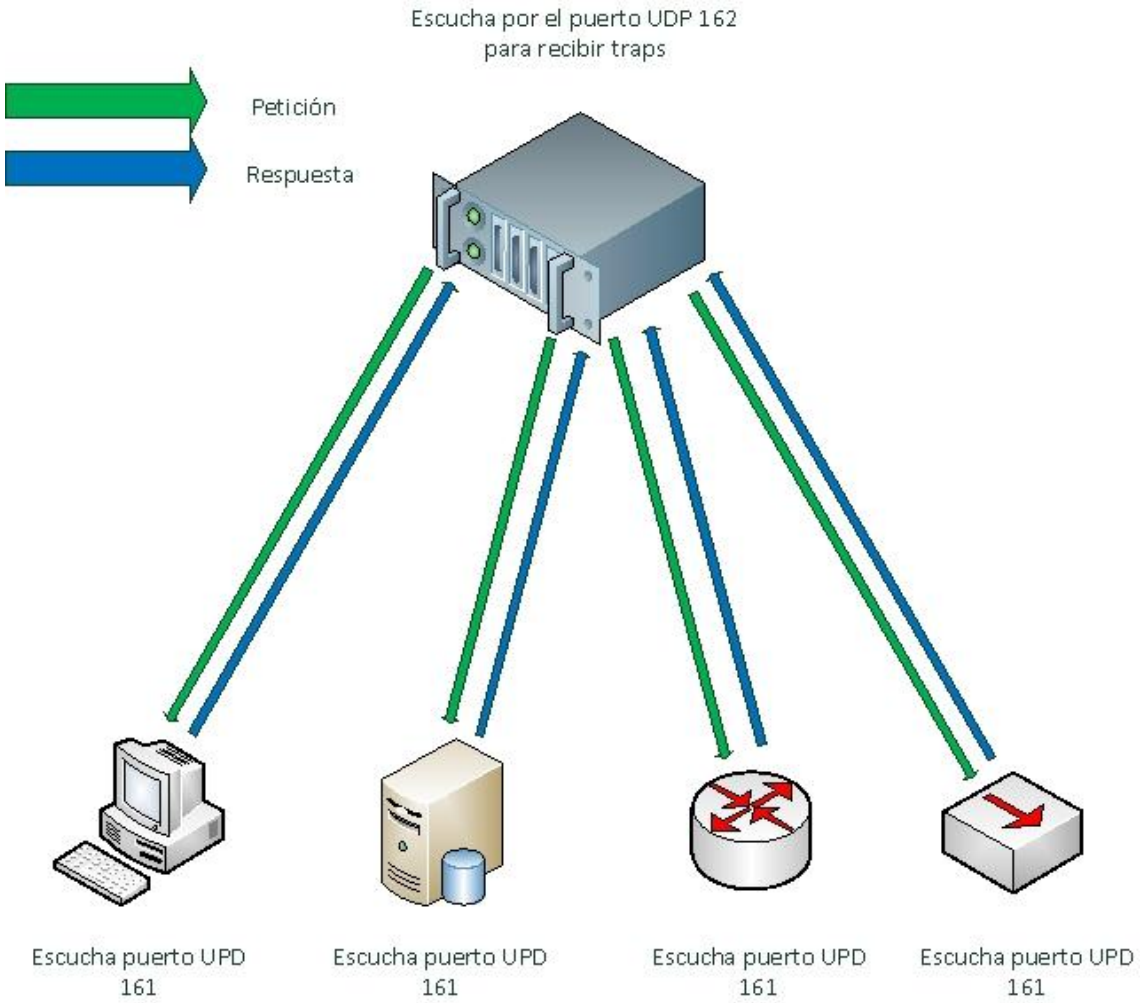


Diagrama representativo SNMP

Configuración en los diferentes equipos

A continuación se detallará la configuración según el sistema operativo.

1. IBM AIX

Para su configuración en sistemas AIX se deben seguir los siguientes pasos:

```
Last unsuccessful login: Thu May 10 09:36:56 GMT+02:00 2012 on ssh from app-miquel01.miquel.es
Last login: Thu May 10 09:36:59 GMT+02:00 2012 on /dev/pts/0 from app-miquel01.miquel.es
*******************************************************************************
*                                                                             *
*                                                                             *
*  Welcome to AIX Version 6.1!                                                *
*                                                                             *
*                                                                             *
*  Please see the README file in /usr/lpp/bos for information pertinent to    *
*  this release of the AIX Operating System.                                  *
*                                                                             *
*                                                                             *
*******************************************************************************

[root@mar3prddgs]:/# uname -a
AIX mar3prddgs 1 6 00C735BA4C00
[root@mar3prddgs]:/#
```

Captura login servidor AIX

1. Localizamos el fichero de configuración de snmp para configurarlo. En el caso de los AIX, el mismo SO ya viene con el software instalado, con los que no será necesario instalar ningún paquete.

```
[root@mar3prddgs]:/# ls -la /etc/snmpdv3.conf
-rw-r-----    1 root     system         2506 Apr 12 17:31 /etc/snmpdv3.conf
[root@mar3prddgs]:/#
```

Fichero configuración snmp

2. Editamos el fichero de configuración.

```
[root@mar3prddgs]:/# vi /etc/snmpdv3.conf
"/etc/snmpdv3.conf" 55 lines, 2506 characters
```

Edición del fichero de configuración

3. Y lo modificamos de la siguiente manera, así es como quedaría en fichero de configuración una vez modificado:

```
VACM_GROUP group1 SNMPv1  public  -
VACM_GROUP group1 SNMPv1  Miquel -
VACM_GROUP group1 SNMPv2c  Miquel  -

VACM_VIEW defaultView 1.3.6.1.4.1.2 - included -
VACM_VIEW defaultView 1.3.6.1.4.1.2.2 - included -
VACM_VIEW defaultView 1.3.6.1.4.1.2.3 - included -
VACM_VIEW defaultView 1.3.6.1.4.1.2.5 - included -
VACM_VIEW defaultView 1.3.6.1.4.1.2.6 - included -
VACM_VIEW defaultView        internet         - included -
VACM_VIEW defaultView        directory                - included -
VACM_VIEW defaultView        mgmt                  - included -
VACM_VIEW defaultView        mib-2                  - included -
VACM_VIEW defaultView        system                  - included -
VACM_VIEW defaultView        aix                   - included -
VACM_VIEW defaultView        xmd                   - included -
VACM_VIEW defaultView        ibm                   - included -
VACM_VIEW defaultView        ibmAgents                - included -

# exclude snmpv3 related MIBs from the default view
VACM_VIEW defaultView        snmpModules      - included -
VACM_VIEW defaultView        1.3.6.1.4          - included -
VACM_VIEW defaultView        1.3.6.1.6          - included -

# exclude aixmibd managed MIBs from the default view
VACM_VIEW defaultView 1.3.6.1.4.1.2.6.191 - included -
VACM_VIEW defaultView 1.3.6.1.6.3.1.1.5   - included -
VACM_VIEW defaultView 1.3.6.1.4.1.2021 - included -
VACM_VIEW defaultView 1.3.6.1.4.1.2.3.1.2.2.2.1.4 - included -

VACM_ACCESS  group1 - - noAuthNoPriv SNMPv1  defaultView - defaultView -
VACM_ACCESS  group1 - - noAuthNoPriv SNMPv2c  defaultView - defaultView -

NOTIFY notify1 traptag trap -

TARGET_ADDRESS Target1 UDP 127.0.0.1 traptag trapparms1 - - -

TARGET_PARAMETERS trapparms1 SNMPv1  SNMPv1  public  noAuthNoPriv -

COMMUNITY public public noAuthNoPriv 127.0.0.1 255.255.255.255  -
COMMUNITY Miquel Miquel noAuthNoPriv 10.100.33.110 255.255.255.255  -
COMMUNITY Miquel Miquel noAuthNoPriv 10.100.7.237 255.255.255.255  -
COMMUNITY Miquel Miquel noAuthNoPriv 10.100.7.236 255.255.255.255  -

DEFAULT_SECURITY no-access - -

logging         file=/usr/tmp/snmpdv3.log       enabled
logging         size=0                          level=0

smux   1.3.6.1.4.1.2.3.1.2.1.2   gated_password  # gated
snmpd smuxtimeout=200 #muxatmd
smux   1.3.6.1.4.1.2.3.1.2.3.1.1   muxatmd_password #muxatmd
smux   1.3.6.1.4.1.2.3.1.2.1.3   xmservd_pw   #xmservd
smux   1.3.6.1.4.1.2.3.1.2.2.1.1.2   dpid_password   #dpid
smux 1.3.6.1.4.1.2.3.1.2.1.5 clsmuxpd_password
```

Fichero de configuración modificado

4. Una vez realizado cualquier cambio en un fichero de configuración, se debe reiniciar el servicio para que los cambios surjan efecto. Usaremos:

   - *stopsrc –s "nombre_servicio"*: Parada del servicio.

   - *lssrc –a | grep "nombre_servicio"*: Comando para ver el estado del servicio, en la captura se puede ver una vez parado y arrancado, en estado "inoperative" y "active" respectivamente.

   - *startsrc –s "nombre_servicio"*: Se inicia el servicio.

```
[root@mar3prddgs]:/#
[root@mar3prddgs]:/# stopsrc -s snmpd
0513-044 The snmpd Subsystem was requested to stop.
[root@mar3prddgs]:/#
[root@mar3prddgs]:/# lssrc -a | grep snmp
 snmpmibd         tcpip            15335432     active
 snmpd            tcpip                         inoperative
[root@mar3prddgs]:/#
[root@mar3prddgs]:/# startsrc -s snmpd
0513-059 The snmpd Subsystem has been started. Subsystem PID is 13172900.
[root@mar3prddgs]:/#
[root@mar3prddgs]:/# lssrc -a | grep snmp
 snmpmibd         tcpip            15335432     active
 snmpd            tcpip            13172900     active
[root@mar3prddgs]:/#
```

Stop/start del servicio

Por último, el protocolo snmp, escucha por el puerto 161 udp. Para comprobar si está escuchando por el puerto usamos el comando "netstat -an", con el cual nos saldrán todos los puertos en "listen".

```
[root@mafirptj]:/# netstat -an | grep 161
udp        0      0  *.161                  *.*
```

Puertos en "listen"

2. SUN SOLARIS

Para su configuración en sistemas Solaris se deben seguir los siguientes pasos:

```
[root@app-miquel1 ~]# ssh madwh
Password:
Last login: Thu May 10 09:34:11 2012

================================================================

  HOSTNAME      ::      madwh
  IP            ::      10.100.1.113
  TYPE          ::      madwh zone
  GLOBAL ZONE   ::      N/A
  NET INTERFACE ::      ce1
  PURPOSE       ::      DataWarehouse
  CPD           ::      Torrejon

 Sun Microsystems Inc.  SunOS 5.10      Generic January 2005
================================================================

You have new mail.
madwh
[madwh:root]/>
```

Ejemplo login sistema Solaris

1. Miramos el servicio para comprobar que no está habilitado, en caso de que no esté disponible hay que instalar el correspondiente a la versión de SO. En este caso está instalado pero no configurado:

```
[madwh:root]/>
[madwh:root]/> svcs -a | grep snmp
disabled       18:32:12 svc:/application/management/snmpdx:default
[madwh:root]/>
```

Muestra servicio "disabled"

2. Configuramos el servicio añadiendo las IPs que queremos que puedan realizar peticiones a través de este protocolo. Se indican con “rocommunity” que significa “Read Only Community” para que solo se puedan consultar datos.

```
[madwh:root]/>
[madwh:root]/> vi /etc/sma/snmp/snmpd.conf
"/etc/sma/snmp/snmpd.conf" 95 lines, 3348 characters

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

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

rocommunity Miquel 10.100.33.110
rocommunity Miquel 10.100.7.236
rocommunity Miquel 10.100.7.237
```

3. Arrancamos el servicio.

```
[madwh:root]/>
[madwh:root]/> svcadm enable snmpdx
[madwh:root]/>
```

Habilitar servicio

4. Comprobamos que se muestra arrancado.

```
[madwh:root]/>
[madwh:root]/> svcs -a | grep snmp
online         18:32:39 svc:/application/management/snmpdx:default
[madwh:root]/>
```

Comprobación servicio activo

5. Y comprobamos que el puerto UDP 161 esté escuchando.

```
[madwh:root]/>
[madwh:root]/> netstat -an | grep 161
      *.161                                Idle
      *.16161                              Idle
[madwh:root]/>
```

Comprobación puerto en “listen”

6. Y comprobamos externamente que la petición se realiza con éxito. En la captura se puede observar cómo se extrae a través de la petición el nombre del equipo y otros datos que posteriormente serán interpretados por los scripts de monitorización.

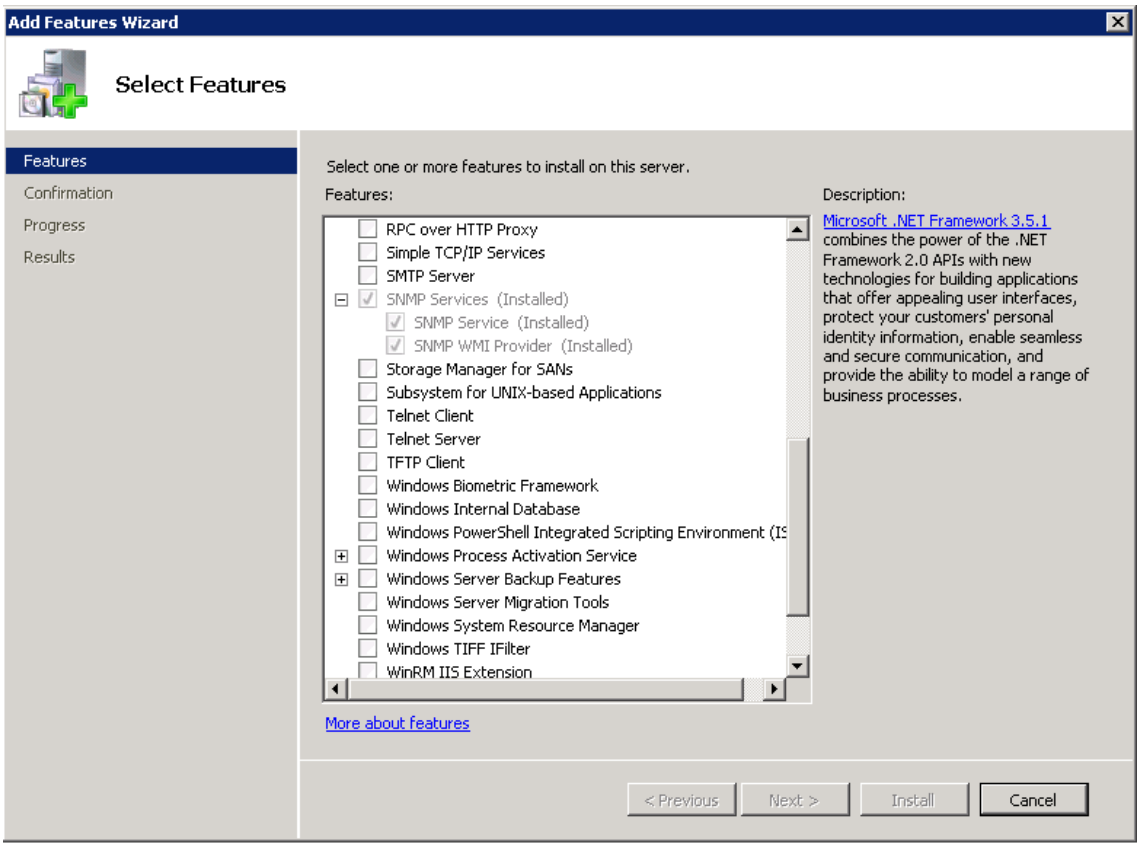
```
[root@app-miquell ~]#
[root@app-miquell ~]# snmpwalk -v2c -c Miquel madwh
SNMPv2-MIB::sysDescr.0 = STRING: SunOS madwh 5.10 Generic_137137-09 sun4u
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.3
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (1351180) 3:45:11.80
SNMPv2-MIB::sysContact.0 = STRING: "System administrator"
SNMPv2-MIB::sysName.0 = STRING: madwh
SNMPv2-MIB::sysLocation.0 = STRING: "System administrators office"
SNMPv2-MIB::sysServices.0 = INTEGER: 72
SNMPv2-MIB::sysORLastChange.0 = Timeticks: (8) 0:00:00.08
SNMPv2-MIB::sysORID.1 = OID: IF-MIB::ifMIB
SNMPv2-MIB::sysORID.2 = OID: SNMPv2-MIB::snmpMIB
SNMPv2-MIB::sysORID.3 = OID: TCP-MIB::tcpMIB
SNMPv2-MIB::sysORID.4 = OID: IP-MIB::ip
SNMPv2-MIB::sysORID.5 = OID: UDP-MIB::udpMIB
SNMPv2-MIB::sysORID.6 = OID: SNMP-VIEW-BASED-ACM-MIB::vacmBasicGroup
SNMPv2-MIB::sysORID.7 = OID: SNMP-FRAMEWORK-MIB::snmpFrameworkMIBCompliance
SNMPv2-MIB::sysORID.8 = OID: SNMP-MPD-MIB::snmpMPDCompliance
SNMPv2-MIB::sysORID.9 = OID: SNMP-USER-BASED-SM-MIB::usmMIBCompliance
SNMPv2-MIB::sysORDescr.1 = STRING: The MIB module to describe generic objects for network interface sub-layers
SNMPv2-MIB::sysORDescr.2 = STRING: The MIB module for SNMPv2 entities
SNMPv2-MIB::sysORDescr.3 = STRING: The MIB module for managing TCP implementations
SNMPv2-MIB::sysORDescr.4 = STRING: The MIB module for managing IP and ICMP implementations
SNMPv2-MIB::sysORDescr.5 = STRING: The MIB module for managing UDP implementations
SNMPv2-MIB::sysORDescr.6 = STRING: View-based Access Control Model for SNMP.
SNMPv2-MIB::sysORDescr.7 = STRING: The SNMP Management Architecture MIB.
SNMPv2-MIB::sysORDescr.8 = STRING: The MIB for Message Processing and Dispatching.
SNMPv2-MIB::sysORDescr.9 = STRING: The management information definitions for the SNMP User-based Security Model.
SNMPv2-MIB::sysORUpTime.1 = Timeticks: (6) 0:00:00.06
SNMPv2-MIB::sysORUpTime.2 = Timeticks: (6) 0:00:00.06
```

Petición SNMP externa

3. WINDOWS

En el siguiente ejemplo se muestra el procedimiento de instalación y configuración en un sistema Windows Server 2008.

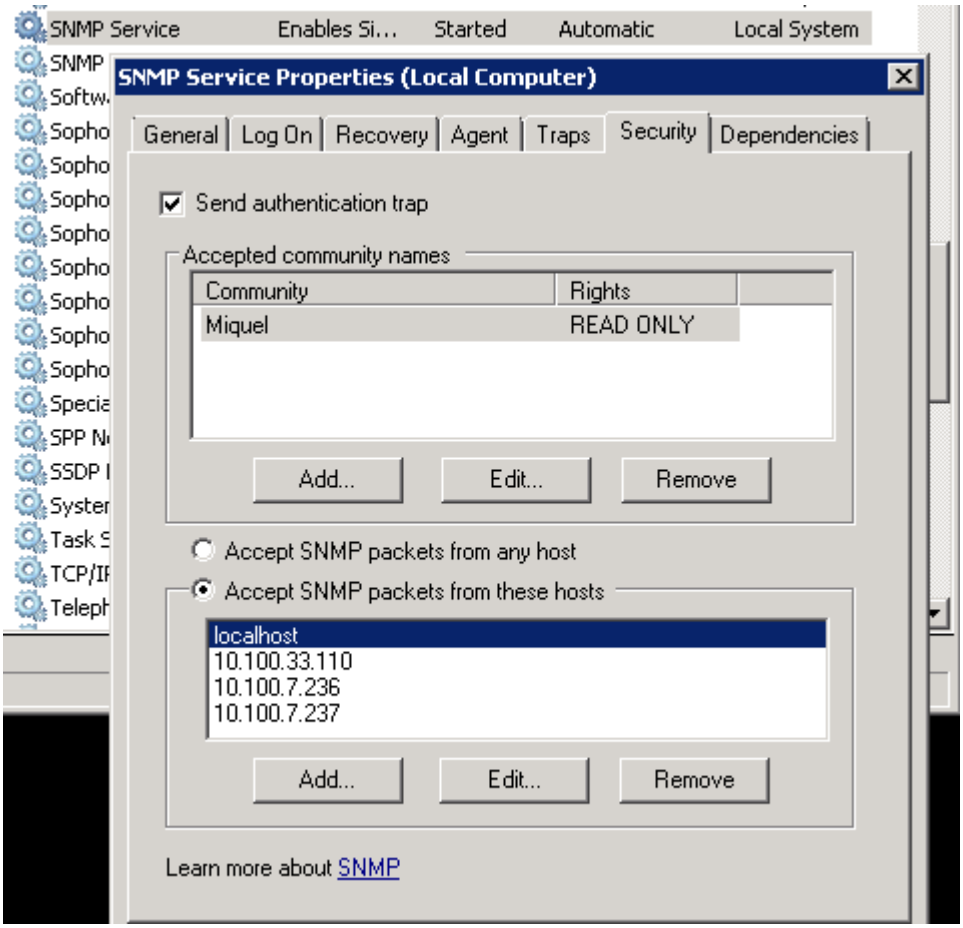
1. Windows server 2008 y Win7 ya tiene en su pre instalación los paquetes del protocolo, por lo que se pueden instalar sin necesidad de insertar ningún CD de instalación de Windows.
   Para ello debemos irnos a “Panel de control” → “Agregar o quitar programas” → “Agregar o quitar software” e instalar las opciones seleccionadas en la siguiente imagen.
   Este procedimiento no es tan sencillo en ninguna de las versiones anteriores de Windows, ya que el paquete no viene pre instalado, y por ello, debemos insertar un CD de Windows, que el programa pedirá automáticamente. Para los casos en los que se ha presentado este problema lo que se ha hecho es compartir el paquete en un servidor central de la red e ir a buscar a esa ruta el paquete cada vez que requería el CD.



Paso 1 en configuración SNMP

2. Una vez finalizado el proceso de instalación nos aparecerá el servicio en la lista de Windows. Para ver que se ha instalado correctamente y acceder a su configuración vamos a MiPC → botón derecho → Administrar → Servicios y aplicaciones → Servicios → SNMP Service. Con estos pasos comprobamos que la instalación fue correcta, por último botón derecho sobre el servicio y propiedades.

3. Seguidamente abrimos el apartado de "Security" y configuramos las IPs remotas que van a poder solicitarle peticiones SNMP. En este caso se han configurado tres de ellas y también se aceptan las peticiones desde el mismo sistema Windows.
   También se debe indicar la "Community", que será el "string" que le pasaremos a la consulta para que sea válida. Los derechos deben estar configurados en solo lectura (RO), también existen del tipo escritura (WO) y lectura/escritura (RW); en este caso solo se verán de lectura debido que lo único que se debe hacer es sacar información del sistema.



Paso 2 en configuración SNMP

4. Reiniciamos el servicio y comprobamos que remotamente acepta las peticiones.

Para los servidores Windows se puede realizar una consulta específica que filtra únicamente por los servicios activos. Para ello indicamos la MIB de servicios que tiene establecida Windows (1.3.6.1.4.1.77.1.2.3.1.X) a través de la cual sólo veremos los servicios.

```
[root@app-miquell /]#
[root@app-miquell /]# snmpwalk -v2c -c Miquel -O a lutetium01 -On 1.3.6.1.4.1.77.1.2.3.1 | cut -d "=" -f 2
 STRING: "Power"
 STRING: "Server"
 STRING: "Netlogon"
 STRING: "IP Helper"
 STRING: "DNS Client"
 STRING: "DHCP Client"
 STRING: "Workstation"
 STRING: "SNMP Service"
 STRING: "Sophos Agent"
 STRING: "Windows Time"
 STRING: "Plug and Play"
 STRING: "Print Spooler"
 STRING: "Task Scheduler"
 STRING: "Windows Update"
 STRING: "Remote Registry"
 STRING: "NSClient++ (x64)"
 STRING: "Windows Firewall"
 STRING: "COM+ Event System"
 STRING: "Sophos Anti-Virus"
 STRING: "Windows Event Log"
 STRING: "IPsec Policy Agent"
 STRING: "Group Policy Client"
 STRING: "Network Connections"
 STRING: "RPC Endpoint Mapper"
 STRING: "Software Protection"
 STRING: "Network List Service"
 STRING: "User Profile Service"
 STRING: "VMware Tools Service"
 STRING: "Base Filtering Engine"
 STRING: "Microsoft FTP Service"
 STRING: "Sophos Message Router"
 STRING: "TCP/IP NetBIOS Helper"
 STRING: "VMware Upgrade Helper"
 STRING: "Application Management"
 STRING: "Cryptographic Services"
 STRING: "COM+ System Application"
 STRING: "Certificate Propagation"
 STRING: "Remote Desktop Services"
```

Comprobación remota de petición SNMP

En la imagen anterior se aprecian los servicios activos que tiene el servidor Windows 2008, preguntando a cualquiera de esos servicios, el estado retornado será “OK”, los que no aparezcan en la lista es debido a que no están en ejecución en el servidor o bien que no están activos debido a que hayan sido parados.

5. Equipos Cisco

Para configurar el protocolo en los sistemas Cisco se deben seguir los siguientes pasos:

1. Accedemos al sistema por “ssh” o “telnet”.
2. Entramos en modo “administrador” realizando un “enable” o “en”.
3. Entramos en el modo “Configure terminal” mediante este comando o “conf t”.
4. Introducimos la siguiente línea “snmp-server community Miquel RO”.
5. Realizamos un “exit” para salir del modo “configure terminal”.
6. Por último, guardamos la configuración mediante un “write memory” o “wr mem” para que la configuración no se pierda en caso de reinicio del equipo.

```
CORE_6500_Torrejon>en
CORE_6500_Torrejon#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
CORE_6500_Torrejon(config)#snmp-server community Miquel RO
CORE_6500_Torrejon(config)#exit
CORE_6500_Torrejon#wr mem
```

Configuración completa en switch Cisco

2.7.2 Configuración de protocolo NRPE

Una vez definido en el capítulo anterior, veremos su configuración y funcionamiento en alguno de los diferentes casos usados.

Este protocolo únicamente se ha usado para sistemas UNIX, debido a su comodidad y fácil distribución de uno a otro, teniendo en cuenta que las configuraciones no varían entre sistemas, es decir, puede configurar un equipo y redistribuir los ficheros a todos los equipos necesarios.

Gráficamente se podría entender de la siguiente manera:

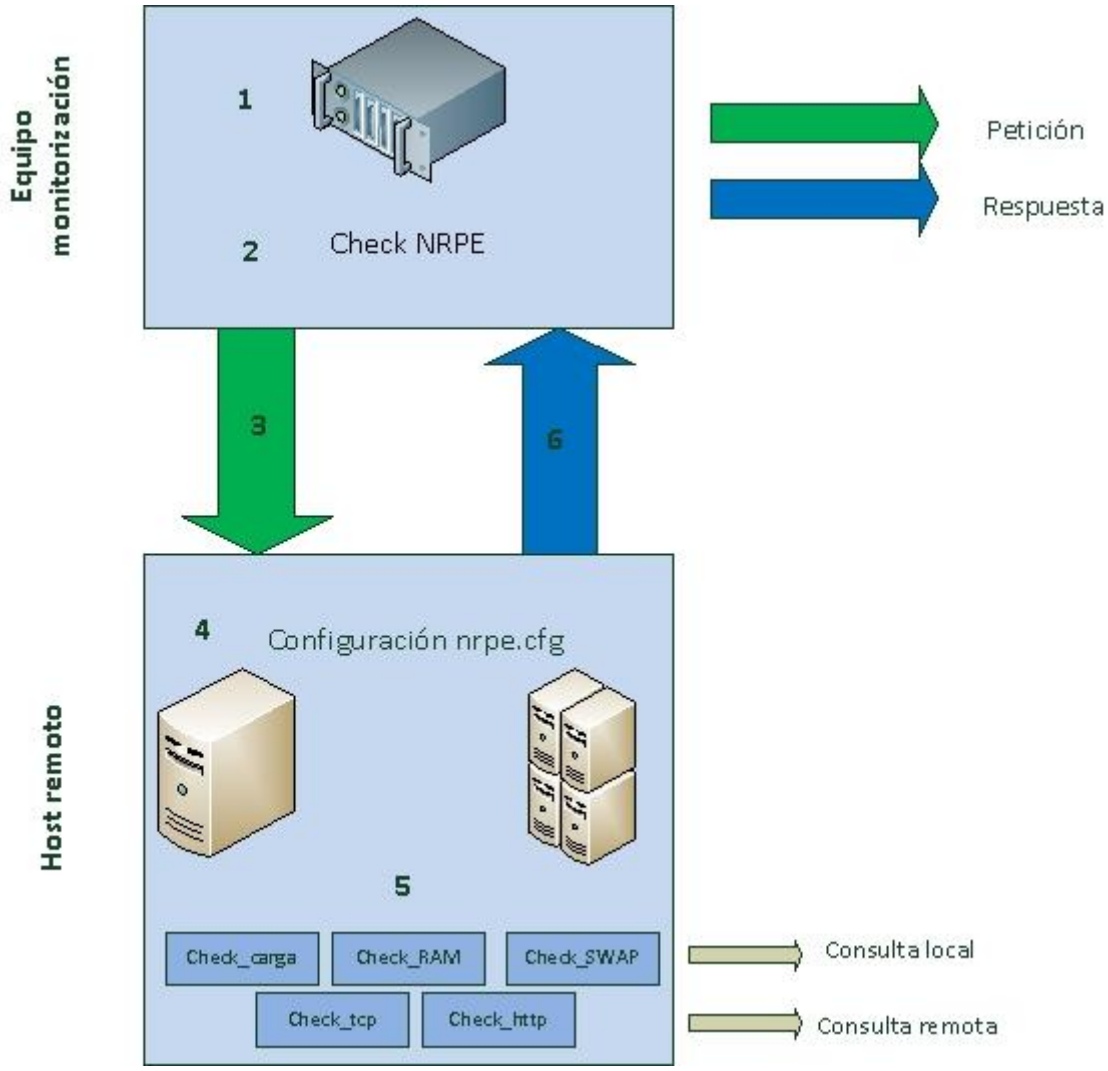


Diagrama funcionamiento NRPE

Cronológicamente, los pasos por los que pasa la petición son los siguientes:

1. El check llega al momento de ejecución en la cola de la aplicación (Nagios).

2. Ejecuta el check_nrpe, alojado en el equipo IBM, y dependiendo del tipo de petición le pasa unos parámetros u otros. Entre los parámetros, se le pasa un nombre único.

3. La petición va camino del host remoto.

4. Ataca al fichero de configuración “nrpe.cfg”.
   Lo primero que comprueba es si la IP/Nombre DNS de la cual proviene la petición está en este fichero en el apartado de “allowed host”. En caso negativo no aceptará la petición y no mostrará ningún dato, *en caso positivo…*

5. A partir del nombre que le hemos pasado como parámetro, miramos en el fichero de configuración y vemos que el nombre nos indica una ruta en la que se ejecuta un script que está también dentro del equipo remoto.
   Ejecutamos el script con los parámetros proporcionados.

6. Devolvemos la salida al equipo que ha realizado la petición.

Configuración en los diferentes equipos.

1. AIX

A continuación se detallan los pasos de configuración en los sistemas AIX.

1. Debemos instalar el paquete que corresponda a nuestra distribución y comprobar que el proceso esté corriendo. La instalación del paquete se realiza con un rpm –ivh "nombre_paquete.rpm". En esta versión de AIX no ha sido necesaria la instalación ya que venía en el sistema por defecto. Para comprobar si existe el proceso hacemos un "ps -ef".

```
[root@mar3prdgs]:/# ps -ef | grep nrpe
    root 26804460 26214560   0 13:55:39  pts/3  0:00 grep nrpe
  svcmon  9306394        1   0   May 07      -  3:15 /opt/nagios/bin/nrpe -n -c /opt/nagios/etc/nrpe.cfg -d
[root@mar3prdgs]:/#
```

PID del proceso NRPE funcionando

2. Con la configuración por defecto del paquete de nrpe tenemos que el fichero de configuración está en /opt/nagios/etc/nrpe.cfg.

```
[root@mar3prdgs]:/#
[root@mar3prdgs]:/# find / -name nrpe.cfg
/opt/nagios/etc/nrpe.cfg
find: 0652-023 Cannot open file /proc/5177578.
find: 0652-023 Cannot open file /proc/9240610.
^C
[root@mar3prdgs]:/# _
```

Ruta fichero de configuración

3. Accedemos al fichero de configuración y modificamos los campos requeridos:

   - *Log_facility=local7*: Nivel de logs por si queremos instalar un agente de syslog que recoja todos los eventos del equipo. Se verá en apartados posteriores ya que se ha implementado como mejoras del proyecto.
   - *pid_file=/var/run/nrpe.pid*: En este fichero se almacena el número de PID asociado al proceso nrpe.
   - *server_port=5666*: Puerto TCP por defecto en el que escucha el servicio.
   - *nrpe_user=svcmon*: usuario encargado de gestionar el servicio.
   - *nrpe_group=svcmon*: igualmente, pero para el grupo que gestiona el servicio.
   - *allowed_hosts=10.100.33.110,10.100.7.236,10.100.7.237*: En este parámetro es donde se indica qué equipos remotos están permitidos para realizar peticiones . En este caso, se indica la IP interna que tiene el equipo IBM (10.100.33.110) y otras dos que serán futuros equipos de backup clonados.

- *debug=0*: Se pone a 0 para que no envíe eventos y a 1 para que sí que los envíe, de momento está deshabilitado.
- *command_timeout=60*: Tiempo antes de hacer timeout de la petición.

4. En el mismo fichero de configuración debemos indicar el nombre con el cual el equipo origen realiza la petición y la ruta a la que apunta y en la cual está el script que realiza el cálculo solicitado.

   command[check_load]=/opt/nagios/libexec/check_load -w 20,19,18 -c 30,29,28
   command[check_SWAP]=/opt/nagios/libexec/check_swap -w 40% -c 30%
   command[check_RAM]=/opt/nagios/libexec/check_aix_mem.sh -c 99 -w 97 -t pctused

5. Ahora ya tenemos que el equipo origen 10.100.33.110, puede realizar la petición a través de los nombres siguientes: check_load, check_SWAP y check_RAM.

6. Una vez recibida la petición debemos saber la ruta a la que enviarla para que realice los scripts, para ello tenemos, en el orden de las anteriores:

   opt/nagios/libexec/check_load
   opt/nagios/libexec/check_swap
   opt/nagios/libexec/check_aix_mem.sh

7. Buscamos o creamos la ruta y movemos estos scripts, que serán globales para todos los equipos, a /opt/nagios/libexec/. Para ello se usa el usuario y grupo svcmon, configurado en el punto número 3.

```
[root@mar3prdgs]:/opt/nagios/libexec# ls -la check_load
-rwxr-xr-x    1 svcmon   svcmon       342290 Sep 03 2009  check_load
[root@mar3prdgs]:/opt/nagios/libexec#
```

Scripts locales en servidor

8. Cada script tendrá su peculiaridad, pero normalmente se le tendrá que pasar unos argumentos que procesará internamente, los cuales corresponderán a umbrales, entre otros datos.

   -w 20,19,18 -c 30,29,28
   -w 40% -c 30%
   -c 99 -w 97 -t pctused

   En estos casos vemos:

   -w (umbral al cual nos enviará un evento warning).
   -c (umbral al cual nos enviará un evento crítico).

9. Cada vez que se realiza un cambio en el fichero de configuración se debe reiniciar el servicio para que surja efecto el cambio. Para ello tenemos que matar el proceso ya creado:

```
[root@mar3prdgs]:/# ps -ef | grep nrpe
    root 47317074 23658972   0 23:53:31  pts/0  0:00 grep nrpe
  svcmon  9306394        1   0   May 07      -  3:19 /opt/nagios/bin/nrpe -n -c /opt/nagios/etc/nrpe.cfg -d
[root@mar3prdgs]:/# kill -9 9306394
```

PID del proceso NRPE

y volver a generar el PID del mismo para que funcione:

```
[root@mar3prdgs]:/# /opt/nagios/bin/nrpe -n -c /opt/nagios/etc/nrpe.cfg -d
```

Ejecución como daemon del servicio

10. El script se ejecuta y envía la respuesta al servidor origen, el cual escucha por el puerto del nrpe el envío que se le hace.
    Tal y como se puede ver en la imagen, el puerto está escuchando peticiones continuamente por el puerto 5666 de nrpe.

```
[root@mar3prdgs]:/# netstat -an | grep 5666
tcp4       0      0  *.5666                 *.*                    LISTEN
[root@mar3prdgs]:/# _
```

Puerto activo en el sistema

2. SUN SOLARIS

Para sistemas Solaris se deben seguir los siguientes pasos para instalar y configurar el protocolo NRPE:

1. Instalar el paquete NRPE en el servidor.

    - Bajarse el paquete de la distribución en /opt.
    - Descomprimirlo en /opt para que cree el directorio /opt/nagios.
    - Crear usuario y grupo svcmon si aún no existe, con “useradd” y “groupadd” con “Uid 101” y “Gid 101”.

```
[:root]/opt> find / -name nrpe.cfg
/opt/nagios/nrpe/etc/nrpe.cfg
```

Directorio NRPE en Solaris

2. Crear el script de arranque:
    - **# ls -la /etc/rc2.d/S77nrpe**
      -rwxr--r-- 1 root root 137 May 2 2011 /etc/rc2.d/S77nrpe
    - **# cat /etc/rc2.d/S77nrpe** e introducir la siguiente línea:
      # Inicio demonio NRPE monitorización Nagios...
      /usr/bin/su svcmon -c "opt/nagios/nrpe/bin/nrpe -n -c /opt/nagios/nrpe/etc/nrpe.cfg -d"

3. Editar el fichero de configuración para permitir las IPs remotas y poner las rutas de los scripts, este paso es idéntico que en el equipo AIX visto anteriormente.

```
[:root]/opt> vi /opt/nagios/nrpe/etc/nrpe.cfg
"/opt/nagios/nrpe/etc/nrpe.cfg" 269 lines, 11604 characters
#############################################################################
# Sample NRPE Config File
# Written by: Ethan Galstad (nagios@nagios.org)
#
# Last Modified: 11-23-2007
#
# NOTES:
# This is a sample configuration file for the NRPE daemon.  It needs to be
# located on the remote host that is running the NRPE daemon, not the host
# from which the check_nrpe client is being executed.
#############################################################################

allowed_hosts=10.100.33.110,10.100.7.237,10.100.7.236


command[check_swap]=/opt/nagios/nrpe/libexec/check_swap -w 10% -c 5%
command[check_load]=/opt/nagios/nrpe/libexec/check_load -w 20,19,18 -c 30,29,28
command[check_mem]=/usr/bin/perl /opt/nagios/nrpe/libexec/check_local_mem.pl -c 10 -w 20
```

4. Al haber realizado cambios en los ficheros de configuración debemos matar el proceso para volverlo a crear y que tengan efecto los cambios.

```
[madwh:root]/> ps -ef | grep nrpe
    root  2114   578   0 23:02:02 pts/44      0:00 grep nrpe
  svcmon  8750  3394   0   Apr 18 ?          5:02 /opt/nagios/nrpe/bin/nrpe -n -c /opt/nagios/nrpe/etc/nrpe.cfg -d
[madwh:root]/>
[madwh:root]/> kill -9 8750
```

Kill del proceso

5. Arrancamos el servicio con el siguiente comando:
"/etc/rc2.d/S77nrpe start" o bien **"/usr/bin/su svcmon -c "opt/nagios/nrpe/bin/nrpe -n -c /opt/nagios/nrpe/etc/nrpe.cfg -d**"

6. Comprobamos el funcionamiento externamente, si aparece la versión es que la configuración se ha realizado correctamente, en caso contrario aparecerá "Connection refused by host". Con este paso también comprobamos que localmente el puerto del NRPE está abierto a peticiones.

```
[root@app-miquell plugins]# ./check_nrpe -n -H madq
NRPE v2.12
```

Comprobación servicio NRPE

2.8 Ejemplos de scripts realizados y cómo se ejecutan internamente

En este apartado se analizarán algunos de los scripts realizados, los cuales solicitan datos a los equipos de comunicaciones y devuelven su estado general o el de sus componentes.

Se debe tener en cuenta que se han recopilado un total de 140 scripts aproximadamente, de los cuales algunos se han modificado para atender a nuestras necesidades, otros se han dejado intactos y algunos se han realizado aprovechando sentencias existentes, como en los casos siguientes.

Se detallará el código de los scripts comentando los apartados más importantes y se realizará una prueba de su funcionamiento.

Se dispone de lo siguiente:

```
[root@app-miquel plugins]# ls -la check_CISCO*
-rwxr-xr-x 1 root root 3956 Jun 11 16:47 check_CISCO_CPU.pl
-rwxr-xr-x 1 root root 3795 Jun 11 17:23 check_CISCO_Fuentes.pl
-rwxr-xr-x 1 root root 3428 Jun 11 18:04 check_CISCO_interface.pl
-rwxr-xr-x 1 root root 4026 Jun 11 18:14 check_CISCO_Memoria.pl
-rwxr-xr-x 1 root root 3152 Jun 11 18:26 check_CISCO_Modulos.pl
-rwxr-xr-x 1 root root 3089 Jun 11 18:30 check_CISCO_Ventilador.pl
[root@app-miquel plugins]#
```

Perl scripts Cisco

En todos los casos se encuentran programados en perl y buscan en las MIBs de equipos Cisco los datos que su nombre indica.

Para ello se ha buscado las MIBs y OIDS de Cisco necesarias para ver el estado en cada una de las consultas realizadas.

Para empezar, ¿qué es una OID?

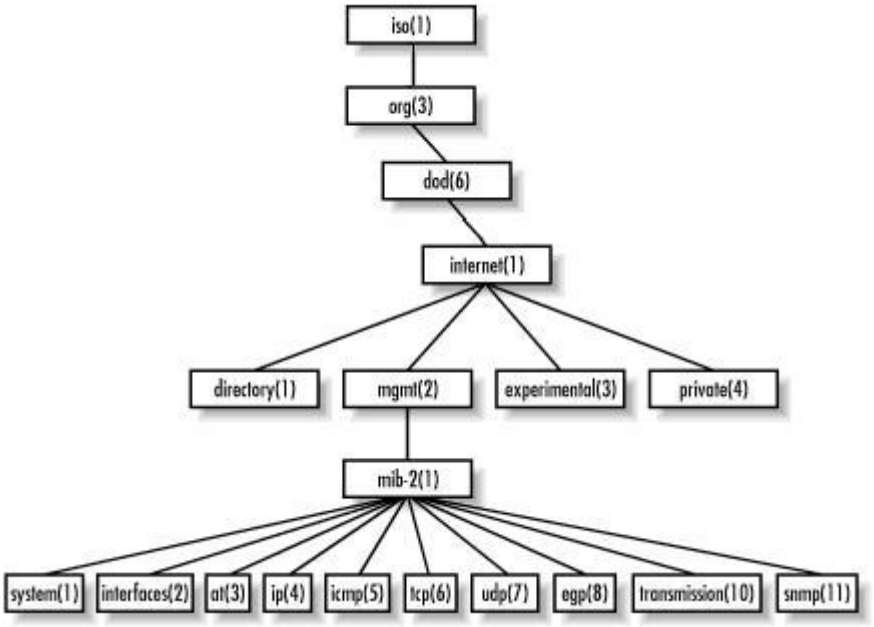
Un OID (Object IDentifier) o Identificador de Objeto en una secuencia de números que se asignan jerárquicamente y que permite identificar objetos inequívocamente en una red.

Las OIDs se utilizan para identificar cualquier objeto, entendiendo como objeto cualquier “cosa”, en gran medida es utilizado para temas tecnológicos, como se ha usado en este proyecto a través del protocolo SNMP.

Se puede entender una OID como un seguido de número enteros que marcan el camino en un árbol decreciente, cada número es una rama de este árbol que indica que has elegido uno de sus caminos para conseguir tu objetivo.

Esta información jerárquica mostrada en forma de árbol se conoce también como MIB (Management Information Base o Base de Información Gestionada) y se puede mostrar con la forma x.x.xx.x.xxx.x.

En la siguiente imagen se muestra un ejemplo de la forma de árbol:



Con este último ejemplo vemos un modelo bastante escueto, pero ya efectivo para saber qué camino coger para monitorizar ciertos aspectos de algunos equipos.

Podemos saber que, por ejemplo:

- Si queremos monitorizar interfaces: 1.3.6.1.2.1.2.x
- Si queremos monitorizar IP: 1.3.6.1.2.1.4.x
- Si queremos aspectos del sistema: 1.3.6.1.2.1.1.x

Y así sucesivamente. Hay que tener en cuenta que en el ejemplo, no se muestra la estructura entera, debido a que algún camino implicaría una raíz de direcciones muy grande para mostrarse.

En este proyecto se han tratado todo tipo de equipos y de servicios obteniendo sus MIBs o OIDs, prestaremos especial atención a los equipos Cisco.

Empezamos, de la lista mostrada anteriormente, en orden:

2.8.1 Check_CISCO_CPU.pl

El código de las OIDs para las diferentes muestras de tiempo se ha obtenido de:

http://www.cisco.com/en/US/tech/tk648/tk362/technologies_tech_note09186a0080094a94.shtml

```
#!/usr/bin/env perl

use strict;
use Net::SNMP;
my $stat;
my $resultado;
my $perf;

# CPU Load OIDs para 5segundos - 1 minuto y 5 minutos
my $S_load_5s = ".1.3.6.1.4.1.9.2.1.56.0";
my $S_load_1m = ".1.3.6.1.4.1.9.2.1.57.0";
my $S_load_5m = ".1.3.6.1.4.1.9.2.1.58.0";

# Iniciamos la sesión SNMP a través de lo que le pasamos por parámetros y lo transformamos en
# snmpwalk IP -v2c -c community
sub _create_session {
      my ($server, $comm) = @_;
      my $version = 1;
      my ($sess, $err) = Net::SNMP->session( -hostname => $server, -version => $version, -community
=> $comm);
      if (!defined($sess)) {
            print "El $server no tiene activo en protocolo SNMP o esta apagado\n";
            exit(1);
      }
      return $sess;
}


#Caso del número incorrecto de parámetros
if($#ARGV != 7) {
      print "Faltan o sobran parametros de entrada\n";
      print "Error en la sintaxis del comando ! Se debe usar:\n";
      print "./check_CISCO_CPU.pl -H [IP Equipo/Nombre DNS] -C [Community] -w [warning] -c
[critical]\n";
      exit(2);
}

#Creamos las variables necesarias
my $switch;
```

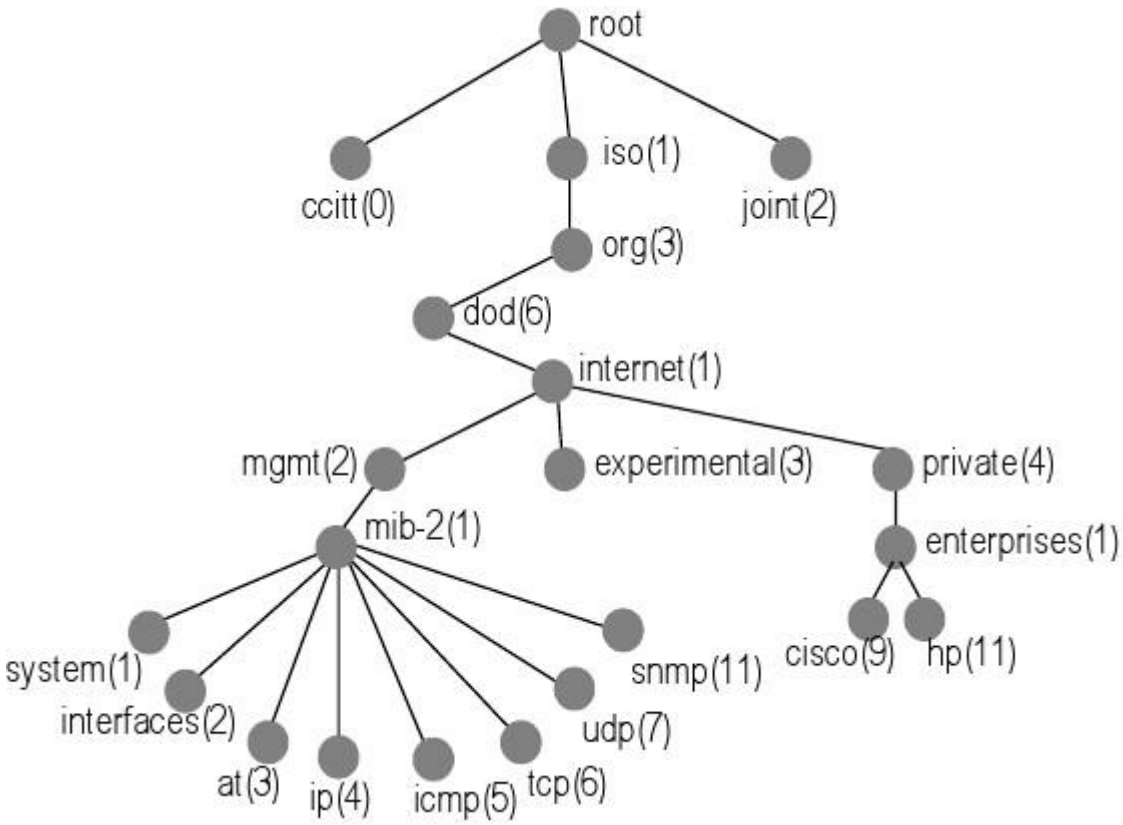
```
my $community;
my $warn = 0;
my $crit = 0;
my $int;
#Recogemos los argumentos introducidos...
while(@ARGV) {
    #Introducimos temporalmente cada argumento introducido en la variable temp
    my $temp = shift(@ARGV);
    #Guardamos el Nombre de host/IP en la variable switch
    if("$temp" eq '-H') {
        $switch = shift(@ARGV);
    #Guardamos la community que viene precedida de la C
    } elsif("$temp" eq '-C') {
        $community = shift(@ARGV);
    #Guardamos el umbral de warning
    } elsif("$temp" eq '-w') {
        $warn = shift(@ARGV);
    #Guardamos el umbral de critical
    } elsif("$temp" eq '-c') {
        $crit = shift(@ARGV);
    #Comprobamos si el numero de argumentos es correcto pero los argumento no son del tipo solicitado
    } else {
        print "Los argumentos introducidos no son validos\n";
        print "./check_CISCO_CPU.pl -H [IP Equipo/Nombre DNS] -C [Community] -w [warning] -c [critical]\n";
        exit(3);
    }
}

# Control errores umbrales
if($warn > $crit) {
    print "Warning debe ser menor que Critical: Valor incorrecto $warn > $crit\n";
    exit(4);
}

# Creamos la sesion SNMP
our $snmp_session = _create_session($switch,$community);

#Datos para la carga de CPU, cogemos el dato que nos muestra la OID seleccionada

    #Recogemos carga en 5 segundos
    my $R_load_5s = $snmp_session->get_request(-varbindlist => [$S_load_5s]);
    my $carga5s = "$R_load_5s->{$S_load_5s}";

    #Recogemos carga en 1 minuto
    my $R_load_1m = $snmp_session->get_request(-varbindlist => [$S_load_1m]);
    my $carga1m = "$R_load_1m->{$S_load_1m}";

    #Recogemos carga en 5 minutos
    my $R_load_5m = $snmp_session->get_request(-varbindlist => [$S_load_5m]);
    my $carga5m = "$R_load_5m->{$S_load_5m}";
```

```
    #Control de estados en nagios, lo comprobaremos todo a partir del dato más pequeña que es el más exacto sin medias
    #Si el dato obtenido está por debajo del umbral de warning almacenamos el estado Ok de nagios = 0
    if($carga5s <= $warn) {
        $stat = 0;
        $resultado = "Cpu OK - Carga actual: $carga5s% Media 1 minuto: $carga1m% Media 5 minutos: $carga5m%";
    #Si el dato obtenido esta entre los dos umbrales podemos considerar que tiene que salir alerta y ponerse en amarillo = 1
    }  elsif($carga5s > $warn and $carga5s < $crit) {
        $stat = 1;
        $resultado = "Cpu Warning - Carga actual: $carga5s% Media 1 minuto: $carga1m% Media 5 minutos: $carga5m%";
    #Si el dato obtenido es mayor que nuestro umbral critico lo pintaremos de rojo y estado = 2
    } elsif($carga5s >= $crit) {
        $stat = 2;
        $resultado = "Cpu Citical - Carga actual: $carga5s% Media 1 minuto: $carga1m% Media 5 minutos: $carga5m%";
    #Error si no se cumplen las anteriores
    } else {
        $stat = 2;;
        exit(5);
    }

    #Mostramos los datos en los 3 intervalos de tiempo para los gráficos, se realiza pase lo que pase
    $perf = "cpu_5s=$carga5s\percent;$warn;$crit cpu_1m=$carga1m\percent cpu_5m=$carga5m\percent";



#Datos para las gráficas
print "$resultado | $perf\n";
#Enviamos el valor a nagios
exit($stat);
```

Antes de pasar a otros scripts, se puede observar la OID utilizada y marcada en rojo en el código anterior.

Para la CPU de un sistema Cisco hemos usado .1.3.6.1.4.1.9.2.1.56.0 para carga en 5 segundos, para carga en 1 minuto .1.3.6.1.4.1.9.2.1.57.0 y para carga en 5 minutos .1.3.6.1.4.1.9.2.1.58.0.

Una vez visto esto nos fijamos en el siguiente árbol.



Para llegar a Cisco debemos pasar por 1.3.6.1.4.1.9, exactamente lo mismo usado en el script.

La segunda parte usada, 2.1.58.0 corresponde únicamente a servicios dentro de la estructura de Cisco que se ha encontrado en la misma página del fabricante.

2.8.2 Check_CISCO_ Fuentes.pl

La información del código de errores y de OIDs se ha obtenido de la siguiente URL:

http://tools.cisco.com/Support/SNMP/do/BrowseOID.do?local=en&translate=Translate&objectInput=ciscoEnvMonSupplyState

```
#!/usr/bin/env perl

use strict;
use Net::SNMP;
my $stat;
my $resultado;
my $perf;

#OIDs de las fuentes de alimentación, nombre y estado
my $S_ps = ".1.3.6.1.4.1.9.9.13.1.5.1";
my $S_ps_name = ".1.3.6.1.4.1.9.9.13.1.5.1.2";
my $S_ps_stat = ".1.3.6.1.4.1.9.9.13.1.5.1.3";

#Posibles estados de una fuente de alimentación según la página de Cisco
my %phy_dev_status = (
	1 => 'normal',
	2 => 'warning',
	3 => 'critical',
	4 => 'shutdown',
	5 => 'notPresent',
	6 => 'notFunctioning',
);

#Creamos la sesion SNMP
sub _create_session {
	my ($server, $comm) = @_;
	my $version = 1;
	my ($sess, $err) = Net::SNMP->session( -hostname => $server, -version => $version, -community
=> $comm);
	if (!defined($sess)) {
		print "El $server no tiene activo en protocolo SNMP o esta apagado\n";
		exit(1);
	}
	return $sess;
}


#Control del número de argumentos
if($#ARGV != 7) {
	print "Numero de argumentos incorrecto\n";
	print "Error en la sintaxis del comando ! Se debe usar:\n";
	print "./check_CISCO_Fuentes.pl -H [IP Equipo] -C [Community] -w [warning] -c [critical]\n";
```

```
	exit(2);
}
#Creamos las variables
my $switch;
my $community;
my $check_type;
my $warn = 0;
my $crit = 0;
my $int;

#Recogemos los parámetros introducidos
while(@ARGV) {
	my $temp = shift(@ARGV);
	#Guardamos el parámetro con la IP/nombre de host en switch
	if("$temp" eq '-H') {
		$switch = shift(@ARGV);
	#Guardamos la comunidad que viene después del parámetro C
	} elsif("$temp" eq '-C') {
		$community = shift(@ARGV);
	#Guardamos la variable de warning
	} elsif("$temp" eq '-w') {
		$warn = shift(@ARGV);
	#Guardamos la variable de critical
	} elsif("$temp" eq '-c') {
		$crit = shift(@ARGV);
	#Error para ninguno de los casos anteriores
	} else {
		print "No has introducido los parametros correctamente\n";
		print "./check_CISCO_Fuentes.pl -H [Nombre host/IP] -C [Comunidad] -w [Warning] -c
[Critical]\n";
		exit(3);
	}
}

# Control errores en los umbrales
if($warn > $crit) {
	print "El valor de warning no uede ser superior al de critico\n";
	exit(4);
}

# Establecemos la sesión SNMP
our $snmp_session = _create_session($switch,$community);

#Calculamos el total de las fuentes y errores
	my $R_tbl = $snmp_session->get_table($S_ps_name);
	my $total_err = 0;
	my $err_msg;
	my $sum = 0;
	foreach my $oid ( keys %$R_tbl) {

		#Calculamos el total de fuentes  erróneas o no
		$sum = $sum + 1;
```

```
        my $name = "$$R_tbl{$oid}";
        my $id = "$oid";
        $id =~ s/$S_ps_name\.//;
        my $R_stat = $snmp_session->get_request(-varbindlist => ["$S_ps_stat.$id"]);
        my $stat = $R_stat->{"$S_ps_stat.$id"};
        #Miramos encada pasada si el estado es diferente de OK y vamos sumando
        if($stat != 1) {
            $total_err = $total_err + 1;
            $err_msg = "$err_msg $name -> $phy_dev_status{$stat}";
        }
    }


    #Control de errores por si el equipo está parado
    if ($sum == 0){
        #Se debe mostrar alerta crítica y mostrar la salida pero sin continuar el código
        $stat = 2;
        $sum="No existen fuentes funcionando en este equipo";
        print"$sum";
        exit($stat);
    }

    #Control de errores para nagios
    #Si el número de errores es inferior al umbral de warning se muestra en verde y estado = 0
    if($total_err <= $warn) {
        $stat = 0;
        $resultado = "Fuentes OK - $sum fuentes funcionando";
    #Si el numero obtenido de fuentes erróneas es superior a umbral de w, lo mostramos en amarillo y estado = 1
    }  elsif($total_err > $warn and $total_err < $crit) {
        $stat = 1;
        $resultado = "Fuentes Warning - $sum/$total_err fuentes funcionando";
    #Si el numero obtenido es superior al umbral c, se muestra en rojo en nagios debido a que estado = 2
    } elsif($total_err >= $crit) {
        $stat = 2;
        $resultado = "Fuentes Criticas - $sum/$total_err fuentes funcionando";
    }

    #Mostramos dos gráficos, uno con fuentes totales y otra con el total de errores
    $perf = "fuentes_totales=$sum fuentes_erroneas=$total_err";


#Mostramos en nagios el estado y enviamos el gráfico
print "$resultado | $perf\n";

#Actualizamos el estado en nagios para el color y estado
exit($stat);
```

2.8.3 Check_CISCO_Interface.pl

Los posibles códigos de la boca y las OIDs correspondiente se han obtenido de:

http://tools.cisco.com/Support/SNMP/do/BrowseOID.do?translate=true&objectInput=1.3.6.1.2.1.2.2.1.8

```
#!/usr/bin/env perl

use strict;
use Net::SNMP;
my $stat;
my $salida;
my $performance_data;


# OIDs estado bocas del equipo
my $S_int_entry = ".1.3.6.1.2.1.2.2.1";
my $S_int_desc = "$S_int_entry.2";
my $S_int_operstatus = "$S_int_entry.8";

# Posibles códigos de la boca
my %int_status_index = (
	1 => 'up',
	2 => 'down',
	3 => 'testing',
	4 => 'unknown',
	5 => 'notPresent',
	6 => 'lowerLayerDown',
);

#Creamos la sesión SNMP
sub _create_session {
	my ($server, $comm) = @_;
	my $version = 2;
	my ($sess, $err) = Net::SNMP->session( -hostname => $server, -version => $version, -community
=> $comm);
	if (!defined($sess)) {
		print "El $server no tiene activo en protocolo SNMP o esta apagado\n";
		exit(1);
	}
	return $sess;
}

#Control de errores por el número de argumentos introducido
if($#ARGV !=4) {
	print "Error en la sintaxis, faltan o sobran argumentos.\n";
	print "Debes usar:\n";
```

```
    print "./check_Cisco_Interface.pl -H [IP Host/Nombre DNS] -C [Community] [Nombre Interface]\n";
    exit(2);

}

my $switch;
my $community;
my $check_type;
my $int;

#Recogemos los parametros introducidos
while(@ARGV) {
    #Variable temporal para ir asignando a variable final
    my $temp = shift(@ARGV);
    #Guardamos el nombre o IP del equipo en la variable switch
    if("$temp" eq '-H') {
        $switch = shift(@ARGV);
    #Guardamos la comunidad SNMP en la variable community
    } elsif("$temp" eq '-C') {
        $community = shift(@ARGV);
        $int = shift(@ARGV);
    #Mostramos el error en caso de no haber introducido los parámetros -H y -C
    } else {
        print "Error en la sintaxis, argumentos mal definidos.\n";
        print "Debes usar:\n";
        print "./check_Cisco_Interface.pl -H [IP Host/Nombre DNS] -C [Community] [Nombre Interface]\n";
        exit(3);
    }
}

# Creamos la sesión SNMP
our $snmp_session = _create_session($switch,$community);

#Estado físico de las bocas
    my $R_tbl = $snmp_session->get_table($S_int_desc);
    my $is_int_exists = 0;
    foreach my $oid ( keys %$R_tbl) {
        #Recogemos el nombre que saca la OID del equipo cisco
        my $name = "$$R_tbl{$oid}";
        #Comparamos el nombre de la OID con la interface pasada por parámetro
        if($name eq $int) {
            $is_int_exists++;
            my $id = "$oid";
            $id =~ s/$S_int_desc\.//;
            my $R_stat = $snmp_session->get_request(-varbindlist => ["$S_int_operstatus.$id"]);
            my $int_stat = $R_stat->{"$S_int_operstatus.$id"};

            #Si la interface buscada esta UP mostramos estado OK y en verde
            if ($int_stat == 1){
                $stat = 0;
```

```
                $salida = "Estado OK: $int  -> $int_status_index{$int_stat}";
                $performance_data = "1";
            #Si la interface está en DOWN mostramos estado crítico y en rojo
            } elsif ($int_stat == 2){
                $stat = 2;
                $salida = "Estado Critico: $int  -> $int_status_index{$int_stat}";
                $performance_data = "0";
            #Si la interface no está en Up o Down mostramos el código de error para su revisión
            } else {
                $stat = 1;
                $salida = "Estado Warning: $int -> $int_status_index{$int_stat}";
            }
            last;
        }

    }
    #No existe en la tabla "sh ip interface brief" o no tiene configurado el SNMP
    if($is_int_exists == 0) {
        $stat = 3;
        $salida = "UNKNOWN: la $int no existe o SNMP no activo";
        $performance_data = "0";
    }


#Datos para graficar 0 inactivo / 1 activo solo en caso de que se requiera
#Por normal general no se grafican estados binarios
print "$salida | $performance_data\n";
exit($stat);
```

2.8.4 Check_Cisco_Memoria.pl

Los códigos para memoria usada y libre se han obtenido de la siguiente URL

http://www.cisco.com/en/US/tech/tk648/tk362/technologies_tech_note09186a0080094a95.shtml

```
#!/usr/bin/perl

use strict;
use Net::SNMP;
my $cisco;
my $community;
my $warning;
my $critical;
my $stat;
my $salida;

# OID de cisco para memoria usada y libre

my $usada = ".1.3.6.1.4.1.9.9.48.1.1.1.5.1";
my $libre = ".1.3.6.1.4.1.9.9.48.1.1.1.6.1";

#Control de errores para un número de argumentos erróneo
if($#ARGV < 7) {
	print "Falta un argumento por introducir\n";
	print "Sintaxis correcta: ./check_CISCO_Memoria.pl -H [IP] -C [Comunidad SNMP] -w [Warning] -c [Critical]\n";
	exit(1);
}elsif($#ARGV > 7){
	print "Sobra un argumento\n";
	print "Sintaxis correcta: ./check_CISCO_Memoria.pl -H [IP] -C [Comunidad SNMP] -w [Warning] -c [Critical]\n";
	exit(2);
}



#Creamos la sesion SNMP v2
sub _create_session {
	my ($server, $comm) = @_;
	my $version = 2;
	my ($sess, $err) = Net::SNMP->session( -hostname => $server, -version => $version, -community => $comm);
	if (!defined($sess)) {
		print "Instalar SNMP o revisar configuracion\n";
		exit(3);
	}
	return $sess;
}
```

```
#Vamos recogiendo los parámetros introducidos
while(@ARGV) {
	my $argumento = shift(@ARGV);
	#Lo que venga después de H lo guardamos en la variable cisco, será la IP/Nombre de host
	if("$argumento" eq '-H') {
		$cisco = shift(@ARGV);
	#Guardamos en la variable community lo que venga a continuación de C
	} elsif("$argumento" eq '-C') {
		$community = shift(@ARGV);
	#Recogemos el umbral warning que será lo que venga después de w
	} elsif("$argumento" eq '-w') {
		$warning = shift(@ARGV);
	#Recogemos el umbral de critical que será lo que venga después de c
	} elsif("$argumento" eq '-c') {
		$critical = shift(@ARGV);
	#Por último revisamos que los parámetros pasados sean los correctos, solo se puede c, w, C y H
	} elsif(("$argumento" ne '-c') || ("$argumento" ne '-w') || ("$argumento" ne '-C') ||
("$argumento" ne '-H')){
		print "Numero de argumentos correctos pero mal definidos\n";
		exit(4);
	}
}

#Validamos los umbrales establecidos
if($warning >= $critical) {
	print "Umbrales mal definidos, warning no puedo ser mayor que critical\n";
	exit(5);
}

#Creamos la sesión contra los parámetros fijados en -H y -C
our $snmp_session = _create_session($cisco,$community);

#Calculamos la salida
	my $tp_usada;

	#Recogemos la memoria usada
	my $temp_usada = $snmp_session->get_request(-varbindlist => [$usada]);
	my $total_usada = "$temp_usada->{$usada}";

	#Recogemos la memoria libre
	my $temp_libre = $snmp_session->get_request(-varbindlist => [$libre]);
	my $total_libre = "$temp_libre->{$libre}";

	#Calculamos la memoria total sumando usada y disponible
	my $total_memoria = $total_libre + $total_usada;

	#Pasamos de bytes a KBytes y a MBytes
	$total_usada = int($total_usada / 1024 / 1024);
	$total_libre = int($total_libre / 1024 / 1024);
	$total_memoria = int($total_memoria / 1024 / 1024);
```

```
    #Datos para los gráficos
    my $performance_data="memoria_total=$total_memoria\MB memoria_usada=$total_usada\MB";

    #Se calcula el tp (%)
    $tp_usada = int($total_usada / $total_memoria * 100);

    #Nagios interpreta:
    #Estado OK (Verde): stat = 0
    #Estado WARNING (Amarillo): stat = 1
    #Estado CRITICAL (Rojo): stat = 2

    #Si el % es menor que el umbral establecido mostramos en nagios estado OK con color verde
    if($tp_usada < $warning) {
        $stat = 0;
        $salida = "Estado OK -> Memoria: $tp_usada%";
    #Si el % está entre el w y el c mostramos stado warning y lo ponemos en amarillo
    }  elsif($tp_usada >= $warning and $tp_usada < $critical) {
        $stat = 1;
        $salida = "Estado Warning -> Memoria: $tp_usada %";
    #Si el % es superior al umbral establecido de critical lo ponemos en rojo
    } elsif($tp_usada >= $critical) {
        $stat = 2;
        $salida = "Estado Critico -> Memoria: $tp_usada %";
    } else {
    #Ultima opción para comprobar posibles errores de código
        print "Error que nunca deberia dar";
    }

#Mostramos la salida y enviamos los datos a las gráficas
print "$salida | $performance_data\n";

#Cada vez que ejecutamos el script devolvemos el estado que interpreta nagios
exit($stat);
```

2.8.5 Check_CISCO_Modulos.pl

La OID de los módulos y los posibles códigos de estado de éstos se han obtenido de la siguiente URL:

http://tools.cisco.com/Support/SNMP/do/BrowseOID.do?local=en&translate=Translate&objectInput=cefcModuleOperStatus

```
#!/usr/bin/env perl

use strict;
use Net::SNMP;
my $stat;
my $salida;
my $performance_data;

# OID Módulos
my $OID_Modulos = ".1.3.6.1.4.1.9.9.117.1.2.1.1.2";

#Posibles códigos de  los módulos
my %estado_modulos = (
	1 => 'unknown',
	2 => 'ok',
	3 => 'disabled',
	4 => 'okButDiagFailed',
	5 => 'boot',
	6 => 'selfTest',
	7 => 'failed',
	8 => 'missing',
	9 => 'mismatchWithParent',
	10 => 'mismatchConfig',
	11 => 'diagFailed',
	12 => 'dormant',
	13 => 'outOfServiceAdmin',
	14 => 'outOfServiceEnvTemp',
	15 => 'poweredDown',
	16 => 'poweredUp',
	17 => 'powerDenied',
	18 => 'powerCycled',
	19 => 'okButPowerOverWarning',
	20 => 'okButPowerOverCritical',
	21 => 'syncInProgress',
);

#Creamos la sesión SNMP
sub _create_session {
	my ($server, $comm) = @_;
	my $version = 1;
	my ($sess, $err) = Net::SNMP->session( -hostname => $server, -version => $version, -community => $comm);
```

```
    if (!defined($sess)) {
        print "El $server no tiene activo en protocolo SNMP o esta apagado\n";
        exit(1);
    }
    return $sess;
}

my $cisco;
my $community;
my $check_type;

#Recorremos los argumentos introducidos que serán solo 2
while(@ARGV) {
    #Creación de variable temporal para ir almacenando los parámetros
    my $temp = shift(@ARGV);
    #Almacenamos en la variable cisco el valor que siga al -H
    if("$temp" eq '-H') {
        $cisco = shift(@ARGV);
    #Almacenamos en la variable community el valor que siga a -C
    } elsif("$temp" eq '-C') {
        $community = shift(@ARGV);
    #En caso de no enviar por parámetros -H o -C mostramos error y salimos
    } else {
        print "Faltan o sobran parametros\n";
        print "./check_CISCO_Modulos.pl -H [IP Cisco] -C [community]\n";
        exit(3);
    }
}


# Creamos la sesión SNMP
our $snmp_session = _create_session($cisco,$community);

# Sumamos i contamos erróneos y correctos
    my $R_tbl = $snmp_session->get_table($OID_Modulos);
    my $errores = 0;
    my $err_msg;
    my $total = 0;
    #Cada vez que entramos en la sentencia es que hay un modulo
    foreach my $oid ( keys %$R_tbl) {
        $total = $total + 1;
        my $codigo_modulo = "$$R_tbl{$oid}";
        my $id = "$oid";
        $id =~ s/$OID_Modulos\.//;
        #El único estado correcto es el 2, si es diferente lo consideramos módulo erróneo
        if($codigo_modulo != 2) {
            $errores = $errores + 1;
            $err_msg = "$err_msg $id -> $estado_modulos{$codigo_modulo}";
        }
    }
```

```
    #Hay switches/routers que no tienen módulos integrados, para eso se implementa este caso
    #No se debe considerar como un error, por eso no se asigna stat
    if($total == 0) {
        print "Este equipo no tiene modulos\n";
        exit(4);
    }


    #Controlamos la salida dependiendo del contador de módulos erróneos
    if($errores == 0) {
        $stat = 0;
        $salida = "Modulos OK: Los $total modulos funcionan correctamente";
    #Si hay algún módulo erróneo se considera crítico y se muestra el motivo de error, en este código la alerta warning no existe
    } elsif($errores >= 1) {
        $stat = 2;
        $salida = "Modulos Criticos: Existen $errores/$total modules sin funcionar. Error: $err_msg";
    }

    #Datos para los gráficos
    $performance_data = "total=$total erroneas=$errores";


#Mostramos los datos en el nagios y enviamos datos a los gráficos
print "$salida | $performance_data\n";

#Enviamos el estado a nagios
exit($stat);
```

2.8.6 Check_CISCO_Ventilador.pl

El estado del ventilador y sus posibles estados se han obtenido del siguiente enlace:

http://tools.cisco.com/Support/SNMP/do/BrowseOID.do?local=en&translate=Translate&objectInput=ciscoEnvMonFanState

```
#!/usr/bin/env perl

use strict;
use Net::SNMP;
my $stat;
my $salida;
my $performance_data;

#OIDs Ventilador Cisco
my $mib_ventilador = ".1.3.6.1.4.1.9.9.13.1.4.1";
my $mib_VNombre = ".1.3.6.1.4.1.9.9.13.1.4.1.2";
my $mib_VStatus = ".1.3.6.1.4.1.9.9.13.1.4.1.3";


# Creamos la sesión SNMP
sub _create_session {
	my ($server, $comm) = @_;
	my $version = 1;
	my ($sess, $err) = Net::SNMP->session( -hostname => $server, -version => $version, -community
=> $comm);
	if (!defined($sess)) {
		print "El $server no tiene activo en protocolo SNMP o esta apagado\n";
		exit(1);
	}
	return $sess;
}

#Controlamos en número de argumentos entrantes
if($#ARGV != 3) {
	print "Faltan o sobran argumentos, debe escribir\n";
	print "$./check_CISCO_Ventilador.pl -H [IP CISCO] -C [community]\n";
	exit(2);
}

#Creamos las variables
my $cisco;
my $community;

#Recogemos los argumentos introducidos
#En este caso solo necesitamos el host y la comunidad snmp
while(@ARGV) {
	my $temp = shift(@ARGV);
```

```
    #Almacenamos el host/IP
    if("$temp" eq '-H') {
        $cisco = shift(@ARGV);
    #Almacenamos la comunidad snmp
    } elsif("$temp" eq '-C') {
        $community = shift(@ARGV);
    #Si no se introducen sólo esos dos datos mostramos error y la sintaxis correcta
    } else {
        print "Error de sintaxis !\n";
        print "$./check_CISCO_Ventilador.pl -H [IP CISCO] -C [community]\n";
        exit(3);
    }
}

#Realizamos la consulta SNMP
our $snmp_session = _create_session($cisco,$community);


#Contamos los ventiladores erróneos y OK
my $R_tbl = $snmp_session->get_table($mib_VNombre);
my $total_err = 0;
my $total = 0;
foreach my $oid ( keys %$R_tbl) {
    #En cada pasada miramos el total de ventiladores
    $total = $total + 1;
    my $name = "$$R_tbl{$oid}";
    my $id = "$oid";
    $id =~ s/$mib_VNombre\.//;
    #Obtenemos el ID del estado, que será 1 si es correcto o <> de 1 si es incorrecto
    my $R_stat = $snmp_session->get_request(-varbindlist => ["$mib_VStatus.$id"]);
    my $stat = $R_stat->{"$mib_VStatus.$id"};
    #Contamos el total que muestra que el estado es diferente a OK
    if($stat != 1) {
        $total_err = $total_err + 1;
    }
}

#Control de errores por si el equipo está parado
if ($total == 0){
    #Se debe mostrar alerta crítica y mostrar la salida pero sin continuar el código
    $stat=2;
    $salida="No existen ventiladores funcionando en este equipo";
    print"$salida";
    exit($stat);
}

#Códigos de errores y salida
#Si no existen errores ponemos el estado a OK y en verde debido a que estado = 0
if($total_err == 0) {
    $stat = 0;
    $salida = "Estado OK: Los $total ventiladores funcionan";
```

```
#Si solo quedase un ventilador en marcha la alerta seria crítica y nagios mostraría color rojo debido a que estado = 2
} elsif($total_err == ($total-1)) {
	$stat = 2;
	$salida = "Estado Critical: Hay ($total-1) de $total ventiladores parados, solo funciona bien 1!";
#Si falla 1 ventilador o funcionan dos o más ventiladores la alerta es menos pero existe, con lo que estado = 1 y se muestra en amarillo
} else {
	$stat = 1;
	$salida = "Estado Warning: Hay ($total_err) de $total ventiladores parados";
}

#Mostramos los datos para los gráficos
$performance_data = "ventiladores_totales=$total ventiladores_erroneos=$total_err";

#Pintamos la salida en nagios y lanzamos lo datos del grafico
print "$salida | $performance_data\n";

#Enviamos el código de estado a nagios
exit($stat);
```

2.9 Análisis de las aplicaciones secundarias instaladas para interface de usuario

Se analizarán funcionalmente las siguientes aplicaciones expuestas en el capítulo anterior.

2.9.1 NDOUtils

Como ya se ha comentado anteriormente, esta utilidad nos permite integrar mysql en la estructura de nagios. Pasamos de configurar todo en texto plano a tener todo enlazado en un BD de mysql.

Ahora se verá cómo se ha construido la estructura interna de la BBDD a partir de esta aplicación, hay que tener en cuenta que la construcción interna de la ER (Entidad Relación) la genera automáticamente la aplicación.

Primero entramos en el mysql, el cual nos ha creado una BD con una serie de tablas y tablas intermedias creando dicha ER.

```
[root@app-miquell ~]# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 442838
Server version: 5.5.17 MySQL Community Server (GPL)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

mysql de aplicación

Las tablas que ha creado son las siguientes:

```
mysql> show tables;
+-------------------------------------------+
| Tables_in_nagiosql                        |
+-------------------------------------------+
| tbl_command                               |
| tbl_contact                               |
| tbl_contactgroup                          |
| tbl_contacttemplate                       |
| tbl_domain                                |
| tbl_group                                 |
| tbl_host                                  |
| tbl_hostdependency                        |
| tbl_hostescalation                        |
| tbl_hostextinfo                           |
| tbl_hostgroup                             |
| tbl_hosttemplate                          |
| tbl_info                                  |
| tbl_lnkContactToCommandHost               |
| tbl_lnkContactToCommandService            |
| tbl_lnkContactToContactgroup              |
| tbl_lnkContactToContacttemplate           |
| tbl_lnkContactToVariabledefinition        |
| tbl_lnkContactgroupToContact              |
| tbl_lnkContactgroupToContactgroup         |
| tbl_lnkContacttemplateToCommandHost       |
| tbl_lnkContacttemplateToCommandService    |
| tbl_lnkContacttemplateToContactgroup      |
| tbl_lnkContacttemplateToContacttemplate   |
| tbl_lnkContacttemplateToVariabledefinition |
| tbl_lnkGroupToUser                        |
| tbl_lnkHostToContact                      |
| tbl_lnkHostToContactgroup                 |
| tbl_lnkHostToHost                         |
| tbl_lnkHostToHostgroup                    |
| tbl_lnkHostToHosttemplate                 |
| tbl_lnkHostToVariabledefinition           |
| tbl_lnkHostdependencyToHost_DH            |
| tbl_lnkHostdependencyToHost_H             |
| tbl_lnkHostdependencyToHostgroup_DH       |
| tbl_lnkHostdependencyToHostgroup_H        |
| tbl_lnkHostescalationToContact            |
| tbl_lnkHostescalationToContactgroup       |
| tbl_lnkHostescalationToHost               |
```

| tbl_lnkHostescalationToHostgroup |
| tbl_lnkHostgroupToHost |
| tbl_lnkHostgroupToHostgroup |
| tbl_lnkHosttemplateToContact |
| tbl_lnkHosttemplateToContactgroup |
| tbl_lnkHosttemplateToHost |
| tbl_lnkHosttemplateToHostgroup |
| tbl_lnkHosttemplateToHosttemplate |
| tbl_lnkHosttemplateToVariabledefinition |
| tbl_lnkServiceToContact |
| tbl_lnkServiceToContactgroup |
| tbl_lnkServiceToHost |
| tbl_lnkServiceToHostgroup |
| tbl_lnkServiceToServicegroup |
| tbl_lnkServiceToServicetemplate |
| tbl_lnkServiceToVariabledefinition |
| tbl_lnkServicedependencyToHost_DH |
| tbl_lnkServicedependencyToHost_H |
| tbl_lnkServicedependencyToHostgroup_DH |
| tbl_lnkServicedependencyToHostgroup_H |
| tbl_lnkServicedependencyToService_DS |
| tbl_lnkServicedependencyToService_S |
| tbl_lnkServiceescalationToContact |
| tbl_lnkServiceescalationToContactgroup |
| tbl_lnkServiceescalationToHost |
| tbl_lnkServiceescalationToHostgroup |
| tbl_lnkServiceescalationToService |
| tbl_lnkServicegroupToService |
| tbl_lnkServicegroupToServicegroup |
| tbl_lnkServicetemplateToContact |
| tbl_lnkServicetemplateToContactgroup |
| tbl_lnkServicetemplateToHost |
| tbl_lnkServicetemplateToHostgroup |
| tbl_lnkServicetemplateToServicegroup |
| tbl_lnkServicetemplateToServicetemplate |
| tbl_lnkServicetemplateToVariabledefinition |
| tbl_lnkTimeperiodToTimeperiod |
| tbl_lnkTimeperiodToTimeperiodUse |
| tbl_logbook |
| tbl_mainmenu |
| tbl_relationinformation |
| tbl_service |
| tbl_servicedependency |
| tbl_serviceescalation |
| tbl_serviceextinfo |

```
| tbl_servicegroup                        |
| tbl_servicetemplate                     |
| tbl_settings                            |
| tbl_submenu                             |
| tbl_tablestatus                         |
| tbl_timedefinition                      |
| tbl_timeperiod                          |
| tbl_user                                |
| tbl_variabledefinition                  |
+-----------------------------------------+
93 rows in set (0.10 sec)
```

La estructura ha quedado definida por un total de 93 tablas que la propia aplicación ha creado automáticamente y se ha encargado de hacer las relaciones internas.

Para ver un ejemplo de una de ellas internamente expongo la tabla de equipos, que coincide con la tabla “hosts.cfg” vista en apartados anteriores.

```
mysql> select host_name, alias, display_name, address from tbl_host where host_name='tin01';
+-----------+-------+--------------+---------------+
| host_name | alias | display_name | address       |
+-----------+-------+--------------+---------------+
| tin01     | tin01 |              | 10.255.255.50 |
+-----------+-------+--------------+---------------+
1 row in set (0.00 sec)
```

Se han seleccionado pocos campos debido a que la tabla contiene muchos para ponerlos en este documento.

2.9.2 NagiosQL

Con este software se hará mucho más fácil la inserción de datos en el sistema de monitorización.

NagiosQL funciona sobre un servidor apache y funciona como un vínculo entre la interface gráfica y las tablas internas mysql o los ficheros de texto plano que tenga nagios, dependiendo de la manera de instalación, en este caso contra las tablas de mysql.

Para acceder a él debemos entrar a **http://IP_NAT** ó **http://IP/nagiosql**.

Una vez dentro podremos ver la estructura del menú:

Opciones menú principal

En cada menú se rellenarán los datos de host, servicio etc, al aplicar aquí los cambios se actualizará la base de datos mysql. La propia aplicación realizará un "update" de cada tabla asociada.

A través del menú anterior tenemos un control total sobre la aplicación. En la siguiente imagen se puede ver un resumen general de servidores y servicios totales puestos a través de la aplicación:

Grupo	Activo	Inactivo
Hosts	398	30
Servicios	898	2
Grupo de Host	42	3
Grupos de servicios	2	0
Plantillas de Host	6	7
Plantillas de servicios	4	1

Su uso y configuración se verán en el capítulo de "Manual", en el cual se verá más a fondo estos aspectos de la aplicación.

2.9.3 PNP4Nagios

Con esta aplicación conseguiremos un histórico de cada equipo o servicio que se configure.

1. Cada script que ejecuta la aplicación internamente muestra una salida (que será la que muestra nagios), en este caso muestra "Cpu OK – Carga actual 3% Media 1 minuto 4% Media 5 minutos 4%".

2. Internamente el script muestra otra salida, que no es visible para el usuario, se verá ejecutando manualmente el código.

```
[root@app-miquel1 plugins]# ./check_CISCO_CPU.pl
Faltan o sobran parametros de entrada
Error en la sintaxis del comando ! Se debe usar:
./check_CISCO_CPU.pl -H [IP Equipo] -C [Community] -w [warning] -c [critical]
[root@app-miquel1 plugins]# ./check_CISCO_CPU.pl -H 10.100.7.3 -C Miquel -w 90 -c 95
Cpu OK - Carga actual: 4% Media 1 minuto: 4% Media 5 minutos: 4% | cpu_5s=4percent;90;95 cpu_1m=4percent cpu_5m=4percent
```

3. En el punto número 2 se observa que hay más datos de los que muestra nagios en el punto 1.

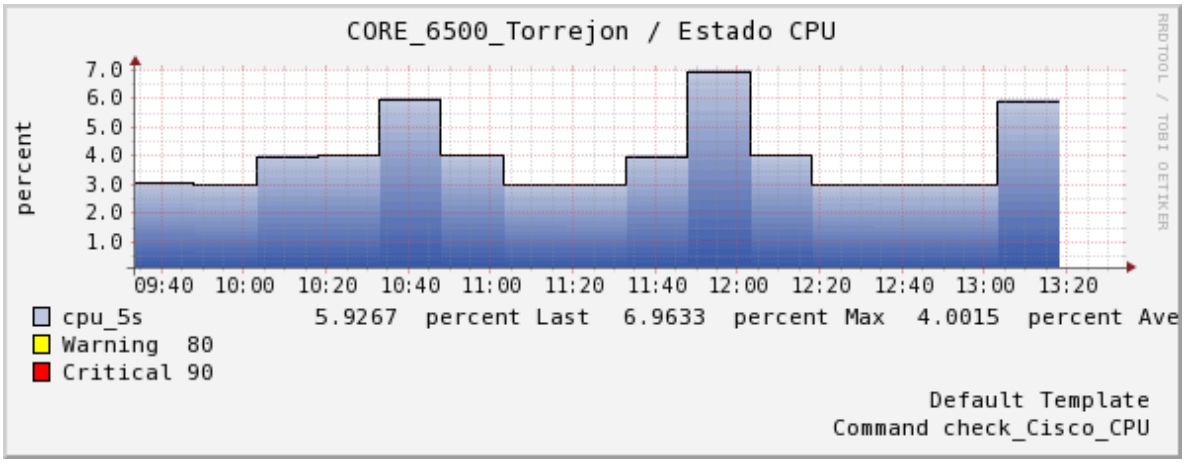
4. El script internamente ejecuta la salida que muestra nagios seguido de una pipe(|), esto lo interpreta el pnp4nagios como que es el dato que debe pintar en el gráfico. Concretamente en este caso vemos que se muestran después de la pipe los tres datos cogidos a intervalos de tiempo diferente (y que cada uno de ellos viene dado por un dato recogido de su MIB como ya se ha visto anteriormente en el código perl del script). Cada vez que se muestra un espacio en blanco, la aplicación interpreta que se debe de crear un nuevo gráfico para representarlo. Su intervalo de actualización en los gráficos vendrá marcado por el tiempo en que se ejecuta cada script, es decir, si se ejecuta el script cada 15 minutos, el dato en el gráfico será actualizado en el mismo intervalo de tiempo.

5. El fichero de configuración del gráfico y de datos lo crea en la ruta de la aplicación con el mismo nombre del equipo y del servicio asociado.

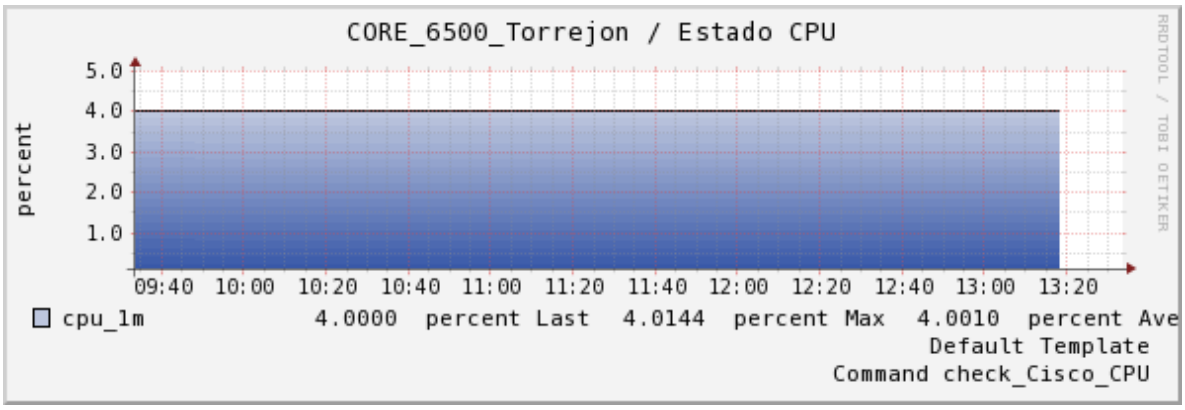
```
[root@app-miquel1 plugins]# locate CORE_6500_Torrejon/Estado_CPU
/usr/local/pnp4nagios/var/perfdata/CORE_6500_Torrejon/Estado_CPU.rrd
/usr/local/pnp4nagios/var/perfdata/CORE_6500_Torrejon/Estado_CPU.xml
```

   El *.rrd* equivale a los datos recogidos por el script, que son escritos en este fichero y minutos más tarde recogidos por la aplicación que lo interpreta en el gráfico.
   El *.xml* es formado por la configuración del gráfico, como son los colores, máximos etc.

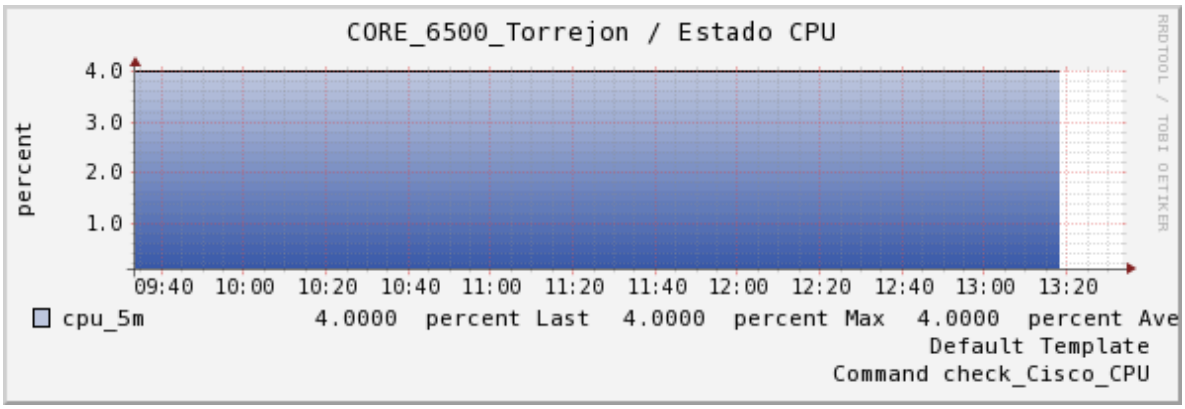
6. Finalmente se recogen los datos del .rrd anterior y muestra lo siguiente.



Actualización cada 15 minutos de estado de CPU



Media de estado de CPU 1 min en las últimas 4 horas



Media de estado de CPU 5 min en las últimas 4 horas

Como se observa, se muestran los tres gráficos que aparecían después de la pipe. En el primer gráfico también se puede diferenciar el intervalo de actualización, que en este caso en particular es de 15 minutos.

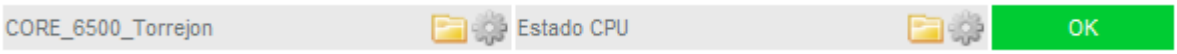
2.9.4 DocuWiki

Con la aplicación DocuWiki conseguimos tener una documentación en texto independientemente asociada a cada servicio o servidor.

La finalidad de instalar este software es poder tener métodos de actuación o notas importantes en cada sistema para poder solventar incidencia que se den en la red.

1. La ruta de instalación de la aplicación es /var/www/html/dokuwiki/data/pages/procedimientos.

2. Cada vez que se añada una nota a uno de los servicios se creará una carpeta nueva que contendrá el texto en un fichero. Se realiza la prueba insertando un comentario haciendo click sobre la imagen de carpeta del servicio CPU



Símbolo de DocuWiki en host y servidor

3. Introducimos un texto: “Prueba de funcionamiento para un servicio”.

4. Una vez guardado comprobamos que ha creado una nueva carpeta para el equipo en el cual se ha puesto la nota.

```
[root@app-miquel1 procedimientos]# pwd
/var/www/html/dokuwiki/data/pages/procedimientos
[root@app-miquel1 procedimientos]# ls -la core_6500_torrejon
total 12
drwxr-xr-x  2 apache apache 4096 May 28 14:00 .
drwxr-xr-x 11 apache apache 4096 May 28 14:00 ..
-rw-r--r--  1 apache apache   42 May 28 14:00 estado_cpu.txt
```

Ruta instalación DocuWiki

5. Entrando en el fichero se puede ver el texto del cual se está alimentando la aplicación.

```
[root@app-miquel1 core_6500_torrejon]# more estado_cpu.txt
Prueba de funcionamiento para un servicio.
```

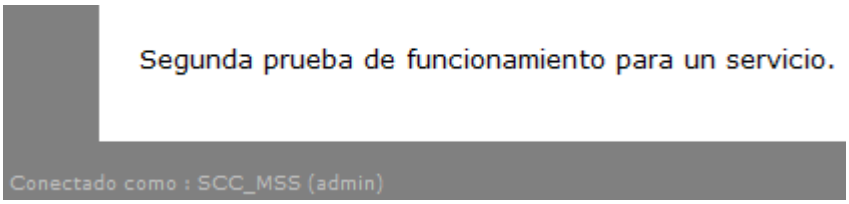
Fichero de texto plano de prueba

6. Ahora se realiza la prueba inversa, haciendo un cambio en el fichero para que lo muestre la aplicación web: “2ª prueba de funcionamiento para un servicio”.

```
[root@app-miquel1 core_6500_torrejon]# vi estado_cpu.txt
Segunda prueba de funcionamiento para un servicio.
~
```

Y guardamos los cambios con un “wq!”.

7. Finalmente entramos en la aplicación y vemos que efectivamente los datos son recogidos.



2.10 Mejoras introducidas

Una vez introducidas todas las aplicaciones en el sistema y estudiadas sus funcionalidades se ha decidido introducir una mejora al sistema para solucionar carencias.

Uno de los problemas con los cuales nos habíamos encontrado era la dificultad de mostrar todo dato relevante a través de la aplicación en los casos en los cuales no se disponía de información suficiente de la tarea que realizaba el equipo.

Debido a que es posible que algunos aspectos hardware como software se escapen de la monitorización, se debe estudiar una manera de recibir toda información del sistema en logs.

Para ello se ha configurado en el servidor central IBM un servidor de “syslog”.

¿Qué aporta un servidor de syslog?

La introducción de este daemon en el sistema nos permite tener un servidor a la espera de recibir logs de cualquier tipo de sistema, que previamente haya sido configurado, y que enviará eventos que ocurran en su funcionamiento, dependiendo del nivel de log que le configuremos.

El software escuchará por un puerto tcp o ud o ambos y escribirá en un directorio lo que el equipo afectado le envíe.

Configuración del servidor syslog

Para instalar el software se ha instalado el paquete correspondiente a la versión de Red Hat, concretamente el paquete de “syslog-ng” que para futuras mejoras nos aporta mayores opciones con respecto al syslog normal.

Una vez instalado se habrá creado en la siguiente ruta el fichero de configuración:

```
[root@app-miquel syslog-ng]# pwd
/etc/syslog-ng
[root@app-miquel syslog-ng]# ls -la syslog-ng.conf
-rw-r--r-- 1 root root 3132 Jun  7 11:39 syslog-ng.conf
```

Ruta instalación syslog-ng

Debemos introducir y modificar los siguientes parámetros en el fichero de configuración.

- Use_dns (yes): Se debe cambiar a "yes" en caso de que no esté configurado ya por defecto. Esta opción permitirá ver los logs que lleguen de equipos remotos con el nombre que resuelva en DNS, en caso de estar en "no" se recibirán con la IP y no resultará cómoda la lectura y comprensión.
- Se deben añadir las siguientes líneas en el fichero:

```
source  s_tcp { tcp(ip(0.0.0.0)  port(514) max_connections(300)); };
source s_udp { udp(ip(0.0.0.0)  port(514)                      ); };
```

Configuración para recepción de logs por tcp y udp

Con las dos líneas anteriores permitimos la recepción de logs mediante conexiones TCP y UDP por el puerto 514.

Comprobamos su funcionamiento, previamente realizando un reinicio del servicio para que los cambios tengan efecto "/etc/init.d/syslog-ng restart", de la siguiente manera:

```
[root@app-miquel syslog-ng]# netstat -an | grep 514
tcp        0      0 0.0.0.0:514                 0.0.0.0:*                   LISTEN
tcp        0      0 10.100.7.236:514            10.100.46.11:13193          ESTABLISHED
tcp        0      0 10.100.7.236:514            10.100.46.11:8379           ESTABLISHED
tcp        0      0 10.100.7.236:514            10.100.46.11:3602           ESTABLISHED
tcp        0      0 10.100.7.236:514            10.100.46.11:62390          ESTABLISHED
tcp        0      0 10.100.7.236:514            10.100.46.11:35943          ESTABLISHED
```

Puerto tcp 514 recibiendo peticiones

Vemos que las conexiones por el puerto 514 se están estableciendo y que además el origen de ellas pertenece a equipos que se han configurado para su uso.

Configuración de envío de logs

Además de instalar y configurar el software en el servidor central para que reciba los logs debemos configurar los equipos que queremos que envíen notificaciones, en una primera implantación de la mejora se han configurado los switches.

Para ello realizamos los siguientes pasos:

1. Entrar por ssh o telnet al equipo que se desee configurar.
2. Entrar en modo administrador a través del comando “enable”.
3. Entrar en modo configurador mediante el comando “configure terminal”.
4. Insertar “logging on”.
5. Insertamos la IP destino de los logs mediante “logging 10.100.33.110”, teniendo en cuenta que la IP destino es el servidor IBM.
6. Insertar el comando “Logging trap informational” para un nivel de logs aceptable y que muestre todo problema registrado y descarte eventos prescindibles.
7. Realizar un “write memory” para que guarde la configuración.

Comprobaciones

Para comprobar un posible caso práctico tenemos el siguiente ejemplo:

1. Se ha configurado en Nagios un switch con 10 interfaces monitorizadas.
2. Por cualquier motivo se han configurado más bocas del equipo y no se han añadido a la monitorización.
3. Algunas de estas bocas, configuradas posteriormente, han generado problemas y Nagios no lo ha podido registrar.

Es en este momento cuando el syslog-ng tendrá la utilidad deseada. Podremos ir al directorio que ha creado y buscar el log del sistema afectado.

```
[root@app-miquel remotos]# pwd
/var/log/remotos
```

Ruta ficheros de log

La carpeta creada de cada equipo que se configure se mostrará en la ruta anterior y contendrá varios ficheros, cada uno con sus datos correspondientes. También se puede observar el funcionamiento de la opción de dns:

```
[root@app-miquel remotos]# pwd
/var/log/remotos
[root@app-miquel remotos]# ls LAN_CORE1
6212  6213  servidor
```

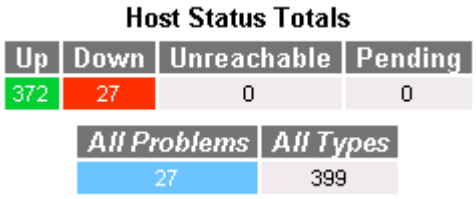
Carpetas que contienen logs del switch

3. Validación

En este capítulo se validará todo el funcionamiento de la aplicación con ejemplos que recojan los aspectos más característicos de ella. Para ello se verá un ejemplo completo de algunas de las peticiones que se ejecutan contra algunos servicios y equipos a través de la interface gráfica para que sea más fácil su comprensión.

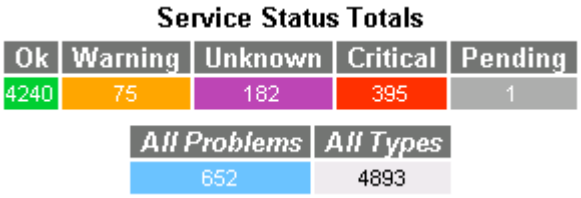
También se comprobará la veracidad de los datos mostrados por el sistema de monitorización comparándolos con el propio sistema físico del cual se extraen los datos.

Una vez finalizada la implantación del servicio se han llegado a un total de 399 equipos de las tecnologías comentadas en capítulos anteriores.



Total host

A esos 399 equipos se les ha asignado un total de 4893 servicios, entre ellos particiones, file systems, servicios, procesos, interfaces, etc.



Total servicios

A continuación se muestran los ejemplos tanto de la aplicación como de la misma consulta realizada desde el equipo, para poder comparar si los datos mostrados por la aplicación son correctos conforme los que muestra el equipo afectado.

4. Caso 1: Comprobación equipo Cisco

El siguiente ejemplo muestra el funcionamiento de un switch cisco con los diferentes servicios que se han implementado para su monitorización:

LAN_COMS	Estado CPU	OK	Cpu OK - Carga actual: 5% Media 1 minuto: 5% Media 5 minutos: 5%
	Estado Memoria	OK	Estado OK -> Memoria: 28%
	Estado fuentes alimentacion	OK	Fuentes OK - 1 fuentes funcionando
	Estado ventiladores	OK	Estado OK: Los 1 ventiladores funcionan
	GigabitEthernet 1/0/21	OK	Estado OK: GigabitEthernet1/0/21 -> up
	GigabitEthernet 1/0/22	OK	Estado OK: GigabitEthernet1/0/22 -> up
	GigabitEthernet 1/0/23	OK	Estado OK: GigabitEthernet1/0/23 -> up
	GigabitEthernet 1/0/24	OK	Estado OK: GigabitEthernet1/0/24 -> up
	Ping	OK	PING OK - Packet loss = 0%, RTA = 17.78 ms
	PortChannel1	OK	Estado OK: Port-channel1 -> up
	PortChannel2	OK	Estado OK: Port-channel2 -> up
	Uptime	OK	Uptime: 306 days, 12:07:29.14

Estado equipo Cisco de muestra en aplicación

3.1.1 Estado CPU

Como se puede observar en las siguientes capturas, los datos sacados tanto del equipo físicamente, de la aplicación como por consola realizando el script son prácticamente iguales.

Hay que tener en cuenta que están sacados con algunos segundos de retardo entre captura y captura:

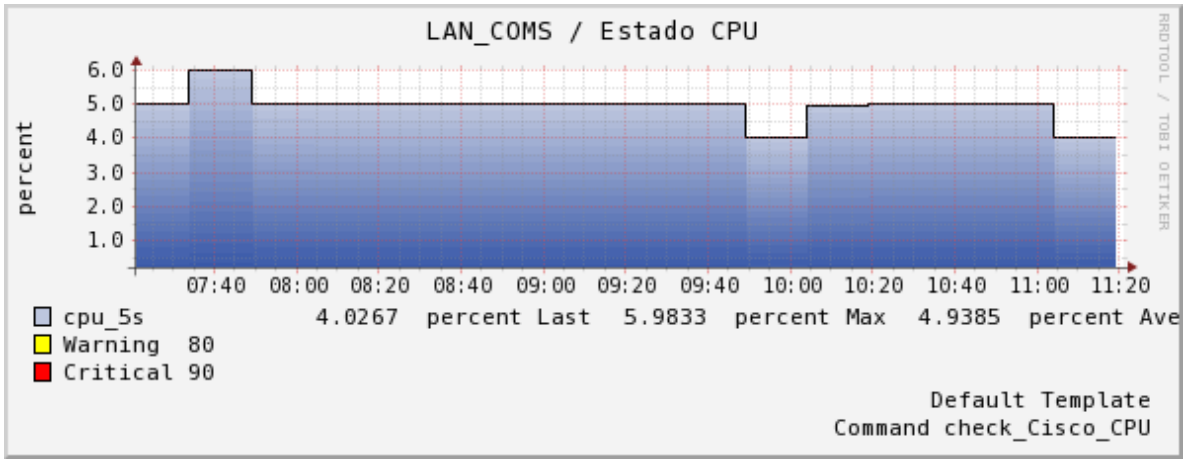
```
[root@app-miquel1 plugins]# ./check_CISCO_CPU.pl -H 10.100.7.243 -C Miquel -w 80 -c 90
Cpu OK - Carga actual: 4% Media 1 minuto: 5% Media 5 minutos: 5% | cpu_5s=4percent;80;90 cpu_1m=5percent cpu_5m=5percent
```

Captura realizada mediante script

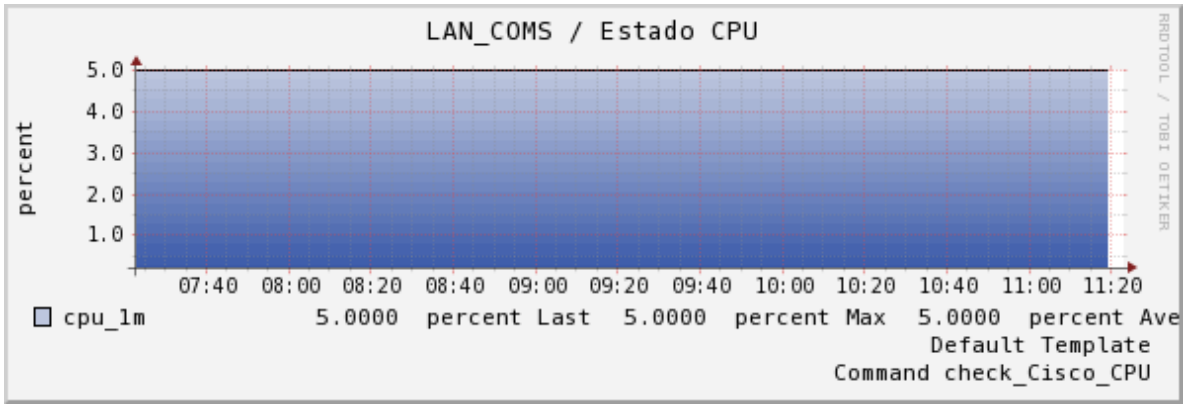
```
LAN_COMS#sh processes cpu
CPU utilization for five seconds: 6%/0%; one minute: 5%; five minutes: 5%
 PID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min TTY Process
   1          76      1249         60  0.00%  0.00%  0.00%   0 Chunk Manager
   2        3895  22441051          0  0.00%  0.00%  0.00%   0 Load Meter
   3        1903      3370        564  0.00%  0.00%  0.00%   0 SpanTree Helper
```

Consulta realizada desde el equipo físico

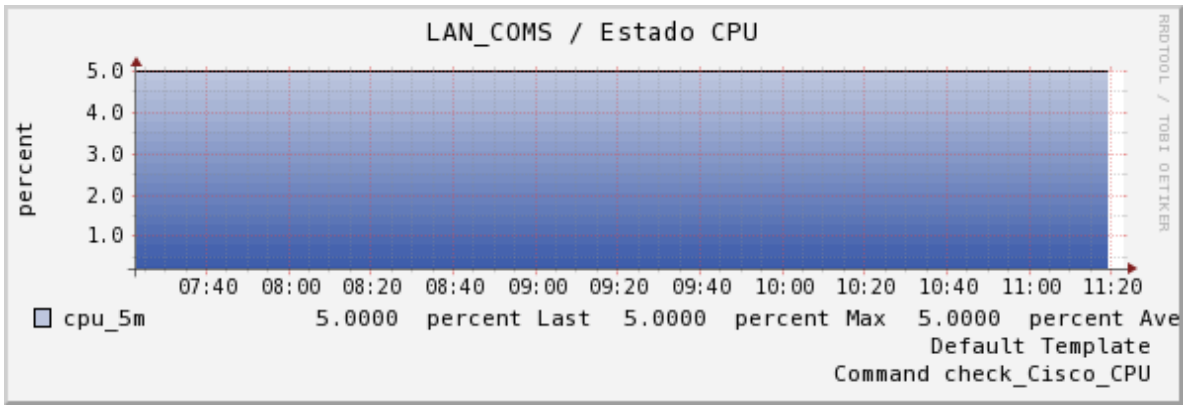
Además, se puede ver que en los gráficos se recogen los mismos datos, tenemos los datos en orden para 5 segundos, 1 minuto y 5 minutos en las últimas 4 horas.



Uso de CPU a tiempo real



Uso de CPU con media de 1 minuto



Uso de CPU con media de 5 minutos

3.1.2 Estado memoria

Las siguientes capturas muestran el estado de la memoria del equipo:

Como se puede observar, el equipo realizando la consulta por SNMP muestra que dispone de 87MB totales de memoria de los cuales tiene 25MB en uso y por lo tanto 62MB libres.

```
[root@app-miquel1 plugins]# ./check_CISCO_Memoria.pl -H 10.100.7.243 -C Miquel -w 80 -c 90
Estado OK -> Memoria: 28% | memoria_total=87MB memoria_usada=25MB
```

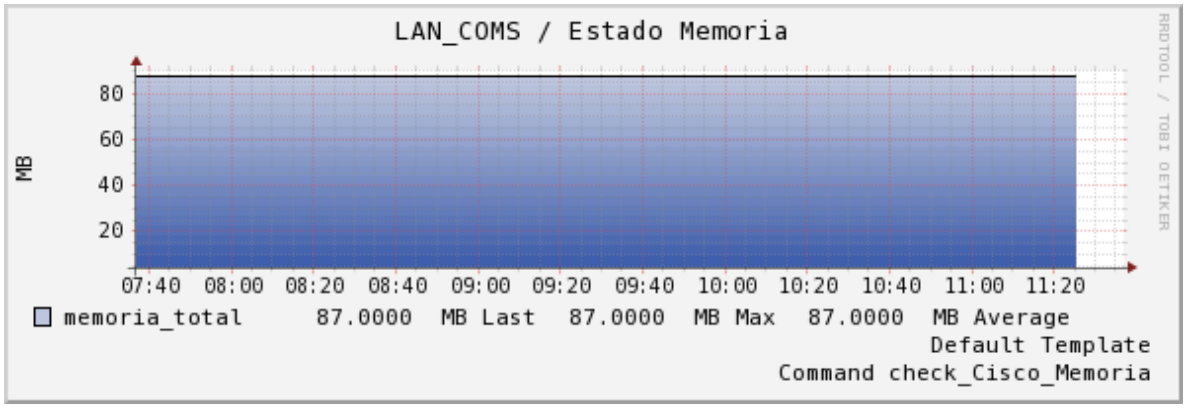
Captura realizada mediante script

Realizando la consulta mediante un "show memory" vemos los datos que nos muestra el dispositivo directamente y que coincide con lo obtenido anteriormente, previamente hay que hacer el factor de conversión de B a MB para que se asemejen.

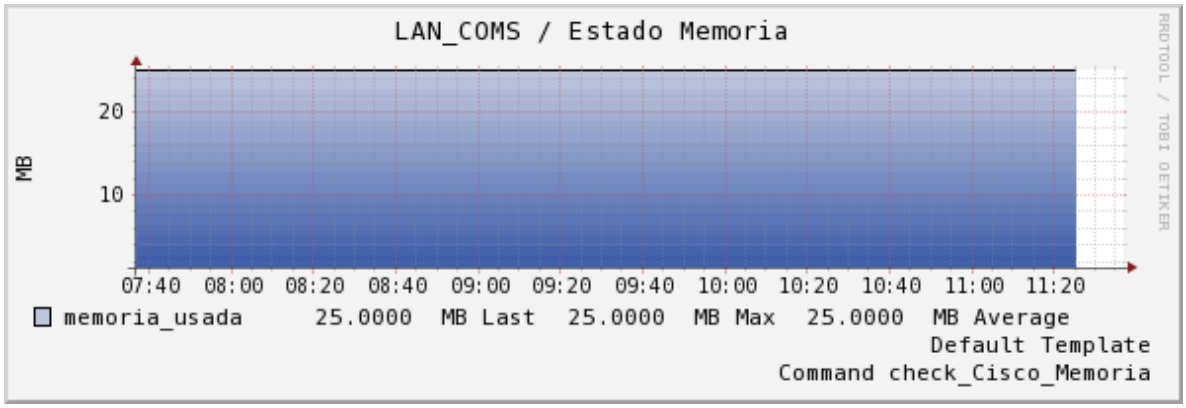
```
LAN_COMS#sh memory
                Head    Total(b)     Used(b)     Free(b)    Lowest(b)  Largest(b)
Processor    1D0AA50    91444656    26613076    64831580     63216432    63622144
      I/O    7400000    12574720     8396216     4178504      4048436     4176224
```

Consulta realizada desde el equipo físico

Gráficamente comprobamos que las muestras también coinciden.



Total memoria dispositivo



Total memoria consumida dispositivo

3.1.3 Estado fuentes alimentación

A continuación se detalla el estado de las fuentes de alimentación en el switch.

En el ejemplo seleccionado se dispone de una única fuente de alimentación disponible y en perfecto funcionamiento si realizamos la consulta por SNMP.

```
[root@app-miquel1 plugins]# ./check_CISCO_Fuentes.pl -H 10.100.7.243 -C Miquel -w 80 -c 90
Fuentes OK - 1 fuentes funcionando | fuentes_totales=1 fuentes_erroneas=0
```

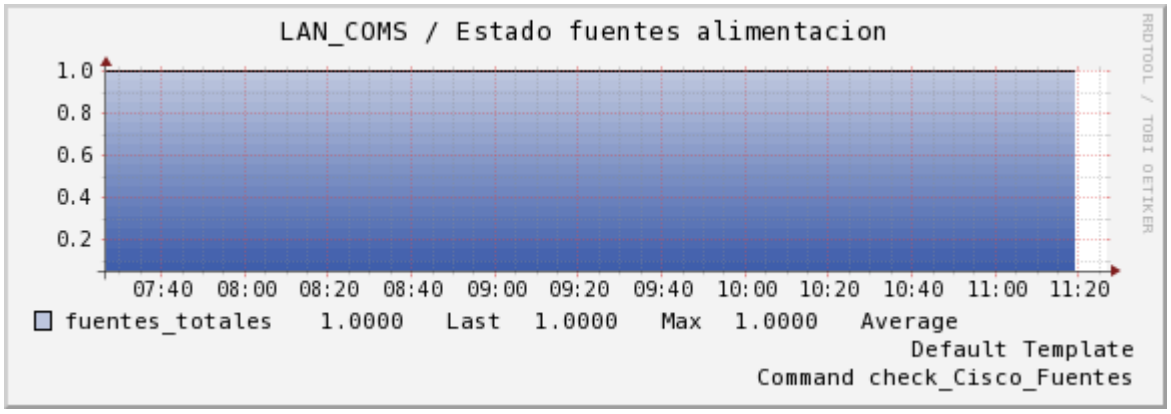
Captura realizada mediante script

A través del comando en el dispositivo vemos que, efectivamente, tenemos una fuente activa.

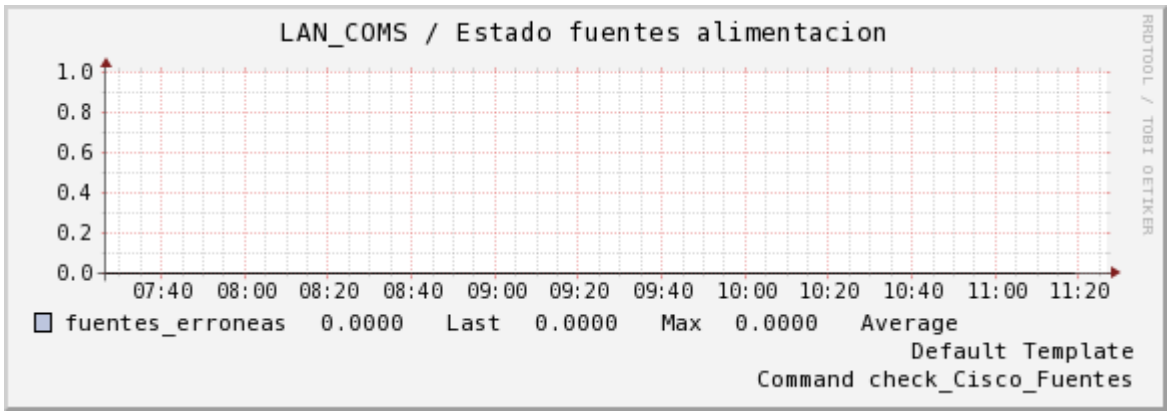
```
LAN_COMS#sh env power
SW  PID                 Serial#     Status           Sys Pwr  PoE Pwr  Watts
--  ------------------  ----------  ---------------  -------  -------  -----
 1  Fixed                                            Good
```

Consulta realizada desde el equipo físico

Del mismo modo, los gráficos son correctos, mostrando el total disponible y total erróneos, respectivamente.



Total fuentes alimentación



Total fuentes alimentación erróneas

3.1.4 Estado ventiladores

En el siguiente caso se muestra el estado de los ventiladores internos del dispositivo.

Realizando la consulta por SNMP vemos que existe un ventilador y está funcionando correctamente.

```
[root@app-miquell plugins]# ./check_CISCO_Ventilador.pl -H 10.100.7.243 -C Miquel
Estado OK: Los 1 ventiladores funcionan | ventiladores_totales=1 ventiladores_erroneos=0
```

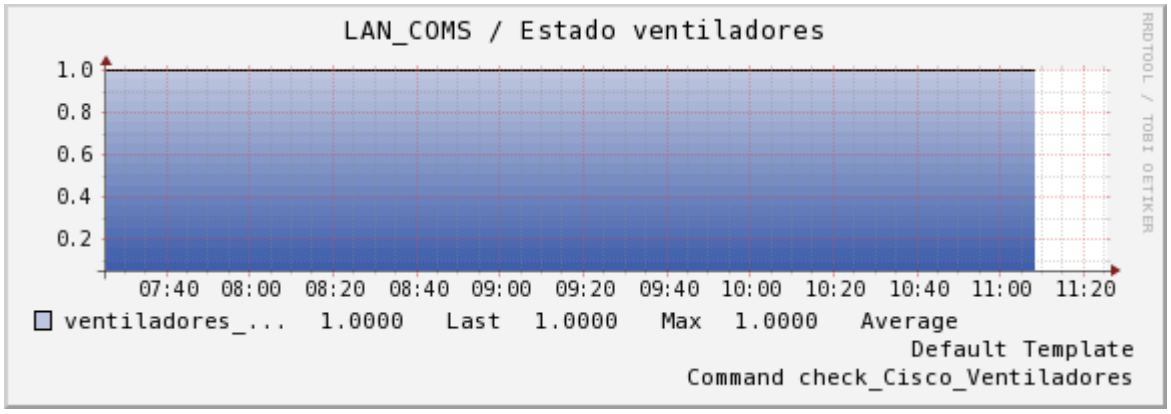
Captura realizada mediante script

Realizando la consulta en el propio equipo se puede confirmar la captura anterior.

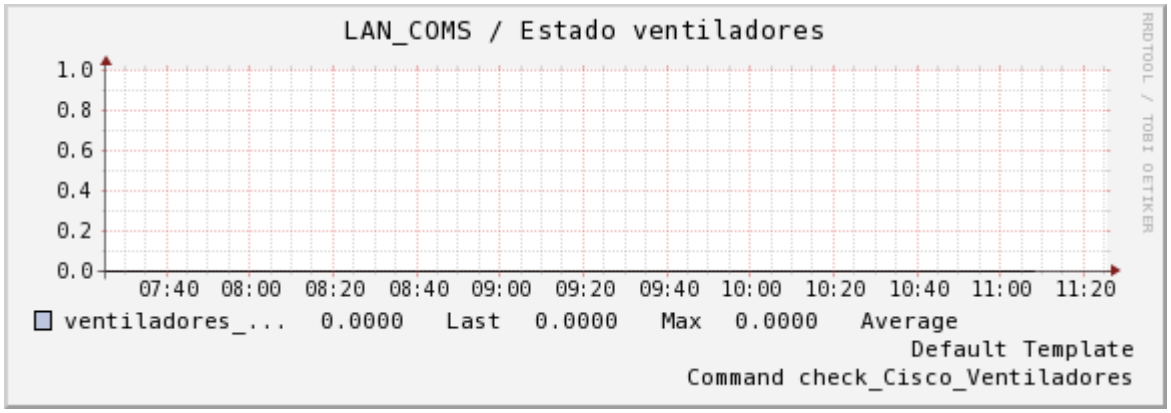
```
LAN_COMS#show env fan
FAN is OK
```

Consulta realizada desde el equipo físico

Y obtenemos los mismos datos capturados por las gráficas.



Total ventiladores



Total ventiladores erróneos

3.1.5 *Estado interfaces*

En las siguientes consultas se identifica el funcionamiento de las interfaces de red activas en el dispositivo.

Se muestran las que son importantes en el switch, para no mostrar todas las que tiene. En este caso no tendrán gráficas ya que se trata de una salida con estado binario 1 (activo) o 0 (inactivo).

```
[root@app-miquel1 plugins]# ./check_CISCO_interface.pl -H 10.100.7.243 -C Miquel GigabitEthernet1/0/21
Estado OK: GigabitEthernet1/0/21  -> up | 1

[root@app-miquel1 plugins]# ./check_CISCO_interface.pl -H 10.100.7.243 -C Miquel GigabitEthernet1/0/22
Estado OK: GigabitEthernet1/0/22  -> up | 1

[root@app-miquel1 plugins]# ./check_CISCO_interface.pl -H 10.100.7.243 -C Miquel GigabitEthernet1/0/23
Estado OK: GigabitEthernet1/0/23  -> up | 1

[root@app-miquel1 plugins]# ./check_CISCO_interface.pl -H 10.100.7.243 -C Miquel GigabitEthernet1/0/24
Estado OK: GigabitEthernet1/0/24  -> up | 1

[root@app-miquel1 plugins]# ./check_CISCO_interface.pl -H 10.100.7.243 -C Miquel Port-channel1
Estado OK: Port-channel1  -> up | 1

[root@app-miquel1 plugins]# ./check_CISCO_interface.pl -H 10.100.7.243 -C Miquel Port-channel2
Estado OK: Port-channel2  -> up | 1
```

Captura realizada mediante script

Revisando el estado de las interfaces a través del comando “show ip interface brief” se muestra el mismo resultado anterior.

```
LAN_COMS#sh ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet1/0/21  unassigned      YES unset  up                    up
GigabitEthernet1/0/22  unassigned      YES unset  up                    up
GigabitEthernet1/0/23  unassigned      YES unset  up                    up
GigabitEthernet1/0/24  unassigned      YES unset  up                    up
Port-channel1          unassigned      YES unset  up                    up
Port-channel2          unassigned      YES unset  up                    up
```

Consulta realizada desde el equipo físico

3.1.6 Estado ping

En el siguiente ejemplo se verá el tiempo de respuesta en mili segundos que tiene el dispositivo desde dentro de su propia red, es decir, un ping desde el equipo de monitorización al switch.

Realizando la consulta mediante el script SNMP se puede ver que el ping es bajo, de unos 18ms, tiempo aceptable para una red interna y que no existen paquetes perdidos en las peticiones realizadas.

```
[root@app-miquell plugins]# ./check_ping -H 10.100.7.243 -w 300,20% -c 500,90% -p 5
PING OK - Packet loss = 0%, RTA = 18.53 ms|rta=18.527000ms;300.000000;500.000000;0.000000 pl=0%;20;90;0
```

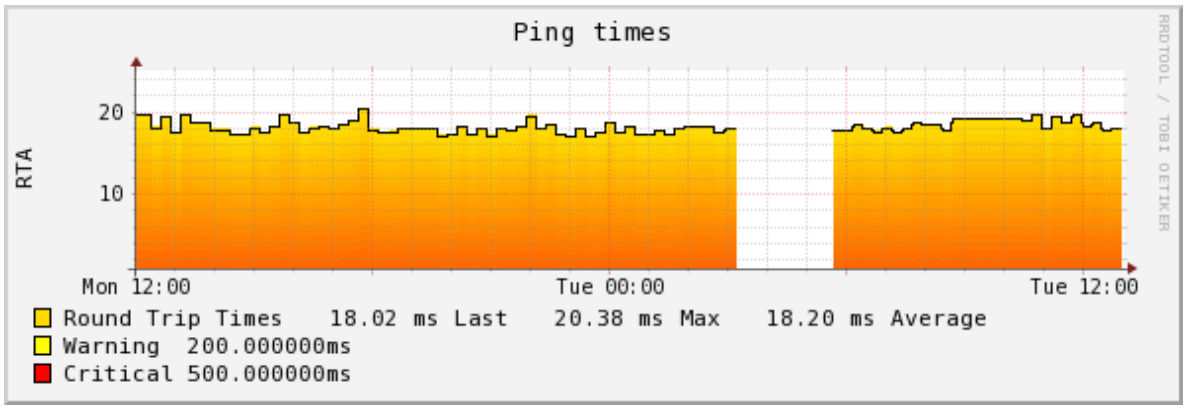
Captura realizada mediante script

Realizamos la consulta manualmente sin necesidad del script para ver el mismo resultado aproximadamente.

```
[root@app-miquell plugins]# ping 10.100.7.243
PING 10.100.7.243 (10.100.7.243) 56(84) bytes of data.
64 bytes from 10.100.7.243: icmp_seq=1 ttl=247 time=19.5 ms
64 bytes from 10.100.7.243: icmp_seq=2 ttl=247 time=17.3 ms
64 bytes from 10.100.7.243: icmp_seq=3 ttl=247 time=17.1 ms
```

Consulta manual

Gráficamente vemos la media en las últimas 12 horas, en las cuales se ve que la media está siempre entorno a los 18ms.



Tiempo en ms de respuesta

Además se puede observar un intervalo sin mostrar su tendencia, esto se debe a que el equipo estuvo parado y por lo tanto no obtuvo muestras. Se confirma así que los datos son totalmente correctos.

4.2 Caso 2: Comprobación equipo Windows

El siguiente ejemplo muestra el funcionamiento de un equipo Windows 2008 R2 con los diferentes servicios que se han implementado para su monitorización:

argon	C:	OK	C: Label: Serial Number 2826092e: 48%used(7437MB/15367MB) (<85%) : OK
	CPU	OK	Carga CPU OK : CPU load 0% < 92%
	Netbackup Client Service	OK	1 services active (matching "NetBackup Client Service$$") : OK
	Ping	OK	PING OK - Packet loss = 0%, RTA = 19.47 ms
	RAM	OK	Physical Memory: 46%used(472MB/1023MB) (<97%) : OK
	Sophos Agent	OK	1 services active (matching "Sophos Agent$$") : OK
	Sophos Anti-Virus	OK	1 services active (matching "^Sophos Anti-Virus$$") : OK
	Uptime	OK	Uptime: 22 days, 0:47:28.20
	Virtual Memory	OK	Virtual Memory: 15%used(367MB/2469MB) (<85%) : OK
	Vnetd	OK	TCP OK - 0.021 second response time on port 13724

Estado equipo Cisco de muestra en aplicación

4.2.1 Estado discos

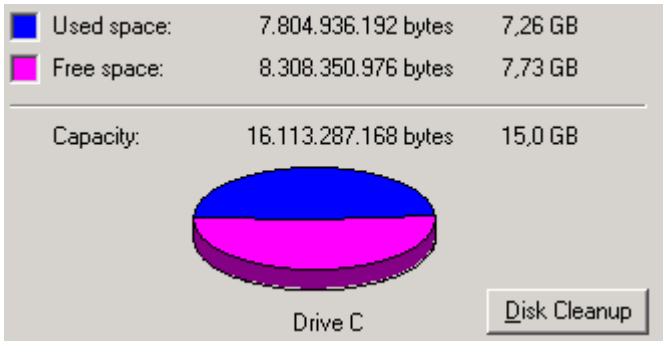
A continuación se mostrará el estado de uno de los discos del equipo, a modo de ejemplo, el funcionamiento de los demás discos será idéntico a modo de consultas y respuestas.

La consulta realizada por SNMP remotamente nos muestra que el nivel de ocupación es del 48% de un total de 15 GB aproximadamente.

```
[root@app-miquel1 plugins]# ./check_snmp_storage.pl -H argon -C Miquel -m ^C -w 85 -c 95 -f
C:\ Label:  Serial Number 2826092e: 48%used(7436MB/15367MB) (<85%) : OK | 'C:\_Label:_Serial_Number_2826092e'=7436MB;13062;14598;0;15367
```

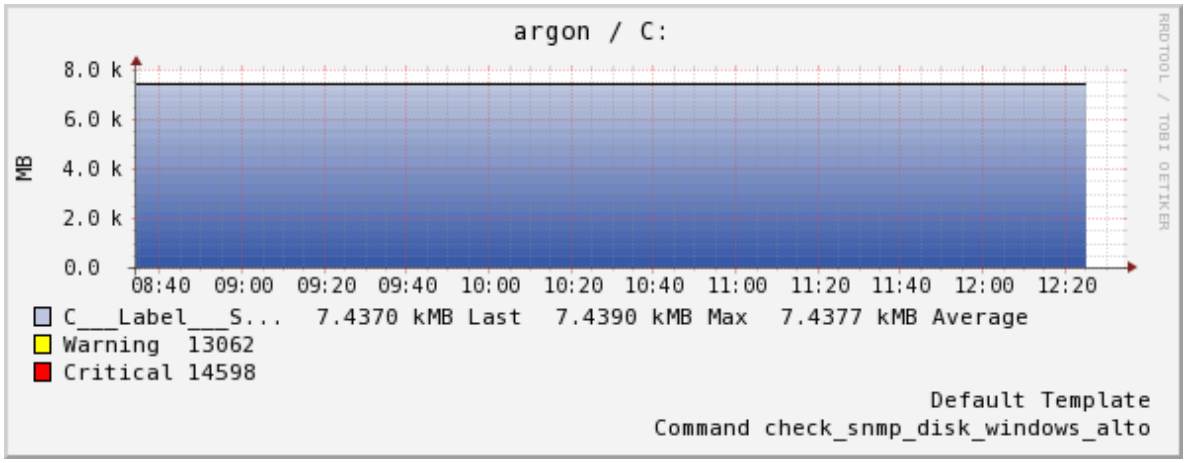
Consulta realizada mediante script

Si se entra al equipo vemos que los datos obtenidos a través de las MIBs son correctos:



Captura desde equipo Windows

En la gráfica siguiente se muestra el estado del disco (memoria restante) en la unidad C en KB, datos que también coinciden con lo obtenido a través de la consulta.



Memoria restante disco C

4.2.2 Estado procesador

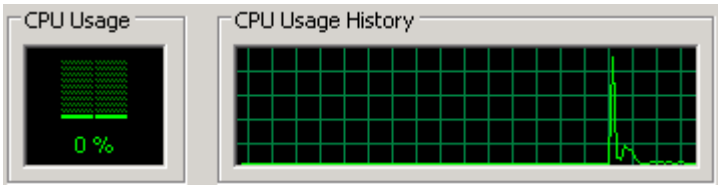
A continuación se mostrará el estado de la CPU del equipo.

En el caso concreto de este equipo vemos que el uso de CPU es del 0%.

```
[root@app-miquel1 plugins]# ./check_win_snmp_cpuload.pl argon Miquel 80 90
Carga CPU OK : CPU load 0% < 80%|cpu_load=0%;80;90[root@app-miquel1 plugins]#
```

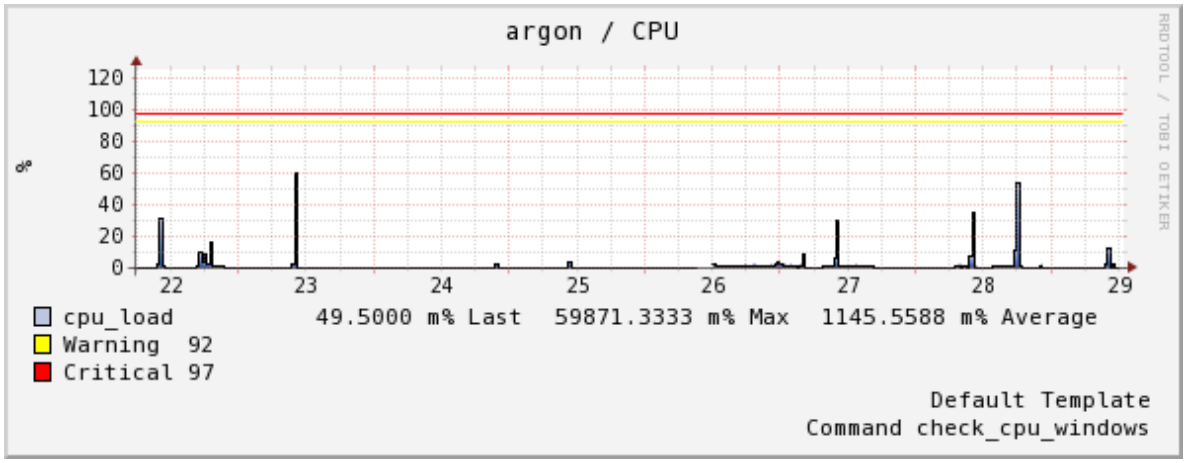
Consulta realizada mediante script

Que sea tan bajo es extraño, pero si entramos en el equipo vemos que es real y que el uso es muy bajo en momentos puntuales, cosa que quedará demostrada en el gráfico posterior a este.



Captura desde equipo Windows

En el gráfico se aprecia el comportamiento anteriormente descrito, con pocos picos de uso por encima del 0%.



Uso de CPU del sistema en 7 días

4.2.3 Estado memoria física

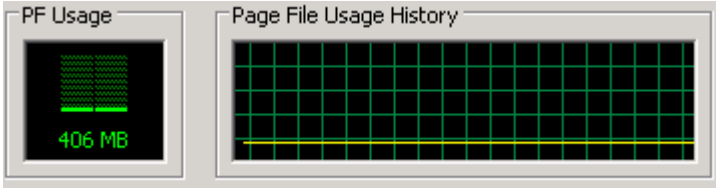
A continuación se mostrará el uso de memora RAM del sistema.

Realizando la consulta a través del script SNMP se muestra que el equipo dispone de un total de 1Gb de RAM y que tiene en uso 525MB, lo que supone un 51% del total usado.

```
[root@app-miquel1 plugins]# ./check_snmp_storage.pl -H argon -C Miquel -m ^Physical -w 80 -c 90 -f
Physical Memory: 51%used(525MB/1023MB) (<80%) : OK | 'Physical_Memory'=525MB;819;921;0;1023
```

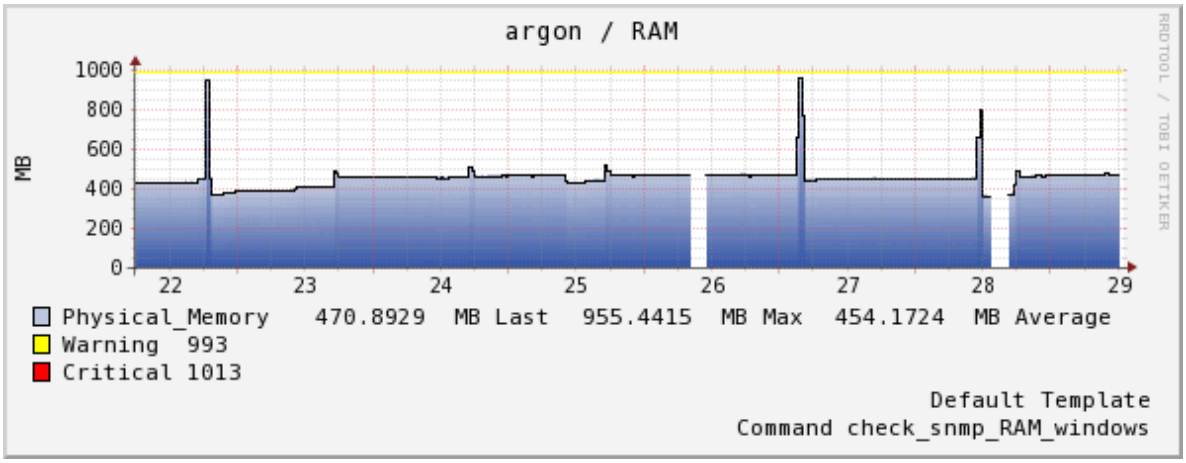
Consulta realizada mediante script

Entrando en el equipo vemos un uso de memoria física que corresponde a la captura, teniendo en cuenta que el uso de RAM es variable, de ahí la pequeña diferencia que se aprecia.



Captura desde equipo Windows

Las gráficas nos muestran la misma información correcta, un uso casi continuo de 400MB aproximadamente.



Uso de RAM del sistema en 7 días

4.2.4 Estado memoria virtual

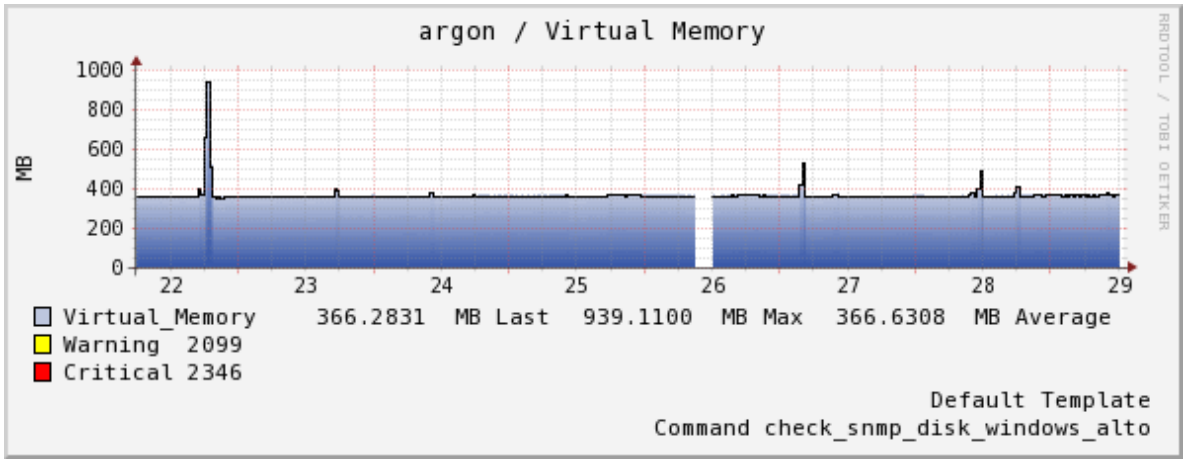
En este apartado se comprobarán las peticiones a la memoria virtual del sistema.

Realizando la petición mediante script comprobamos que, se dispone de un total de 2GB de memoria, de los cuales se ha consumido 406MB

```
[root@app-miquel1 plugins]# ./check_snmp_storage.pl -H argon -C Miquel -m ^Virtual -w 85 -c 95 -f
Virtual Memory: 16%used(406MB/2469MB) (<85%) : OK | 'Virtual_Memory'=406MB;2099;2346;0;2469
```

Consulta realizada mediante script

En el gráfico se recoge la información mostrada anteriormente.



Uso de memoria virtual en los últimos 7 días

4.2.5 Estado puertos

Ahora se comprobará el estado de los puertos por los cuales está escuchando el servidor debido a que alguna aplicación lo requiere.

En este caso veremos el puerto 13724 que es un puerto requerido por un software de backup de Symantec, si le hacemos la petición mediante el script vemos que abre el socket TCP, por lo tanto, el servicio está activo y escuchando por ese puerto.

```
[root@app-miquel1 plugins]# ./check_tcp -H argon -p 13724
TCP OK - 0.038 second response time on port 13724|time=0.038330s;;;0.000000;10.000000
```

Consulta realizada mediante script

Haciendo la consulta en el propio servidor a través de un "netstat" vemos los puertos por los cuales está esperando alguna conexión y por los cuales ya se ha establecido (estos últimos se han descartado). En la lista se muestra que efectivamente, el puerto 13724 está en "listening".

```
R:\>netstat -an

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1025           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1057           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1058           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1124           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1556           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3389           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3460           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:5666           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:5800           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:5900           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:6129           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:8192           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:8193           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:8194           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:12489          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:13722          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:13724          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:13782          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:13783          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:21010          0.0.0.0:0              LISTENING
  TCP    10.100.2.134:139       0.0.0.0:0              LISTENING
```

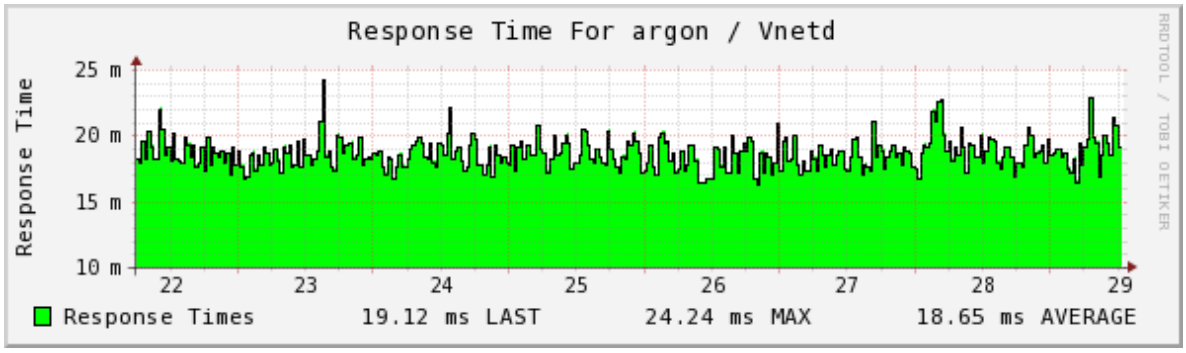
Puertos en escucha del servidor

Del mismo modo probamos que cualquier puerto no indicado en la lista no está escuchando en el equipo.

```
[root@app-miquell plugins]# ./check_tcp -H argon -p 12121
Connection refused
```

Prueba de conexión fallida a puerto

El gráfico recoge el tiempo en mili segundos que tarda en establecerse la conexión por el puerto, se mantiene a una media de 20 ms o 0,020 segundos.



Tiempo de respuesta (ms) al abrir conexión TCP

4.3 Caso 3: Comprobación equipo AIX

Para el siguiente ejemplo se muestra un equipo AIX 6.1 con algunos file systems. Se ha escogido un equipo sin muchos servicios en el que se contemple el funcionamiento del protocolo SNMP y NRPE funcionando al mismo equipo.

maboprdgs	/	OK	/dev/hd4: 18%used(552MB/3072MB) (<80%) : OK
	/admin	OK	/dev/hd11admin: 0%used(0MB/128MB) (<80%) : OK
	/home	OK	/dev/hd1: 1%used(8MB/1024MB) (<90%) : OK
	/opt	OK	/dev/hd10opt: 63%used(242MB/384MB) (<90%) : OK
	/tmp	OK	/dev/hd3: 26%used(1083MB/4096MB) (<90%) : OK
	/usr	OK	/dev/hd2: 90%used(2777MB/3072MB) (<95%) : OK
	/var	OK	/dev/hd9var: 17%used(692MB/4096MB) (<90%) : OK
	CPU load	OK	OK - load average: 0.02, 0.08, 0.23
	Cluster Synchronisation	OK	Cluster Synchronisation OK Mon May 28 14:00:01
	NetBackup Client Service	OK	TCP OK - 0.018 second response time on port 13782
	Ping	OK	PING OK - Packet loss = 0%, RTA = 17.12 ms
	RAM	OK	MemUsed = 76%:
	Swap	OK	SWAP OK - 99% free (28385 MB out of 28672 MB)
	Uptime	OK	Uptime: 76 days, 23:32:26.00
	Vnetd	OK	TCP OK - 0.017 second response time on port 13724

Estado equipo Solaris en la aplicación

4.3.1 Estado file systems

En el siguiente apartado se verá el estado del punto de montaje como ejemplo para un fs.

Se realiza la consulta mediante el script remoto y se muestra que la "/" tiene un total de 3GB asignados, con unos 552MB usados, lo que equivale a un 18% de uso.

```
[root@app-miquell plugins]# ./check_snmp_storage_aix.pl -H maboprdgs -C Miquel -r -m /dev/hd4 -w 80 -c 90 -f
/dev/hd4: 18%used(552MB/3072MB) (<80%) : OK | '/dev/hd4'=552MB;2458;2765;0;3072
```

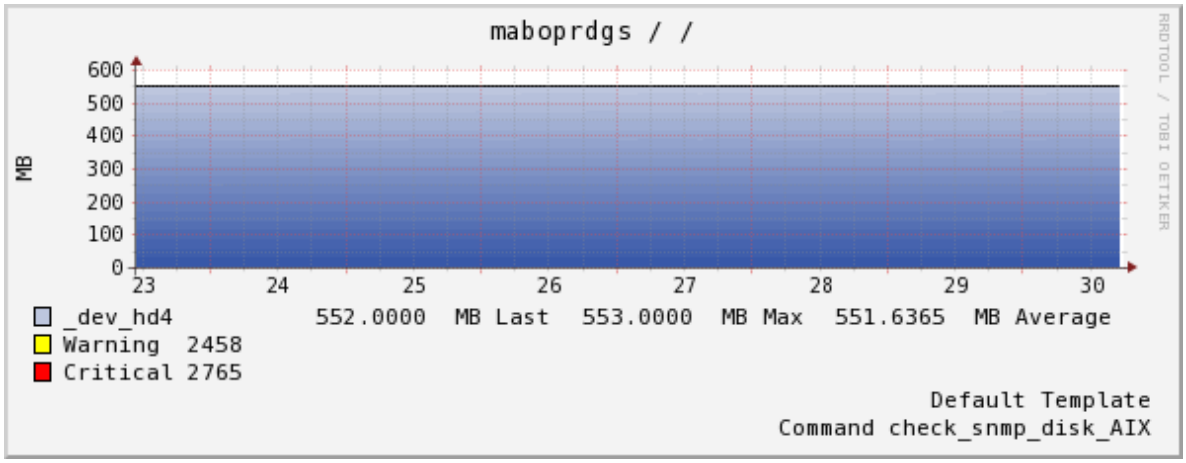
Consulta realizada mediante script

Si se realiza la consulta con un "df -k" en el propio AIX, vemos que los datos obtenidos a través de las MIBs son correctos tanto en porcentaje como en MB disponibles.

```
[root@maboprdgs]:/# df -k
Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4          3145728   2580624   18%    15849     3% /
/dev/hd2          3145728    301916   91%    60541    42% /usr
/dev/hd9var       4194304   3484728   17%    13398     2% /var
/dev/hd3          4194304   3085776   27%     2288     1% /tmp
/dev/hd1          1048576   1040240    1%       55     1% /home
/dev/hd11admin     131072    130692    1%        5     1% /admin
/proc                   -         -    -         -     -  /proc
/dev/hd10opt       393216    145844   63%     9176    22% /opt
```

Consulta de FS del servidor

En el gráfico se observan los mismos datos anteriormente comentados con sus mismos valores.



Uso de filesystem "/" en 7 días

4.3.2 Estado CPU

A continuación se detalla el consumo de CPU del sistema a través de NRPE.

Realizamos la consulta mediante el script NRPE y obtenemos lo siguiente:

```
[root@app-miquell plugins]# ./check_nrpe -n -H maboprdgs -c check_load -t 120
OK - load average: 0.19, 0.12, 0.07|load1=0.190;20.000;30.000;0; load5=0.120;19.000;29.000;0; load15=0.070;18.000;28.000;0;
```

Consulta realizada mediante script

A la consulta se le ha pasado como parámetros en nombre del equipo, debido a que resuelve por dns, aunque también se podría haber hecho por IP; el nombre del script que tiene la máquina remota en uno de sus directorios, y el –t 120 que es un parámetro de "time out" para que en caso de no respuesta, deje de hacer la petición en 2 segundos.

Como se ha comentado en capítulos anteriores, esta petición busca en el fichero "nrpe.cfg" del servidor AIX al cual le realizamos la petición y busca en la ruta a la que apunta "check_load" para ejecutar el script.

```
#SISTEMA
command[check_load]=/opt/nagios/libexec/check_load -w 20,19,18 -c 30,29,28
command[check_SWAP]=/opt/nagios/libexec/check_swap -w 20% -c 10%
command[check_RAM]=/opt/nagios/libexec/check_aix_mem.sh -c 99 -w 90 -t pctused
```

Fichero nrpe.cfg

En la imagen anterior se puede comprobar la ejecución del script que se realizará al pasar como parámetro "check_load". Al igual que en los anteriores casos, realizamos la consulta desde la propia máquina.

```
[root@maboprdgs]:/# /opt/nagios/libexec/check_load -w 20,19,18 -c 30,29,28
OK - load average: 0.12, 0.12, 0.20|load1=0.120;20.000;30.000;0; load5=0.120;19.000;29.000;0; load15=0.200;18.000;28.000;0
```

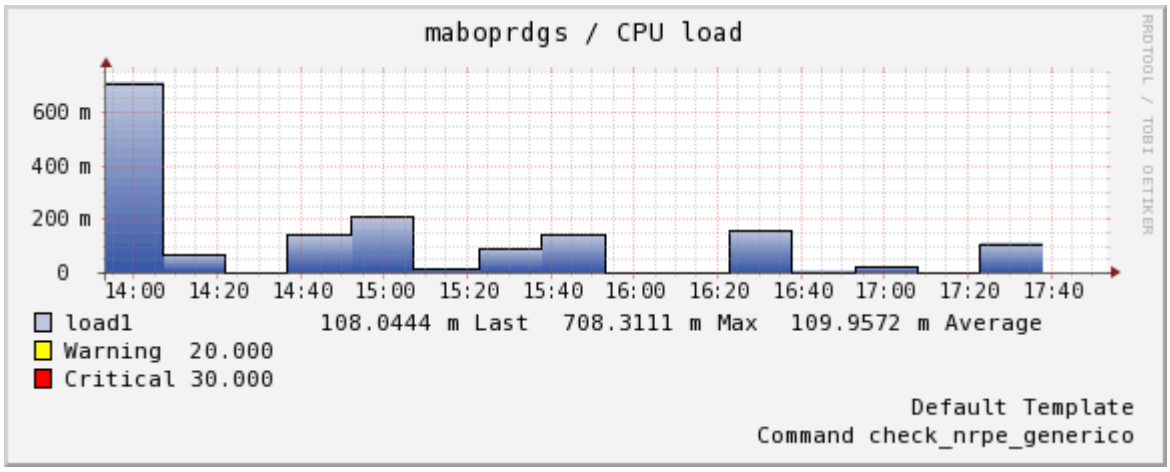
Y que a su vez será devuelta al servidor IBM que ha solicitado la petición.

También podemos comprobar si los datos son correctos realizando la consulta a través de comandos de sistema sin necesidad de scripts, comprobamos que todo es correcto:

```
[root@maboprdgs]:/# uptime
  05:43PM   up 84 days, 34 mins,  1 user,  load average: 0.12, 0.20, 0.12
```

Carga real del equipo

En el siguiente gráfico se muestra la carga del sistema en las últimas horas.



Carga del AIX en últimas 4 horas

4.3.3 Estado swap

En este punto se recogen los datos relacionados con el uso de *swaping* que pueda tener el equipo.

Realizamos de nuevo una consulta por NRPE y obtenemos que tiene asignado casi 3GB y de ellos el 99% está libre de uso. En el caso de la memoria swap, las lecturas siempre se muestran en porcentaje libre, cosa que también se reflejará en el gráfico de este apartado.

```
[root@app-miquell plugins]# ./check_nrpe -n -H maboprdgs -c check_SWAP -t 120
SWAP OK - 99% free (28385 MB out of 28672 MB) |swap=28385MB;5734;2867;0;28672
```

Consulta realizada mediante script

A través del comando “swap –l” se muestra en tiempo real sobre el equipo.

```
[root@maboprdgs]:/# swap -l
device              maj,min        total       free
/dev/hd6              10,  2     28672MB     28388MB
```

Uso de swap a tiempo real

Vemos que el uso de swap no varía en los últimos días y que se muestra el total libre, tal y como se ha visto en la consulta mediante NRPE.

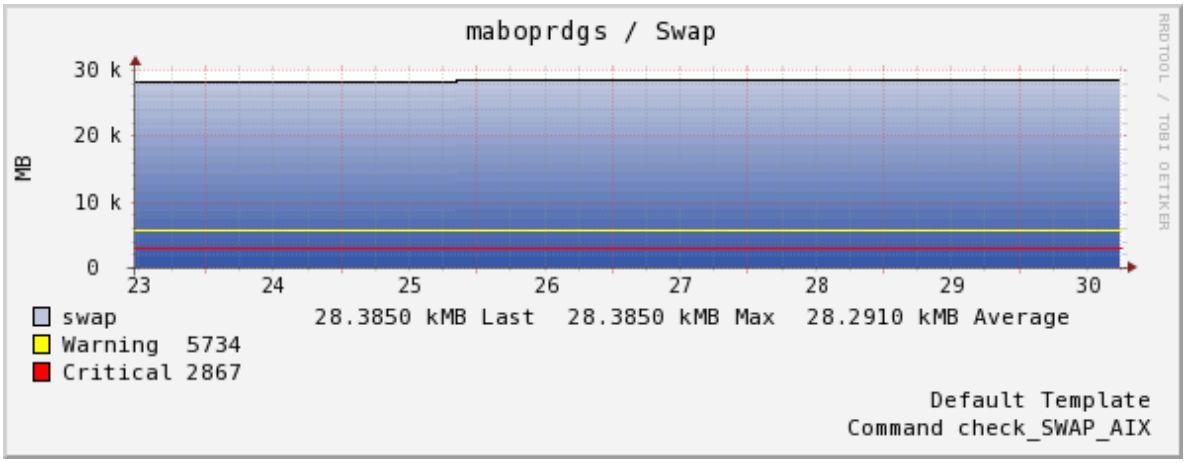


Grafico últimos 7 días de uso de swap

4.3.4 Estado puertos

A continuación se muestra uno de los checks a los puertos tcp activos en el equipo.

Existe un cliente de netbackup funcionando por el puerto tcp 13782, lo comprobamos a través de una consulta tcp.

```
[root@app-miquell plugins]# ./check_tcp -H maboprdgs -p 13782
TCP OK - 0.038 second response time on port 13782|time=0.037816s;;;0.000000;10.000000
```

Consulta realizada mediante script

Que equivale a realizar una consulta de telnet remota, viendo como efectivamente abre el socket:

```
[root@app-miquell ~]# telnet maboprdgs 13782
Trying 10.100.1.65...
Connected to maboprdgs.miquel.es (10.100.1.65).
Escape character is '^]'.
```

Socket aceptado puerto 13782

Realizamos la consulta localmente y vemos si el puerto está “escuchando” posibles conexiones, viendo que así es.

```
[root@maboprdgs]:/# netstat -an | grep 13782
tcp4       0      0  *.13782                *.*                    LISTEN
```

Puerto a la escucha de peticiones

En este caso no se ha considerado importante el tiempo que tarda la petición tcp en realizarse y por lo tanto, no se ha introducido gráfico debido a que solo se considera un estado binario de activo/inactivo.

4.4 Caso 4: Comprobación equipo Linux

Se ha escogido un equipo Linux sin muchos servicios en el que se contemple el funcionamiento del protocolo SNMP y NRPE atacando al mismo equipo.

bustiesprd01	/	OK	/: 19%used(4669MB/24920MB) (<75%) : OK
	/busties	OK	/busties: 97%used(148496MB/153593MB) (<97%) : OK
	/busties_backup	OK	/busties_backup: 72%used(150788MB/208368MB) (<80%) : OK
	/var	OK	/var: 11%used(647MB/6150MB) (<80%) : OK
	CPU	OK	OK - load average: 0.28, 0.36, 0.32
	FTP	OK	TCP OK - 0.019 second response time on port 21
	NetBackup Client Service	OK	TCP OK - 0.019 second response time on port 13782
	Ping	OK	PING OK - Packet loss = 0%, RTA = 17.74 ms
	Procesos vsftpd	OK	PROCS OK: 7 processes with command name 'vsftpd'
	Swap	OK	SWAP OK - 100% free (3074 MB out of 3074 MB)
	Uptime	OK	Uptime: 1 day, 3:13:30.63
	Vnetd	OK	TCP OK - 0.018 second response time on port 13724

Estado equipo linux en la aplicación

4.4.1 Estado file system

Se mostrará el estado del file system "/" del equipo como ejemplo.

Realizando la consulta SNMP obtenemos un 19% de uso total de la partición, con un total de casi 25GB asignados y 5GB usados.

```
[root@app-miquell plugins]# ./check_snmp_storage.pl -H 10.100.46.11 -C Miquel -r -m / -w 80 -c 90 -f
/: 19%used(4668MB/24920MB) (<80%) : OK | '/'=4668MB;19936;22428;0;24920
```

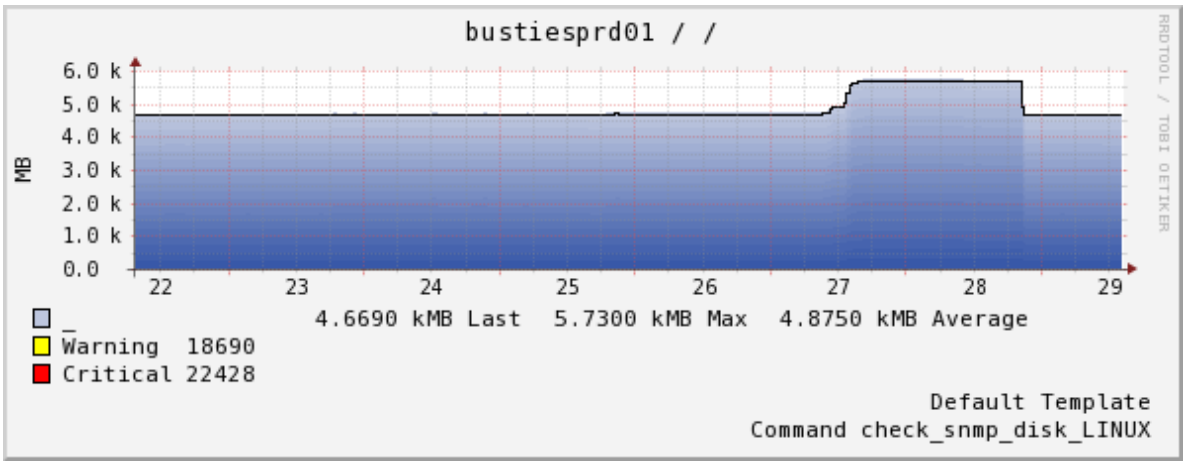
Consulta realizada mediante script

Si se miran los datos desde el equipo obtenemos los mismos valores en la "/".

```
bustiesprd01:~ # df -k
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/cciss/c0d0p4     25518448   4781436  20737012  19% /
udev                    777688       152    777536   1% /dev
/dev/cciss/c0d0p1       514028     60296    453732  12% /boot
/dev/cciss/c0d1p1    213368780 154407120  58961660  73% /busties_backup
/dev/cciss/c0d0p3      6297272    662924   5634348  11% /var
/dev/sda1            157279484 152060456   5219028  97% /busties
```

Tabla de FS en servidor Linux

Las estadísticas de los últimos 7 días muestran los datos anteriores.



Progreso fs "/" en servidor linux

4.4.2 Estado CPU

A continuación se mostrará el estado de la CPU del equipo por NRPE.

Realizamos la petición solicitando la ejecución remota del script "check_load" y vemos que la carga del equipo es muy baja.

```
[root@app-miquel1 plugins]# ./check_nrpe -n -H bustiesprd01 -c check_load -t 120
OK - load average: 0.80, 0.81, 0.67|load1=0.800;15.000;30.000;0; load5=0.810;10.000;25.000;0; load15=0.670;5.000;20.000;0;
```

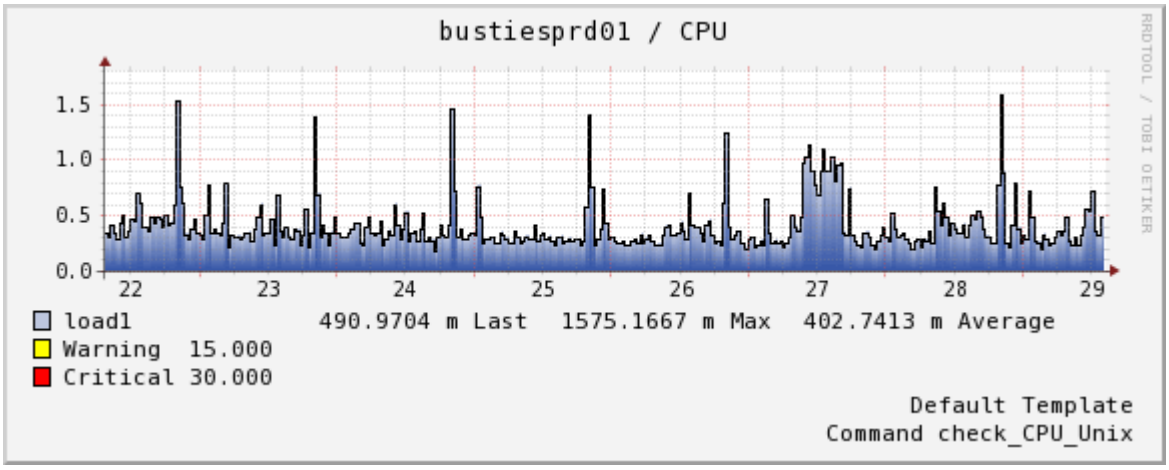
Consulta realizada mediante script

Con el comando "top" comprobamos que lo datos anteriores mostrados mediante el script son correctos.

```
bustiesprd01:~ # top
top - 08:52:43 up 440 days, 19:20,  1 user,  load average: 0.26, 0.29, 0.27
```

Carga CPU desde el propio sistema

El gráfico verifica el correcto funcionamiento de las salidas anteriores, se muestra la media en los últimos 7 días.



Consumo últimos 7 días de CPU del sistema

4.4.3 Estado swap

Para ver el estado de la memoria swap se realizan los mismos pasos.

Las pruebas mediante el script muestran que se dispone de 3GB totales, de los cuales está libre el 100% de ellos.

```
[root@app-miquel1 plugins]# ./check_nrpe -n -H bustiesprd01 -c check_swap -t 120
SWAP OK - 100% free (3074 MB out of 3074 MB) |swap=3074MB;922;614;0;3074
```

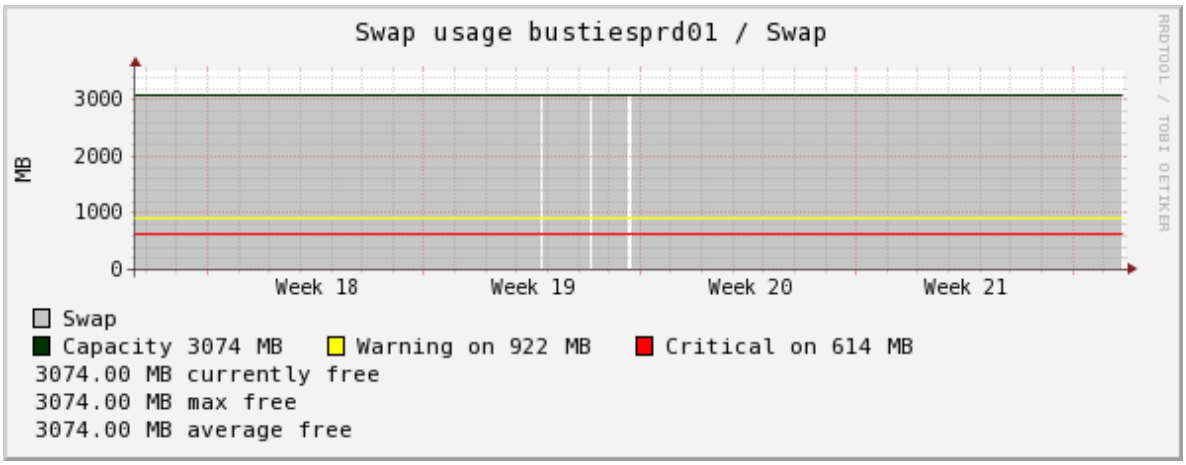
Consulta realizada mediante script

La consulta al equipo físico muestra que efectivamente tiene 3GB de swap (3148732 K) y usado nada prácticamente (160 K), equivalente a un 0%.

```
bustiesprd01:~ # top
top - 12:34:49 up 440 days, 23:02,  2 users,  load average: 0.28, 0.41, 0.36
Tasks:  99 total,   1 running,  98 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.5%us,  0.2%sy,  0.0%ni, 98.7%id,  0.5%wa,  0.0%hi,  0.2%si,  0.0%st
Mem:   1555380k total,  1440724k used,   114656k free,   590516k buffers
Swap:  3148732k total,      160k used,  3148572k free,   447664k cached
```

Asignación swap y uso desde el sistema

Al igual que en todos los casos, la memoria se muestra en forma de gráfico de manera correcta.



Estadística últimas 4 semanas de swap libre

4.4.4 Estado puertos

A continuación se demostrará el correcto funcionamiento de la escucha de puertos en el equipo.

Realizamos la petición en remoto a través de los puertos 13782 y 13724 correspondientes a la aplicación de backup interna. Comprobamos que la conexión es correcta y por lo tanto significa que el software funciona correctamente.

```
[root@app-miquell plugins]# ./check_tcp -H bustiesprd01 -p 13782
TCP OK - 0.033 second response time on port 13782|time=0.032961s;;;0.000000;10.000000
```

Petición puerto 13782 mediante script

```
[root@app-miquell plugins]# ./check_tcp -H bustiesprd01 -p 13724
TCP OK - 0.033 second response time on port 13724|time=0.033169s;;;0.000000;10.000000
```

Petición puerto 13724 mediante script

```
[root@app-miquell plugins]# ./check_tcp -H bustiesprd01 -p 21
TCP OK - 0.033 second response time on port 21|time=0.033343s;;;0.000000;10.000000
```

Petición puerto 21 mediante script

Las capturas anteriores muestran la petición a través del script, que equivale a hacer un telnet al puerto:

```
[root@app-miquel1 ~]# telnet bustiesprd01 13782
Trying 10.100.46.11...
Connected to bustiesprd01.miquel.es (10.100.46.11).
```

Petición al puerto 13782

```
[root@app-miquel1 ~]# telnet bustiesprd01 13724
Trying 10.100.46.11...
Connected to bustiesprd01.miquel.es (10.100.46.11).
```

Petición al puerto 13724

```
[root@app-miquel1 ~]# telnet bustiesprd01 21
Trying 10.100.46.11...
Connected to bustiesprd01.miquel.es (10.100.46.11).
Escape character is '^]'.
220  - Bienvenido al servicio FTP de                          - ( -- 2009 -- ) Nodo 2
```

Petición al puerto de ftp

Ahora entramos en el equipo afectado y lo comprobamos internamente, una vez hecha la petición vemos que efectivamente el equipo está escuchando por esos puertos posibles conexiones.

```
bustiesprd01:~ # netstat -an | grep 13782
tcp        0      0 0.0.0.0:13782           0.0.0.0:*               LISTEN
```

Puerto 13782 a la escucha de peticiones

```
bustiesprd01:~ # netstat -an | grep 13724
tcp        0      0 0.0.0.0:13724           0.0.0.0:*               LISTEN
```

Puerto 13724 a la escucha de peticiones

En el caso de este puerto 21 (ftp), se puede comprobar que existen conexiones activas por ese puerto en este preciso momento.

```
bustiesprd01:~ # netstat -an | grep :21
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN
tcp        0      0 10.100.46.11:21         10.100.46.17:44920      TIME_WAIT
tcp        0      0 10.100.46.11:21         10.100.46.17:44947      TIME_WAIT
tcp        0      0 10.100.46.11:21         83.56.44.80:3479        TIME_WAIT
tcp        0      0 10.100.46.11:21         10.100.46.17:44933      TIME_WAIT
tcp        0      0 10.100.46.11:21         10.100.46.17:44990      TIME_WAIT
tcp        0      0 10.100.46.11:21         10.100.46.17:44975      TIME_WAIT
tcp        0      0 10.100.46.11:21         10.100.46.17:44962      TIME_WAIT
tcp        0      0 10.100.46.11:2161       10.100.1.134:8000       ESTABLISHED
tcp        0      0 10.100.46.11:2160       10.100.1.134:8000       ESTABLISHED
tcp        0      0 10.100.46.11:2162       10.100.1.134:8000       ESTABLISHED
tcp        0      0 10.100.46.11:2169       10.100.1.134:8000       ESTABLISHED
tcp        0      0 10.100.46.11:2168       10.100.1.134:8000       ESTABLISHED
tcp        1      0 10.100.46.11:2156       10.100.1.134:8000       CLOSE_WAIT
tcp        1      0 10.100.46.11:2159       10.100.1.134:8000       CLOSE_WAIT
tcp        1      0 10.100.46.11:2158       10.100.1.134:8000       CLOSE_WAIT
tcp        1      0 10.100.46.11:2155       10.100.1.134:8000       CLOSE_WAIT
tcp        0      0 10.100.46.11:21         88.22.136.151:3257      ESTABLISHED
```

Puerto ftp a la escucha de peticiones

4.4.5 Estado procesos

A continuación se estudiará cómo interpreta los procesos que están corriendo en el servidor.

La consulta a través del check NRPE realiza un "count" de los procesos con descripción "vsftpd" (very secure FTP daemon) y se muestran 7 en este momento.

```
[root@app-miquel1 plugins]# ./check_nrpe -n -H bustiesprd01 -c check_vsftpd -t 120
PROCS OK: 7 processes with command name 'vsftpd'
```

Consulta realizada mediante script

Haciendo la misma comprobación dentro de la máquina se confirma que la salida anterior era correcta, ya que existen 7 procesos con esa descripción activos.

```
bustiesprd01:~ # ps -ef | grep vsftpd
ftp      10876 26185  1 14:21 ?        00:00:00 /usr/sbin/vsftpd
c84284   10880 10876  0 14:21 ?        00:00:00 /usr/sbin/vsftpd
ftp      10951 26185  2 14:22 ?        00:00:00 /usr/sbin/vsftpd
c84141   10994 10951  0 14:22 ?        00:00:00 /usr/sbin/vsftpd
ftp      11034 26185  0 14:22 ?        00:00:00 /usr/sbin/vsftpd
ftpprd   11036 11034  0 14:22 ?        00:00:00 /usr/sbin/vsftpd
root     11038  9986  0 14:22 pts/0    00:00:00 grep vsftpd
root     26185     1  0  2011 ?        00:54:02 /usr/sbin/vsftpd
```

Estado de procesos vsftpd activos

4.5 Caso 5: Comprobación equipo Solaris

Para el siguiente ejemplo se muestra un equipo Solaris con algunos file systems. Se ha escogido un equipo sin muchos servicios en el que se contemple el funcionamiento del protocolo SNMP y NRPE funcionando al mismo equipo.

A nivel de peticiones SNMP es idéntico a los IAX, con la única diferencia que, a los AIX se les pregunta no por el file system, sino por el punto de montaje de la partición, mientras que a los Solaris directamente le preguntaremos por el propio file system.

El estudio del estado de la memoria en sistemas Solaris sería igual que en sistemas AIX, por lo que se descartan las capturas que reflejen su uso.

madq	/	OK	/: 87%used(8780MB/10085MB) (<90%) : OK
	/opt/openv	OK	/opt/openv: 72%used(7300MB/10069MB) (<90%) : OK
	/var	OK	/var: 79%used(4802MB/6053MB) (<80%) : OK
	/var/run	OK	/var/run: 0%used(0MB/124418MB) (<90%) : OK
	CPU	OK	OK - load average: 0.14, 0.16, 0.15
	NetBackup Client Service	OK	TCP OK - 0.017 second response time on port 13782
	Netbackup Master Server	OK	TCP OK - 0.016 second response time on port 1556
	Ping	OK	PING OK - Packet loss = 0%, RTA = 17.50 ms
	RAM	OK	MEM OK - Memory free 80.00%
	Swap	OK	/tmp: 0%used(0MB/124419MB) (<90%) : OK
	Uptime	OK	Uptime: 32 days, 4:26:05.72
	Vnetd	OK	TCP OK - 0.016 second response time on port 13724

Estado equipo Solaris en la aplicación

4.5.1 Estado file systems

Se mostrará el estado del file system "/" del equipo como ejemplo.

Realizando la consulta SNMP obtenemos un 87% de uso total de la partición, con un total de casi 25 GB asignados y 8,7 GB usados.

```
[root@app-miquel1 plugins]# ./check_snmp_storage.pl -H madq -C Miquel -r -m / -w 90 -c 95 -f
/: 87%used(8780MB/10085MB) (<90%) : OK | '/'=8780MB;9077;9581;0;10085
```

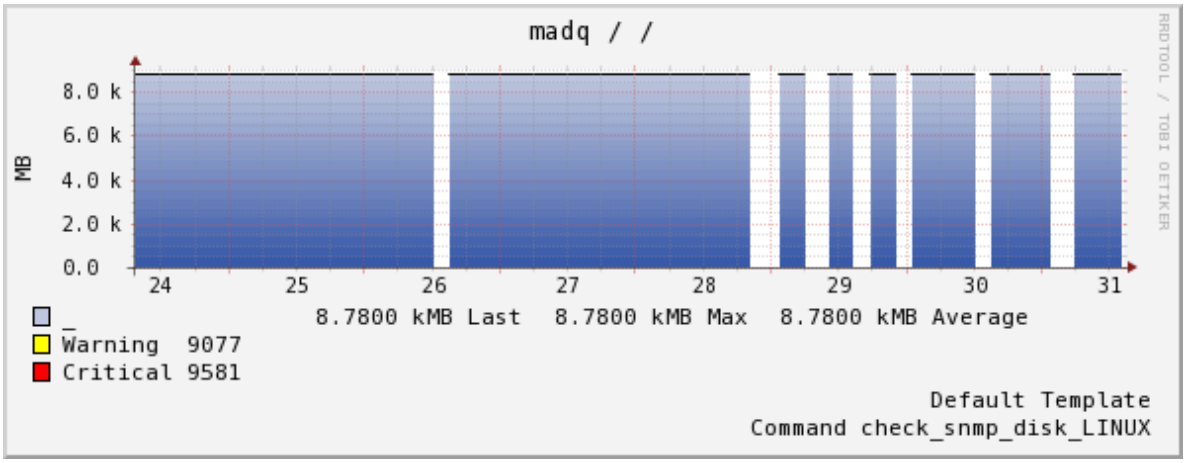
Consulta realizada mediante script

La consulta desde el equipo corrobora el resultado anterior.

```
[madq:root]/> df -k
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/md/dsk/d0       10327372 8990400 1233699    88%    /
/devices                 0       0       0     0%    /devices
ctfs                     0       0       0     0%    /system/contract
proc                     0       0       0     0%    /proc
mnttab                   0       0       0     0%    /etc/mnttab
swap                127406480    1128 127405352     1%    /etc/svc/volatile
objfs                    0       0       0     0%    /system/object
fd                       0       0       0     0%    /dev/fd
/dev/md/dsk/d3       6198606 4917204 1219416    81%    /var
swap                127405552     200 127405352     1%    /tmp
swap                127405464     112 127405352     1%    /var/run
```

Tabla de file systems montados

El gráfico siguiente muestra la evolución de la "/" en los últimos 7 días. También se pueden apreciar cortes intermitentes, que coinciden con problemas puntuales en el equipo de rendimiento, como ya se vio en el estudio del SNMP, este tráfico es descartado en primera instancia en un equipo si necesita de sus pocos recursos (los que consume SNMP) para funcionar mejor.



Capacidad de fs "/" durante los últimos 7 días

4.5.2 Estado puertos

Por último se verá el resultado de las peticiones a los puertos activos del sistema.

Realizamos las peticiones externamente a partir del check tcp.

```
[root@app-miquell plugins]# ./check_tcp -H madq -p 13782
TCP OK - 0.033 second response time on port 13782|time=0.033128s;;;0.000000;10.000000
```

Petición tcp puerto 13782

```
[root@app-miquell plugins]# ./check_tcp -H madq -p 1556
TCP OK - 0.042 second response time on port 1556|time=0.042060s;;;0.000000;10.000000
```

Petición tcp puerto 1556

```
[root@app-miquell plugins]# ./check_tcp -H madq -p 13724
TCP OK - 0.033 second response time on port 13724|time=0.033137s;;;0.000000;10.000000
```

Petición tcp puerto 13724

El siguiente paso es comprobar que desde dentro del equipo los puertos se muestran activos, como se ha demostrado en las conexiones tcp anteriores.

```
[madq:root]/> netstat -an | grep 13782
      *.13782              *.*                  0      0 49152      0 LISTEN
```

Puerto 13782 en escucha en el equipo

```
[madq:root]/> netstat -an | grep 1556
10.100.1.50.60543    10.100.1.237.1556    64305      0 64240      0 ESTABLISHED
10.100.1.50.60545    10.100.1.237.1556    65031      0 64240      0 ESTABLISHED
10.100.1.50.60552    10.100.1.237.1556    64319      0 64240      0 ESTABLISHED
10.100.1.50.1556     10.100.1.237.4219    65461      0 64240      0 ESTABLISHED
      *.1556               *.*                  0      0 49152      0 LISTEN
10.100.1.50.1556     10.100.1.236.1686    65535      0 49640      0 TIME_WAIT
10.100.1.50.1556     10.100.1.236.1687    65535      0 49640      0 TIME_WAIT
10.100.1.50.38301    10.100.1.237.1556    65535      0 64240      0 ESTABLISHED
10.100.1.50.38304    10.100.1.237.1556    64754      0 64240      0 ESTABLISHED
10.100.1.50.1556     10.100.1.236.1933    65535      0 49640      0 TIME_WAIT
10.100.1.50.1556     10.100.1.236.1934    65535      0 49640      0 TIME_WAIT
10.100.1.50.38306    10.100.1.237.1556    64095      0 64240      0 ESTABLISHED
10.100.1.50.38308    10.100.1.237.1556    64893      0 64240      0 ESTABLISHED
10.100.1.50.60596    10.100.1.237.1556    65087      0 64240      0 ESTABLISHED
10.100.1.50.58593    10.100.1.237.1556    65384      0 64240      0 ESTABLISHED
```

Puerto 1556 en escucha en el equipo

```
[madq:root]/> netstat -an | grep 13724
      *.13724              *.*                0      0 49152      0 LISTEN
10.100.1.50.13724    10.100.1.237.4541    63341      0 49640      0 TIME_WAIT
10.100.1.50.13724    10.100.1.237.4539    64234      0 49640      0 TIME_WAIT
10.100.1.50.13724    10.100.1.237.4619    64234      0 49640      0 TIME_WAIT
10.100.1.50.13724    10.100.1.237.4621    63342      0 49640      0 TIME_WAIT
10.100.1.50.13724    10.100.1.237.4716    64234      0 49640      0 TIME_WAIT
10.100.1.50.13724    10.100.1.237.4717    63342      0 49640      0 TIME_WAIT
10.100.1.50.13724    10.100.1.237.4796    62934      0 49640      0 ESTABLISHED
```

Puerto 13724 en escucha en el equipo

Con las comprobaciones anteriores realizadas en todas las tecnologías que se han estudiado, se demuestra el correcto funcionamiento de los scripts elegidos como muestra. Todo script realizado remotamente ha presentado un valor exacto o muy aproximado a lo que la realidad indica al proceder a hacer la misma consulta localmente.

4.6 Caso 6: Prueba de host caído y notificaciones

En este punto se verá un ejemplo del comportamiento de un equipo cuando va a generar una alerta.

Para ello se parará un switch y se analizará todo el proceso que realiza la aplicación desde que se pierde la señal con él, pasando por las comprobaciones que realiza, notificaciones que envía y posterior recuperación.

Para el ejemplo partiremos del siguiente switch:

CORE6500-VM_MA_CORE_1	Estado CPU	OK	Cpu OK - Carga actual: 5% Media 1 minuto: 5% Media 5 minutos: 5%
	Estado Memoria	OK	Estado OK -> Memoria: 52%
	Estado fuentes alimentacion	OK	Fuentes OK - 2 fuentes funcionando
	Estado ventiladores	OK	Estado OK: Los 2 ventiladores funcionan
	Ping	OK	PING OK - Packet loss = 0%, RTA = 2.87 ms

Ahora pararemos el equipo físicamente para ver si el sistema muestra el estado erróneo del mismo, hay que tener en cuenta que el estado del switch se comprueba cada 5 minutos, por lo que tendremos un retardo desde que se apague entre 0 a 5 minutos.

Los tiempos siguientes han sido sacados de la cola de procesos que se están generando y que está gestionando nagios, como se puede ver, en la primera línea se muestra el check del estado del switch, que es de 5 minutos, debido a que se prioriza sobre cualquier otra comprobación ver que el estado de cualquier equipo esté bien. Los demás tiempos que se muestran son de los servicios, que son superiores siendo de 15 minutos cada uno.

Host	Service	Last Check	Next Check
CORE6500-VM_MA_CORE_1		03-06-2012 14:07:03	03-06-2012 14:12:16
FW-NOKIA-GS-01	Vrrp	03-06-2012 13:54:35	03-06-2012 14:09:35
CORE6500-VM_MA_CORE_1	Ping	03-06-2012 13:54:35	03-06-2012 14:09:35
seleniumB2B		03-06-2012 14:04:04	03-06-2012 14:09:36
MA_ADMIN_DISTRIB_01	Estado ventiladores	03-06-2012 13:23:54	03-06-2012 14:23:54
CORE6500-VM_MA_CORE_1	Estado CPU	03-06-2012 13:55:25	03-06-2012 14:10:25
CORE6500-VM_MA_CORE_1	Estado Memoria	03-06-2012 14:09:51	03-06-2012 14:24:51
CORE6500-VM_MA_CORE_1	Estado fuentes alimentacion	03-06-2012 14:10:06	03-06-2012 14:25:06

En el momento en que se compruebe el primer check, es decir, una vez han pasado el máximo de los 5 minutos, vemos lo siguiente en el sistema de monitorización. El color rojo muestra que existe algún problema, en este caso es que no se puede conectar con el switch, esto es debido a que el check ha enviado la variable “stat” con valor 2 a nagios.

Estado de un switch parado

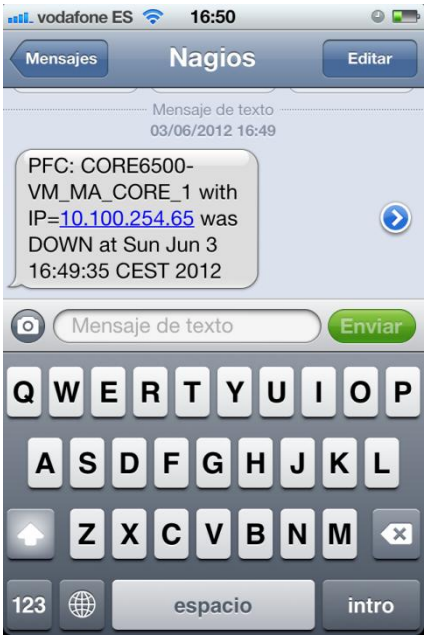
El proceso anterior generará a nivel visual una alerta y paralelamente el equipo habrá enviado las notificaciones configuradas (correo, SMS y llamada).

Internamente se han generado las siguientes líneas de comandos:

1. SMS: /usr/bin/ssh 10.0.7.195 "/bin/echo 'PFC: $HOSTNAME$ with IP=$HOSTADDRESS$ was $NOTIFICATIONTYPE$ at `date`' | /usr/local/bin/gnokii --sendsms $CONTACTPAGER$"

   Donde nagios toma como argumentos sus datos que tiene almacenados en la base de datos, dejando la notificación, según el ejemplo mostrado, de la siguiente manera:

   /usr/bin/ssh 10.0.7.195 "/bin/echo 'PFC: CORE6500-VM_MA_CORE_1 with IP=10.100.254.65 was DOWN at `date`' | /usr/local/bin/gnokii --sendsms 616109238"



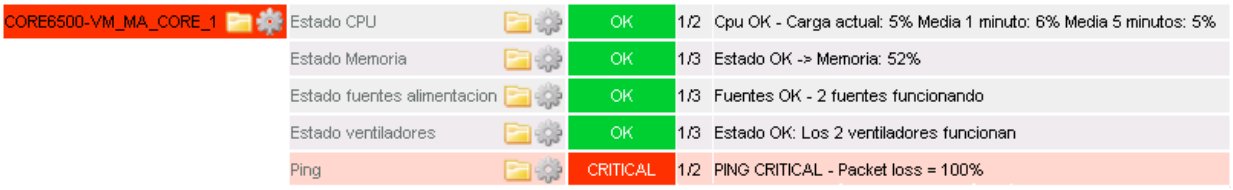
Prueba recepción SMS

2. Email: /usr/bin/ssh 10.0.7.195 '/usr/bin/mailx -r "Alert SCC <alert@mss.scc.com>" -s "PFC: $HOSTNAME$ $HOSTADDRESS$ Estado del host: $HOSTSTATE$" $CONTACTEMAIL$ </dev/null'

Igual que en el caso anterior de SMS, nagios pasará como parámetro las variables necesarias para que la línea de comando quede de la siguiente manera:

/usr/bin/ssh 10.0.7.195 '/usr/bin/mailx -r "Alert SCC <alert@mss.scc.com>" -s "PFC: CORE6500-VM_MA_CORE_1 10.100.254.65 Estado del host: DOWN" manolo.mg@gmail.com </dev/null'

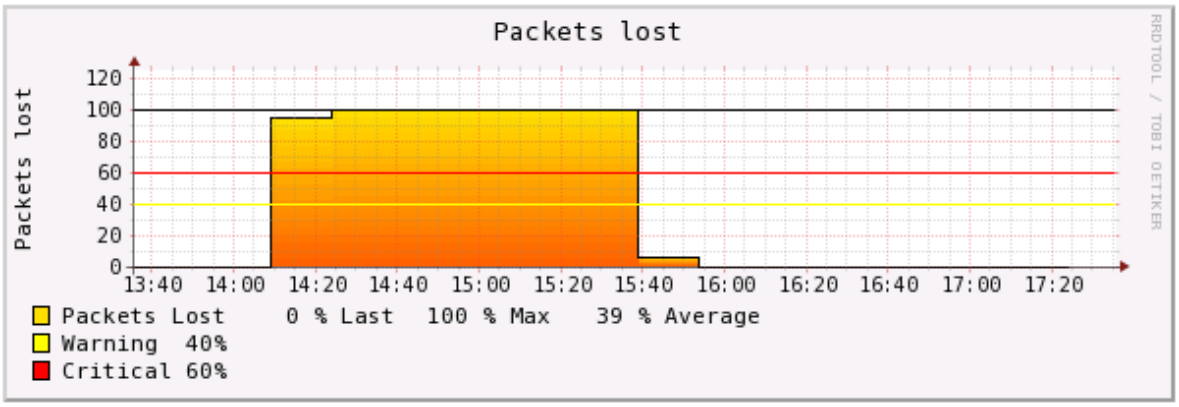
Posteriormente, se realizarán los demás checks que irán mostrando que todos los servicios están parados, debido a que la IP del equipo asociado no responde, poco a poco podremos ver:



Captura switch parado a nivel de ping

Y para todos los servicios sucesivamente, quedando cada uno de ellos en rojo.

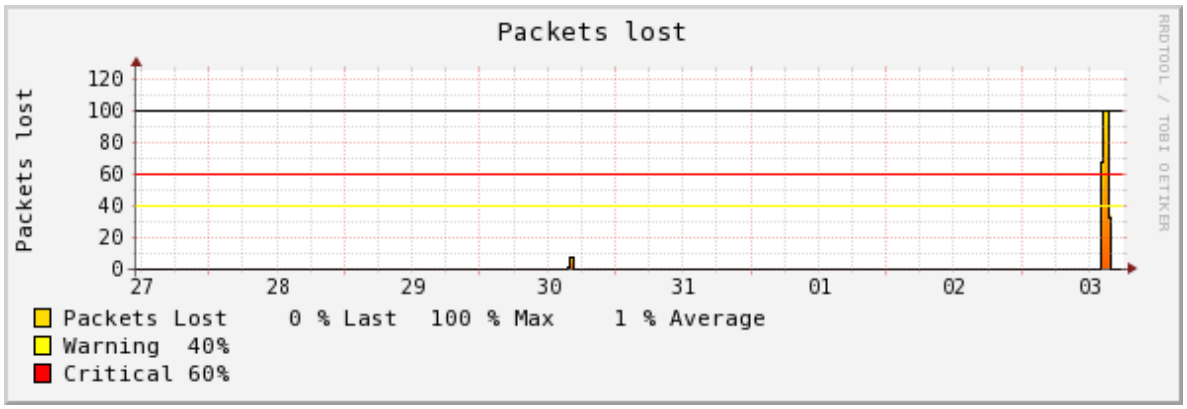
Ahora trataremos de localizar la parada en el histórico del PNP4Nagios, es decir, en las gráficas, de esta manera veremos una de las principales utilidades de la aplicación.



Paquetes perdidos del switch

Tal y como se observa, el equipo contestaba a ping correctamente hasta la parada programada, en cuanto el sistemas ha detectado el cambio de estado ha pasado a mostrar que el switch tenía un 100% de paquetes perdidos, debido a que no contestaba a ping.

Si nos alejamos más en el tiempo podemos ver este corte del servicio y mirar si se han producido otros, la siguiente imagen muestra el estado en los últimos 7 días, en los cuales se muestra el corte producido y un pequeño pico de nivel de paquetes perdidos (día 30), que en ningún momento supuso una caída del equipo.



Captura paquetes perdidos últimos 7 días

5. Anexo: Manual de usuario

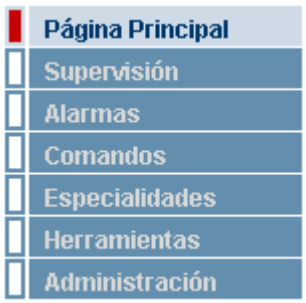
En este capítulo se verá cómo se configuran los servicios básicos a través de NagiosQL para dejar la aplicación funcional con las funciones básicas.

Se verá un ejemplo completo de:

- Manual de instalación de la aplicación.
- Configurar los comandos de notificación.
- Configuración comandos de servicio.
- Creación de un periodo de tiempo.
- Creación de un contacto.
- Creación de una plantilla de host.
- Creación de una plantilla de servicio.
- Creación de un host.
- Creación de un servicio.
- Aplicar cambios.
- Comprobación en interface de usuario Nagios.

En primer lugar se debe entrar a http://IP_Interna/nagiosql para entrar en la administración de la aplicación.

A partir de este menú se navegará por todas las opciones, tenemos:



- Supervisión: Se configurarán equipos y servicios.
- Alarmas: Apartado en el cual se configuran las notificaciones.
- Comandos: Submenú en el cual se crean los tipos de consulta (snmp o nrpe) y parámetros que se pasarán a la hora de configurar el servicio.
- Especialidades: No se requiere de este apartado para realizar una configuración básica.
- Herramientas: Apartado para realizar backups, exportaciones e importaciones y en la cual se aplican los cambios realizados.
- Administración: Gestión de usuarios, password, permisos etc.

Vistas las utilidades del menú empezamos la configuración, se verán los pasos de una manera lógica:

6.1 Manual de instalación de la aplicación

La aplicación está almacenada en un disco USB con formato del software “Clonezilla”, por lo que se debe instalar a través de este programa. Se deben seguir los siguientes pasos:

1. Introducir en el equipo IBM el CD de Clonezilla v1.2.11.23, preferentemente.
2. Reiniciar el sistema y arrancar desde la unidad de CD/DVD.
3. Dentro del menú del software Clonezilla ir dando a “siguiente” elegir instalar una imagen desde dispositivo extraíble USB.
4. Introducir el pen drive USB en cualquier puerto del servidor IBM.
5. Seleccionar la imagen que aparecerá en el disco externo.
6. Seguir pasos de instalación.
7. La instalación se completará.
8. Las credenciales de sistema son root/root.

Con los pasos anteriores tendremos la aplicación lista y preparada para realizar los siguientes pasos de configuración.

6.2 Configurar los comandos de notificación

Se deben de crear unas políticas con un nombre específico que realicen los envíos cuando ocurra algún evento en la red, para ello se configurará la petición ssh a través de la IP de NAT configurada y así poder asignarle a todo servicio o equipo esta notificación para que la ejecute en caso de error.

Para ello nos vamos al menú “Comandos” → “Definiciones” → “Agregar” y configuramos los siguientes comandos:

6.2.1 Notificación SMS

Hay que configurar dos comandos en cada caso, uno corresponderá a la notificación para el host y otra a la del servicio, ya que las variables que envía a través del SMS son diferentes y, en este caso, las del servicio llevan más información.

Hay que sustituir “X”s por la IP de NAT e “Y”s por el nombre que se quiera mostrar en las notificaciones.

Ejemplo de comando para notificación de Host por SMS

Notificación del host:

/usr/bin/ssh XXX.XXX.XXX.XXX. "/bin/echo 'YYY: $HOSTNAME$ with IP=$HOSTADDRESS$ was $NOTIFICATIONTYPE$ at `date`' | /usr/local/bin/gnokii --sendsms $CONTACTPAGER$"

Notificación del servicio:

/usr/bin/ssh XXX.XXX.XXX.XXX "/bin/echo 'YYY: $SERVICEDESC$ in $HOSTNAME$ with IP=$HOSTADDRESS$ was $SERVICESTATE$ at `date`' | /usr/local/bin/gnokii --sendsms $CONTACTPAGER$"

6.2.2 Notificación email

Al Igual que en la configuración de SMS, se deberá realizar un comando para equipo y otra para servicio:

Ejemplo de comando para notificación de Host por email

Notificación del host:

/usr/bin/ssh XX.XX.XX.XX '/usr/bin/mailx -r "Alert <alert@mss.scc.com>" -s "YYY: $HOSTNAME$ $HOSTADDRESS$ Estado del host: $HOSTSTATE$" $CONTACTEMAIL$ </dev/null'

Notificación del servicio:

/usr/bin/ssh XX.XX.XX.XX '/usr/bin/mailx -r "Alert <alert@mss.scc.com>" -s "YYY: $HOSTNAME$ $HOSTADDRESS$ Servicio $SERVICEDESC$: $SERVICESTATE$ - $SERVICEOUTPUT$" $CONTACTEMAIL$ </dev/null'

6.2.3 Notificación mediante llamada

Para el funcionamiento implementado de las llamadas solo es necesario un comando independientemente de si se trata de un equipo o servicio:

Ejemplo de comando para notificación por llamada

Notificación:
/usr/bin/ssh XX.XX.XX.XX "/bin/echo 'Llamada' | /usr/local/bin/gnokii --dialvoice $CONTACTPAGER$"

6.3 Configuración comandos de servicio

En el apartado anterior se exponían los comandos que se deben configurar para enviar notificaciones de alerta. Existen otro tipo de comandos y son los que necesitan los servicios (discos, particiones, servicios, memoria etc.) para que se comprueben en la aplicación.

Se pondrá un ejemplo y se expondrán en una tabla todos los creados en el proyecto que existe funcional.

Para ver un ejemplo veremos el de los discos de Windows y de un unix, que son los más usados.

6.3.1 Comando disco Windows SNMP

Para realizar el comando que compruebe los discos de Windows debemos crear el siguiente comando como por ejemplo, con el nombre "*check_snmp_disk_windows*":

/usr/bin/perl $USER1$/check_snmp_storage.pl -H $HOSTADDRESS$ -C $ARG1$ -m $ARG2$ -w 80 -c 90 –f

Desglosamos el comando para ver su funcionamiento:

- /usr/bin/perl: Ruta del binario de perl para que ejecute el script.

- $USER1$: Nos encontramos con la primera variable (**que estará en todo comando creado**), viene definida por la aplicación y apunta a la ruta en la que están todos los scripts almacenados.

```
# Sets $USER1$ to be the path to the plugins
$USER1$=/usr/lib64/nagios/plugins
```

Ruta variable $USER1$

- -H $HOSTADDRESS$: El "-H" indica que la siguiente variable que se pondrá es el nombre/IP del equipo, en todos los casos esta variable vendrá dado por nagios, es decir, no se deberá de especificar manualmente.
- -C $ARG1$: Nos encontramos el primer ARG (argumento) que habrá que especificar manualmente. En este caso "–C" especifica que el siguiente campo especificado será la comunidad configurada para el SNMP.
- -m $ARG2$: Es el segundo argumento que habrá que especificar manualmente. "-m" indica que el siguiente campo a especificar será la unidad que se requiera (disco windows) C, D, E o el que corresponda.
- -w 80: "-w" especifica que la siguiente variable es el umbral que si se traspasa mostrará un evento (que se enviará por SMS, email o llamada) y que se representará en nagios con un cambio de estado que requerirá de nuestra atención. Significa % de espacio de disco ocupado, en este caso.
- -c 90: "-c" especifica que la siguiente variable es el umbral que si se traspasa mostrará un evento crítico en nagios (que enviará por SMS, email o llamada), quiere decir % de uso de disco ocupado.
- -f: Este parámetro especifica que se deben de representar los datos para que se muestren los gráficos. Como ya se ha visto en capítulos anteriores, se usa para que se muestre la pipe "|" seguida de los datos a representar gráficamente.

El resultado de un ejemplo funcional sería:

/usr/bin/perl $USER1$/check_snmp_storage.pl -H IPsrv_windows -C ComunidadSNMP -m ^C -w 80 -c 90 –f

6.3.2 Comando disco Unix SNMP

A continuación se detalla un comando de monitorización para un disco de un sistema Unix, como se puede ver por su estructura, es idéntico al de Windows, la única diferencia que se aprecia es la siguiente:

- -r –m $ARG2$: "-r -m" especifica que el siguiente argumento que se debe introducir es el file system de Unix que se quiera monitorizar, pudiendo ser cualquiera de su tabla de particiones "/", "/var", "/tmp" etc.

$USER1$/check_snmp_storage.pl -H $HOSTADDRESS$ -C $ARG1$ -r -m $ARG2$ -w $ARG3$ -c $ARG4$ -f

El resultado de un ejemplo funcional sería:

$USER1$/check_snmp_storage.pl -H IPsrv_unix -C ComunidadSNMP -r -m / -w 80 -c 90 -f

6.3.3 Comando memoria UNIX NRPE

A continuación se muestra un ejemplo de la configuración de un comando de memoria RAM de un servidor Unix:

$USER1$/check_nrpe -n -H $HOSTADDRESS$ -c check_mem -t 120

En este caso no se debe de introducir ningún argumento, el único caso diferente a los otros ejemplos es saber identifica qué significa "–c check_mem –t 120".

- -c check_mem: "-c" especifica que el siguiente argumento será el script que se debe de ejecutar en el servidor al cual estamos realizando la petición (ver página 53).
- -t 120: "-t" especifica el tiempo de timeout en caso de que la petición no tenga éxito.

El resultado de un ejemplo funcional sería:

$USER1$/check_nrpe -n -H IPsrv_Unix -c check_mem -t 120

Una vez vistos algunos de los ejemplos más utilizados en cualquier sistema, se exponen a continuación todos los creados para el proyecto expuesto, con todos sus argumentos y estructura que se debe de crear:

Nombre comando	Definición comando
Check_particionesSCC	check_nrpe!check_particionesSCC!"-w 80 -c 90 -l solo (cambiar)"
check_nrpe_generico	$USER1$/check_nrpe -n -H $HOSTADDRESS$ -c $ARG1$ -t 120
check_servicio_windows	$USER1$/check_snmp_win.pl -H $HOSTADDRESS$ -C $ARG1$ -n $ARG2$
check_snmp_disk_AIX	/usr/bin/perl $USER1$/check_snmp_storage_aix.pl -H $HOSTADDRESS$ -C $ARG1$ -r -m $ARG2$ -w $ARG3$ -c $ARG4$ -f
check_http_con_dominio_url_y_texto	$USER1$/check_http -H $ARG1$ --port=$ARG2$ --url=$ARG3$ --string=$ARG4$ -w 10 -c 15 -t 65
check_http_con_dominio	$USER1$/check_http -H $ARG1$ --port=$ARG2$ -w 10 -c 15 -t 65
host-notify-by-SMS	/usr/bin/ssh XX.XX.XX.XX. "/bin/echo 'XXX: $HOSTNAME$ with IP=$HOSTADDRESS$ was $NOTIFICATIONTYPE$ at `date`' | /usr/local/bin/gnokii --sendsms $CONTACTPAGER$"
service-notify-by-SMS	/usr/bin/ssh XX.XX.XX.XX "/bin/echo 'XXX: $SERVICEDESC$ in $HOSTNAME$ with IP=$HOSTADDRESS$ was $SERVICESTATE$ at `date`' | /usr/local/bin/gnokii --sendsms $CONTACTPAGER$"

process-host-perfdata-file	/bin/mv /usr/local/pnp4nagios/var/host-perfdata /usr/local/pnp4nagios/var/spool/host-perfdata.$TIMET$
process-service-perfdata-file	/bin/mv /usr/local/pnp4nagios/var/service-perfdata /usr/local/pnp4nagios/var/spool/service-perfdata.$TIMET$
notify-service-by-email	/usr/bin/ssh XX.XX.XX.XX '/usr/bin/mailx -r "Alert SCC <alert2@mss.scc.com>" -s "MAG: $HOSTNAME$ $HOSTADDRESS$ Servicio $SERVICEDESC$: $SERVICESTATE$ - $SERVICEOUTPUT$" $CONTACTEMAIL$ </dev/null'
oracle_tablespace_usage	/usr/bin/sudo su - nagios -c "$USER1$/check_oracle_health $ARG1$ $ARG2$ $ARG3$ --warning=$ARG4$ --critical=$ARG5$"
check_RAM_AIX	$USER1$/check_nrpe -n -H $HOSTADDRESS$ -c check_RAM -t 120
notify-host-by-email	/usr/bin/ssh XX.XX.XX.XX '/usr/bin/mailx -r "Alert SCC <alert2@mss.scc.com>" -s "MAG: $HOSTNAME$ $HOSTADDRESS$ Estado del host: $HOSTSTATE$" $CONTACTEMAIL$ </dev/null'
check-host-alive	$USER1$/check_ping -H $HOSTADDRESS$ -w 3000.0,80% -c 5000.0,100% -p 5
check_udp	$USER1$/check_udp -H $HOSTADDRESS$ -p $ARG1$ $ARG2$
check_tcp	$USER1$/check_tcp -H $HOSTADDRESS$ -p $ARG1$ $ARG2$
check_ssh	$USER1$/check_ssh $ARG1$ $HOSTADDRESS$
check_snmp	$USER1$/check_snmp -H $HOSTADDRESS$ $ARG1$
check_smtp	$USER1$/check_smtp -H $HOSTADDRESS$ $ARG1$
check_nt	$USER1$/check_nt -H $HOSTADDRESS$ -p 12489 -v $ARG1$ $ARG2$ -t 240 -u
check_ping	$USER1$/check_ping -H $HOSTADDRESS$ -w $ARG1$ -c $ARG2$ -p 5
check_pop	$USER1$/check_pop -H $HOSTADDRESS$ $ARG1$
check_local_mrtgtraf	$USER1$/check_mrtgtraf -F $ARG1$ -a $ARG2$ -w $ARG3$ -c $ARG4$ -e $ARG5$
check_local_procs	$USER1$/check_procs -w $ARG1$ -c $ARG2$ -s $ARG3$
check_local_swap	$USER1$/check_swap -w $ARG1$ -c $ARG2$
check_local_users	$USER1$/check_users -w $ARG1$ -c $ARG2$
check_local_disk	$USER1$/check_disk -w $ARG1$ -c $ARG2$ -p $ARG3$
check_local_load	$USER1$/check_load -w $ARG1$ -c $ARG2$
check_dhcp	$USER1$/check_dhcp $ARG1$
check_ftp	$USER1$/check_ftp -H $HOSTADDRESS$ $ARG1$
check_hpjd	$USER1$/check_hpjd -H $HOSTADDRESS$ $ARG1$
check_http	$USER1$/check_http -H $HOSTADDRESS$ -u $ARG1$ -p $ARG2$
check_imap	$USER1$/check_imap -H $HOSTADDRESS$ $ARG1$
check_interficies	$USER1$/check_ifoperstatus -H $HOSTADDRESS$ -C $ARG1$ -k $ARG2$
check_cpu_windows	/usr/bin/perl $USER1$/check_win_snmp_cpuload.pl $HOSTADDRESS$ $ARG1$ $ARG2$ $ARG3$
check_windos_disk	/usr/bin/perl $USER1$/check_win_snmp_storage.pl $HOSTADDRESS$ $ARG1$ $ARG2$ $ARG3$ $ARG4$
check_cpu_windows3	$USER1$/check_snmp_load.pl -H $HOSTADDRESS$ -C $ARG1$ -w $ARG2$ -c $ARG3$
check_snmp_uptime	$USER1$/uptime_by_snmp.sh $HOSTADDRESS$ $ARG1$

check_snmp_disk_windows	/usr/bin/perl $USER1$/check_snmp_storage.pl -H $HOSTADDRESS$ -C $ARG1$ -m $ARG2$ -w 80 -c 90 -f
check_cpu_aix	/usr/bin/perl $USER1$/check_aix_snmp_cpuload.pl $HOSTADDRESS$ $ARG1$ $ARG2$ $ARG3$
check_trafico	$USER1$/check_snmp_trafic.pl $HOSTADDRESS$ $ARG1$ $ARG2$ " < $ARG3$ " " > $ARG4$ "
check_interficies_libres	/usr/bin/perl $USER1$/check_cisco_freeInterfaces.pl -H $HOSTADDRESS$ -C $ARG1$ -d $ARG2$ -e
check_daemons_unix	/usr/bin/perl $USER1$/check_exit_status.pl -s $ARG1$
check_snmp_disk_LINUX	$USER1$/check_snmp_storage.pl -H $HOSTADDRESS$ -C $ARG1$ -r -m $ARG2$ -w $ARG3$ -c $ARG4$ -f
check_SWAP_AIX	$USER1$/check_nrpe -n -H $HOSTADDRESS$ -c check_SWAP -t 120
check_CPU_Unix	$USER1$/check_nrpe -n -H $HOSTADDRESS$ -c check_load -t 120
check_swap_Linux	$USER1$/check_nrpe -n -H $HOSTADDRESS$ -c check_swap -t 120
check_snmp_RAM_windows	/usr/bin/perl $USER1$/check_snmp_storage.pl -H $HOSTADDRESS$ -C $ARG1$ -m ^Physical -w $ARG2$ -c $ARG3$ -f
oracle_connection_time	/usr/bin/sudo su - nagios -c "$USER1$/check_oracle_health $ARG1$ $ARG2$ --warning=$ARG3$ --critical=$ARG4$"
check_CPU_Solaris	$USER1$/check_nrpe -n -H $HOSTADDRESS$ -p 5665 -c check_load
check_SWAP_Solaris	$USER1$/check_nrpe -n -H $HOSTADDRESS$ -p 5665 -c check_swap
check_RAM_Solaris	$USER1$/check_nrpe -n -H $HOSTADDRESS$ -c check_mem -t 120
check_RAM_Solaris2	$USER1$/check_nrpe -n -H $HOSTADDRESS$ -p 5665 -c check_mem
check_nt_counter	$USER1$/check_nt -H $HOSTADDRESS$ -p 12489 -v $ARG1$ $ARG2$ -t 240
host-llamame	/usr/bin/ssh 10.0.7.195 "/bin/echo 'UPF: $HOSTNAME$ with IP=$HOSTADDRESS$ was $NOTIFICATIONTYPE$ at `date`' | /usr/local/bin/gnokii --dialvoice $CONTACTPAGER$"
service-llamame	/usr/bin/ssh 10.0.7.195 "/bin/echo 'XXX: $SERVICEDESC$ in $HOSTNAME$ with IP=$HOSTADDRESS$ was $SERVICESTATE$ at `date`' | /usr/local/bin/gnokii --dialvoice $CONTACTPAGER$"
check_undo_AIX	$USER1$/check_undo_tbsp.sh $ARG1$
check_bloqueos	$USER1$/check_block_sessions.sh $ARG1$
check_tcp_remoto	$USER1$/check_tcp -H $ARG1$ -p $ARG2$
check_snmp_disk_windows_alto	/usr/bin/perl $USER1$/check_snmp_storage.pl -H $HOSTADDRESS$ -C $ARG1$ -m $ARG2$ -w 85 -c 95 -f
check_vrrp	$USER1$/check_snmp_vrrp.pl -H $HOSTADDRESS$ -C n3tw0rk -s $ARG1$ -T nokia
chcek_mysql_health	$USER1$/check_mysql_health --hostname $HOSTNAME$ --username $ARG1$ --password $ARG2$ --mode $ARG3$
check_snmp_disk_windows_muyAlto	/usr/bin/perl $USER1$/check_snmp_storage.pl -H $HOSTADDRESS$ -C $ARG1$ -m $ARG2$ -w 90 -c 95 -f
check_Cisco_CPU	$USER1$/check_CISCO_CPU.pl -H $HOSTADDRESS$ -C

	$ARG1$ -w $ARG2$ -c $ARG3$
check_Cisco_InterfaceActiva	$USER1$/check_CISCO_interface.pl -H $HOSTADDRESS$ -C $ARG1$ $ARG2$
check_Cisco_Memoria	$USER1$/check_CISCO_Memoria.pl -H $HOSTADDRESS$ -C $ARG1$ -w $ARG2$ -c $ARG3$
check_Cisco_Modulos	$USER1$/check_CISCO_Modulos.pl -H $HOSTADDRESS$ -C $ARG1$
check_Cisco_Ventiladores	$USER1$/check_CISCO_Ventilador.pl -H $HOSTADDRESS$ -C $ARG1$
check_Cisco_Fuentes	$USER1$/check_CISCO_Fuentes.pl -H $HOSTADDRESS$ -C $ARG1$ -w $ARG2$ -c $ARG3$
check_snmp_disk_AIX_errores	/usr/bin/perl $USER1$/check_snmp_storage_sin.pl -H $HOSTADDRESS$ -C $ARG1$ -r -m $ARG2$ -w $ARG3$ -c $ARG4$ -f
check_swap_Linux_noSSL	$USER1$/check_nrpe -H $HOSTADDRESS$ -c check_swap -t 120
check_CPU_Unix_noSSL	$USER1$/check_nrpe -H $HOSTADDRESS$ -c check_load -t 120
check_RAM_Linux_noSSL	$USER1$/check_nrpe -H $HOSTADDRESS$ -c check_mem -t 120
check_snmp_disk_windows_critico	/usr/bin/perl $USER1$/check_snmp_storage.pl -H $HOSTADDRESS$ -C $ARG1$ -m $ARG2$ -w 96 -c 98 -f
check_datastore	$USER1$/check_esx_datastore $HOSTADDRESS$ $ARG1$ $ARG2$ $ARG3$
check_multi	$USER1$/check_multi $ARG1$ $ARG2$ $ARG3$ $ARG4$
check_esx	$USER1$/check_esx3.pl -D $ARG1$ -H $HOSTADDRESS$ -u $ARG2$ -p $ARG3$ -l $ARG4$ -s $ARG5$ -w $ARG6$ -c $ARG7$

6.4 Creación de un periodo de tiempo

Para crear una plantilla con las horas a las cuales queremos recibir las notificaciones debemos irnos a Menú → Alarmas → Periodos de tiempo.

En este proyecto se ha requerido que las notificaciones se envíen las 24 horas del día los 7 días de la semana, así pues se ha creado de la siguiente manera:

Configuración de horario de alarmas

Se pueden configurar diferentes horarios dependiendo de nuestras necesidades, como por ejemplo sólo recibirlas los fines de semana, recibirlas únicamente en horario laboral, etc.

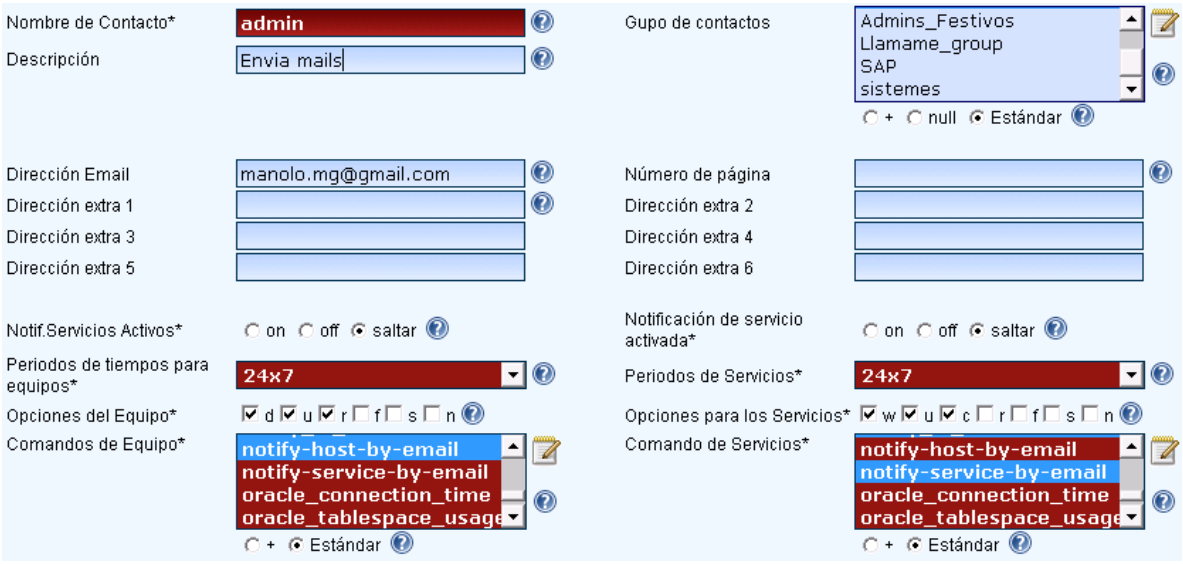
También se pueden configurar diferentes plantillas de horarios si se desea configurar uno para cada equipo.

º

6.5 Creación de un contacto

En este punto se verá cómo crear una dirección de email para las notificaciones, en el caso de los SMS o llamadas será de la misma manera.

Vamos al menú Alarmas → Datos de contacto



Captura de datos de contacto

Campos requeridos:

- Nombre: Se especifica un nombre para la notificación.
- Dirección de Email: Se especifica la dirección de correo electrónico.
- Periodos de tiempo para equipos: Se indica la franja de horas a las que se desea recibir los correos de problemas en los servidores.
- Comando de equipo: Se especifica el comando definido en apartados anteriores.
- Periodos de servicio: Se indica la franja horaria en la cual se quiere recibir notificaciones de los servicios asociados a los servidores cuando generen algún evento.
- Comando de servicio: Se especifica el comando definido en apartados anteriores.
- Opciones del equipo/servicios: Se indica el tipo de eventos que se quiera recibir, para entenderlos:
  - D: Significa “Down” y se activa en caso de que se requiera una alerta cuando el servidor deje de responder a ping (imprescindible activar).
  - U (host): Significa “Unreachable” y se activa en caso de que se quieran recibir notificaciones de los equipos a los cuales se pierde la señal por tema de routing (no necesaria).
  - R: Significa “Recovery” y se debe activar si se desea recibir la notificación de que el servidor vuelve a estar activo después de haberse “caído” (Recomendable activar).
  - F: Significa “Flapping” y se activa si se desea descartar si un servidor cambio de estados continuamente (No recomendable).

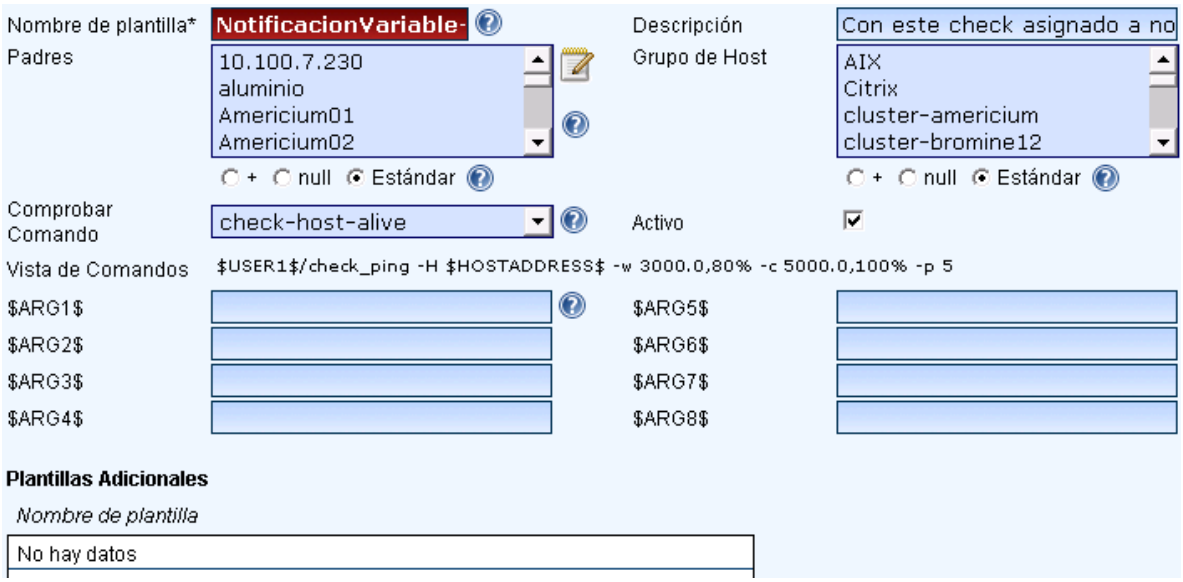
- S: Significa “Send” y se activa si se desea recibir notificación de cuando se activa una “pausa” en la monitorización de un servidor (No necesaria).
- W: Significa “Warning” y se habilita si se desea recibir una notificación de este tipo de un servicio cuando se genere (muy aconsejable activar).
- C: Significa “Critical” y se activa si se desea recibir eventos graves en los servicios (imprescindible activar).
- U (servicio): Significa “Unknown” y se habilita si se desean habilitar estados “desconocidos” para los servicios. Esto ocurre cuando, por ejemplo, deja de funcionar el protocolo SNMP (recomendable activar).

6.6 Creación de una plantilla de host

Este apartado en uno de los puntos más importantes qué definirá el comportamiento de la aplicación, en él se podrá ver cómo se configura el intervalo en que se comprueba que los equipos funcionen correctamente, el periodo de las notificaciones, de los recordatorios de las notificaciones, etc.

Esta plantilla, para el proyecto implementado, se ha asignado a todos los equipos, ya que todos tienen la misma prioridad y criticidad de comprobación.

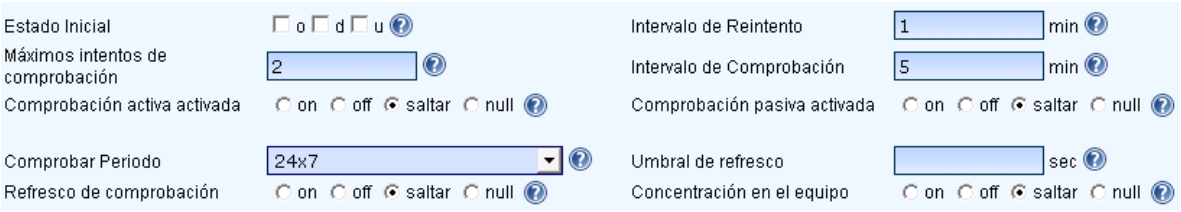
Se empieza su configuración en Menú → Supervisión → Plantilla de host y configuramos los siguientes datos:



Ejemplo de plantilla de host

- Nombre de plantilla: Nombre que se asociará al equipo, tal y como se verá en el apartado 4.7 posterior.
- Comprobar comando: Se especifica siempre “check-host-alive”, es un comando que ya viene creado en nagios y es el que comprueba que el equipo esté activo simplemente generando tráfico ICMP hacia él, es decir, haciéndole un ping cada X minutos.

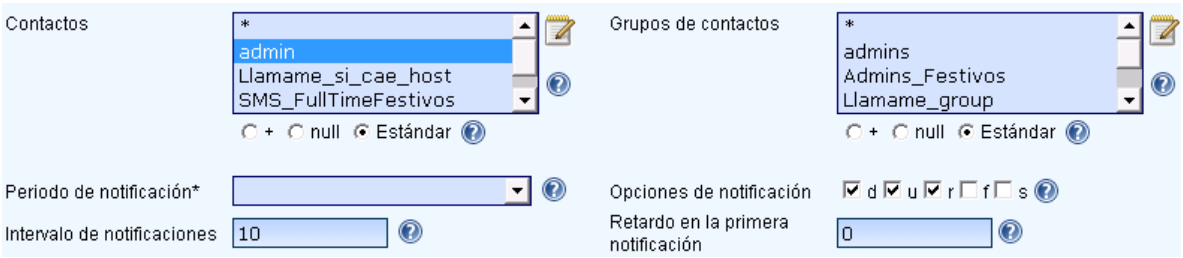
Cambiamos de pestaña y nos dirigimos a la que indica “Comprobar opciones”, en la cual se especificarán los siguientes parámetros:



Ejemplo de comprobaciones de servidor

- Máximos intentos de comprobación: Es el número de veces que se comprobará el estado del equipo antes de que se envíe el evento generado, por ejemplo:
  1- Tenemos un servidor parado.
  2- Llega el momento de realizar su comprobación y el sistema ve que está parado en su 1ª comprobación *(ver diagrama de flujo en página 37).*
  3- Guarda el estado y entra de nuevo en cola.
  4- Cuando vuelve a comprobarse y se detecta de nuevo que el equipo está parado, es en este momento cuando se enviará la notificación y el sistema generará la alerta. Se pueden poner tantos intentos como se desee, teniendo en cuenta que contra más intentos más se retardará la generación del evento de alerta.
- Comprobar Periodo: Se asigna el periodo de tiempo de comprobación en el cual se quiere que se compruebe el estado.
- Intervalo de Reintento: Es el tiempo que se adjudica para, en este caso, la segunda comprobación y si hubiese, para posteriores.
- Intervalo de Comprobación: Establece el tiempo, en minutos, entre la comprobación de estados de cada equipo. En este caso, indica que el estado de cada servidor se comprobará cada 5 minutos. Se recomienda un valor bajo, ya que el estado de los servidores se debe de comprobar con frecuencia.

Por último nos dirigiremos a la pestaña “Opciones Alarmas”, en ella:



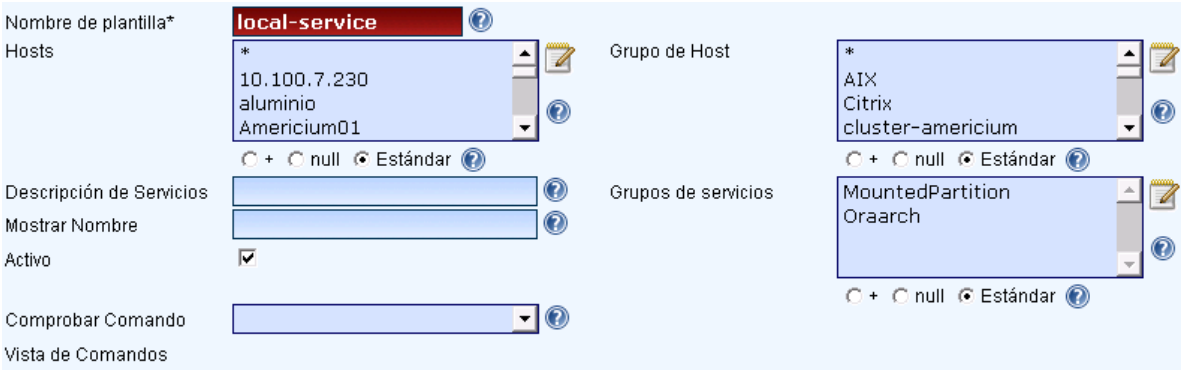
Configuración de Opciones de Alarma

- Contactos: Se debe especificar el “contacto” creado en puntos anteriores (ver apartado 4.4 de este capítulo).
- Intervalo de notificaciones: Se trata de un tiempo definido en minutos, en este caso 10 minutos, y significa que cada 10 minutos se enviará una repetición del evento generado como alerta en caso de que no se solucione.
- Opciones de notificación: Queremos que nos llegue cuando “d” (se para un equipo), u (se pierde el routing hacia un equipo) o “r” (se restablece un equipo).

6.7 Creación plantilla de servicios

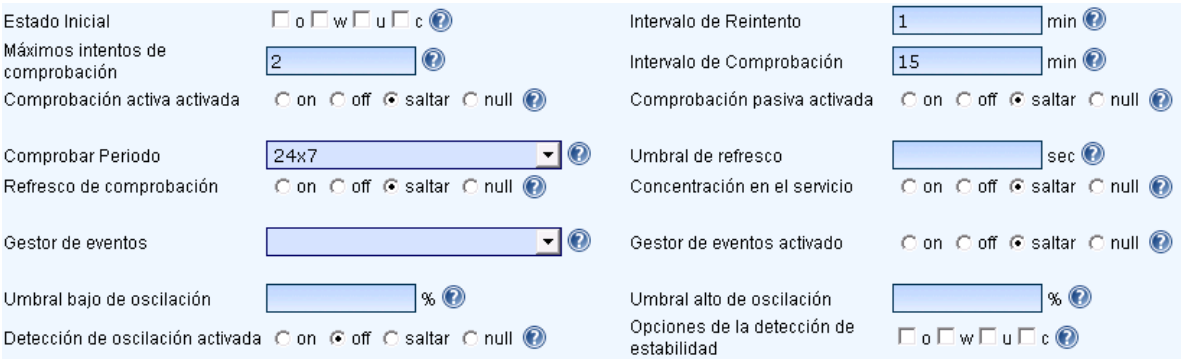
Se realizará los mismo que en el apartado anterior, pero esta vez orientado a los servicios:

Para ello Menú → Supervisión → Plantillas de Servicios y nos quedamos en la pestaña “Configuración Común”.



Ejemplo de plantilla de servicio

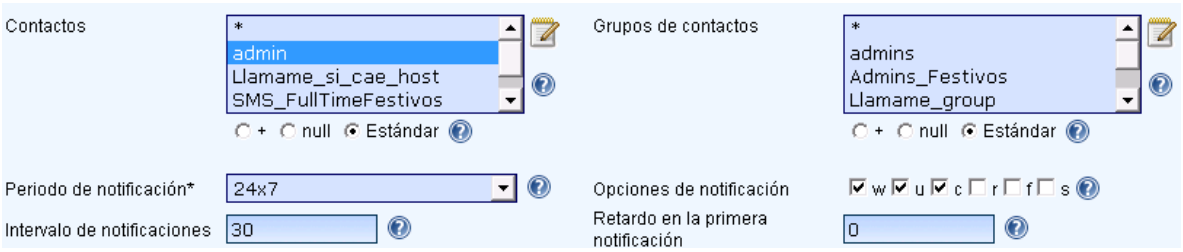
Solo debemos configurar el nombre del servicio y pasar a la siguiente pestaña “Comprobar Opciones”.



Ejemplo de comprobación de servicio

Se configura de igual manera que se ha visto en el punto anterior con el servidor, exceptuando el “Intervalo de comprobación”, ya que en este caso se ha configurado a 15 minutos, esto es debido a que no se prioriza tanto la comprobación de un servicio como de un servidor y 15 minutos es un tiempo más que aceptable para cada intervalo de comprobación.

Seguidamente pasamos a la pestaña “Opciones Alarmas” y lo configuramos adaptado a un servicio con las mismas directrices que se ha hecho con el servidor.



Ejemplo de alarmas en servicio

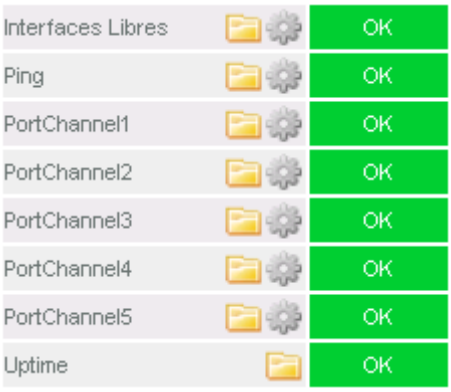
Por último entramos en la pestaña “Ajustes Adicionales” y lo configuramos de la siguiente manera:

Ejemplo ajustes servicios

En el apartado anterior es donde se configurará que el PNP4Nagios y la documentación aparezca en cada servicio, para ello debemos configurarlo de la siguiente manera:

- Notas: */dokuwiki/doku.php?id=procedimientos:$HOSTNAME$:$SERVICEDESC$*
- Notas URL: */pnp4nagios/index.php/graph?host=$HOSTNAME$&srv=$SERVICEDESC$' class='tips' rel='/pnp4nagios/popup?host=$HOSTNAME$&srv=$SERVICEDESC$*

Para que los iconos se muestren en la aplicación.



Enlaces documentación y PNP4Nagios

De esta manera nos habrá aparecido, tal y como se muestra en la imagen anterior, un símbolo de una carpeta en todo servicio o equipo que requiera documentación (DocuWiki) y la imagen de la "tuerca" en todo host o servicio que tenga la necesidad de registrar sus valores en un gráfico (PNP4Nagios).

6.8 Creación de un host

En este apartado se verá la creación de un equipo para su monitorización, para ello vamos a Menú → Supervisión → Host y lo rellenamos, por poner un ejemplo, de la siguiente manera:

Creación de un host

Se deben rellenar los datos siguientes:

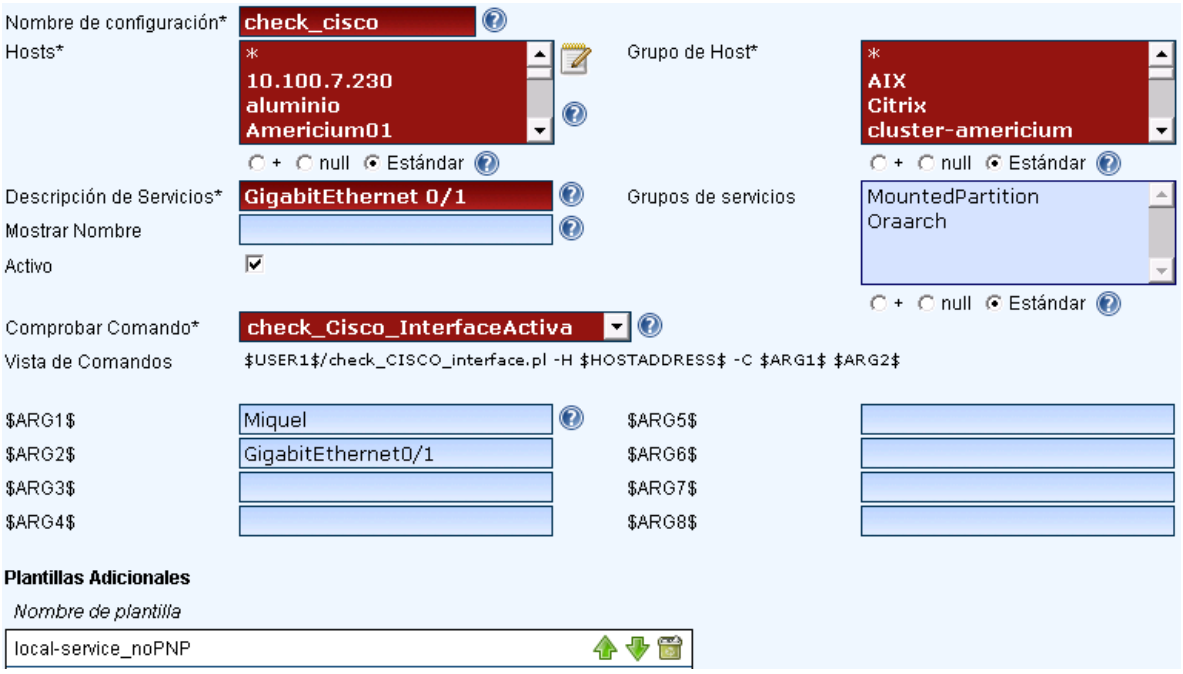
- Nombre de Host: Nombre con el cual se visualizará en nagios.
- Descripción: Pequeña descripción de qué hace o a qué se dedica el servidor.
- Dirección: IP o nombre del equipo (si está en DNS) para hacer las peticiones.
- Grupo de Host: Se pueden agrupar por grupos para su agrupación (Windows, Linux, comunicaciones, exchange, etc.).
- Nombre de plantilla: En este apartado se asigna la configuración que se ha definido anteriormente, con tipos de notificaciones, horarios, direcciones de envío y número de teléfono, etc.

En las demás pestañas se pueden configurar los tiempos de comprobación y demás que se han visto en la creación de la plantilla, por si se quiere cambiar en algún caso en particular.

6.9 Creación de un servicio

En este punto se definirán los servicios que posteriormente deberán ser asociados a los equipos, es decir, se crearán los discos que se asociarán a un Windows, los files systems que se asociarán a un Linux, las interfaces que se asociarán a un switch, etc.

Para ello vamos a Menú → Supervisión → Servicios.



Ejemplo de creación de un servicio

Los campos requeridos son:

- Nombre de configuración: Se usa para definir conjuntos de servicios, en ningún caso será mostrado en Nagios.
- Hosts: Se indica en qué equipos será asociado este servicio, pueden ser a varios o únicamente a uno.
- Grupo de host: Se puede aplicar a un grupo entero, por ejemplo, un disco "C:" se puede aplicar al grupo Windows con total seguridad y se aplicará a todos los equipos definidos en este grupo.
- Comprobar comando: Es la opción más importante de todas, se asigna el comando, creado anteriormente y que será el que solicite los argumentos y muestre una salida coherente. En este caso, al tratarse de una interface de un switch, hemos escogido el comando "check_cisco_InterfaceActiva", el cual solicita dos argumentos (ARG1 y ARG2).
- Argumentos 1 a n: En estas casillas se indican los argumentos necesarios para que el comando tenga efecto, se deben especificar el mismo número de argumentos que defina el comando, en este caso nos solicitaba la comunidad SNMP y el nombre de la interface.

Ya tenemos listo un equipo con un servicio funcional con las siguientes características en conjunto:

- Notificará las 24 horas del día.
- Notificará los 7 días de la semana.
- Notificará correos a la dirección de "manolo.mg@gmail.com", a la espera de configurar otras notificaciones si se requiriese.
- Notificará, para el servidor, en caso de que se pierda la comunicación con él, se restablezca la comunicación o se deje de llegar por tema de "routing".
- Notificará, para el servicio, en caso de que se genere una alerta "warning", una "critical" o no respondan los servicios.

6.10 Aplicar cambios

Una vez realizado cualquier cambio, se deben de aplicar para que se muestren en la aplicación Nagios. Se deben seguir los siguientes pasos:

1- Menú → Herramientas → Control Nagios.
2- Escribimos datos de configuración.
3- Escribimos datos adicionales.
4- Comprobar los ficheros de configuración.
5- Reiniciar Nagios: Esto realiza un "/etc/init.d/nagios restart" en la aplicación y será el paso que muestre los cambios.

6.11 Comprobación en interface de usuario Nagios

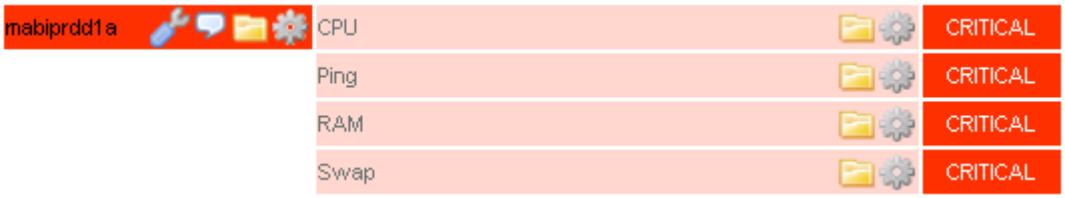
Cuando se haya acabado la configuración de todo equipo y servicio a través de NagiosQL, los cambios se mostraran en Nagios, debemos acceder ara ello a la http://IP_Interna/nagios/

Una vez dentro se visualizarán los equipos según su estado:

1- Equipos correctos: Los equipos sin ningún problema registrado de muestran en color verde con estado OK.

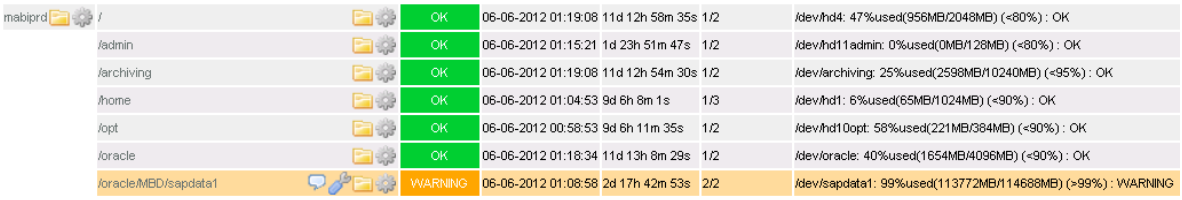
Ejemplo de equipo en estado "OK"

2- Equipos parados: Los equipos que están parados por cualquier motivo se muestran en rojo con el mensaje "Critical" en sus servicios.



Ejemplo de servidor parado

3- Servicios con problemas: Lo servicios con problemas dentro de un servidor activo se muestran en color naranja con el evento que haya generado y una descripción del problema.



Equipo en estado "Ok" con servicios alertados

7. Conclusiones y mejoras

En este capítulo se expondrán las mejoras y conclusiones a las cuales se han llegado a los largo de todo el desarrollo del proyecto, detallando los aspectos más importantes que se han estudiado y descubierto ante los problemas que han ido apareciendo.

En la primera idea de realizar el proyecto se tenía claro que lo que se iba a desarrollar era la monitorización de una red.

En esta idea se planteó inicialmente el desarrollo para sistemas únicamente de comunicaciones, mirando el estado de los equipos que comunican delegaciones y que conectan los servidores internos de una delegación. Debido a este planteamiento inicial, solo se iban a estudiar routers y switches, limitando mucho lo que una infraestructura de red aporta a nivel de sistemas, ya que el estudio se limitaba solo al análisis de una tecnología entre toda la diversidad que se puede insertar a la hora del diseño funcional de una red empresarial.

A medida que el proyecto avanzó se planteó la posibilidad de implementar en el sistema de monitorización todo equipo de comunicaciones y sistemas que existía en una red, con el aumento de complejidad que suponía para el estudio de cada tipo de sistema.

Las mayores dificultades encontradas a la hora de desarrollar el proyecto han sido las siguientes:

El primer lugar se destacaría la necesidad del estudio de cada equipo individualmente, que en total han sido unos 400 aproximadamente y distribuidos en 5 entornos principales como han sido Windows, Linux, englobando Solaris y AIX, y comunicaciones. Solo se disponía de la funcionalidad de cada equipo, es decir, si se trataba de un servidor DNS, servidor web, servidor SAP, etc. y a partir de esa información se debía extraer todo dato relevante del equipo que indicara cualquier tipo de problema que pudiese generar, por poner varios ejemplos encontrados en cada entorno:

1. Servidor web Linux: No basta con controlar los datos de sus file systems y saber si se llenan. Al tratarse de un servidor web que recibirá peticiones http, debemos controlar que el número de sesiones web no crezca hasta un número insostenible para el equipo. Además para saber el máximo número de sesiones permitidas se le deben realizar unas pruebas de “stress” al servidor. En este caso se ha concluido que cuando el número de sesiones aumentaba entorno a las 500 conexiones, la RAM consumida por el equipo llegaba al 100% y posteriormente el SWAP de equipo también llegaba al 100% impidiendo que se liberasen las sesiones web creadas. Por lo tanto se deben controlar los procesos http.

Contador procesos http

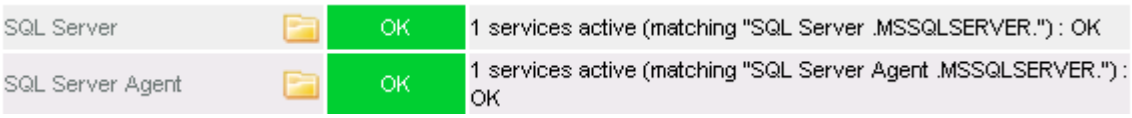
2. Servidor Oracle Solaris: En casi la totalidad de los equipos Solaris existen bases de datos de Oracle. Ocurre lo mismo que en el caso anterior, no es suficiente comprobar el equipo a nivel de sistema, sino que lo más importante es comprobar el funcionamiento de la aplicación, para ello se mira el tiempo de conexión a la base de datos a través de una consulta SNMP remota y es necesario crear un usuario para realizarla con este fin.

Conexión a BBDD con usuario DBSNMP

3. Servidor SAP AIX: En los equipos AIX que corre un SAP dentro, debemos vigilar con especial atención los llamados "oraarch" que son los logs de transacciones que se realizan en dicha aplicación. Son pequeñas líneas de texto plano que se generan cada vez que un usuario interactúa con la aplicación, de tal manera que se generan miles de líneas y ocupan un espacio muy elevado de disco dentro del servidor. Con la infraestructura estudiada, no se disponía de disco suficiente para aguantar un día entero de estos logs y si se llenan estos file system se para la aplicación, por lo que se deben realizar backups cada X horas, así que es imprescindible ver que el agente de backup esté activado, además del file systems que contiene estos "oraarch", para no darnos cuenta de que dicho backup no se ha hecho una vez ya ha fallado.

Comprobación puertos aplicación de backup

4. Servidores SQL Windows: En los equipos Windows que soportan base de datos SQL se deben mirar todos los servicios con descripción de SQL para revisar el estado de la aplicación, simplemente se ha mirado que exista un servicio activo que contenga el string "SQL".



Servicios SQL activos

5. Equipos de comunicaciones Cisco: En los sistemas Cisco no existe nada que diferencie a un equipo de otro, es decir, en todos se mirará la memoria e interfaces por igual, junto con sus componentes hardware.

En general los sistemas más difíciles de descubrir y analizar todo su funcionamiento han acabado siendo los Unix. Windows, en mi opinión, es un sistema más sencillo a nivel de usuario y administrador, mucho más visual y fácil de analizar sus aplicaciones instaladas.

Lo más fácil de implantar han sido los sistemas Cisco, simplemente instalando el SNMP a través de un sencillo comando tienes las configuración realizada.

En segundo lugar, se destacaría la necesidad de implementar la aplicación que recopilase todos los programas y los unificara para tenerlo todo centralizado en un único equipo e interface. Se debía crear una utilidad, con libre elección de software, de una aplicación sencilla de controlar tanto para un administrador de sistemas como para un usuario con conocimientos bajos-medios en los entornos. La mayor dificultad podría haberse dado en el hecho de no haber seleccionado bien ni las utilidades ni los protocolos para poder extraer de cada equipo todo lo requerido.

Si este proyecto se quiere implantar en un futuro en otra red de sistemas, se puede tener instalado el Sistema Operativo con todas las utilidades en un máximo de 15-20 minutos, lo que ahorrará mucho trabajo de investigación y desarrollo en hacer una buena elección de aplicaciones.

Al final y una vez realizado todo el proyecto, a la espera de futuras mejoras, se puede afirmar que todas las decisiones han sido acertadas y recogen las necesidades propuestas en el inicio del proyecto.

En tercer lugar se destacaría la problemática de distribuir o aplicar todo script necesario a los cerca de 400 equipos estudiados. Este apartado se ha concluido con más de 4.800 servicios monitorizados y la implantación en la imagen de múltiples checks que ya para futuras implantaciones en otras redes facilitarán mucho el avance de un nuevo proyecto.

Los checks SNMP resultan muy fáciles de utilizar ya que se ejecutan desde el servidor central IBM y no necesita de distribución alguna, en cambio, tiene el inconveniente de que se debe configurar e instalar en cada equipo para que acepte las peticiones, lo que se ha realizado en un total de unos 160 equipos. A nivel de monitorizar servicios es sencillo y rápido, teniendo en cuenta que es el único protocolo usado en Windows:

- Todo equipo Windows tiene una unidad “C:”.
- En todo sistema Windows la memoria (RAM, SWAP y CPU) se comprueba de la misma manera.
- Los servicios son muy sencillos de comprobar, simplemente debemos ir al listado de servicio donde se nos muestran en claro los que están activos y funcionando en el sistema.

Los checks NRPE han resultado más laboriosos de distribuir y configurar, ya se requiere tener cada script localmente en los servidores Unix, que han supuesto un número aproximado de unos 220.

En todos ellos se ha realizado un “scp” (Secure Copy), que es un protocolo de copia remota de ficheros o archivos, desde una máquina que ya tuviese estos scripts hacia una remota que los necesitase. Igualmente, se ha aplicado este procedimiento para configurar los ficheros de configuración del protocolo (nrpe.cfg), distribuyéndolo desde una máquina en la que ya funcionase hacia otra que lo

necesitara. Esto ha supuesto ir realizándolo, a medida que se disponía de la información de los equipos en los 220 servidores e ir entrando en cada uno para reiniciar el servicio y que así se tuviesen efecto los cambios.

Con respecto al apartado anterior, se puede destacar la gran cantidad de tiempo que se emplea en todo el proceso y lleva a exponer una de las posibles mejoras que se podría haber realizado, como es el uso, dejando de banda el "scp" (que por otra parte cada vez que se ejecuta pide la contraseña del servidor al cual se quiere enviar el fichero) y utilizando "dcp" (Distributed Copy).

¿Qué hubiese aportado el "dcp"?

Con este protocolo se define un servidor cualquiera como centro de distribución, realizando una relación de confianza con todo los equipos de la red. A continuación se realizará una "copia distribuida" o "distributed copy" de los ficheros que nos interese, con la ventaja de que al realizar este comando y tener la relación de confianza:

- No se solicitará contraseña al realizar la copia.
- Se puede distribuir un fichero con un solo comando a todos los equipos requeridos, sin necesidad de hacerlo uno a uno.
- Se pueden distribuir comandos, acción muy útil para reiniciar los servicios (para que lo cambios en los ficheros de configuración tengan efecto) también usando solo una línea de comando.

Funcionaría de la siguiente manera:

***dcp -h dcp -V dcp -q dcp** [-a] [--all-nodes* context_list*] [-A] [--all-devices* context_list*] [-n* node_list*] [-N* nodegroups*] [-d* device_list*] [-D* devicegroups*] [-C* context*] [-f* fanout*] [-l* user_ID*] [-o* node_options*] [-O* device_options*] [-p] [-P] [-Q] [-r* node_remote_copy*] [--**device-rcp** device_remote_copy*] [-R] [-t* timeout*] [-X* env_list*] [-T] [-v]* source_file... target_path

Por último, comentar que tras varias semanas de funcionamiento de la aplicación con un entorno, en mi opinión, muy grande a nivel de servicios y número de equipos, se ha mostrado muy estable no generando ningún tipo de problema de carga y sosteniendo múltiples conexiones simultaneas tanto por http como por ssh.

Esto lleva a concluir, que tanto a nivel de hardware como de software elegido e instalado, se ha hecho de una manera correcta y eficiente para redes de una capacidad entre 100 y 600 servidores, que es lo que se buscaba en un principio.

Una de las mejoras ya realizada es que, actualmente, existen dos equipos trabajando en paralelo, cada uno de ellos en un CPD diferente, con los mismos datos por si se estropease uno de ellos. Para ello semanalmente, se extraen los datos de la base de datos de un equipo (siempre del que está en producción) y se exporta a un servidor central, esta base de datos estará disponible en caso de tener que “volcar” los datos actuales en el otro.

8. Bibliografía

Wikipedia
Enciclopedia libre online editada por la comunidad de internautas.

Cisco
https://sso.cisco.com
http://www.cisco.com/web/ES/index.html

Nagios
http://www.nagios.org/
http://exchange.nagios.org/

Comparativa software de monitorización
http://es.wikipedia.org/wiki/Anexo:Comparaci%C3%B3n_de_sistemas_de_monitorizaci%C3%B3n_de_redes

PNP4Nagios
http://www.pnp4nagios.org/

DocuWiki
http://www.dokuwiki.org/dokuwiki

NagiosQL
http://www.nagiosql.org/

NDOUtils
http://nagios.sourceforge.net/docs/ndoutils/NDOUtils.pdf

NDOUtils. Modelo de base de datos
http://nagios.sourceforge.net/docs/ndoutils/NDOUtils_DB_Model.pdf

MySQL Workbench
http://www.mysql.com/products/workbench/

Perl
http://www.cbkihong.com/download/perltut.pdf
http://support.microsoft.com/kb/324263/es

Red Hat
http://es.redhat.com/
http://rpm.pbone.net/

CPAN
http://www.cpan.org/

Apache
http://httpd.apache.org/

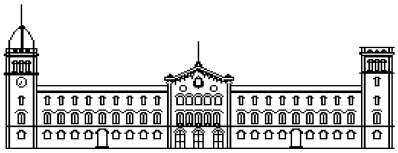
Introducción al NRPE
http://nagios.sourceforge.net/docs/nrpe/NRPE.pdf

Características IBM x3550 M3
http://www-03.ibm.com/systems/x/hardware/rack/x3550m3/

SNMP
http://www.unainet.net/documents/SNMP.pdf

NRPE
http://www.hcastelli.com.ar/nagios/instalacion-de-nrpe/

Syslog-ng Cisco
http://www.cisco.com/en/US/products/sw/cscowork/ps2073/products_tech_note09186a00800a7275.shtml



ENGINYERIA TÈCNICA EN INFORMÀTICA DE SISTEMES

UNIVERSITAT DE BARCELONA

Treball fi de carrera presentat el dia de de 200

a la Facultat de Matemàtiques de la Universitat de Barcelona,

amb el següent tribunal:

Dr. President

Dr. Vocal 1

Dr. Secretari

Amb la qualificació de: