UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

“PROPUESTA DE DESARROLLO DE UN SISTEMA DE GRAFICACIÓN DE LA RED DEL DEPARTAMENTO DE SERVIDORES DE LA DGSCA A TRAVÉS DEL PROTOCOLO DE GESTIÓN DE REDES SNMP”

TESIS

QUE PARA OBTENER EL TÍTULO DE
INGENIERO EN COMPUTACIÓN
P R E S E N T A
ALFONSO ABRAHAM AGUILERA CASTELLANOS

DIRECTOR DE TESIS
ING. RODOLFO VÁZQUEZ MORALES

ESTADO DE MÉXICO, AGOSTO 2007

Agradecimientos

Hoy que empiezo esta palabras, me doy cuenta que por fin completo un circulo que jamás pensé cerrar, sin duda, el hecho de embarcarme en la aventura de un trabajo de tesis, me hizo comprender que en la vida es importante concluir las metas, y llegar al final de mi camino en la vida universitaria era una de ellas.

Muchas personas han aparecido a lo largo de mi vida, me han apoyado y aconsejado para que ande por el sendero correcto, y en este sentido, mi familia ha sido una pieza fundamental, ellos me han guiado y heredado su buen ejemplo, me han apoyado incondicionalmente para brindarme las oportunidades que ellos no pudieron tener, pero sobre todo una persona, María, mi madre quien merece una mención especial, yo no sería el hombre que soy ahora si ella no me hubiera dado su cariño, dedicación y apoyo, solo puedo decirle que la amo mucho y que estoy orgulloso de ella.

No puedo dejar de mencionar a algunas personas con las que tuve la experiencia de transitar el largo camino de la carrera y con los cuales compartí las alegrías y sin sabores del andar universitario, Alejandra, Rene Lemuet, Ernesto y otros compañeros que espero con todos mis deseos se encuentren bien y cumpliendo sus sueños.

Y como no agradecer a los integrantes del Departamento de Administración de Servidores de la DGSCA que me ofrecieron su hospitalidad y apoyo para concebir y completar mí trabajo de tesis, en especial a Jesús Fernández Rauda quien me dio esta oportunidad y fue mi guía para completar esta meta. También hacer extensivo mi agradecimiento a mi asesor de tesis Rodolfo Vázquez por su tiempo y experiencia que me facilitaron la conformación de este trabajo.

Por último quiero agradecer a mi máxima casa de estudios 2 cosas, la primera por brindarme toda su maquinaría para hacer mi un profesionista que hará siempre su máximo esfuerzo por dejar en alto a la UNAM y la segunda por haberme dado la oportunidad de experimentar tantas experiencias en sus aulas, que por desgracia, se ahora mas que nunca que no las volveré a vivir, experiencias que me permitieron conocer a lo mejor que me ha pasado en muchos años, estefani, quien me ha dado su amor, apoyo y compañía sin esperar a nada a cambio y que han hecho de mi un hombre muy feliz, y que independientemente de lo que pase en el futuro siempre estará presente en mi mente.

Desde hace varios años el desarrollo de la informática ha permitido a ésta involucrarse en casi todos los aspectos del quehacer humano, ya desde los años 70, el empleo de las computadoras se había extendido a la mayoría de los ámbitos de la vida cotidiana, empezando por la computadora personal que de manera masiva irrumpió en muchos hogares y que hoy en día forma parte ya de uno de los artículos de uso cotidiano como lo son la televisión o la radio.

Es así como rápidamente la computadora paso a ser de un juego de video casero a una herramienta poderosa para la solución de problemas reales de la vida diaria, la implantación de estas tecnologías de la información en las organizaciones o empresas y su aplicación cada vez más extensa en diversos aspectos de las actividades humanas, ha permitido el desarrollo de grandes avances en áreas tan diversas como la economía, los negocios, la cultura, las ciencias, las relaciones humanas y en general la creación y difusión de la información.

En la actualidad sería prácticamente imposible concebir el crecimiento y desarrollo de las actividades de cualquier institución o empresa que prescindiera de la tecnología informática, no importando lo pequeña o grande que fuera ésta. Es así que cada vez es más común que las organizaciones realicen mayores inversiones en materia de tecnologías de la información para sistematizar sus procesos de negocio; procurando con esto contar con sistemas informáticos que proporcionen altos niveles de disponibilidad, desempeño y una mejor respuesta a las demandas de nuevos servicios.

Ante este panorama y debido a la importancia que representa la Universidad Nacional Autónoma de México para el desarrollo del país, la UNAM cuenta con una no menos importante infraestructura informática que le permite la prestación de servicios y

recursos informáticos en beneficio de la comunidad universitaria, planeación y desarrollo de proyectos enfocados a la divulgación de actividades culturales, científicas, deportivas así como otras tantas relacionadas con la vida académica de la universidad.

En virtud de este escenario, los administradores de sistemas se enfrentan a retos tecnológicos cada vez más complejos que requieren el manejo de conceptos y habilidades diversas, para el desarrollo de nuevas y mejores prácticas relacionadas con la administración y planeación de redes de datos y servidores que traigan como consecuencia mayores beneficios para la institución o empresa, independientemente de la actividad o ramo de la misma.

Este trabajo de tesis plantea una solución, basada en la integración de metodologías y herramientas de software de libre distribución que permita al departamento de servidores de la Dirección General de Sistemas y Cómputo Académico contar con información del estado y comportamiento que guardan los equipos y servicios de cómputo ante la demanda de los usuarios, datos que permitan prever y justificar planes o estrategias para satisfacer las necesidades que la universidad le demande al departamento.

De esta manera se contarán con herramientas que permitan incrementar la disponibilidad y rapidez de los recursos y servicios brindados por el área, con lo cual la comunidad universitaria se verá beneficiada en diversos aspectos ya que el Departamento de Administración de Servidores lleva a cabo tareas como: administración de la mayor parte del correo institucional de la UNAM, administración de base de datos ,administración de sitios Web los cuales contribuyen a la difusión de actividades culturales, científicas, entre otras relacionadas con la actividad académica.

Capitulo 1
Antecedentes

En este primer capitulo se explicará un poco de la historia del departamento de Administración de Servidores y su posición dentro del organigrama de la Dirección General de Servicios y Computo Académico[1]. También se expondrán brevemente los servicios y actividades que se realizan así como la infraestructura de la que se vale para poder ofrecer estas importantes actividades a la UNAM. Es así como podremos iniciar la descripción de los esquemas de monitoreo y graficación actuales del departamento, y por consiguiente plantear las necesidades que se presentan en dichos esquemas. Por último se menciona la necesidad de completar dichos esquemas en base a una nueva herramienta de monitoreo, la cual fue propuesta al final de este capitulo.

1.1 Antecedentes históricos

1.1.1Origenes de Red UNAM

Hacia final de los años 60's y el principio de los años 70's marcaron para la UNAM, una etapa de inicio en lo referente a las comunicaciones telefónicas y de datos. Es en este lapso cuando se realizaron las primeras tareas de conexión de teletipos hacia una computadora central, utilizando líneas telefónicas de cobre, de la recién instalada red telefónica dentro de la institución.

De manera vertiginosa este tipo de tecnología fue usada al interior de la UNAM y difundida al exterior, pero es a partir de la segunda parte de la década de los años 80's cuando la UNAM realiza cambios en las comunicaciones.

Es así como en el año de 1987, la UNAM pudo establecer la primera conexión a

[1] De ahora en adelante se hará referencia a esta dirección por su acrónimo de DGSCA

la Red Académica de C ó BITNET, solo mediante enlaces telefónicos, desde la Ciudad Universitaria hasta el Instituto Tecnológico de Estudios Superiores de Monterrey y de ahí hasta San Antonio, Texas en los Estados Unidos.

No fue sino hasta el año de 1989, cuando la UNAM a través del Instituto de Astronomía establece un convenio de enlace a la red de la NSF[2] en Estados Unidos, el cual se realizó gracias a un satélite mexicano denominado Morelos II entre el Instituto de Astronomía en la UNAM y el UCAR-NCAR[3] con residencia en Boulder Colorado. También se llevó a cabo el primer enlace para conectar las redes de área local, entre el Instituto de Astronomía y la DGSCA, utilizando enlaces de fibra óptica.

De esta manera se inició dentro de la UNAM una revolución en las comunicaciones, dando lugar a una gran adquisición de computadoras personales y su interconexión e intercomunicación en redes de área local, lo que permitió desarrollar la infraestructura de comunicaciones con fibra óptica, y establecer más enlaces satelitales hacia Cuernavaca, Mor., y San Pedro Mártir en Ensenada, Baja California Norte, así como la creación del primer enlace de microondas de alta velocidad entre la Torre II de Humanidades y la DGSCA. Es así como en 1990 la UNAM, fue la primera institución en Latinoamérica que se incorpora a la red mundial Internet, que ya en ese momento enlazaba a millones de máquinas y decenas de millones de usuarios en todo el mundo.

A finales de 1989 se estableció un proyecto que tenía como objetivo reemplazar los viejos conmutadores para renovar de forma total el sistema telefónico de la UNAM, de acuerdo con los estándares más modernos y con capacidad de crecer conforme a las necesidades de la institución. Debido a ello se creo la Dirección de Telecomunicaciones Digitales (DTD) cuyo objetivo sería la creación de la Red Integral de Telecomunicaciones de la UNAM, la cual debería ser capaz de transmitir indistintamente datos e imágenes entre las dependencias universitarias

[2] NFS: *National Science Foundation* (Fundación de Ciencia Natural)
[3] UCAR-NCAR: *Early Career Scientists Assembly*

independientemente de su ubicación geográfica.

En 1990, con la creación del laboratorio de Red UNAM, la Universidad se convierte en la primera Institución de Latinoamérica en incorporarse a la Red mundial de Internet enlazando a millones de maquinas y decenas de millones de usuarios de todo el mundo, ofreciendo una gran cantidad de servicios a la comunidad universitaria, como correo electrónico, hospedaje de páginas web, ftp, entre otros servicios.

1.1.2 La Dirección de Telecomunicaciones Digitales (DTD)

Con el fin de renovar la Infraestructura y los Sistemas de Telecomunicaciones se creo, a finales de 1989, la DTD para a su vez, propiciar la creación de la Red Integral de Telecomunicaciones de la UNAM. Todo esto con el fin de recabar y transmitir datos de interés general entre las diferentes dependencias universitarias. Hoy en día el 90% de la Comunidad Universitaria tienen acceso a esta Red.

El siguiente organigrama (figura 1.0) es la estructura de las diferentes subdirecciones y coordinaciones en que se divide la DTD de la UNAM para una mejor optimización de los recursos y servicios que ofrece al público en general, así como a la comunidad universitaria.

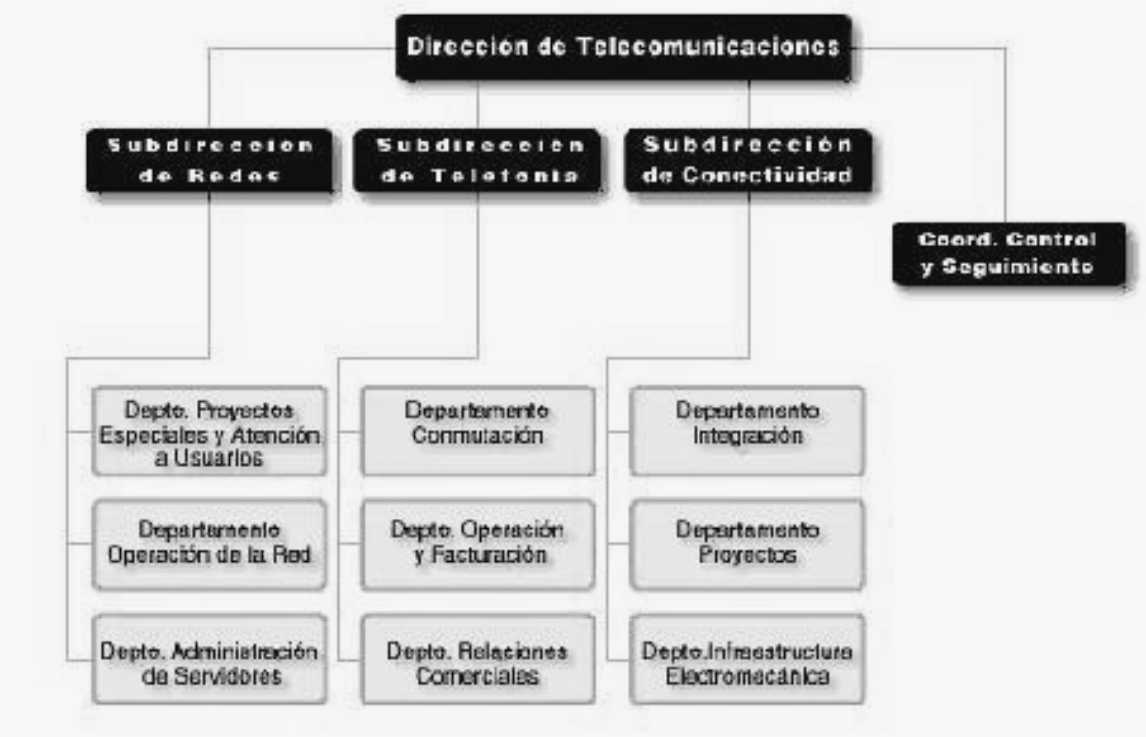


Figura 1.0 Organigrama de la DTD

1.1.3 Subdirección de Redes

Esta subdirección opera, coordina y mantiene a la Red Universitaria de Datos (RedUNAM) a nivel técnico y administrativo. Analiza y dirige los lineamientos a seguir en la evolución de la misma Red. Esta integrada por los departamentos de:

- Proyectos Especiales y Atención a Usuarios
- Operación de la Red
- Administración de Servidores

1.1.4 Departamento de Administración de Servidores

El Departamento de Administración de Servidores fue creado en el año de 1997 como parte de la descentralización de la Coordinación de Servicios de Red (CSR),

perteneciendo desde esa fecha a la Subdirección de Redes.

El Departamento es responsable del mantenimiento y supervisión de servicios de vital importancia para la universidad tales como el correo electrónico, hospedaje de sitios web, ftp anónimo y comercio electrónico[4]. A través del tiempo el departamento ha ido reestructurado su forma de operación en función de los equipos de cómputo que ha tenido a su disposición para llevar a cabo las tareas antes mencionadas, de esta manera el Departamento de Administración de Servidores se convirtió en el responsable de los servidores centrales de Red UNAM con el fin de brindar servicio a la comunidad universitaria.

Debido a que el objetivo del presente trabajo es la implementación de un sistema de graficación y monitoreo para la supervisión del correcto funcionamiento de los servidores y servicios que estos ofrecen, en el siguiente apartado se dará una descripción breve de los servidores y servicios más importantes a cargo del Departamento de Administración de Servidores que serán graficados y monitoreados.

1.2 Servidores y Servicios

El Departamento de Servidores cuenta con una infraestructura que le permite llevar a cabo las tareas que le son encomendadas. Los servidores proporcionan una serie de servicios que con el tiempo han llegado a cobrar gran importancia para el desarrollo de las actividades universitarias. Dentro de esta serie de servicios podemos mencionar dos de los ejemplos más importantes como son:

- **Hospedaje de sitios web.** Principalmente juegan el papel de la divulgación de actividades culturales y científicas no solo de la propia universidad sino también de ésta con otras instituciones educativas, aun que existen otros portales web más sofisticados que mediante la incorporación de bases de datos tienen como finalidad otros objetivos, tales como la pagina de la bolsa de trabajo

[4] La parte de comercio electrónico se refiere a las diferentes tiendas que pertenecen a la UNAM

de la UNAM o la incorporación de la universidad al comercio electrónico con el desarrollo de portales de Internet para la venta de productos y servicios de la cadena de tiendas que pertenecen a la UNAM.

- **Correo electrónico.** Tal vez es el servicio informático más demandante y por lo consiguiente el más importante para el área de Administración de Servidores. El departamento es responsable de proporcionar y mantener el correo para cuentas con dominio @servidores.unam.mx, @correo.unam.mx. También es responsable de crear y mantener cuentas de correo electrónico para cualquier dominio que se requiera tener bajo el dominio unam.mx.

Sin embargo el Departamento de Administración de Servidores constantemente incorpora nuevos servicios informáticos conforme a las nuevas necesidades que la propia comunidad universitaria le demanda. A continuación se brindará una tabla donde se describe el nombre de los servidores, los servicios que estos ofrecen así como algunas características del equipo.

Servidor	***Servicios***	***Equipo***
uranio.servidores.unam.mx	Proporciona el servicio de correo con la capacidad de múltiples dominios. Utiliza qmail, vpopmail, apache y sqwebmail como interfaz gráfica.	SunBlade100
oxigeno.servidores.unam.mx (web)	Proporciona la página y base de datos de www.correo.unam.mx, además de otras páginas web. Utiliza apache con ssl, horde y oracle de ayuda y correo. Contiene un ldap de correo.unam.mx	Xeon a 3.6 GHZ 20 GB en disco y 4 GB de memoria.
servidor.servidores.unam.mx	Servidor con LDAP, no contiene correo qmail, solo es consultado para reenviar a los usuarios.	Armada

Servidor	***Servicios***	***Equipo***
serv1.servidores.unam.mx	Servidor de listas de correo @servidor.unam.mx. Utiliza majordomo y qmail. (apache para los cgi.)	AMD a 1.2 GHZ con 37 GB en disco y 256 MB de memoria.
servidor1.servidores.unam.mx	Servidor de correo. Utiliza qmail y ldap. (Apache para generar estadísticas con mrtg para los cgi).	Xeon a 2.4 GHZ 150 GB en disco y 2 GB de memoria.
servidor2.servidores.unam.mx	Servidor de correo. Utiliza qmail y ldap. (apache para generar estadísticas con mrtg y los cgi).	Xeon a 2.4 GHZ 150 GB en disco y 2 GB de memoria.
servidor3.servidores.unam.mx	Servidor de correo. Utiliza qmail y ldap. (Apache para generar estadísticas con mrtg y cgi).	Xeon a 2.4 GHZ 150 GB en disco y 2 GB de memoria.
servidor4.servidores.unam.mx	Servidor de correo. Utiliza qmail y ldap. (Apache para generar estadísticas con mrtg y cgi).	Xeon a 2.4 GHZ 150 GB en disco y 2 GB de memoria.
cs (Comunicación social)	Tiene una página web de la Dirección General de Comunicación Social con base de datos. Utiliza apache y postgres.	Xeon a 2.4 GHZ 30 GB en disco y 2 GB de memoria.
correo.servidores.unam.mx	Servidor de correo. Utiliza qmail y ldap.	Xeon a 3.6 GHZ 128 GB en disco y 4 GB de memoria.
cuarentena.servidores.unam.mx	Proporciona la página de la cuarentena y la base de datos. Utiliza tomcat y mysql.	Es la misma máquina. Pentium 4 a 3.4 GHZ. 200 Gb en Disco y 1 Gb en RAM.

Servidor	***Servicios***	***Equipo***
cuarentena.correo.unam.mx	NOTA: esta cuarentena es para cuentas @servidor.unam.mx	
velvet.servidores.unam.mx	Proporciona páginas web con ssl y bases de datos. Utiliza apache con ssl, tomcat, sybase, postgres, mysql.	Dominio de un servidor Sun Fire E6900
issyk.servidores.unam.mx	Proporciona páginas web con ssl y bases de datos. Utiliza apache con ssl, sybase, postgres, mysql, ssh2 puerto 8022 y Real Audio.	Dominio de un servidor Sun Fire E6900
erannis.servidores.unam.mx	Proporciona páginas web y bases de datos. Utiliza apache, tomcat y postgres.	Sun Fire V20z
azuki.servidores.unam.mx	Respalda bases de datos ldap de correo, riu, servidor y base de datos Utiliza ldap	Sun Fire V20z
aletia.servidores.unam.mx	Tiene implementado un MON con los mismos monitores que los usados en junin. Es tambien el servidor de graficación ya que tiene instalado CACTI con una base de datos mysql y un apache para consultar el front -end	Sun Fire V20z
Aura	Utilizada para pruebas de alta disponibilidad. Esta implementado heartbeat para un failover en conjunto con la maquina kaori. A su vez esta implementado un servidor de traps que se encarga de recibir las alertas SNMP y reenviarlas a aletia para que notifique por medio de MON.	Sun Fire V20z

Servidor	***Servicios***	***Equipo***
junin.servidores.unam.mx	Proporciona el servicio de monitoreo y alarma de todos los servidores y servicios. Utiliza mon y apache. También tiene la base de datos de la RIU, ésta utiliza ldap.	Pentium 4 a 3.4 GHZ, 200 GB en disco y 1 GB de memoria.
Kaori	Utilizada para pruebas de alta disponibilidad. Esta implementado heartbeat para un failover en conjunto con la máquina aura. Es utilizada para desplegar una página web temporal en caso de que no este funcionando el apache de un servidor productivo.	Pentium 4 a 3.4 GHZ, 200 GB en disco y 1 GB de memoria.
Hanza	Es el servidor de bitacoras de la system controller de la 6900. Este realiza un monitoreo de las bitacoras y manda una alerta SNMP cuando el site tiene alta temperatura.	Sun Blade
Kenai	Tiene un apache con soporte SSL y Sybase. Se espera que los sitios de las tiendas sean migrados a este servidor.	Armada
Gypsy	Esta máquina establece una VPN y un bridge para accesar a la red privada del site. Se realiza la comunicación con la maquina lecna.	Pentium 4 a 3.4 GHZ, 200 GB en disco y 500 Mb de memoria.
Lecna	Esta máquina establece una VPN y un bridge con la maquina gypsy. Establece un firewall para impedir que una maquina de la red privada de su segmento pueda accesar a una maquina del otro lado del bridge	Pentium 3, 200 GB en disco y 500 Mb de memoria.
Mailflow	Maquina utilizada para realizar búsquedas en el ironport por medio de una herramienta web llamada mailflow	Pentium 4 a 3.4 GHZ, 200 GB en disco y 1 GB de memoria.

Servidor	***Servicios***	***Equipo***
dragon.dgsca.unam.mx	Servidor apache que contiene la página de la UNAM entre otros. Tiene instalado Sybase, mysql, postgres.	Dominio Sun 10000 en DGSCA
einstein.servidores.unam.mx	Servidor apache con soporte SSL para los sitios de las tiendas. Manejadores BD Sybase. Un servidor servlet TOMCAT.	Dominio Sun 1000 en pitagoras
cobalto.servidores.unam.mx	Tiene Apache, y manejadores BD Sybase, postgres y mysql.	Dominio Sun 10000 en DGSCA
euler.servidores.unam.mx	Nosotros no supervisamos ningún servicio	Dominio Sun 1000 en pitagoras
newton.servidores.unam.mx	Nosotros no supervisamos ningún servicio (Dominio de redes)	Dominio Sun 1000 en pitagoras
Cromosoma	Máquina que tiene configurados los robots para respaldos (Dell y el sun) 1N, 2N y 3N	Sun 3500
Pine	EL servicio de PINE para el doctor pisanty	Armada
pine.servidores.unam.mx	Servicio de correo PINE para los usuarios	Armada
litio.servidores.unam.mx	Consola SSP para plataforma silicio de la 1000 de DGSCA	Sun ultra Sparc
pitagoras.servidores.unam.mx	Consola SSP para plataforma silicio de la 1000 de Pitágoras	Sun ultra Sparc
ironport.servidores.unam.mx	Cajas anti-spam	No se cuenta con la especificación
ironport2.servidores.unam.mx	Cajas anti-spam	No se cuenta con la especificación
xalumno.servidores.unam.mx	Servidor de correo con LDAP y QMAIL para el dominio exalumno.servidores.unam.mx	Servidor HP

Servidor	***Servicios***	***Equipo***
neon.servidores.unam.mx	Ya fue migrado los correo y LDAP a xalumno.servidores.unam.mx. Lo único que sigue en producción es el horde.	Armada
fenix	Utilizado para monitorear la temperatura del nuevo site de la supercomputadora.	Sun 3500

1.3 Monitoreo y Graficación actual en el Departamento

El desarrollo del presente trabajo viene a complementar las actuales herramientas de graficación y monitoreo con las que el Departamento de Administración de Servidores cuenta. En la parte de monitoreo el Departamento se vale de la herramienta de monitoreo llamada MON[5] para vigilar los servicios críticos en los servidores más importantes del área, como son: servidores web Apache (httpd) ,correo, base de datos (mysql, Sybase, Postgres, Oracle) así como verificar la disponibilidad de estos recursos en la red publica(ping), entre otros.

La parte de graficación de los recursos y servicios es la más carente en el departamento , casi no se cuentan con gráficas de los servidores, actualmente solo se dispone de esta característica en los 4 servidores de correo principales del dominio servidor.unam.mx, los cuales son servidor1.servidores.unam.mx, servidor2.servidores.unam.mx, servidor3.servidores.unam.mx y servidor4.servidores.unam.mx que son generadas a través de una herramienta de graficación denominada MRTG y solo se grafican aspectos como son: entrada de paquetes de correo, Bits Trasferidos, consumo de CPU, Carga del sistema ,Paquetes recibidos y enviados por las interfaces de red así como los consumos de memoria RAM

[5] MON: *Service Monitoring Daemon* (Demonio de Monitoreo de Servicio)

y SWAP del equipo. Estas gráficas son acumuladas y presentadas en forma diaria, semanal, mensual y anual.

1.3.1 Herramienta de monitoreo MON

MON es el programa utilizado en el Departamento de Administración de Servidores, esta es una herramienta de monitoreo que es utilizada para verificar la disponibilidad de cualquier tipo de servicios sobre cualquier numero de servidores. MON es un útil sistema de monitoreo quien se encargará de realizar una notificación tan pronto se presente algún problema en algún servidor o dispositivo de red que este siendo monitoreado.

MON fue diseñado bajo Linux aunque se ha comprobado que trabaja sin problema bajo Solaris, también soporta la comunicación y recepción de otros métodos de alertas, todas ellas pueden ser fácilmente implementadas con programas como C, Perl, shell scripts, alertas SNMP y alertas especiales de MON. Esta herramienta lleva a cabo dos tareas fundamentales, la comprobación de alguna condición que refleje la presencia de algún error y disparar alguna acción definida en caso de haber presentado alguna falla.

MON ya cuenta con un conjunto de programas para monitorear diversos servicios en diferentes aspectos así como una serie de alertas que se encargarán de hacer las notificaciones correspondientes en caso de alguna falla. A continuación se hará mención de algunos monitores y alertas.

1.3.1.1 Tipo de Monitores

- **ICMP echo (ping)**

Es un monitor que tiene como finalidad enviar un paquete de datos (comúnmente llamados pings) a uno o varios servidores vía ICMP[6] para verificar la disponibilidad del servidor en la red de datos.

- **SMTP**

Este monitor comprueba la conectividad vía SMTP[7] a uno o varios servidores esperando una respuesta que comprobará la disponibilidad de este servicio.

- **TELNET**

Este monitor comprueba la disponibilidad del servicio Telnet[8].

- **FTP**

Este monitor comprueba la disponibilidad del servicio FTP[9].

- **HTTP**

Este monitor se comunica con el protocolo HTTP[10] para consultar una dirección web en un servidor remoto. Este monitor puede ser usado para verificar la velocidad y reportar fallas en caso de presentarse una baja velocidad de transmisión con uno o varios servidores de Internet, aun que hay muchas variantes de monitores para verificar diversas características del servicio HTTP.

- **POP-3**

Este monitor comprueba la conectividad vía POP3[11] a uno o varios servidores esperando una respuesta que comprobará la disponibilidad de este servicio.

---

[6] ICMP: *Internet Control Message Protocol* (Protocolo de Control de Mensajes de Internet)
[7] SMTP: *Simple Mail Transfer Protocol* (Protocolo se Transferencia de Correo Simple)
[8] Telnet: Protocolo de red que sirve para acceder y controlar remotamente a otra computadora.
[9] FTP: *File Transfer Protocol* (Protocolo de Transferencia de Archivos).
[10] HTTP: *Hyper Text Transfer Protocol* (Protocolo de Transferencia de Hipertexto)
[11] POP3: *Post Office Protocol* 3 (Protocolo de Oficina Postal versión 3)

- **IMAP**

Este monitor comprueba la conectividad vía IMAP[12] a uno o varios servidores esperando una respuesta que comprobará la disponibilidad de este servicio.

- **Disk space**

Monitor utilizado para monitorear el espacio disponible en los discos físicos de uno o varios servidores vía NFS[13].

- **SNMP monitos**

Existen múltiples monitores para establecer comunicación con agentes SNMP[14]. Debido a las capacidades del protocolo se pueden monitorear diversas aspectos en los dispositivos de red que implemente el agente SNMP, algunas son el tiempo que ha estado prendido el dispositivo de red, el estado que guarda alguna impresora en red, la cantidad de procesos del algún servidor , el espacio consumido y disponible de los sistemas de ficheros, etc.

- **LDAP**

Este monitor comprueba la conectividad de LDAP[15] a uno o varios servidores esperando una respuesta que comprobará la disponibilidad de este servicio.

- **Dial-in terminal servers**

Monitor que verifica periódicamente la conexión a un servidor modem.

En caso de presentarse algún problema con alguno de estos monitores, MON cuenta con alertas que avisan a través de notificaciones de correo electrónico, mensajes vía modem usando un programa llamado QuickPage[16] para desplegar la información en una pagina web, entre otros más.

[12] IMAP: *Internet Message Access* Protocol (Protocolo de Acceso de Mensajes de Internet)
[13] NFS: *Network File System* (Sistema de Archivo de Red)
[14] SNMP: *Simple Network Management Protocol* (Protocolo Simple de Administración de Red)
[15] LDAP: *Lighweight Directory Access Protocol* (Protocolo de Acceso a Directorio de Bajo Peso)
[16] *QuickPage*: Diminuto y eficiente editor de lenguaje HTML que permite crear paginas web

1.3.2 Esquema de monitoreo en el Departamento de Administración de Servidores

Como ya se ha mencionado anteriormente, MON es la herramienta utilizada para monitorear los servidores del área, auxiliándose del conjunto de monitores que la herramienta trae consigo, algunos de los cuales ya fueron descritos en el apartado anterior. De esta manera se han configurado una serie de grupos de servidores para los cuales se han definido una serie de monitoreos periódicos para verificar la disponibilidad de servicios como son: http ,ldap(correo), base de datos (mysql, Sybase, Postgres, Oracle), disponibilidad en la red (ping), POP3(correo),IMAP(correo), entre otros más.

En caso de existir falla en alguno de los monitoreos, las notificaciones son enviadas tanto a una lista de correo llamada "staff@servidor.unam.mx", en la cual están inscritos las direcciones de correo electrónicos de todos los responsables del Departamento de Administración de Servidores, así como a un servicio de Sky Tel que es recibido en un aparato de mensajes de texto llamado *bipper* propiedad del Departamento.

MON permite presentar la información en una interfaz web conectándose como cliente al servidor MON (a través de la url http://132.248.120.101/cgi-bin/mon.cgi), de esta manera el Departamento de Administración de Servidores cuenta con una pagina web que muestra el detalle de los grupos de servidores que están siendo monitoreados (figuras 1.1, 1.2 y 1.3), que tipo de monitores se están usando, la frecuencia en que se están realizando los monitoreos así como la representación de forma gráfica mediante el uso colores para indicar si se han presentado fallas y alertas.

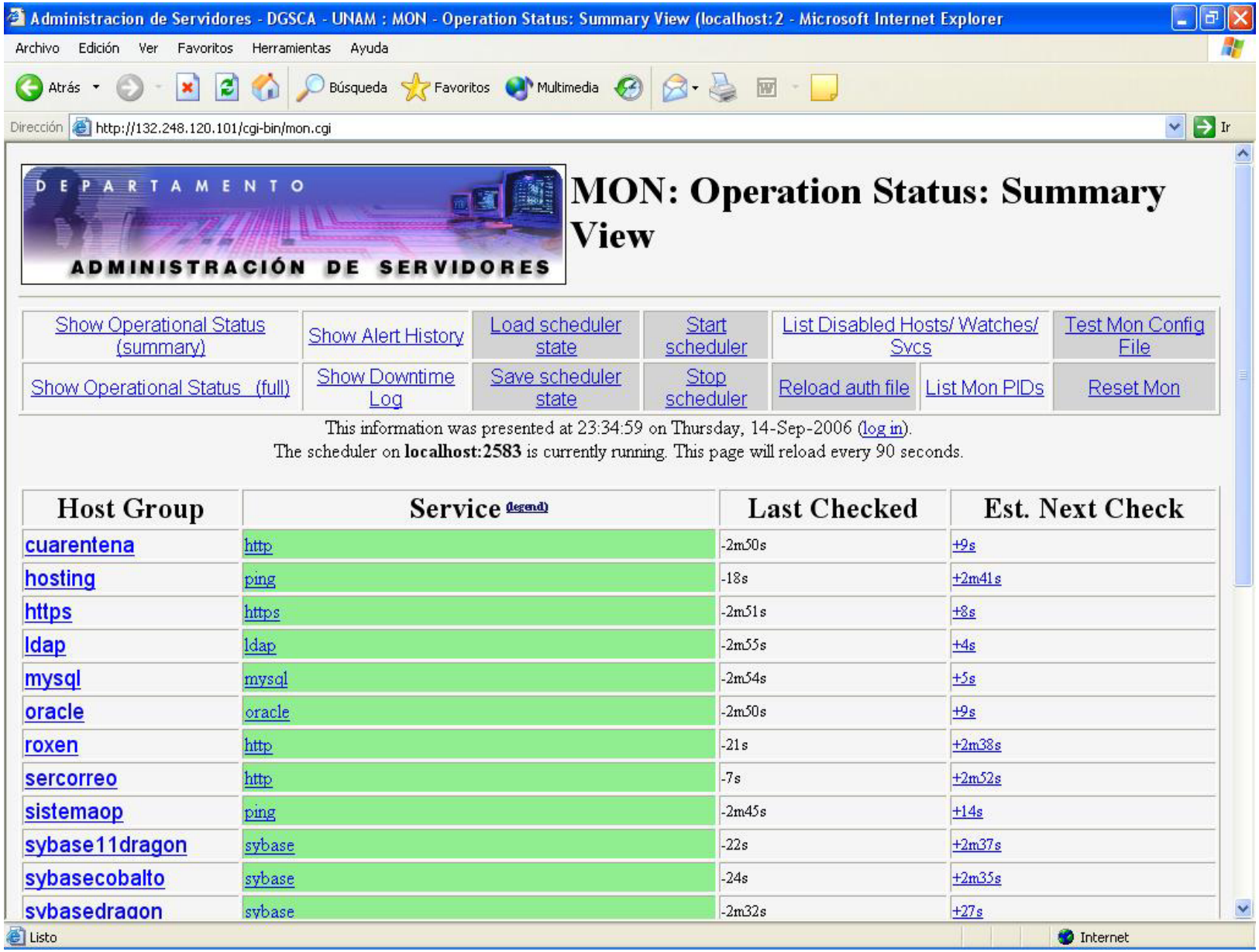


Figura 1.1 Pagina principal del cgi de MON

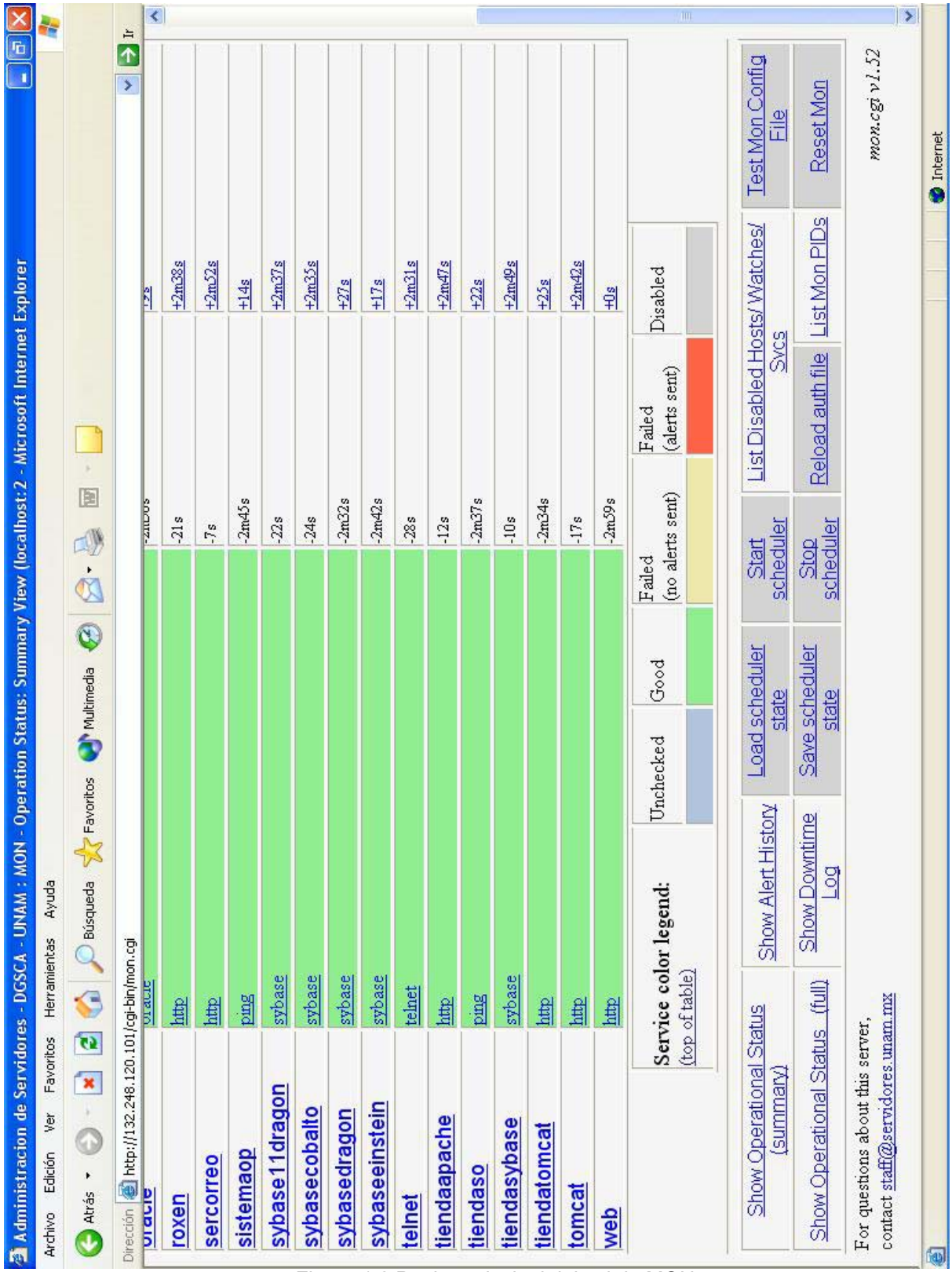


Figura 1.2 Pagina principal del cgi de MON

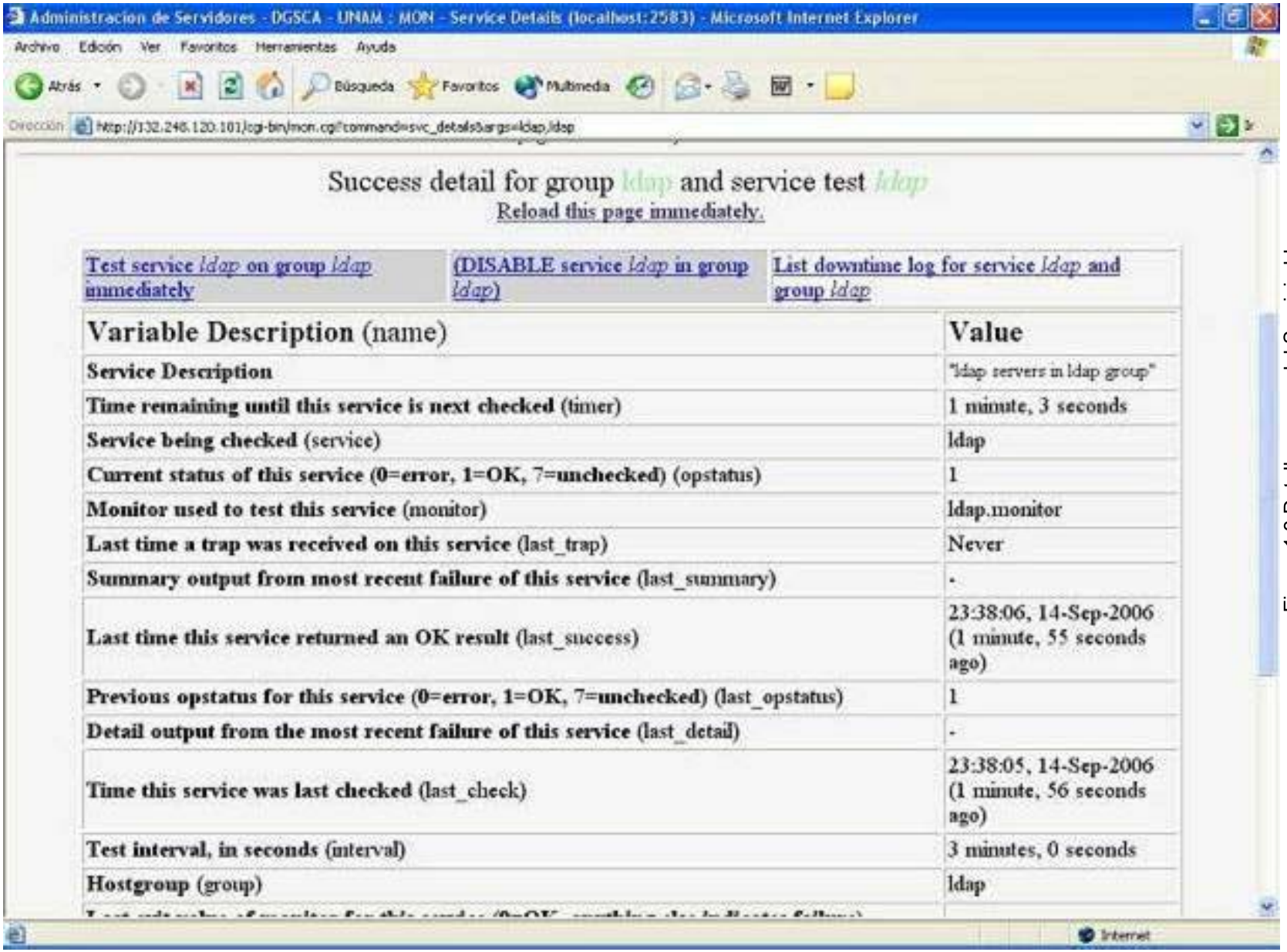


Figura 1.3 Detalle acerca del Servicio ldap

1.3.3 Herramienta MRTG

MRTG[17] es una utilidad que nos sirve para representar datos. Fue una herramienta que inicialmente fue creada para representar de forma gráfica el tráfico que atravesaba las interfaces de los routers, pero su implementación se puede utilizar para representar prácticamente cualquier dato.

MRTG genera paginas HTML que contienen imágenes en formato PNG[18] las cuales proveen una representación visual bastante buena del tipo de información que se este graficando.

MRTG ha sido probado con éxito en la mayoría de los sistemas operativos derivados de UNIX y Windows NT. Esta herramienta captura los datos de dos formas, mediante el uso del protocolo de gestión de red SNMP ó el uso de scripts de usuario.

1.3.4 Graficas en el Departamento

El Departamento de Administración de Servidores solo estaba llevando a cabo la graficación en los servidores principales de correo, se están graficando diversos aspectos como son: entrada de paquetes POP3, POP3 SSL, IMAP, IMAP SSL,Bits Trasferidos, consumo de CPU, Carga del sistema ,paquetes recibidos y enviados por las interfaces de red así como los consumos de memoria RAM y SWAP del equipo.

Estas gráficas son acumuladas y presentadas en forma diaria, semanal, mensual y anual. Los datos entregados a MRTG son a través de una serie de scripts de usuario, estos se encargan de recopilar la información de interés y entregarla a MRTG para que lleve a cabo la creación de las gráficas. Desgraciadamente solo se cuentan con gráficas de este tipo en solo 4 servidores (figuras 1.4, 1.5 y 1.6).

[17]MRTG: *Multi Router Trafic Graph* (Graficador de Tráfico de Múltiples Enrutadores)
[18] PNG: *Portable Network Graphics (Graficas de Red Portátil).*

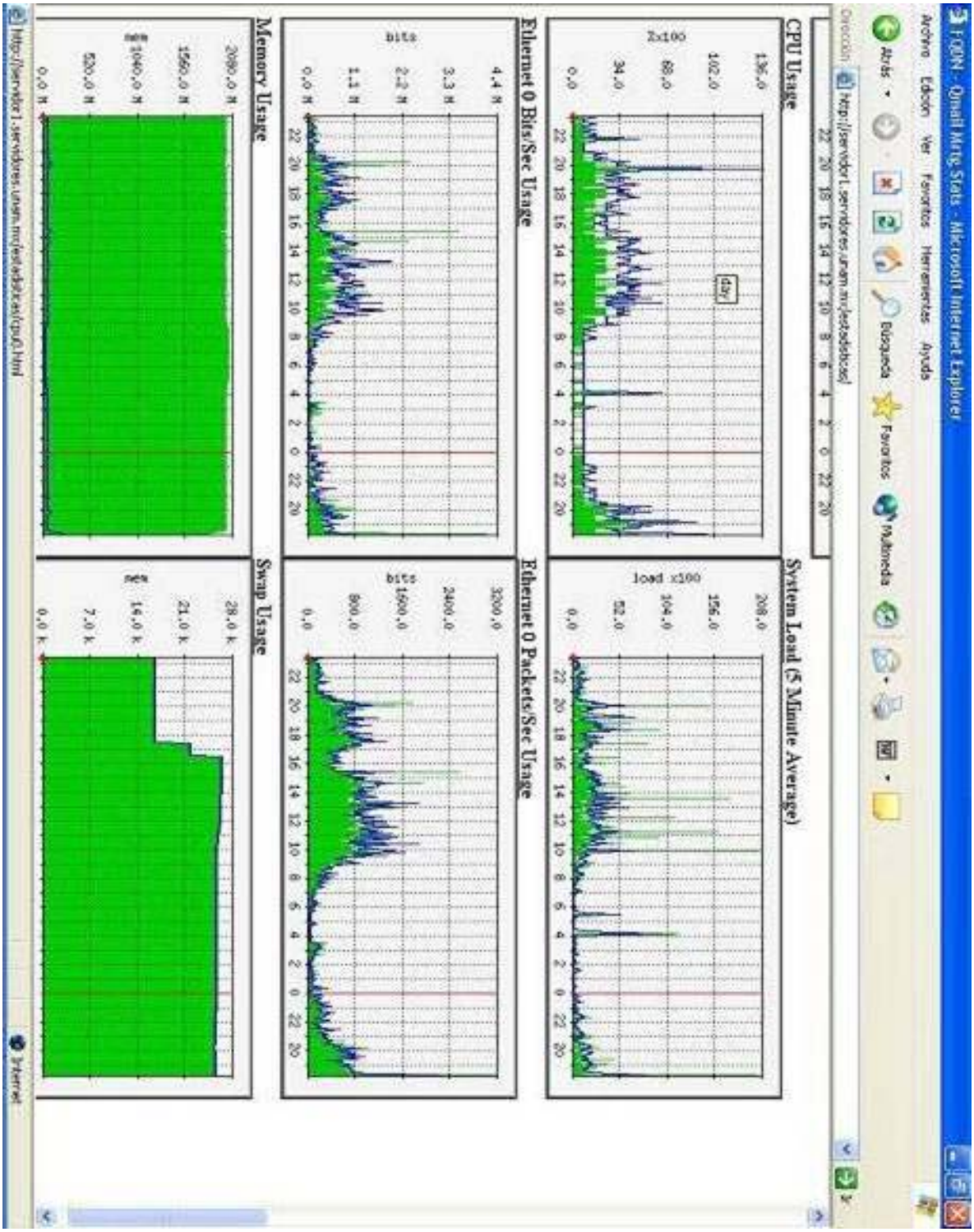


Figura 1.4 Graficas de los recursos del servidor1

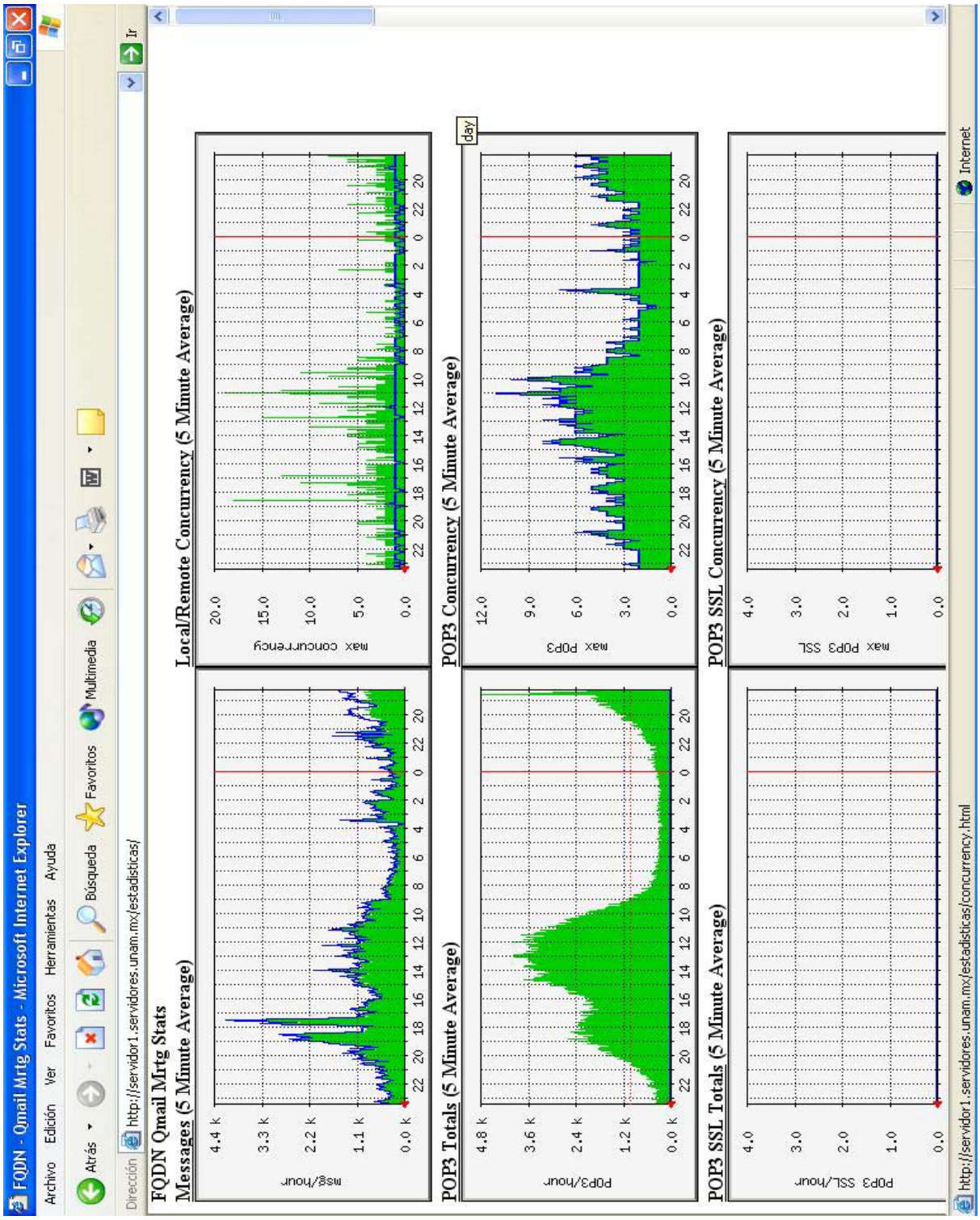


Figura 1.5 Graficas POP3 y POP3 ssl en Servidor1

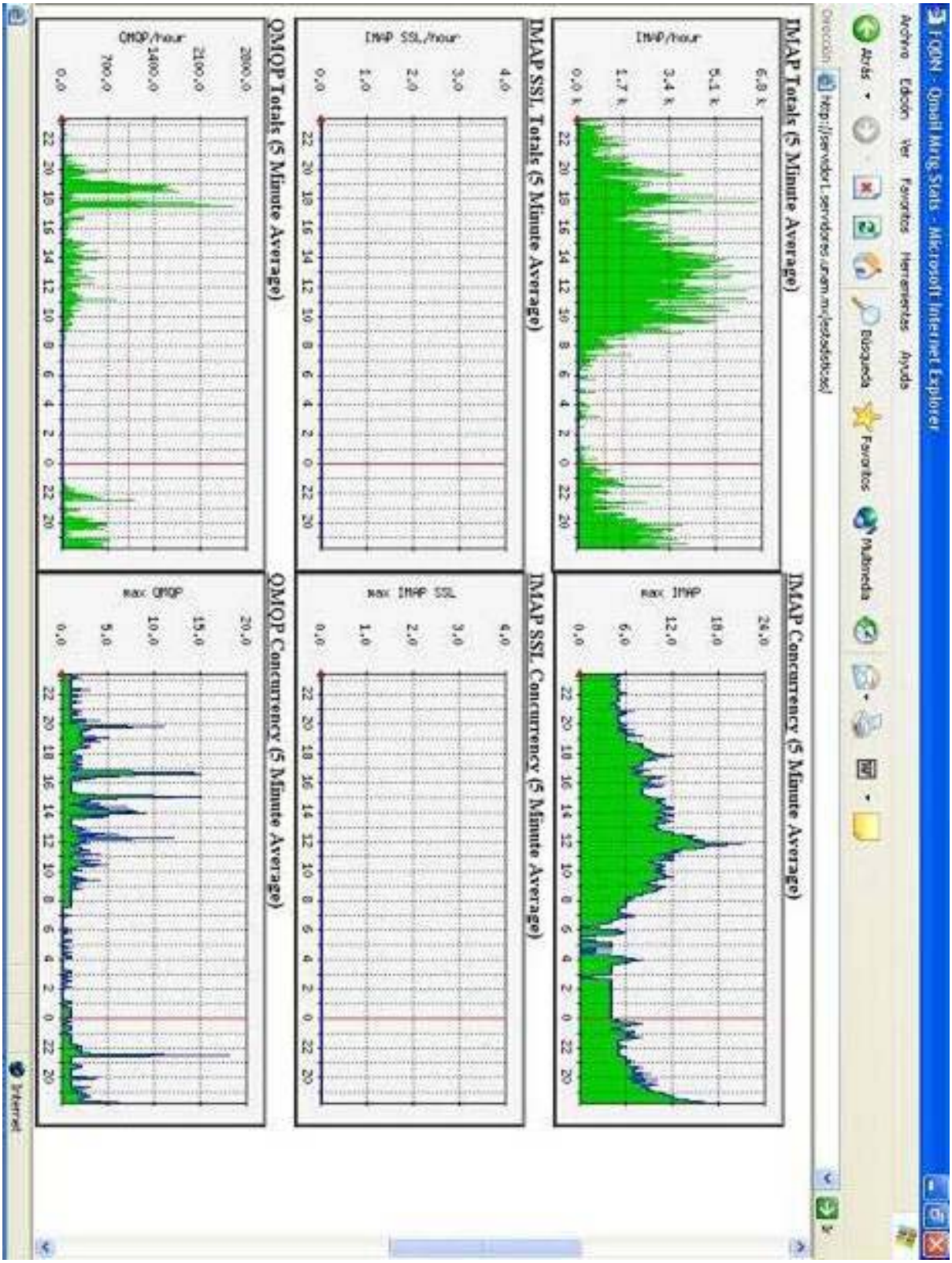


Figura 1.6 Graficas de los paquetes IMAP y IMAP ssl del servidor1

1.4 Necesidades en los esquemas de Graficación y monitoreo

El Departamento de Administración de Servidores tiene a su cargo el mantenimiento y administración de un gran número de equipos. En tiempos recientes la infraestructura del departamento ha venido creciendo y con ello también la importancia de los servicios así como la responsabilidad de mantenerlos disponibles.

Los servidores y los servicios que estos brindan son de vital importancia para la UNAM, servicios como correo electrónico, hospedaje de un gran numero de sitios web, comercio electrónico, entre otros más son indispensables en la vida cotidiana de la Universidad. Debido a estas razones fue necesario complementar los actuales esquemas de graficación y monitoreo, con la finalidad de contar con una supervisión y monitoreo más eficiente en la totalidad de los servidores.

Ante este panorama el departamento tenía 2 objetivos principalmente:

- Contar con una herramienta que permita la acumulación y almacenamiento de información significativa de todos y cada uno de los servidores del área, la cual deberá ser representada mediante graficas y mostrará diversos aspectos como carga de procesamiento de CPU, promedio de carga del sistema, número de procesos, consumo de memoria RAM y SWAP , tráfico de entrada y salida, consumo de los sistemas de archivos, estadísticas de base de datos, entre otros aspectos que se tenga necesidad de analizar. Todo este cúmulo de información nos permitirá tener un panorama global del comportamiento de la infraestructura informática del departamento ante la demanda de los usuarios, situación que nos dotará de la posibilidad de prever y justificar planes o estrategias para satisfacer las necesidades del departamento para con la comunidad universitaria. De esta forma se tendría una útil herramienta que principalmente tendrá su razón de ser en la ayuda que pueda brindar para la toma de decisiones en miras de mejorar la prestación de servicios. En la siguiente tabla se resumen algunas necesidades de reportes

gráficos que el departamento tenía antes de desarrollar el presente trabajo de tesis, graficas que nos permitirían analizar la información y actuar en consecuencia.

Necesidad	Servidores Implicados	Gráficos Requeridos	Acciones a tomar
Mejorar el rendimiento de los servidores de correo electrónico.	servidor1.servidores.unam.mx servidor2.servidores.unam.mx servidor3.servidores.unam.mx servidor4.servidores.unam.mx correo.servidores.unam.mx exalumno.servidores.unam.mx	Carga de procesamiento Consumo de File systems Consumo de Memoria SWAP Consumo de Memoria RAM Número de usuarios de correo	Las graficas nos permitirían tener un histórico del rendimiento de cada uno de los servidores de correo, de manera que podamos establecer el correcto balanceo en cuanto al número de usuarios en cada servidor, así como indagar si el almacenamiento y procesamiento disponible será suficiente en los siguientes meses o años.
Medir el rendimiento en la prestación del Hosting	velvet.servidores.unam.mx issyk.servidores.unam.mx	Carga de procesamiento Consumo de File systems Consumo de Memoria RAM Consumo de Memoria SWAP Cantidad de procesos de Apache Peticiones por segundo al servidor Apache Consultas a base de datos	Todo el análisis de esta información iría enfocado al mejoramiento de la prestación del servicio, determinando si la plataforma actual es suficiente para implementar nuevos requerimientos en Hosting o cualquier otra aplicación que se pretenda incorporar.
Justificación de Estrategias de Migración	Todos los del área	Carga de procesamiento Consumo de File systems Consumo de Memoria RAM Consumo de Memoria SWAP Cualquier gráfico para ver la demanda de algún servicio en particular	Todo el análisis de esta información, será de gran ayuda para justificar si se requiere hacer una migración de datos así como determinar las características de la nueva plataforma en los aspectos de procesamiento y almacenamiento.
Viabilidad de Planes de Implementación	Todos los del área	Carga de procesamiento Consumo de File systems Consumo de Memoria SWAP Consumo de Memoria RAM Cualquier gráfico que nos permita observar la demanda de algún servicio en particular	Sin duda toda esta información podrá brindar la certeza de que si el servidor involucrado en la implementación tendrá la capacidad de prestar el servicio de manera óptima.

- Complementar el esquema actual de monitoreo basado en el programa MON, con fines de mejorar esta tarea en lo referente al envío de alertas.

1.4.1 Problemática ante el nuevo panorama

Dada las condiciones que se tenían en los sistemas de graficación y monitoreo, surgió la necesidad de implementar nuevas herramientas que permitieran solventar la problemática que se presentaba con los esquemas de solución anteriores, sobre todo en la parte de generación de gráficas estadísticas.

Graficación

La implementación anterior de graficación, que solo se esta llevando a cabo en 4 servidores, no resultaba la más viable a seguir debido a que conlleva a un gran esfuerzo técnico y humano para poderlo configurar en la totalidad de los servidores.

Algunas de sus problemáticas eran las siguientes:

- La herramienta esta basada en MRTG de manera que este software tendrá que ser instalado en cada servidor.
- MRTG captura los datos a graficar por medio de scripts de usuario, lo que implica que se tendrán que copiar y adecuar estos programas, dependiendo del sistema operativo, en cada uno de los servidores.
- Para que las gráficas pudieran ser consultadas desde cualquier equipo en la red era necesario instalar un servidor web, dicho de otra manera, será necesario instalar un servidor web en cada uno de los servidores.
- El proceso de graficación al tener un esquema local, consumía, aún que no considerablemente, tiempo de CPU en cada servidor.

Alertas de Monitoreo

Era necesario solventar algunos problemas que se estaban presentando con las alertas de monitoreo. Debido a que MON trabaja bajo un esquema cliente/servidor en donde los programas monitores establecen una conexión como cliente a cada uno de los servidores monitoreados, se han presentado algunas anomalías bajo ciertas circunstancias.

Esto ha traído como consecuencia el envió de alertas falsas, alertas que se han disparado debido a que el servidor MON no puede establecer una sesión con el equipo ya sea por tráfico excesivo de la red de datos o por que el equipo se encontraba saturado por carga de trabajo. Ante estas condiciones un gran número de alertas que no tienen que ver con problemas de conexión, eran enviadas aun que no se hayan presentado las fallas.

Por ello se planteó la necesidad de auxiliarse de otra herramienta de monitoreo para complementar y mejorar el envió de alertas para determinados tipos de monitoreos.

1.5 Protocolo SNMP como propuesta de solución

Ante los objetivos anteriormente planteados, era necesario buscar la herramienta adecuada que permitiera cumplir con las metas mencionadas. Debido a que el Departamento de Administración de Servidores no puede hacer gastos excesivos en la compra de algún software propietario para este cometido, la filosofía fue auxiliarse de software libre.

La idea fue implementar un esquema de graficación centralizado, es decir tener

un solo servidor que tenga la tarea de realizar las gráficas para todos los servidores del Departamento, para ello la herramienta debería tener la capacidad de guardar información relevante de cada uno de los dispositivos de la red, de manera tal que pueda ser consultada de forma fácil y rápida.

Desde el punto de vista de las alertas, la herramienta debía tener la capacidad de revisar de manera autónoma la información recolectada del dispositivo con el objetivo de encontrar alguna anomalía y mandar algún tipo de mensaje haciendo saber esta condición.

Debido a que la solución se inclinaba a usar herramientas de software libre, permitió inclinarse a la implementación de un protocolo de gestión de redes denominado SNMP. Las características y la forma en que trabaja este protocolo se adecuaron perfectamente a las necesidades y objetivos planteados en el presente trabajo de tesis, y fue ideal para tomarlo como base , en conjunción con otras herramientas, para llegar a la solución deseada.

El protocolo SNMP permite recolectar casi cualquier tipo de información en el dispositivo, almacenando la información en una base de datos maestra. De esta manera se tiene una forma estandarizada de guardar la información, consultarla y modificarla.

El dispositivo que implemente el protocolo SNMP también tendrá la capacidad de consultar su misma información de la base de datos maestra y podrá mandar una alerta en caso de encontrar alguna condición que le haga saber de alguna falla. En el siguiente capitulo se describirá a detalle las características de este protocolo.

Capitulo 2
Protocolo Simple de Gestión de Redes SNMP

A través de este capitulo de ahondará en la forma de operación del Protocolo Simple de Gestión de Redes SNMP, revisando su forma de operación, componentes que lo conforman así como el tipo de operaciones de las que se vale para la administración. Se tratarán las características de las diferentes versiones del protocolo así como las herramientas de las cuales echaremos mano para reforzar el esquema de alertas del departamento. Por último, se cerrará con una serie de conclusiones acerca de los puntos más importantes de todo lo expuesto, situación que permitió dictaminar que versión y características de SNMP fueron utilizadas para dicho proyecto de tesis.

2.1 Administración de Redes TCP/IP y SNMP

Los protocolos de administración de redes fueron desarrollados para permitir a los administradores manejar los dispositivos, dar seguimiento a los eventos críticos de la red y recolectar información relacionada con las tendencias de crecimiento de las rutas de comunicación así como del desarrollo de la red, todo ello desde una estación de administración centralizada (figura 2.1). El primer protocolo de administración de redes no propietario que ha sido ampliamente aceptado fue desarrollado por la comunidad de Internet para el uso del conjunto de protocolos TCP/IP. En un principio solo se crearon para satisfacer necesidades muy básicas, como era el manejo centralizado del crecimiento local de direcciones IP asociadas a ruteadores en una red global Internet.

El grupo de estudio conocido como IETF[18] fue asignado al problema del manejo de ruteadores en Internet. Este grupo diseño una plataforma de trabajo que vino a constituir la fundación del conjunto de protocolos de manejo, lo que ahora conocemos como SNMP.

[18] IETF: Fuerza de Trabajo de Ingeniería en Internet

Dos de los criterios más importantes que son utilizados en los protocolos de manejo de red, y que son parte del diseño de la plataforma SNMP son:

1. La implementación del protocolo no debe aumentar de manera significativa el tráfico de la red para satisfacer las necesidades de administración.

2. El agente de protocolo, en el dispositivo de manejo, no debe disminuir las capacidades de operación básicas o primarias que deba satisfacer el dispositivo en el trabajo que tenga asignado. De manera tal que un mínimo de ciclos de CPU y de memoria deben ser utilizados para propósitos de manejo o administración.

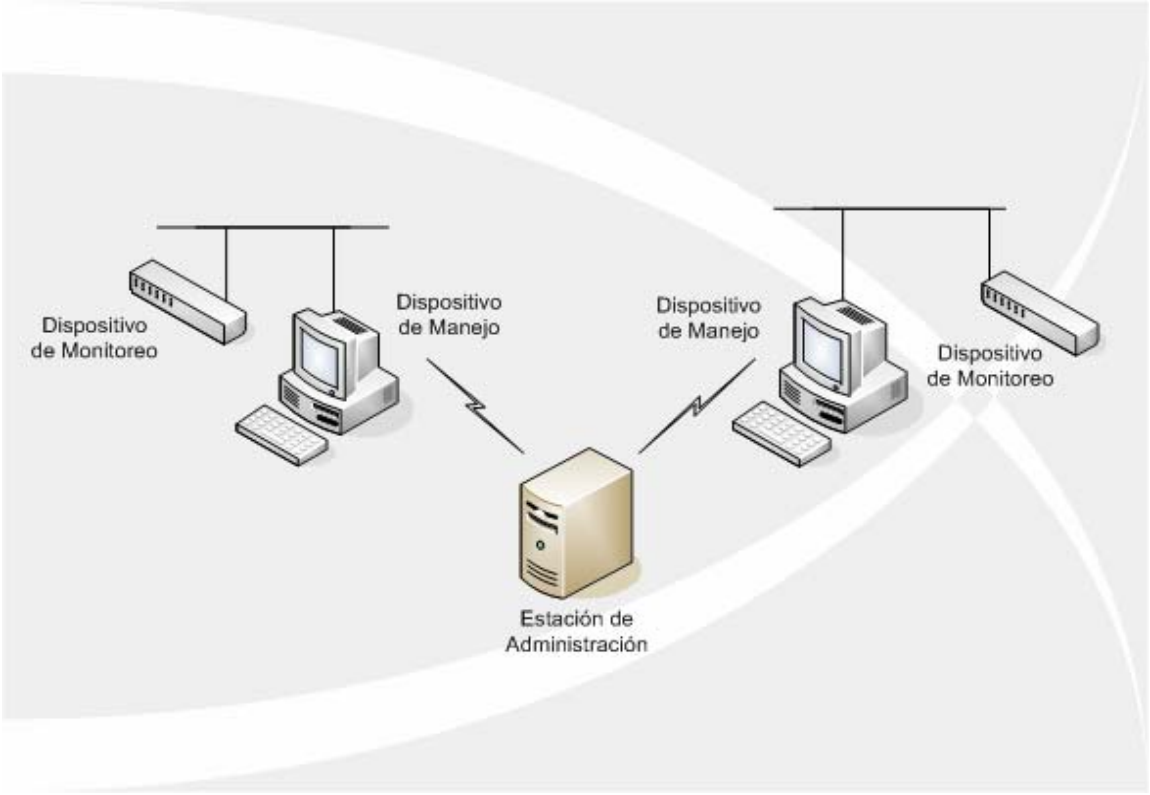


Figura 2.1 Organización y Administración

2.2 Sistema de Administración de Redes (NMS)

Un sistema de administración de red es una colección de herramientas para el monitoreo y control de redes. El esquema de este sistema esta

compuesto por los siguientes elementos (figura 2.2):

Estación de Administración: Es un dispositivo que sirve como interfase entre la persona encargada de la administración de la red y el sistema de administración de red, esta estación debe tener como mínimo:

- Un conjunto de aplicaciones de administración para análisis de datos, recuperación de fallas, detección de alarmas, estadísticas, etc.

- Una interfase a través de la cual la persona encargada de la administración de la red pueda monitorear y controlar la red.

- La capacidad de trasladar los requerimientos del administrador a los dispositivos remotos que conforman la red.

- Una base de datos de información de administración de la red extraída a partir de las bases de datos de todas las entidades administradas en la red.

Agentes: Son un elemento clave en la plataforma, pueden ser: concentradores, ruteadores, puentes y cualquier nodo que este equipado con un agente de "software", el cual puede ser manejado desde una estación remota de administración.

Base de Información de Administración: Las características de los componentes de la red son representadas mediante objetos, normalmente son enteros, pero también pueden almacenar cadenas de caracteres o estructuras más complejas como son las tablas. Cada objeto es, en esencia, una variable de datos que representa un aspecto del agente en cuestión. La colección de estos objetos es denominada MIB.

Es así como la estación de administración ejecuta las funciones de monitoreo recuperando el valor de los objetos MIB, además, la estación de administración puede ejecutar una acción que se ejecute en un agente, o puede cambiar la configuración original de un agente modificando el valor de variables especificas.

Las variables MIB registran el estado de cada red conectada, las estadísticas de tráfico, el total de errores encontrados y el contenido actual de las estructuras internas de datos.

Protocolo de Administración de Red: Es el encargado de enlazar la estación que actúa como administradora y los agentes. El protocolo mayormente usado en la actualidad para la administración de redes TCP/IP es SNMP.

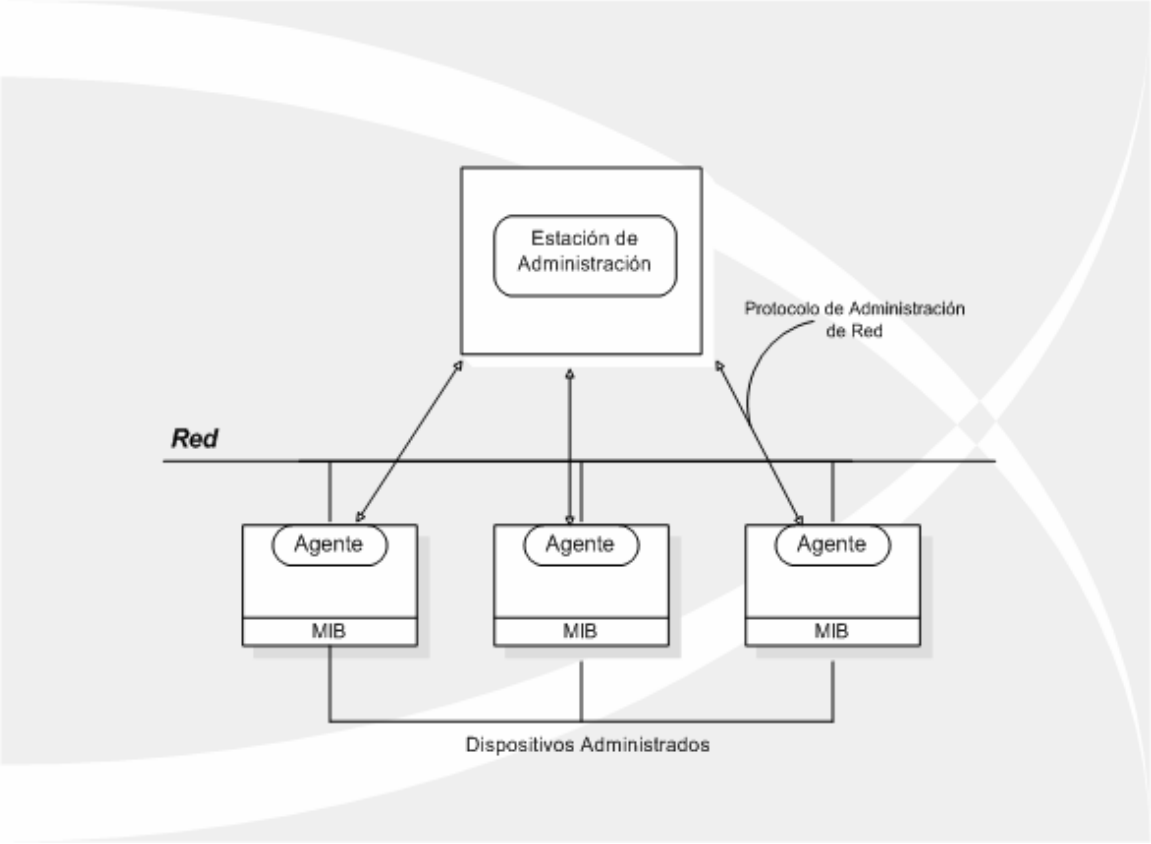


Figura 2.2 Sistema de Administración de Redes NMS

2.2.1 Protocolo Simple de Administración de Red (SNMP)

Diseñado en los años 80`s, su principal objetivo fue el integrar la administración de diferentes tipos de redes mediante un diseño sencillo, y que produjera poca sobrecarga en la red de datos. Se creó para permitir que sistemas heterogéneos y equivalentes hablasen entre sí, generasen informes y permitiesen modificaciones de sus configuraciones sobre una red TCP/IP. El SNMP ha llegado a

ser un protocolo estándar de Internet para la administración de dispositivos sobre redes IP. Actualmente muchas clases de dispositivos soportan el protocolo SNMP, incluyendo ruteadores, switches, servidores, estaciones de trabajo, impresoras, modems y fuentes ininterrumpidas de poder (UPS).

SNMP puede ser usado tanto para una tarea muy simple hasta situaciones muy específicas en cuestiones de administración y monitoreo. El protocolo permite monitorear el estado de los ruteadores, servidores y otros dispositivos conectados a la red, pero también puede ser utilizado para controlarlos o incluso ejecutar automáticamente, de un conjunto definido, alguna tarea en caso de que se presente algún problema en los dispositivos. El rango de información que se puede monitorear es relativamente simple y en términos estandarizados, cuestiones como la cantidad de tráfico de red fluyendo por alguno de los ruteadores o alguna interfaz de red, así como datos específicos del "hardware" en términos específicos del fabricante como puede ser la temperatura dentro del algún ruteador, etc.

A lo largo de los años, desde su diseño original en los años 80 han surgido otras 2 versiones el protocolo, siendo la versión SNMPv3 las más actual y que implementa una serie de características en cuanto a seguridad que las dos anteriores versiones no tienen, aunque fue creada a partir de las versiones anteriores SNMPv1 y SNMPv2. Todas las versiones de SNMP tienen la misma estructura básica y los mismos componentes y conservan la misma arquitectura.

De esta forma el protocolo SNMP ha contribuido en facilitar la cada vez más tediosa y laboriosa tarea del administrador de redes, debido a los adelantos tecnológicos en esta materia las redes actuales son cada vez más complejas al incluir una gran cantidad y variedad de dispositivos, los cuales no solo deben estar funcionando sino trabajando de manera óptima. SNMP proporciona a los usuarios una manera simple de realizar operaciones que permiten a los dispositivos ser administrados remotamente.

2.2.1.1 Capa de operación

SNMP opera en el nivel de aplicación del modelo OSI, utilizando el protocolo de transporte TCP/IP, característica que le permite ignorar todos los aspectos específicos relacionados con el "hardware" sobre el que esta funcionando (fig. 2.3).

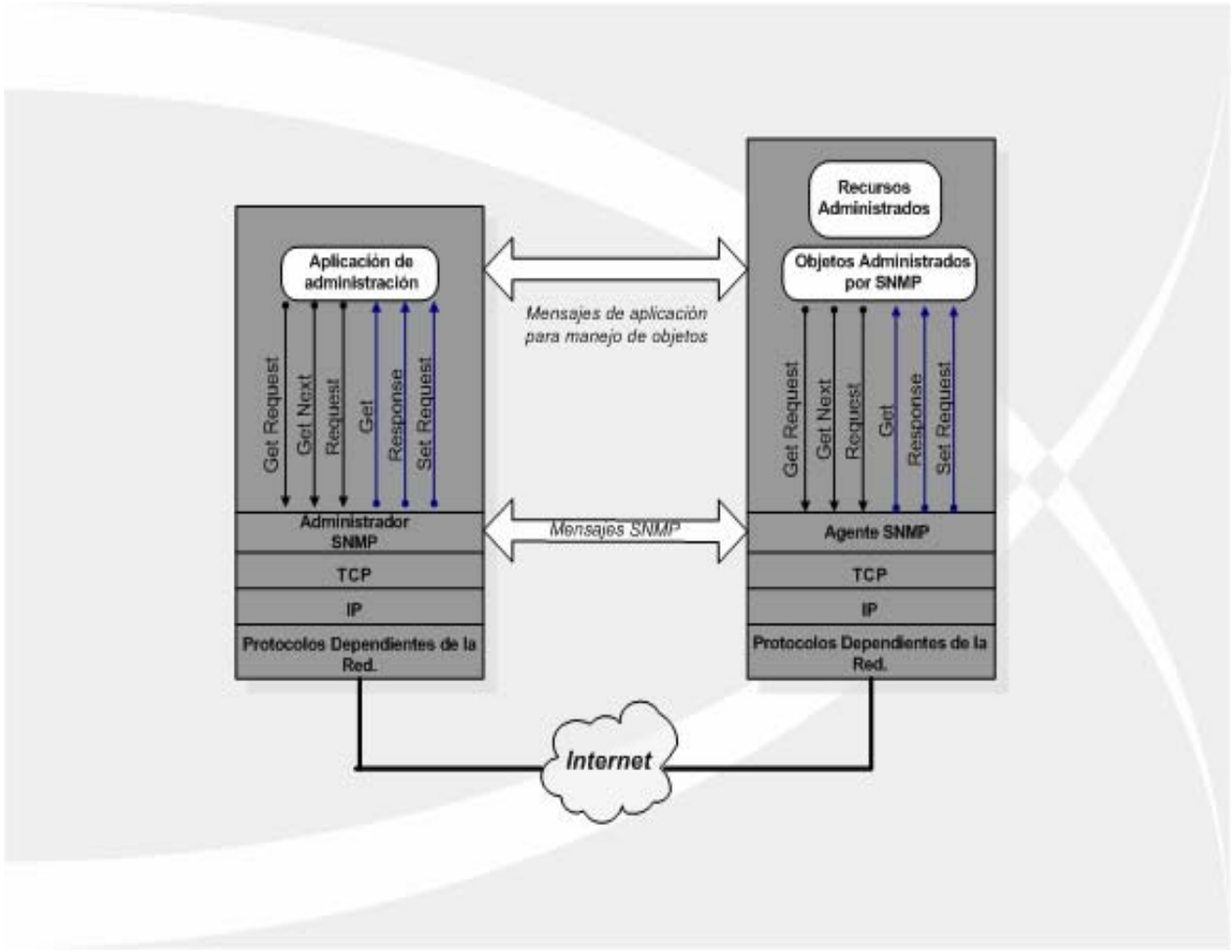


Figura 2.3 Funcionamiento de SNMP

SNMP usa el protocolo UDP[19] como transporte para intercambiar información entre el administrador y los agentes (figura 2.4). Este protocolo fue usado en lugar del protocolo TCP[20] debido a que no es orientado a conexión, esto significa que cuando el NMS cuestiona a alguno de los agentes no puede determinar el número de datagramas que se han perdido en la comunicación. Sin embargo SNMP por medio de un parámetro llamado tiempo de espera agotado puede ejecutar una

[19] UDP: *User Datagaram Protocol* (Protocolo de Datagrama de Usuario)

retransmisión en caso de requerirlo, es decir el NMS al enviar un mensaje UDP al agente espera por una respuesta, en caso de agotar el tiempo establecido el NMS vuelve a transmitir el mensaje. El tiempo de espera así como el número de intentos son parámetros configurables en el NMS.

A pesar de las características del protocolo UDP en cuanto a su no fiabilidad en la entrega de la información, esto no se considera ningún problema en el funcionamiento de SNMP. Esto es debido a que el peor escenario que podría pasar es que la estación de administración cuestione a alguno de los dispositivos de red y no reciba una respuesta, nada impide realizar una nueva petición. Para el caso de las alertas ("traps") la situación es algo diferente. Si un agente envía una alerta ("trap") y ésta nunca llega, el NMS no tiene forma de saber si la alerta alguna vez fue enviada. El agente no sabe si la alerta necesita ser reenviada, por que el NMS no requiere enviar una respuesta de regreso al agente.

La naturaleza del protocolo UDP en cuanto a su no fiabilidad, no repercute en el incremento del tráfico de la red, por lo que el rendimiento de la misma se ve poco afectado ante la implementación del protocolo. SNMP ha sido también implementado sobre TCP, pero solo se utiliza este comportamiento en casos especiales en los que alguien esta desarrollando un agente para un equipo propietario. En redes muy grandes y altamente congestionadas por el intenso tráfico de red, sería muy mala idea implementar algún monitoreo con SNMP utilizando el protocolo TCP para la comunicación, debido a que es un protocolo orientado a conexión.

SNMP usa el puerto 161 del UDP para enviar y recibir peticiones, mientras que el puerto 162 es utilizado para recibir alertas ("traps") de los dispositivos administrados (figura 2.4).

[20] TCP: *Transport Control Protocol* (Protocolo de Control de Transporte)

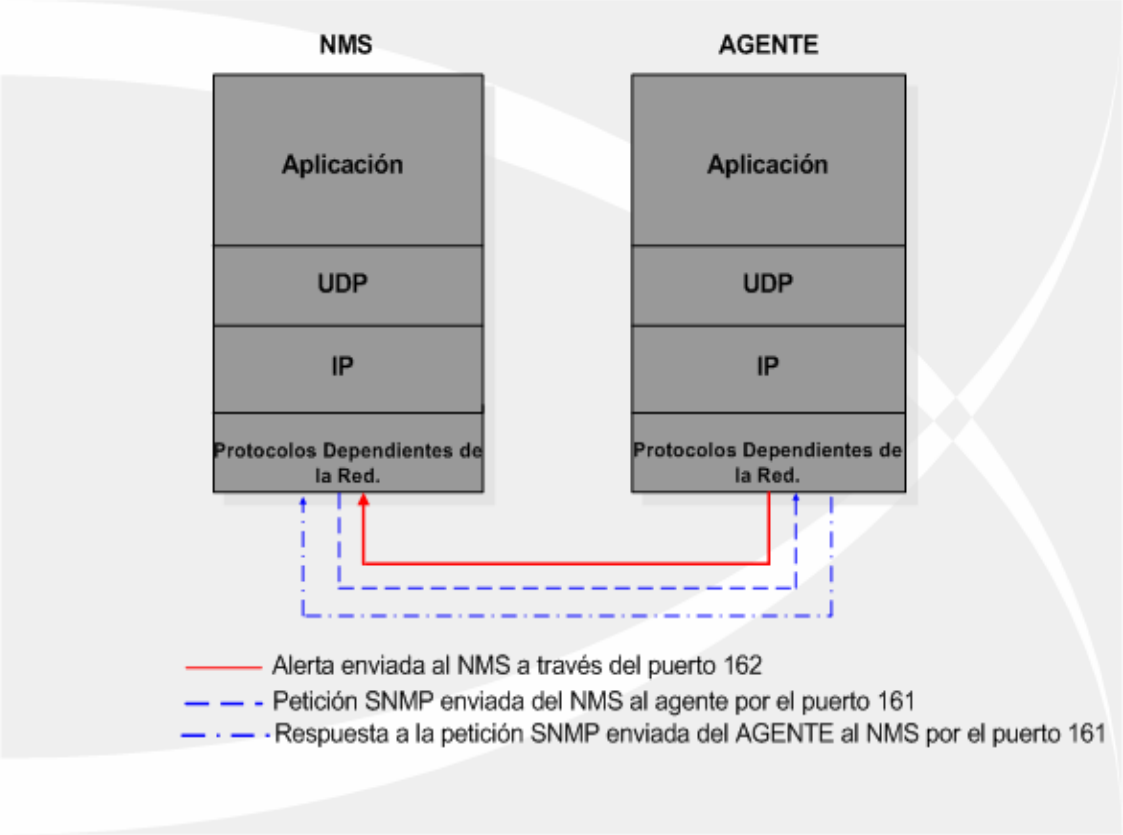


Figura 2.4 Modelo de comunicación TCP/IP y SNMP

La administración se lleva a cabo a nivel de IP, por lo que se pueden controlar dispositivos que estén conectados en cualquier red accesible desde la Internet, y no únicamente aquellos localizados en la propia red local. Evidentemente, si alguno de los dispositivos de encaminamiento con el dispositivo remoto a controlar no funciona correctamente, no será posible su monitorización ni re-configuración.

2.2.1.2 Componentes básicos de SNMP

El protocolo SNMP esta compuesto esencialmente por dos elementos clave: el agente ("*agent*"), y el *administrador* o gestor ("*manager*"). Es una arquitectura cliente – servidor, en el cual el agente desempeña el papel de servidor, mientras que el administrador hace el del cliente (observar la figura 2.2).

Agente: Este es un programa que ha de ejecutarse en cada nodo de red que se desea monitorear o administrar. Ofrece una interfase de todos los elementos que se pueden monitorear. Estos elementos se almacenan en unas estructuras denominadas MIB (Base de Información de Administración). El agente representa la parte del servidor, en la medida que tiene la información que se desea administrar y espera comandos por parte del cliente.

Cabe mencionar los dispositivos administrados, en algunas ocasiones llamados elementos de red, pueden ser ruteadores, servidores de acceso, switches, puentes, concentradores, servidores, computadoras, impresoras o cualquier elemento que tenga instalado el "*software"* de agente SNMP.

Administrador: Es el "*software*" que se ejecuta en la estación encargada de monitorear la red, y su tarea consiste en consultar los diferentes agentes que se encuentran en los de la red y los datos que estos han ido obteniendo.

La posibilidad de ampliación del protocolo va en relación directa con la capacidad de la MIB de almacenar nuevos elementos. Si un fabricante quiere dotar de un nuevo comando a un dispositivo, como puede ser un ruteador, tan sólo tiene que añadir las variables correspondientes a su base de datos (MIB). Casi todos los fabricantes implementan versiones agente de SNMP en sus dispositivos: ruteadores, concentradores, sistemas operativos, etc.

2.2.1.3 Comandos básicos de SNMP

Los dispositivos administrados son monitoreados y controlados usando 4 comandos básicos de SNMP: lectura ("*read*"), escritura ("*write*"), alerta ("*trap*") y operaciones transversales.

El comando *read* es usado por el NMS para monitorear los dispositivos administrados. EL NMS examina diferentes variables que son mantenidas por los dispositivos.

El comando *write* es usado por el NMS para controlar los dispositivos administrados. El NMS tendrá la capacidad de cambiar los valores de las variables almacenadas dentro de los dispositivos.

El comando *trap* es usado por los dispositivos administrados para reportar eventos de forma asíncrona al NMS. Cuando cierto tipo de eventos se manifiestan en el dispositivo administrado, éste envía una alerta al NMS.

Los comandos de *operación transversal* son usados por el NMS para determinar que tipo de variables soporta un dispositivo que esta siendo administrado y para recolectar de forma secuencial la información en tablas de variables, algo parecido a las tablas de ruteo.

2.2.1.4 Versiones del Protocolo SNMP

La IETF es el responsable de definir el estándar de los protocolos que gobiernan el tráfico en la Internet, incluyendo también al protocolo SNMP. Este organismo publica los RFC's que no son mas que documentos de carácter técnico donde se describen los estándares de gran cantidad de protocolos. En dichos documentos se describen las diferentes versiones que han surgido desde la aparición de SNMP, las cuales suman 3 diferentes estándares, SNMPv1, SNMPv2 y SNMPv3. La versión SNMPv3 es el estándar mas actual, ésta versión a diferencia de las 2 anteriores implementa una serie de características en relación a la seguridad que las anteriores no tienen, aunque no hay que soslayar que fue creada a partir de las primeras versiones, SNMPv1 y SNMPv2 y por ésta razón tienen la misma estructura básica y los mismos componentes, además de eso conservan la misma arquitectura. A continuación se entrará más a detalle en cada una de las versiones de SNMP.

2.2.1.4.1 Versión SNMPv1

SNMP versión 1 fue la implementación inicial del protocolo SNMP. Los

estándares relacionados con esta versión están descritos en los RFC's 1155 (Estructura e Identificación de la información de administración para Internet basado sobre el protocolo TCP – IP), 1156 (Base de Información de Administración) y la 1157 que describe al propio protocolo simple de administración de red SNMP. SNMP es un protocolo de aplicación, el cual es encapsulado en el protocolo UDP y fue diseñado para cumplir un propósito en particular, la gestión de la red.

Concepto de comunidad en SNMPv1

Las comunidades en SNMP sirven para establecer una relación de confianza entre el gestor y los agentes, de manera tal que se logra una comunicación para el intercambio de información entre estas dos entidades. Las características de esta relación definen aspectos como la autenticación y el control de acceso a la MIB por parte de un agente.

Un agente puede ser configurado con 3 tipos de comunidades: Solo lectura ("*read-only*"), Lectura escritura ("*read - write*") y alerta ("*trap*"). Los nombres de las comunidades son esencialmente contraseñas, por lo que entre mas complicado sea el nombre de la comunidad puede dificultar que estas sean descubiertas con facilidad, pero es importante mencionar que esto no asegura que las comunidades puedan ser interceptadas debido a que estas se envían en texto plano ya que esta versión del protocolo no implementa ningún tipo de protocolo de autenticación o cifrado para asegurar la confidencialidad de la información.

El agente establece una comunidad para cada combinación deseada de autentificación y control de acceso, y a cada comunidad se le da un nombre único dentro del agente ("*community name"*) Las estaciones de gestión pertenecientes a una comunidad deben emplear ese nombre en todas las operaciones *get* y *set* mediante el uso de comunidades, un agente puede limitar el acceso a su MIB en cuanto a si se puede leer información de la MIB (Vista de la MIB: subconjunto de los objetos de la MIB) y el tipo de modo de acceso: Solo lectura ("*read-only*"), Lectura escritura ("*read - write*").

La combinación de las dos (Nombre de Comunidad y Políticas de acceso) se le denomina perfil de comunidad SNMP ("*SNMP community profile*").

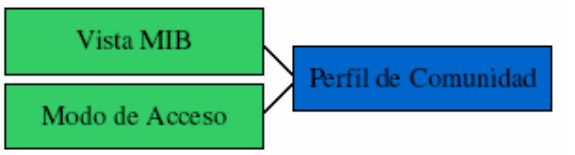


FIGURA 2.6 Perfil de comunidad SNMP

A cada comunidad se le asigna un perfil, denominándose a esta asociación política de acceso SNMP ("*SNMP access policy*").

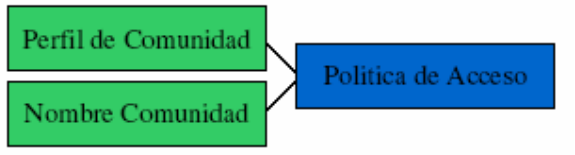


FIGURA 2.7 Política de acceso SNMP

Operaciones en SNMPv1

SNMP es un simple protocolo de petición y respuesta, el cual tiene 2 tipos fundamentales de solicitudes de información:

- Petición (encuesta) de gestor a agente
- Notificación no solicitada (Alertas) de agente a gestor

SNMP realiza en general 4 operaciones en base a las peticiones que realiza a los dispositivos administrados. Este comportamiento es implementado usando una de las operaciones:

- **GET**. Leer información de un agente
- **GET NEXT**. Leer la siguiente información especificada en un agente
- **SET**. Modificar la información de un agente

- **TRAP**. Enviar alertas ("*traps*") del agente al gestor para notificar de algún evento significativo

Tipos de mensajes en SNMPv1

- **GetRequest.** Gestor solicita datos al agente.
- **GetNextRequest .** Gestor solicita el siguiente dato al agente.
- **SetRequest.** Gestor solicita al agente que modifique un valor.
- **GetResponce.** Respuesta del agente al gestor a un 'setrequest'
- **Trap.** Agente que manda datos hacia el gestor.
- **Polling.** Sondeo del gestor a un grupo de agentes.
- **Sondeo dirigido por traps (trap-directed polling) -** Es cuando el gestor se espera hasta que aparezca una alerta ("*trap*") y después inicia a hacer el sondeo en esa determinada zona. También suelen hacerse peticiones a intervalos de tiempo regulares.

Formato de mensajes en SNMPv1

El formato de los mensajes para todas las versiones SNMP son similares excepto en el formato de la PDU[21]. Todos los paquetes contienen dos campos: El numero de versión de SNMP y un nombre de comunidad. El resto depende del tipo del mismo, y se denomina PDU de SNMP.

Para SNMPv1 existen 5 tipos diferentes de PDU: *GetRequest, GetNextRequest, GetResponse, SetRequest, y Trap*.

El formato para las PDU's *GetRequest, GetNext Request, GetResponse y SetRequest* es descrito en la siguiente figura (figura 2.8):

[21] PDU: Unidad de Datos de Protocolo

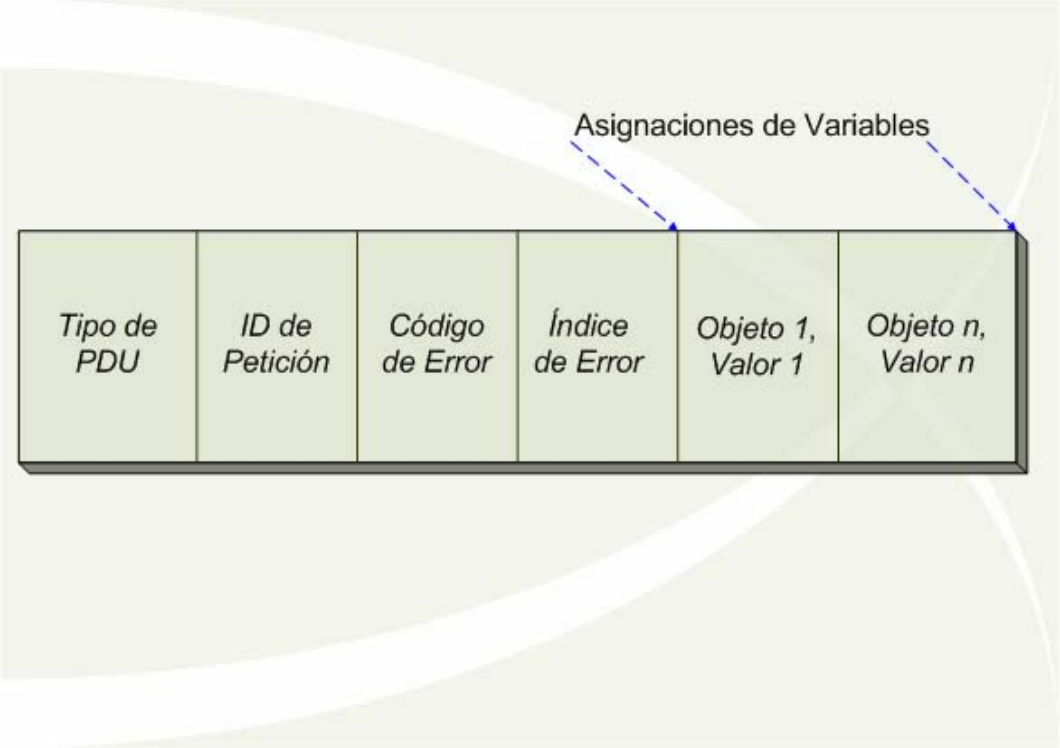


Figura 2.8 Formato de la PDU en SNMPv1

- **Tipo PDU.** Especifica el tipo de PDU transmitida, 0 *GetRequest*, 1 *GetNextRequest*, 2 *GetResponse* y 3 *SetRequest*
- **ID de petición.** Asocia las peticiones SNMP con las respuestas
- **Código de error.** Indica unos de los números de error así como el tipo de error definidos en el protocolo. Solo la operación de respuesta ("SetRequest") modifica la información en este campo, otras operaciones asignan el valor 0
- **Índice de error.** Asocia el error a una instancia de objeto en particular. Solo la operación de respuesta ("*SetRequest*") modifica la información en este campo, otras operaciones asignan el valor 0
- **Asignaciones de variables.** Sirve como un campo de datos en el formato de la PDU de SNMPv1. Las Asignaciones de variables se conforman de nombre variable1 + valor variable1, nombre variable2 + valor variable2, y así sucesivamente.

El formato de la PDU de la alerta (“trap”) es descrito en la siguiente figura (figura 2.9):

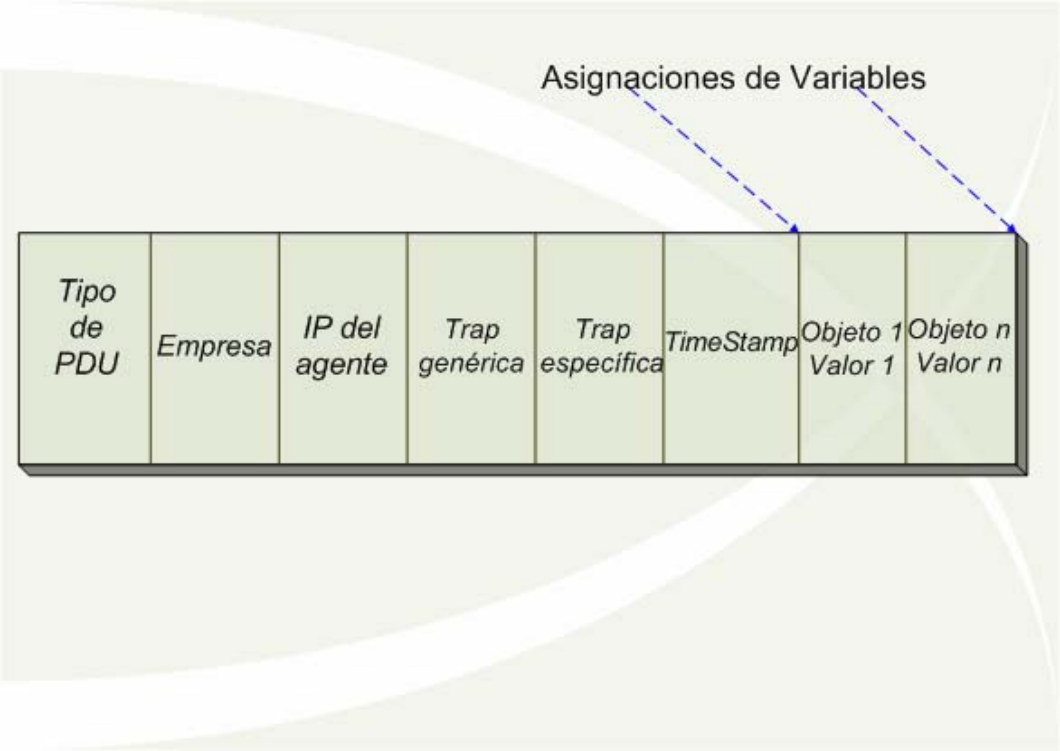


Figura 2.9 Formato de la PDU Trap de SNMPv1

- **Tipo PDU.** Especifica el tipo de PDU, 4 para la alerta (“*trap*”)
- **Empresa.** Identifica el administrador corporativo bajo el cual esta la autoridad de registro de la trap que fue definida
- **IP del Agente.** Es la ip del agente, con lo cual se asegura la identificación
- **Trap Generica.** Campo descriptor del evento que esta siendo reportado
- **Trap Específica.** Usado para describir una trap que no es genérica, identificando a esta como una trap especifica definida por una empresa.
- **TimeStamp.** Tiempo que ha transcurrido entre el último reinicio del sistema y la ultima generación de la trap

Ventajas e inconvenientes de la versión SNMPv1

Ventajas:

- Es un protocolo maduro, estándar aceptado por la industria.
- Esta disponible en gran cantidad de productos.
- Es fácil de implementar y requiere pocos recursos del sistema.

Inconvenientes:

- **Falta de seguridad:** Cualquier estación puede resetear variables con SetRequest, por lo que muchos fabricantes no implementan este comando en sus productos.
- **No hay control de acceso:** Al recibir un PDU un agente no comprueba si ha sido enviado por una estación autorizada. La identificación de comunidad viaja tal cual.
- **Mala utilización del ancho de banda:** No existe la posibilidad de transferir información por bloques.
- **Limitaciones en el mecanismo de traps:** Solo se puede informar de algunos eventos previstos.
- **Limitaciones de tamaño:** No es apropiado para gestionar redes muy grandes (por el sondeo).

2.2.1.4.2 Versión SNMPv2

SNMPv2 fue resultado de la evolución de la primera versión del protocolo. Los RFC's que están relacionados con esta versión es el 1902 (Estructura de la Información de Administración), 1213 (Base de Información de Administración II) y los 1441,1901, 1910 que hace referencia a las diferentes vertientes de la versión SNMPv2.

Con las deficiencias y prestaciones que serian heredadas de la primera versión, en julio del año 1992 se abordaron 2 nuevos protocolos en busca de solucionar los problemas existentes:

- **S-SNMP**. SNMP seguro ("*Secure SNMP*"), mejora en cuanto a los aspectos de seguridad

- **SMP.** Protocolo simple de gestión, trata de mejorar en algunos otros aspectos de la versión 1 y algunos referentes a la seguridad.

Tras la aparición de estos 2 protocolos se decide combinarlos para presentar en Marzo del año 1993 la versión SNMPv2. En esas fechas solo fue propuesto como una recomendación para un estándar de Internet, por lo que necesito de pruebas, concesiones y otros tramites adicionales.

No fue hasta el año de 1996 que la versión SNMPv2 salió completamente, incluyendo nuevas mejoras en cuanto a eficiencia y compatibilidad.

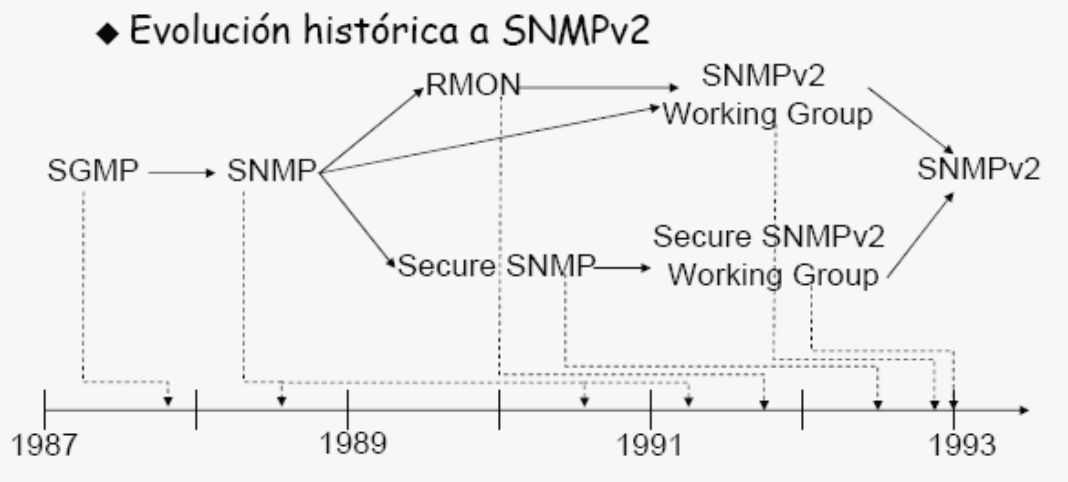


FIGURA 2.10 Evolución histórica a SNMPv2

A pesar de que se contemplaron requerimientos de seguridad, en la definición del protocolo SNMPv2 no aparece ningún estándar asociado a esto, por lo que se decidió asociarlo con otros modelos administrativos referentes a este tema, y es así

que aparecen otras versiones como: SNMPv2c, SNMPv2u y SNMPv2*.

La versión SNMPv2c (basada en comunidades) no incluye mecanismos de seguridad puesto que implementa el mismo modelo administrativo que la versión SNMPv1. Las mejoras fueron una mayor flexibilidad en la administración del control de acceso, permitiendo definición de políticas de acceso, perfiles de comunidad, vistas y derecho de acceso de lectura o escritura a éstas.

La versión denominada SNMPv2u (Versión SNMPv2 basada en usuarios) reutiliza algunos conceptos de la versión SNMPsec, introduciendo la noción de usuario. Con esta versión la comunicación se establece a través de usuarios y no comunidades.

La versión SNMPv2* implementa mecanismos de seguridad, pero no alcanzó el nivel de estandarización y aceptación por el IETF.

Operaciones en SNMPv2

Las operaciones *GET, GETNEXT*, y *SET* usadas en esta versión son exactamente las mismas que las de la primera versión. Sin embargo la versión SNMPv2 incorpora y mejora alguna otras operaciones. La operación de alerta ("*trap*") en SNMPv2 tiene la misma funcionalidad que su antecesora en la versión 1 del protocolo, pero ésta usa un formato diferente de mensaje y su diseño tuvo como objetivo reemplazar la *trap* de la versión SNMPv1.

SNMPv2 define 2 nuevas operaciones de protocolo:

- **GetBulk.** Esta operación es usada por el NMS para recuperar de una manera mucho más eficiente largos bloques de datos que son tratados como múltiples filas en una tabla. *GetBulk* llena el mensaje de respuesta con tantos datos como le sea indicado.
- **Inform.** Esta operación permite a uno de los NMS enviar la

información de alguna alerta ("*trap*") hacia otro NMS, y a su vez recibir una confirmación de recepción por parte del NMS receptor.

Formato de mensajes en SNMPv2

El formato de los mensajes en las diferentes versiones del protocolo es muy similar, la única variante es el formato de la PDU.

Para SNMPv2 los formatos PDU de los mensajes de las operaciones *GetNext, Inform, Response, Set,* y *Trap* están descritos en la figura 2.11.

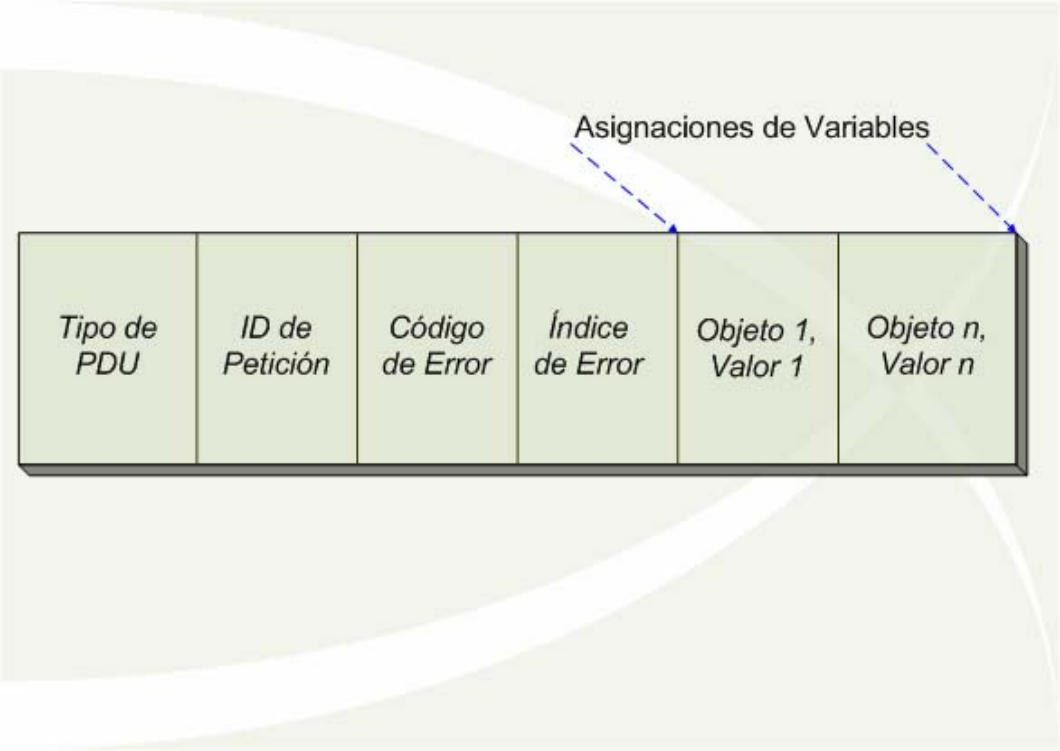


Figura 2.11 Formato de la PDU en SNMPv2

- **Tipo PDU.** Especifica el tipo de PDU transmitida (GetRequest, GetNextRequest, GetResponse, SetRequest, Inform o Trap)

- **ID de petición.** Asocia las peticiones SNMP con las respuestas

- **Código de error.** Indica unos de los números de error así como el tipo de error definidos en el protocolo. Solo la operación de respuesta ("*SetRequest*") modifica la información en este campo, otras operaciones asignan el valor 0
- **Índice de error.** Asocia el error a una instancia de objeto en particular. Solo la operación de respuesta ("*SetRequest*") modifica la información en este campo, otras operaciones asignan el valor 0
- **Asignaciones de variables.** Sirve como un campo de datos en el formato de la PDU de SNMPv2. Las Asignaciones de variables se conforman de nombre variable1 + valor variable1, nombre variable2 + valor variable2, y así sucesivamente.

El formato de la PDU de la operación *GetBulk* esta descrito en la figura 2.12:

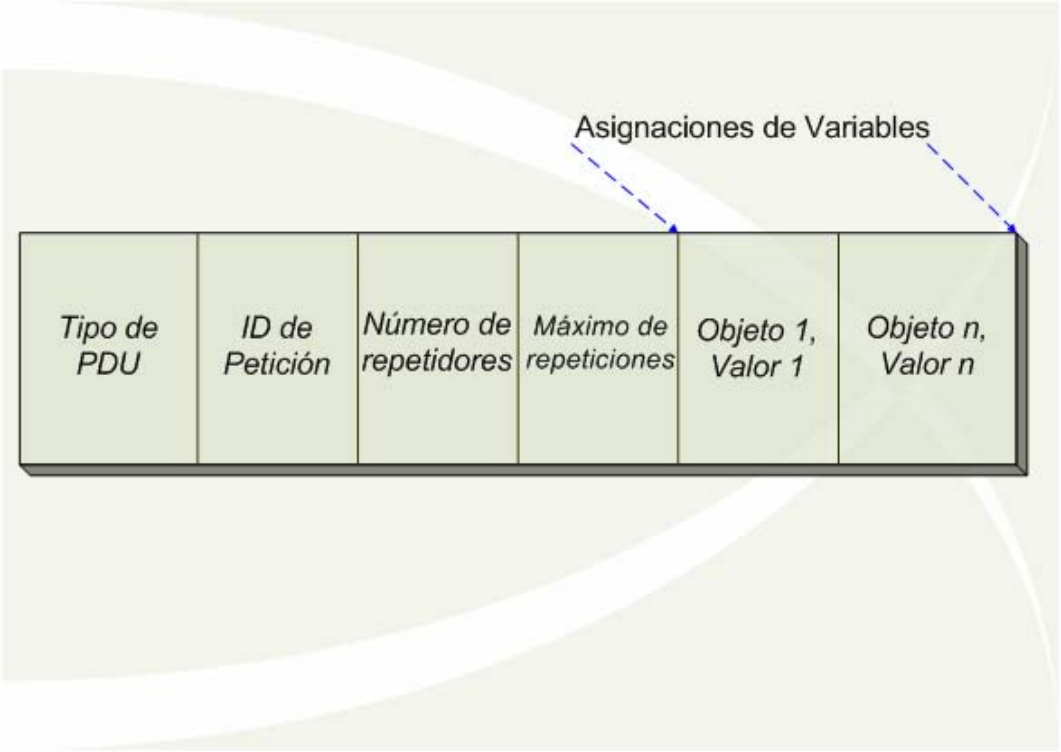


Figura 2.12 Formato de la PDU GetBulk en SNMPv2

- **Tipo PDU.** Especifica el tipo de PDU transmitida (GetRequest, GetNextRequest, GetResponse, SetRequest, Inform o Trap)

- **ID de petición.** Asocia las peticiones SNMP con las respuestas
- **Numero de repetidores.** Específica el numero de instancia de objetos en el campo de variables que deberán ser recuperadas, no más de una al principio de la petición. Este campo es utilizado cuando alguna de las instancias son objetos escalares con solo una variable.
- **Máximo de repeticiones.** Define el número máximo de veces que otras variables de las especificadas por el campo "No repeticiones" deberán ser recuperadas.
- **Asignaciones de variables.** Sirve como un campo de datos en el formato de la PDU de SNMPv2. Las Asignaciones de variables se conforman de nombre variable1 + valor variable1, nombre variable2 + valor variable2, y así sucesivamente.

Ventajas de SNMPv2

Algunas de las ventajas de esta versión con respecto a la primera son:

- Se mejoran los tipos de datos (Se mejoran contadores de 64 bits, gauge).
- Se mejoran la documentación de cada objeto (Se añade la cláusula *Units*)
- Se crean nuevas convenciones para crear y eliminar filas en una tabla.
- Se implementan dos tipos de MIB.s:

1) MIB SNMP v2 (Información de agentes, protocolos, configuración, agentes, gestores) Incluye 5 grupos:

i. Estadísticas SNMPv1
ii. Estadísticas SNMPv2
iii. Recursos de objetos (objetos que son configurados por el gestor, y ayudan a que una entidad trabaje como agente)

iv. Grupo de *Traps* (objetos que permiten a un agente generar PDU's del tipo *traps* SNMPv2)
v. Permite que un grupo de gestores se comuniquen y colaboren para el uso de una operación *SET*.

2) M2M (Manager To Manager) Para sistemas de gestión distribuidos, incluye 2 grupos:

i. snmp-ALARM, son objetos que permiten definir y configurar umbrales de alarmas SNMPv2, tanto para agentes y gestores.

ii. Snmp-EVENT, son objetos que permiten definir y configurar eventos en una entidad realizando un papel doble.

- Se definieron dos nuevas PDU

1) Intercambio de información entre gestores: *InformRequest*

2) Transferencia eficiente de grandes bloques de datos: *GetBulkRequest*

Inconvenientes de SNMPv2

A pesar que las mejoras en referencia a la primera versión del protocolo son significativas, el principal inconveniente es el tema de la seguridad que implementa el protocolo.

Debido a la falta de consenso en cuanto al método a utilizar, no fueron implementadas las mejoras de seguridad con respecto a la autenticación y encriptación de los datos, por lo que SNMPv2 no ofrece la confianza de asegurar la confidencialidad de los datos, por lo que es necesario auxiliarse de otras herramientas para lograr un esquema de seguridad aceptable.

2.2.1.4.3 Versión SNMPv3

Debido a que en la versión final de SNMPv2 no se incluyeron las mejoras de seguridad, la principal preocupación al desarrollar la siguiente versión del protocolo fue la incorporación de estas mejoras. Es así como a finales del año 1998 surge SNMPv3, la cual es la versión más actual de SNMP (figura 2.13).

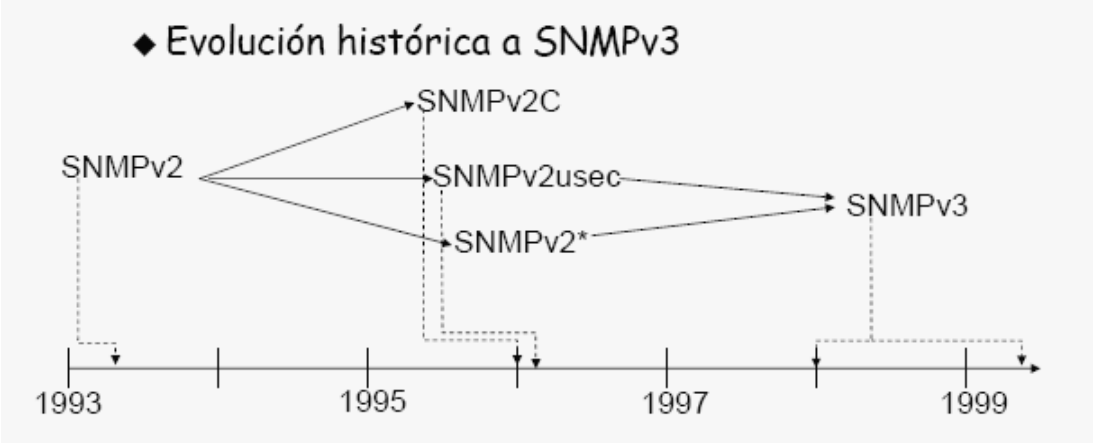


FIGURA 2.13 Evolución a snmpv3

SNMPv3 incorpora los soportes de seguridad que no se incluyeron en SNMPv2 , no existen cambios en el protocolo ni tampoco se definen nuevas operaciones, esta versión del protocolo SNMP hereda muchos componentes que fueron creados para las versiones SNMPv1 y SNMPv2, incluyendo las operaciones de protocolo , y los tipos y formatos de PDU.

SNMPv3 también introduce la habilidad para configurar dinámicamente al agente SNMP usando los comandos SNMP – SET sobre los objetos de la MIB que representan la configuración del agente. Este soporte de configuración dinámica permite la adición, supresión y modificación de los parámetros de configuración ya sea en forma local o remota.

A pesar de que en SNMPv3 no se realizaron cambios del lado del protocolo, si se agregó seguridad criptográfica, esto trajo consigo una serie de desarrollos que han introducido una serie de nuevos conceptos, convenciones textuales y terminología.

La máquina de SNMPv3

El cambio más importante es que la versión SNMPv3 abandona la idea de los administradores y agentes. Ambos, tanto administradores como agentes son llamados entidades SNMP. Cada entidad consiste en una máquina SNMP y una o más aplicaciones SNMP.

La máquina SNMP esta compuesta de cuatro piezas (figura 2.14): el Distribuidor (*Dispatcher)*, el Subsistema de Procesamiento de Mensaje (*Message Processing Subsystem)*, el Subsistema de Seguridad (*Security Subsystem)* y el Subsistema de Control de acceso (*Access Control Subsystem)*. El trabajo del *Dispatcher* es enviar y recibir mensajes. Esta parte intenta determinar la versión de cada mensaje recibido (ya sea la v1, v2 o v3) y si esta es soportada, controla el mensaje y lo envía al *Message Processing Subsystem*. El *Dispatcher* también envía mensajes SNMP a otras entidades.

El *Message Processing Subsystem* prepara el mensaje para ser enviado y extraer los datos de los mensajes recibidos.

El *Security Subsystem* proporciona los servicios de autentificación y privacidad. La autentificación usa ya sea las cadenas de comunidad tanto para la versión 1 y 2 así como usuarios basados en autentificación de la versión 3. Los usuarios basados en autentificación utilizan los algoritmos MD5[22] o SHA[23] para autenticar. El servicio de privacidad utiliza el algoritmo DES[24] para encriptación de los mensajes SNMP.

El Access Control *Subsystem* es el responsable del controlar el acceso a los objetos de la MIB. Con esto se pude controlar que objetos un usuario puede acceder así como que tipo de operaciones puede ejecutar sobre ellos.

[22] MD5: *Message-Digest Algorithm* 5 (Algoritmo de Resumen del Mensaje 5)
[23] SHA: *Secure Hash Algorithm* (Algoritmo de Hash Seguro)
[24] DES: *Data Encryption Standard* (Estándar de Encriptación de Datos)

Las aplicaciones SNMPv3

Las aplicaciones SNMP están divididas en (figura 2.14): generador de comando (*command generator*), contestador de comando (*command responder*), creador de notificación (*notification originator*), receptor de notificación (*notification receiver*) y expedidor del apoderado (*Proxy forwarder*).

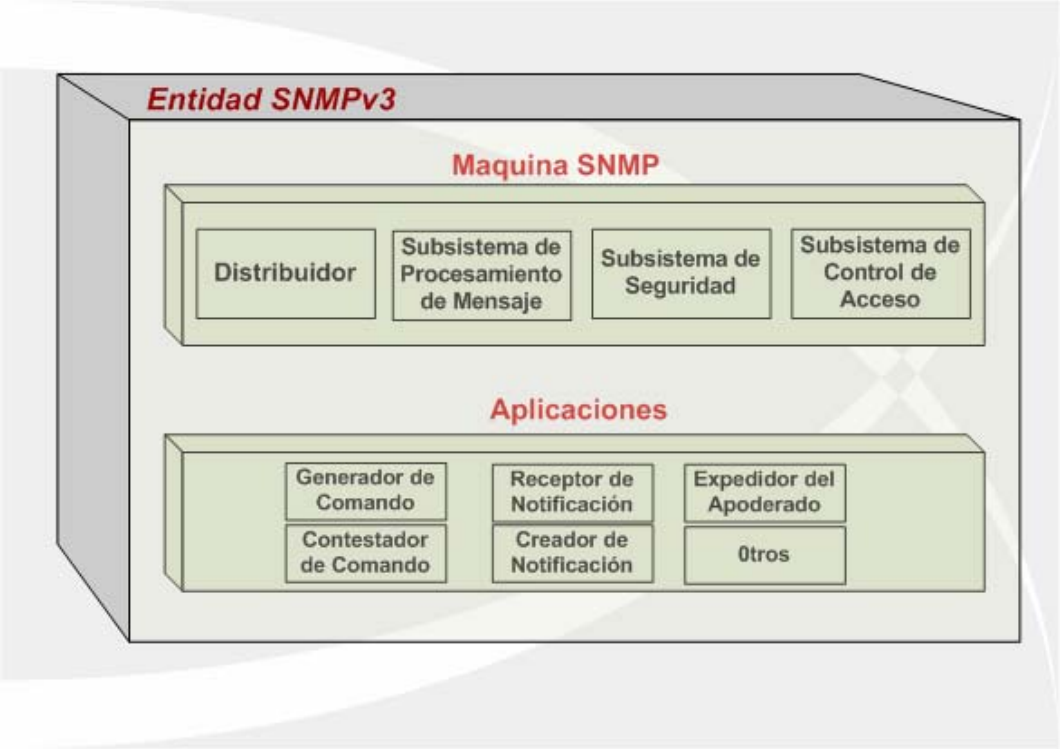


Figura 2.14 Entidad SNMPv3

La seguridad en SNMPv3

Entre los cambios más significativos hechos en la versión SNMPv3 incluyen una manera más flexible de definir métodos de seguridad y parámetros para permitir la coexistencia de múltiples técnicas de seguridad.

SNMPv3 incluye los siguientes mecanismos de seguridad:

- Protección con enmascaramiento de datos

- Modificación de flujo de mensajes
- Revelación de contenidos
- Modificación de mensajes.

Sin embargo no incluye protección a ataque DoS[25] y el análisis del tráfico.

La arquitectura de SNMPv3 introduce 2 nuevas herramientas:

1. Modelo de seguridad orientado a usuarios ("*USM User-Based Security Model*")que se ejemplifica en la figura 2.15.
    - Seguridad ante la Modificación de información, Suplantación, Modificación del flujo de mensaje, Revelación de información.
    - Funciones de cifrado (CBC-DES) y autentificación (MD5 o SHA)
    - Motores autorizados y no autorizados
    - Mecanismos de temporización

2. Modelo de configuración orientado en vistas ("*VACM View-Based Access Control Model*") ejemplificado en la figura 2.16.

    - Administración para crear diversos perfiles de acceso a la MIB del agente

[25] DoS: Denegación de servicio

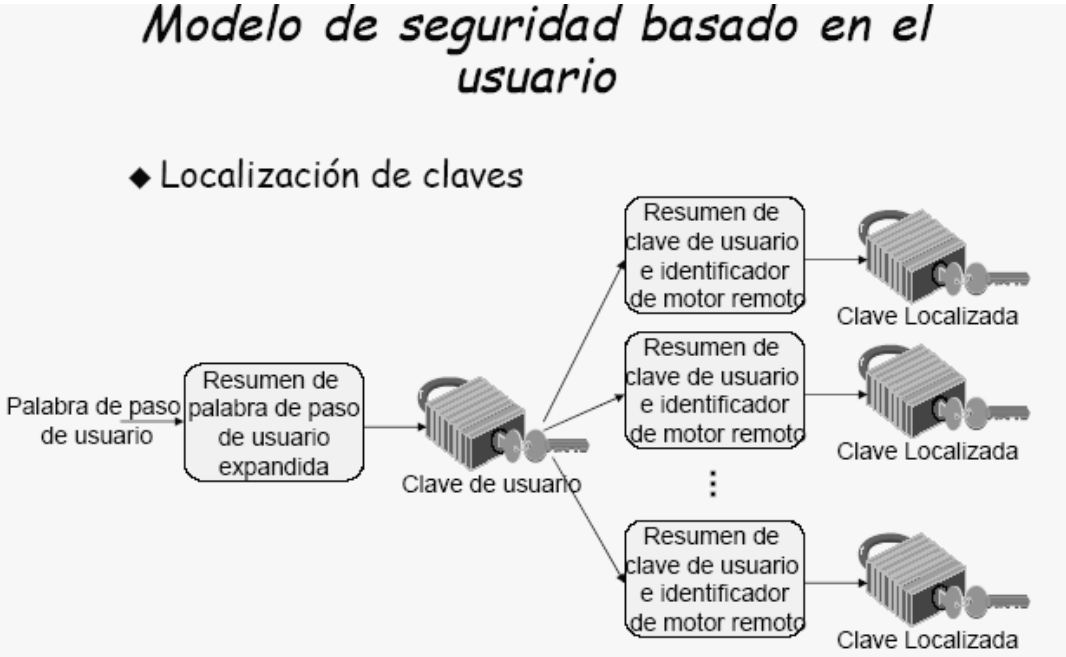


FIGURA 2.15 MODELO USM

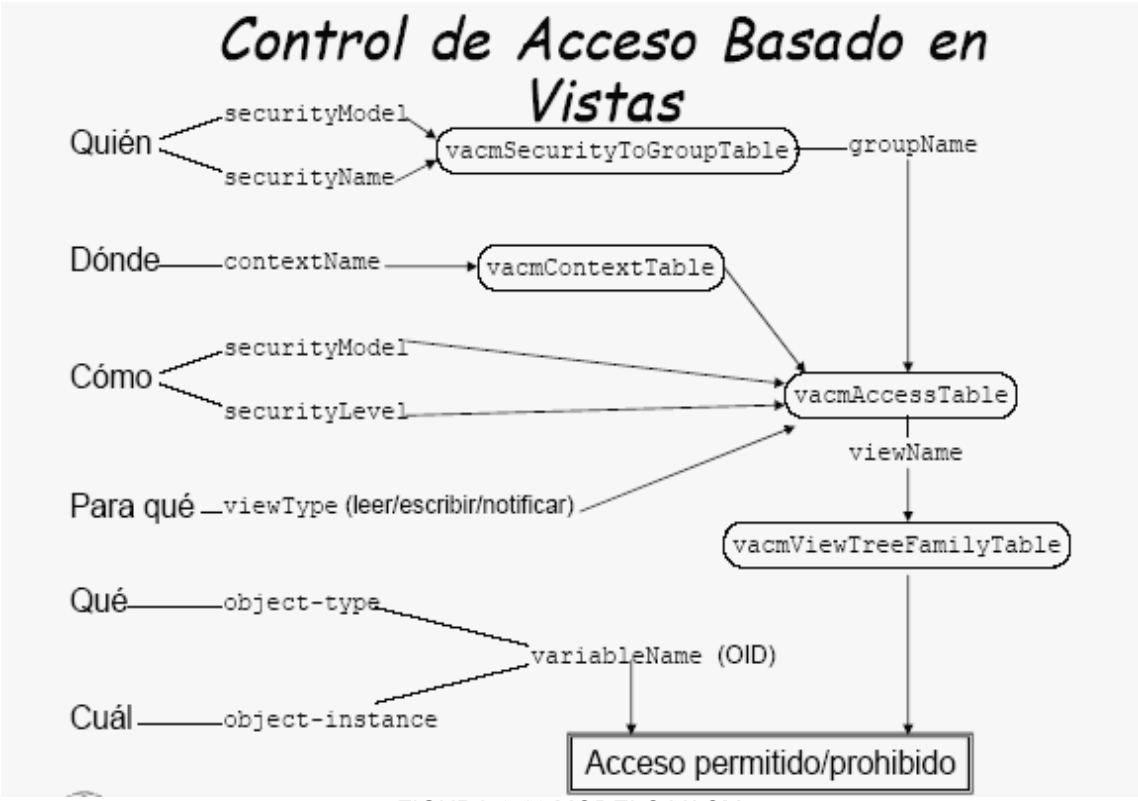


FIGURA 2.16 MODELO VACM

Formato de mensajes en SNMPv3

El formato en general de los mensajes en esta versión, sigue la misma idea que en las versiones anteriores, es decir es una envoltura que contiene una cabecera y una PDU encapsulada (figura 2.17). Sin embargo en SNMPv3 este concepto es fuertemente redefinido. Los campos en la cabecera han sido divididos tanto en aquellos que son distribuidos con características de seguridad así como los que no tratan con materias de seguridad. Los campos de no seguridad (*non-security*) son comunes para todas las implementaciones de SNMPv3, mientras que el uso de los campos de seguridad pueden ser asociados por cada modelo de seguridad de SNMPv3, y procesado por cada módulo en una entidad SNMP que distribuya con seguridad.

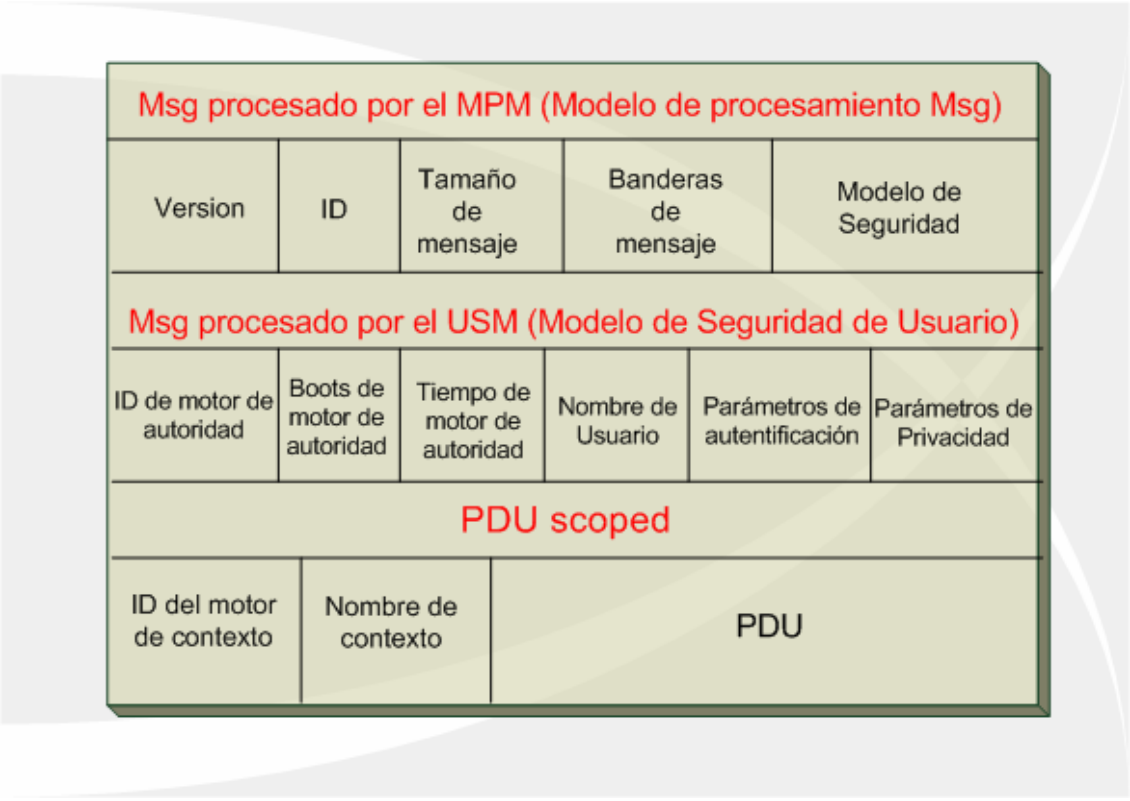


Figura 2.17 Formato de mensaje SNMPv3

- **Versión.** Para SNMPv3 es el número 3

- **ID.** Un identificador único usado entre dos entidades SNMP para coordinar la petición y respuesta de mensajes
- **Tamaño de mensaje.** Tamaño máximo de un mensaje, especificado en octetos, soportado por el emisor del mensaje.
- **Banderas de mensaje.** Una cadena de caracteres, especificando en octetos tres banderas, siendo los últimos 3 bits los más significativos. Con estas banderas se establece el proceso de control del mensaje que se llevará a cabo, las banderas son las siguientes: *Reserved, ReportaFlag, privFlag, authFlag*
- **Modelo de seguridad.** Un identificador único para identificar que modelo de seguridad fue usado por el emisor y por consiguiente que modelo de seguridad deberá se usado por el receptor para procesar el mensaje
- **ID de motor de autoridad.** Es el **"***snmpEngineID*" del motor de autoridad de SNMP involucrado en el intercambio de este mensaje. Así que éste valor se refiere a la fuente de procedencia de una *Trap, Response o Report*. Y hacia que destino para un *Get, GetNext, GetBulk, Set o Inform*.
- **Boots de motor de autoridad.** Es el valor "*snmpEngineBoots*" del motor de autoridad de SNMP involucrado en el intercambio de este mensaje.
- **Tiempo de motor de autoridad.** Es el valor del "*snmpEngineTime*" del motor de autoridad de SNMP involucrado en el intercambio de este mensaje.
- **Nombre de usuario.** Es el principal usuario sobre el cual el mensaje esta siendo intercambiado.
- **Parámetros de autentificación.** Este valor es nulo si la autentificación no esta siendo utilizada, de otra manera este es el parámetro de autentificación para el intercambio del mensaje.
- **Parámetros de privacidad.** Este valor es nulo si la privacidad no esta siendo utilizada, de otra forma este es el parámetro de privacidad para el intercambio de información.
- **ID del motor de contexto.** Usada para identificar que aplicación

procesará la PDU que será enviada.

- **Nombre de contexto.** Un identificador de objeto especificando el contexto en particular asociado con la PDU.
- **PDU.** Son los tipos de PDU para SNMPv3, éstos son los mismos del protocolo SNMPv2.

2.2.2 Base de Información de Administración MIB

La información que puede ser recolectada a través del protocolo SNMP, en los diferentes dispositivos administrados de la red es almacenada y descrita en una Base de Información de Administración, la cual es una base de datos jerárquica utilizada por SNMP con el propósito de almacenar un conjunto de objetos que representarán las características del componente gestionado.

2.2.2.1 Estructura de Información de Administración SMI

Las MIB's son determinadas por lo que se denomina Estructura de Información de Administración, mejor conocida como SMI, la cual representa el marco general dentro del cual se define y se construye la MIB, identifica los tipos de datos que son utilizados en el MIB y cuáles recursos dentro del MIB son representados y nombrados. Existen tres elementos claves en la especificación del MIB, éstos son:

- En el nivel más bajo, especifica los tipos de datos que pueden ser almacenados.
- En el nivel medio, especifica una técnica formal para definir objetos y tablas de objetos.
- En el nivel más alto, provee un esquema para asociar un identificador único con cada objeto en el sistema.

El SMI proporciona una manera de definir los objetos que pretendan ser administrados, mientras que la MIB es la definición (usando la sintaxis SMI) de los

propios objetos. Esto es parecido a un diccionario, en el cual muestras como deletrear una palabra y entonces proporcionas su significado o definición, una MIB define el nombre textual para un objeto administrado y explica a su vez su significado.

La Estructura de Información de Administración versión 1 conocida como SMIv1, es descrita en el RFC 1155, esta define de forma precisa como los objetos administrados son nombrados y especificados con un tipo de dato en particular. La Estructura de Información de Administración versión 2 conocida como SMIv2, es descrita en el RFC 2578, y proporciona una serie de mejoras con respecto a la primera versión.

La definición de los objetos que pretenden ser administrados puede ser partida en 3 atributos:

Nombre

El nombre o identificador de objeto (denominado OID), únicamente define un objeto administrado. Los nombres comúnmente aparecen en 2 formas: numérico y el "humanamente legible". En cualquiera de los casos son largos e inconvenientes.

Tipo y sintaxis

El tipo de dato de un objeto administrado es definido usando un subconjunto de Sintaxis de Notación Abstracta versión 1 (denominada ASN.1). ASN.1 es la manera de especificar como los datos son representados y transmitidos entre los administradores y agentes dentro de un contexto SNMP. La buena noticia acerca de ASN es que la notación es independiente de la máquina. Esto significa que 2 maquinas con sistemas operativos completamente diferentes pueden comunicarse sin riesgos tales como la incompatibilidad en el ordenamiento o clasificación de bytes.

Codificación

Una sola instancia de un objeto administrado es puesta en código dentro de una cadena de octetos usando las normas BER[26]. BER define como los objetos son codificados y descifrados, de esta manera pueden ser transmitidos por un medio de transporte como el Ethernet.

2.2.2.2 Nombrando y definiendo OID's

Los objetos administrados son organizados dentro de una estructura arborescente. Esta estructura es la base para el esquema de nombramiento de SNMP. Un identificador de objeto esta compuesto por una serie de enteros basándose en los nodos del árbol, separados mediante puntos (.). A pesar de que existe un formato por medio del cual se hace amigable la lectura un OID, no es otra cosa más que una cadena de nombres separadas mediante puntos, donde cada nombre representa un nodo del árbol. Así que es indiferente usar números enteros o nombres para encontrar un nodo. La figura muestra una estructura arborescente con un nodo raíz y las ramas que se encuentran por debajo de este.

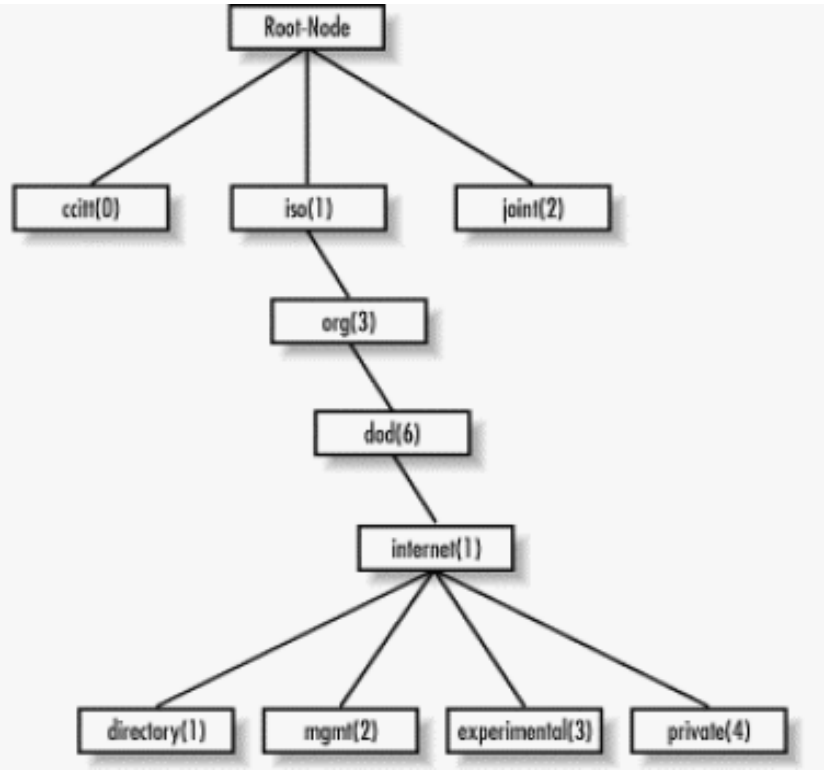


FIGURA 2.18 Estructura arborescente de la MIB

Es este objeto de árbol el nodo de donde parten todas las demás ramas es llamado nodo raíz (root). Un subárbol esta formado por ccitt (0), iso (1) y joint (2). En esta figura, iso (1) es el único nodo que contiene un subárbol más, mientras que los otros 2 son únicamente nodos terminales. El resto del árbol que será de interés para ilustrar el ejemplo es el subárbol formado por los nodos iso (1).org (3).dod (6).internet (1) el cual es representado mediante el OID de la forma numérica 1.3.6.1 o en su forma amigable iso.org.dod.internet. Cada objeto administrado tiene un OID numérico que a su vez esta asociado a un nombre textual. Aquí hay un ejemplo que trata de la definición del subárbol Internet:

```
internet      OBJECT IDENTIFIER ::= { iso org(3) dod(6) 1 }
directory     OBJECT IDENTIFIER ::= { internet 1 }
mgmt          OBJECT IDENTIFIER ::= { internet 2 }
experimental  OBJECT IDENTIFIER ::= { internet 3 }
private       OBJECT IDENTIFIER ::= { internet 4 }
```

[26] BER: *Basic Encoding Rules* (Reglas Básicas de Codificación)

La primera línea declara a internet como la OID formada por la trayectoria 1.3.6.1, la cual es definida como un subárbol de iso.org.dod o 1.3.6. Las últimas 4 declaraciones son similares, pero estas definen las otras ramas que pertenecen a internet. Para la rama directory, la notación " { Internet 1 } " significa que es parte del subárbol Internet y su OID es 1.3.6.1.1. El OID para mgmt es el 1.3.6.1.2 , etc.

Actualmente existe una rama bajo el subárbol privado (private). Este es usado para proporcionar a los vendedores de hardware y software la capacidad de definir sus propios objetos privados para cualquier tipo de hardware y software que ellos deseen administrar bajo SNMP. La definición SMI es la siguiente:

enterprises OBJECT IDENTIFIER ::= { private 1 }

Podemos citar el ejemplo del número de empresa privado reservado para los sistemas de la compañía Cisco, dicho numero es el 9 de manera que OID base para el espacio de sus objetos privados es definido como iso.org.dod.internet.private.enterprises.cisco o 1.3.6.1.4.1.9. Cisco es libre de hacer todo lo que desee dentro de su rama privada. De esta manera se enriquece la información de administración que puede ser manejada en los dispositivos gestionados.

El atributo SYNTAX proporciona para la definición de objetos administrados a través de un subconjunto de ASN.1. En el caso de la SMIv1 mucho tipos diferentes de datos que son superiores para la administración de redes y dispositivos de red. Es importante tener en cuenta que esos tipos de datos son simplemente una manera de definir que tipo de información un objeto administrado puede manejar. A continuación se presenta una tabla descriptiva de los tipos de datos soportados por la SMIv1:

Tipo de Dato	Descripción
INTEGER	Un número de 32 bits usualmente usado para especificar los tipos enumerados dentro del

Tipo de Dato	Descripción
	contexto de un solo objeto administrado
OCTET STRING	Una cadena de 0 o más octetos (comúnmente conocidos como bytes) generalmente usado para representar cadenas de texto, pero también algunas veces usado para representar direcciones físicas.
Counter	Un número de 32 bits con un mínimo valor 0 y como máximo valor 2 a la 32. Principalmente es usado para manejar información como el número de octetos enviados y recibidos en una interfaz o el número de errores y desechos observados sobre una interfaz de red. Cuando un agente es reiniciado, todos los valores de tipo Counter deben ser iniciados a 0.
OBJECT IDENTIFIER	Una cadena de números decimales, separada por puntos que representa un objeto administrado del objeto árbol. Por ejemplo 1.3.6.1.4.1.9 representa el OID de empresa privado de los sistemas Cisco.
NULL	Actualmente no usado en SNMP.
SEQUENCE	Define una lista que contiene cero o más tipos de datos ASN.1.
SEQUENCE OF	Define un objeto administrado que esta hecho de una secuencia de tipos ASN.1.
IpAddress	Representa una dirección IPv4 de 32 bits. Ni la SMIv1 o SMIv2 discuten las direcciones IPv6 de 128 bits.
NetworkAddress	Igual que el tipo de dato IpAddress, pero puede representar diferentes tipos de dirección de red.
Gauge	Un número de 32 bits con un mínimo valor 0 y como máximo valor 2 a la 32. A diferencia de Counter, el tipo Gauge puede aumentar o disminuir a voluntad, pero no puede exceder del máximo valor. La velocidad de la interfaz de red de un ruteador es medida con un tipo de dato Gauge.
TimeTicks	Un número de 32 bits con un mínimo valor 0 y como máximo valor 2 a la 32. Este tipo mide el tiempo en centésimas de segundo. El tiempo que ha estado funcionando un equipo es medido con este tipo de dato.
Opaque	Permite a cualquier código ASN.1 ser llenado dentro de un OCTECT STRING.

El propósito de estos tipos de datos es colaborar en la definición de los objetos administrados.

Para la SMIv2 se incorporaron nuevos tipos de datos, los cuales son descritos en la siguiente tabla.

Tipo de Dato	Descripción
Integer32	Lo mismo que un tipo INTEGER.
Counter32	Lo mismo que un tipo Counter.
Gauge32	Lo mismo que un tipo Gauge.
Unsigned32	Representa valores decimales en el rango de 0 a 2 a la 32.
Counter64	Similar al tipo Counter32, pero su máximo valor es 18,446,744,073,709,551,615.
BITS	Una enumeración de los bits nombrados no negativos.

Tabla de los tipos de datos soportados por la SMIv2

Cada objeto en la SMIv1 tiene el siguiente formato:

<name> OBJECT-TYPE
SYNTAX <datatype>
ACCESS <cualquiera de las siguientes read-only, read-write, write-only o not accessible>
STATUS <cualquiera de las siguientes mandatory, optional o obsolete>

DESCRIPTION
"Una descripción textual del objeto administrado en particular"
::= { <OID único que define este objeto> }

La definición de objetos en la SMIv2 tiene un ligero cambió con respecto a la

primera, existen nuevos campos opcionales, dando la posibilidad de tener más control sobre el acceso a un objeto. A continuación se presenta la sintaxis para definir un objeto en la SMIv2, los cambios fueron puestos en letras negritas:

<name> OBJECT-TYPE
SYNTAX <datatype>
UnitsParts <Optional, ver la tabla descriptiva>
MAX-ACCESS <Ver la tabla descriptiva>
STATUS <Ver la tabla descriptiva>

DESCRIPTION
"Una descripción textual del objeto administrado en particular"
AUGMENTS { <nombre de la tabla> }
::= { <OID único que define este objeto> }

La siguiente tabla muestra una breve descripción de las mejoras en la definición de los objetos hechas en la SMIv2.

Definición de Objeto Mejorada	Descripción
UnitsParts	Una descripción textual de las unidades (por ejemplo, segundos, milisegundos, etc) usados para representar los objetos.
MAX-ACCESS	Un ACCESS de un OBJECT-TYPE puede ser un MAX-ACCESS en SNMPv2. Las opciones válidas para MAX-ACCESS son read-only, read-write, read-create, not-accessible y accessible-for-notify.
STATUS	Esta cláusula ha sido extendida para permitir las palabras claves current, obsolete y deprecated. current in SNMPv2 es lo mismo que mandatory en una MIB SNMPv1.
AUGMENTS	En algunos casos es útil agregar una columna a una tabla ya existente. La cláusula AUGMENTS permite extender una tabla agregando una o más columnas, representadas por algunos otros objetos. Esta cláusula requiere el nombre de la tabla del objeto que será aumentado.

Tabla de las mejoras de los objetos de la SMIv2

Podemos ejemplificar la definición de un objeto administrado tanto en la SMIv1 así como en la SMIv2, este objeto será "*sysContact*" el cual esta definido dentro de un subárbol denominado system, este objeto guarda información acerca de la persona de contacto, la cual es la encargada de administrar este dispositivo, junto con la información de cómo contactar a esta persona.

Ejemplo de definición de objeto administrado en SMIv1:

sysContact	OBJECT-TYPE --SMI v1
SYNTAX	DisplayString (SIZE (0..255))
ACCESS	read - write
STATUS	mandatory
DESCRIPTION	
"Es la identificación textual de la persona de contacto quien administra este nodo, junto con la información de cómo contactar a esta persona"	
::= { system 4 }	

Ejemplo de definición de objeto administrado en SMIv2:

sysContact	OBJECT-TYPE -- SMI v2
SYNTAX	DisplayString (SIZE (0..255))
MAX-ACCESS	read-write
STATUS	current
DESCRIPTION	
"Es la identificación textual de la persona de contacto quien administra este nodo, junto con la información de cómo contactar a esta persona. Si no existe información de contacto el valor de longitud de cadena es cero"	
::= { system 4 }	

Un agente puede implementar más de una MIB, pero todos los agentes traen consigo una MIB en particular, ésta es denominada MIB II. El principal objetivo de este estándar es proporcionar información en general para propósitos de

administración sobre TCP/IP. Obviamente esto no podría incluir todas las posibles características que un fabricante quisiera administrar en su dispositivo, por ello existe la posibilidad de que un agente pueda implementar varias MIB's.

En los siguientes apartados se describirán los dos principales estándares que hacen referencia a las MIB's, MIB I y MIB II.

2.2.2.3 Concepto de MIB I

La Base de Información de Administración 1 (conocida como MIB I), es un conjunto de variables, llamados objetos, almacenadas y organizadas de manera jerárquica definido en el RFC 1156. Los MIB's son accedidos por protocolos de red, como es el caso de SNMP. La comunicación entre el protocolo y el MIB se da en un esquema de consulta – respuesta.

Es así como a través del MIB se tiene acceso a la información para la administración, contenida en la memoria interna del dispositivo en cuestión. MIB es una base de datos completa y bien definida, con una estructura arborescente, una manera adecuada para manejar diversos grupos de objetos (información sobre variables/valores que se pueden adoptar), con identificadores de objetos únicos.

La MIB I constituye la primera MIB normalizada. Está formada con objetos de los protocolos de TCP/IP. En la siguiente tabla se especifican los grupos que la forman, con el número de objetos que forma cada grupo.

Grupo	Numero
System	3
Interfaces	22
ATT	3
IP	33
ICMP	26
TCP	17
UDP	4
EGP	6

Tabla del Esquema de grupos de la MIB I

El objeto (o variable) administrado, es una característica específica de algún elemento administrado. Existen dos tipos de objetos administrados:

- Escalares: definen una sola instancia del objeto.
- Tabulares: definen múltiples instancias de objetos que tienen una relación entre sí y están agrupados en el MIB.

Con el motivo de identificar a cada uno de los objetos administrados se creó un identificador único del objeto. Este OID se crea siguiendo el árbol jerárquico del MIB, de manera descendente. Se toma el número de cada punto intermedio, hasta llegar al objeto o instancia de objeto que se desea obtener. Este OID puede ser proporcionado en forma numérica o con los nombres de cada objeto (a esta forma se le denomina simbólica), pero es mucho más común con números.

Árbol MIB I

El árbol MIB I (figura 2.19), comienza en un punto indefinido, seguido de tres opciones que son: CCITT, ISO e ISO-CCITT, que representan las principales Organizaciones de Estándares. La mayor parte del "*software*" necesita un punto raíz (.) para localizar el objeto en la MIB, si no se incluye este punto, se asume que la ruta es relativa desde ISO, por ser la opción más usual, la cual tiene otras 4 opciones. DoD se refiere al Departamento de Defensa de Estados Unidos. Así mismo, éste tiene una opción, que es Internet, del cual se desprenden 6 opciones donde se encuentra una, llamada Privado. Dentro de esta opción encontramos una más, Empresa, dentro de la cual están definidos los MIB's de todas las empresas que los han diseñado para sus productos. Dentro de este grupo de empresas se encuentra Cisco, 3com, HP, Microsoft, Sun, etc.

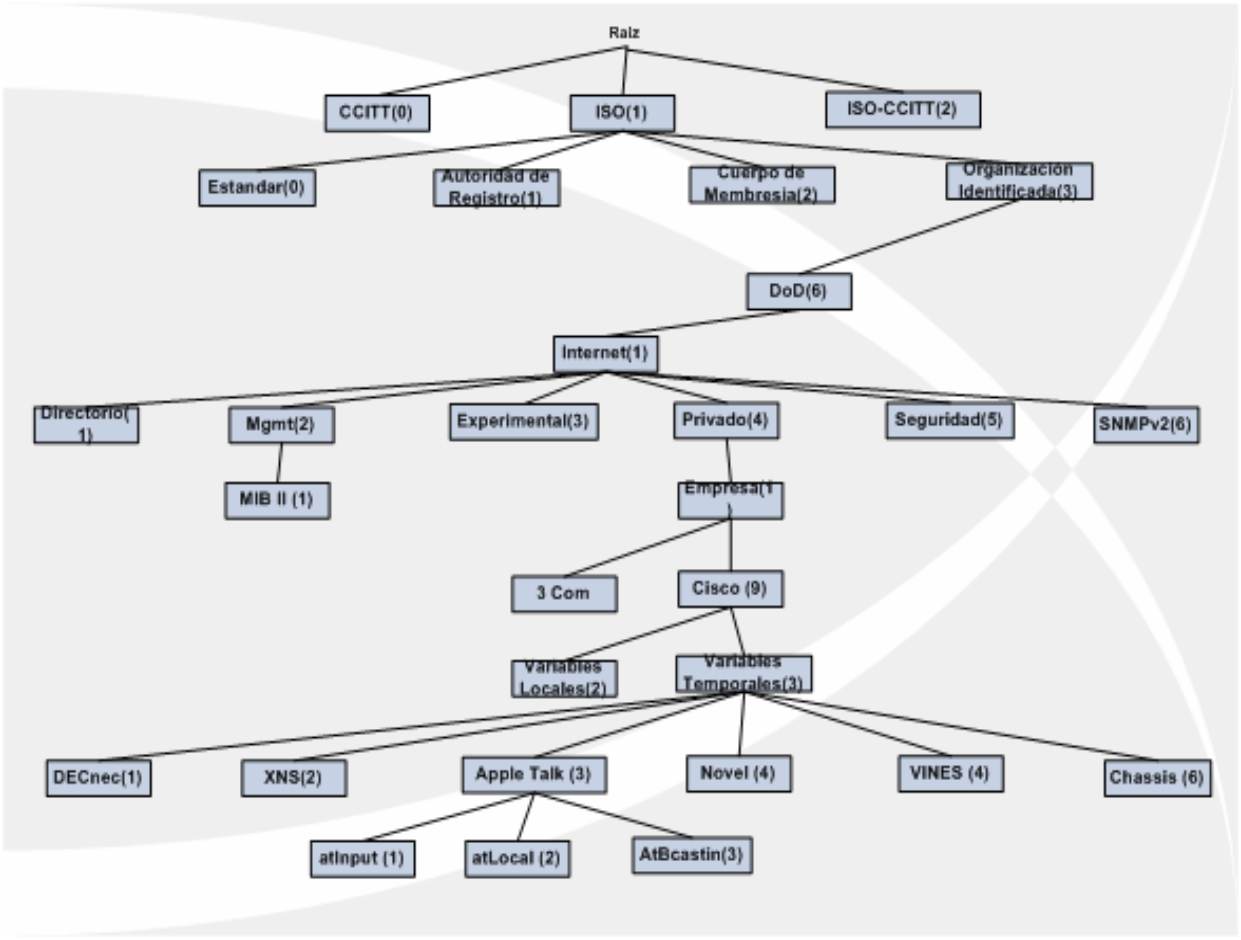


Figura 2.19 Estructura del árbol MIB I

Cabe destacar que una opción bastante importante dentro de la ramificación de la MIB I es la de Internet, puesto que dentro de ésta se encuentra Mgmt ("*Management*" o Administración), dentro de la cual, está definido el MIB II.

2.2.2.4 RFC 1156

El RFC 1156, llamado Base de Información Administrable para Redes en Internet basadas en TCP/IP, es un documento donde se define la primera versión de la Base de Información Administrable MIB I .

Además de este documento, existen otros dos que complementan la definición de una estructura para administrar redes basadas en TCP/IP, que son: el documento que define la Estructura de Información Administrable (SMI) definida en el RFC 1155, y el documento que define al protocolo inicial de administración de redes TCP/IP (SNMP) definido en el RFC 1157.

En esta especificación del MIB se definen las variables necesarias para efectos del monitoreo y control de varios componentes de redes. Cabe mencionar que no todos los grupos de variables definidas son obligatorios para todos los componentes de la red.

No se puede definir la información que formará la MIB de forma arbitraria. Así que, para verificar que la información es útil, se definieron una serie de criterios:

- Un objeto necesita ser esencial para identificar una falla o configuración de administración.
- Solo se permiten objetos de control que no manejen información confidencial. Tal vez con la versión 3 del protocolo esto puede ser posible gracias a sus características de seguridad.
- Debe existir evidencia de que la información fue requerida por su uso actual y su utilidad.
- Se hizo el intento de limitar el número de objetos a 100, para facilitar a los proveedores definir dichos objetos dentro de su “software” y “hardware”.
- Para evitar variables redundantes, no se permitió que objetos que pueden ser derivados de otros objetos fueran incluidos.
- Fue excluida la implementación de objetos específicos.

Los objetos dentro del MIB son definidos usando la Notación Uno de Sintaxis Abstracta (ASN.1). Cada objeto tiene un nombre (“name”), una sintaxis (“syntax”) y un código (“encoding”). El nombre es un identificador del objeto, y es asignado administrativamente, el cual especifica el tipo de objeto. La sintaxis de un tipo de objeto define la estructura de datos abstractos correspondiente al tipo de dato. Por último, el código de un tipo de objeto es simplemente el tipo de objeto representado usando la sintaxis del tipo de objeto.

2.2.2.5 Concepto de MIB II

En marzo de 1991 hubo nuevas revisiones que dieron pie a una nueva especificación de base de datos denominada MIB II. Fue a partir de ese momento que los diversos fabricantes se dedicaron a la obtención de MIB's particulares que permitían la compatibilidad entre plataformas de administración de entornos de red heterogéneos.

Un proveedor también puede definir objetos para introducirlos a la MIB y de esta manera administrar sus propios productos. MIB II define aspectos generales de productos de cualquier empresa, y está definido en el RFC 1213.

Árbol de MIB II

La colección de objetos administrables por SNMP está definida en el MIB. Estos objetos están agrupados en 10 categorías, las cuales corresponden a los 10 nodos definidos bajo el lenguaje ANS.1 en el árbol del MIB II (figura 2.20).

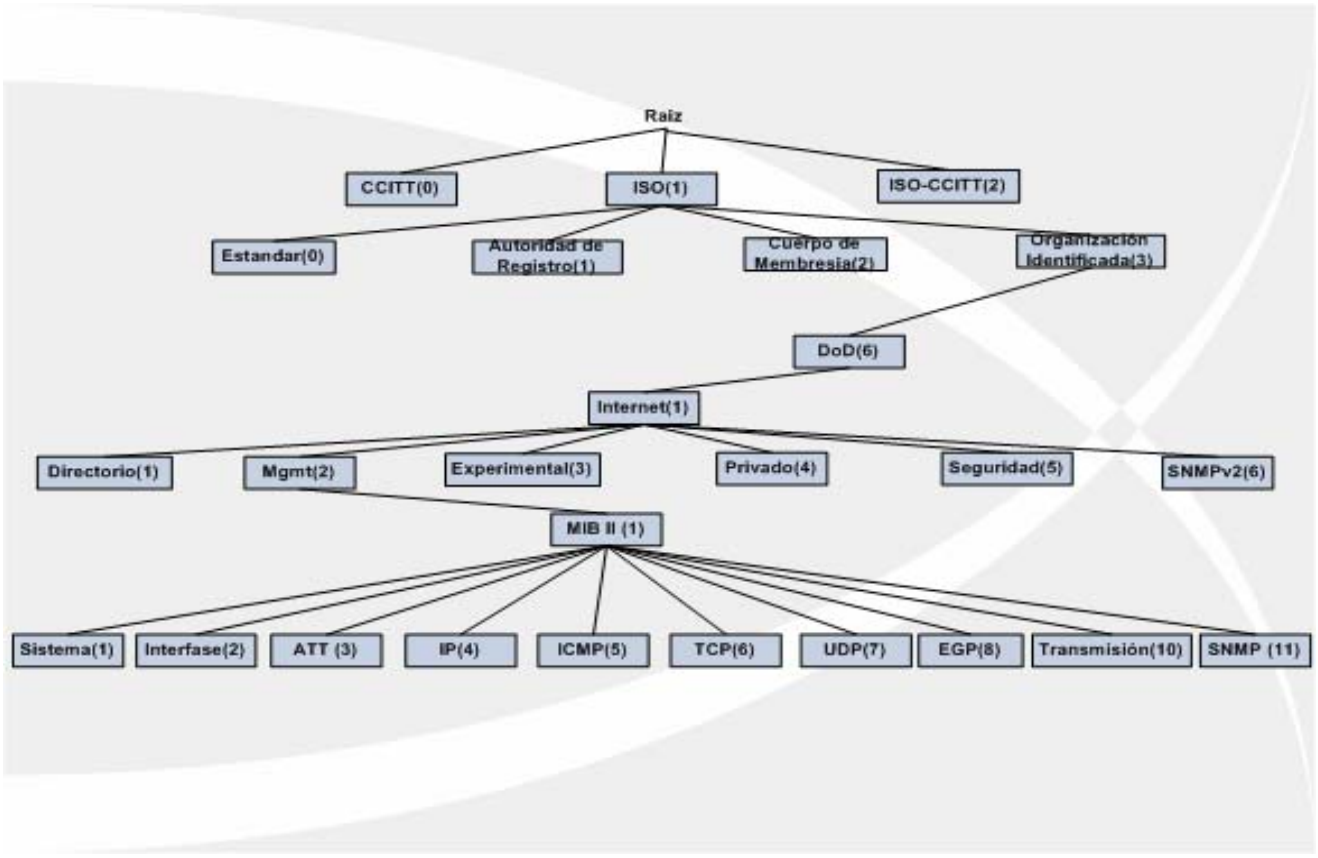


Figura 2.20 Estructura del árbol MIB II

En la MIB II se realizaron una serie de modificaciones sobre la primera versión.

También se define un nuevo grupo para cada tipo específico de interfase, así como un nuevo grupo con objetos de SNMP.

Grupo	Numero
System	7
Interfaces	23
ATT	3
IP	38
ICMP	26
TCP	19
UDP	7
EGP	18
Transmission	0
SNMP	30

Tabla de los Grupos de la MIB II

A continuación se presenta una breve descripción de los objetos definidos en el extremo inferior del árbol, estos objetos son lo más utilizados en esta nueva definición de MIB:

- **Sistemas ("Systems"):** Incluye nombre, localización y descripción del equipo, la identidad del vendedor y el tiempo desde la última inicialización del sistema.
- **Interfases ("Interfaces):** Una única o múltiples interfases de red y sus medidas de tráfico local o remota.
- **ATT ("Address Translation Table"):** Contiene la dirección de la red y las equivalencias con las direcciones físicas.
- **IP ("Internet Protocol"):** Proporciona las tablas de rutas, y mantiene estadísticas sobre los datagramas IP recibidos.
- **ICMP ("Internet Communication Management Protocol"):** Cuenta el número de mensajes ICMP recibidos y los errores.

- **TCP ("Transmission Control Protocol"):** Facilita información acerca de las conexiones TCP, retransmisiones, etc.
- **UDP ("User Datagram Protocol"):** Cuenta el número de datagramas UDP, enviados, recibidos y entregados.
- **EGP ("Exterior Gateway Protocol"):** Recolecta información sobre el número de mensajes EGP recibidos, generados, etc.
- **Transmisión ("Transmission"):** Reservado para MIB's específicos.
- **SNMP:** Estadísticas de tráfico SNMP.

2.2.2.6 RFC 1213

Este RFC fue titulado como Base de Información Administrable para Redes basadas en TCP/IP, pero es comúnmente conocido como MIB II. En este RFC se describen los cambios que han surgido para cada objeto definido en el RFC 1155, así también se describen los nuevos objetos que han sido definidos.

La definición de la estructura de la MIB II esta basada fuertemente en los criterios considerados para su antecesora MIB I, mientras que para la MIB I el limite de objetos definido fue de 100, para facilitar a los proveedores definir dichos objetos dentro de su "*software*" y "*hardware*", en la MIB II este limite fue superado para proporcionar una base tecnológica más amplia, incluyendo lo considerado en la MIB I. También fue acordado evitar la instrumentación de seccione de código critico.

Algunos de los cambios realizados en el MIB II son los siguientes:

- Para prevenir futuros cambios en el MIB, se ha implantado un nuevo término que puede ser usado cuando se describe un objeto: "*deprecated*", que no significa cosa que no aprobado. Esto quiere decir que el objeto no aprobado puede mantenerse en la MIB, pero es uno de los que probablemente será removido en la siguiente versión del MIB.

- En el MIB I se mal interpretó la manera en que los datos definidos como cadenas serían desplegados al usuario. Se tenía la idea de desplegar la información como un texto convencional en el MIB. Pero como era introducido como tipo de dato "*datatype*", no funcionó como se esperaba.
- En cuanto a la dirección física de los equipos, permite regresar las direcciones físicas tal cual son.
- Para cada objeto del MIB I se han definido nuevos objetos (o instancias) y se han corregido errores detectados.
- Se agregaron nuevos grupos de objetos como por ejemplo: el grupo SNMP, donde se definen los objetos que describen el funcionamiento del protocolo SNMP en cada equipo.

2.2.2.7 Categorías de MIB

MIB describe los objetos que deben ser incluidos en las bases de datos del agente SNMP para efectos de la administración, por esta razón, los agentes SNMP son referidos algunas veces como MIB's. Los objetos en un MIB deben estar definidos de la manera en que los desarrolladores del "*software*" de la estación conocen a disposición (nombres y valores de los objetos).

En la actualidad existen 3 categorías de MIB:

1. **Estándar:** Ésta especificación incluye un conjunto común de objetos aceptados y ratificados por el grupo de estándares de Internet. El primer estándar MIB I que se dio a conocer constaba de 114 objetos, este fue mejorado posteriormente y presentado como MIB II conteniendo 172 objetos. La información que proveen estas especificaciones MIB esta dirigida a ruteadores de manejo IP.

2. **Experimental:** Ésta categoría incluye información específica relacionada con otros aspectos de la red y de los dispositivos de manejo, estos datos son considerados como muy importantes y no existe en otros estándares

MIB. Una vez que este tipo de especificación experimental ha sido reconsiderada y evaluada pasando por una serie de niveles de pruebas de competencia y eficiencia, será reclasificada como estándar.

3. **Privado (o de empresa):** Ésta se ha diseñado para uso individual de compañías que requieren colectar datos particulares de sus propios dispositivos de red. Permite que se definan objetos propios, que pueden ser específicos y no estar definidos en la categoría estándar.

2.2.3 Operaciones SNMP

El protocolo PDU es el formato del mensaje utilizado tanto por los administradores y agentes para el envío y recepción de la información.

En un apartado anterior solo se hizo mención de los tipos de comandos básicos aplicables en SNMP, de éstos comandos se desprenden los tipos de operaciones que pueden ser enviados a través del protocolo.

Hay un estándar de formato PDU para cada uno de los siguientes mensajes en SNMP.

2.2.3.1 Get Request (realizar petición)

Es la solicitud enviada desde un administrador hacia el agente SNMP para que éste envíe los valores contenidos en el MIB de una o más variables. El agente recibe esta petición y tratará de procesarla de la mejor manera que su estado lo permita (figura 2.21). Algunos dispositivos que se encuentran bajo una gran carga de trabajo, como pueden ser los routers, podrían no contestar la petición del administrador, pero si ésta es atendida con éxito, el agente responderá mediante una petición de respuesta (*get – response*).

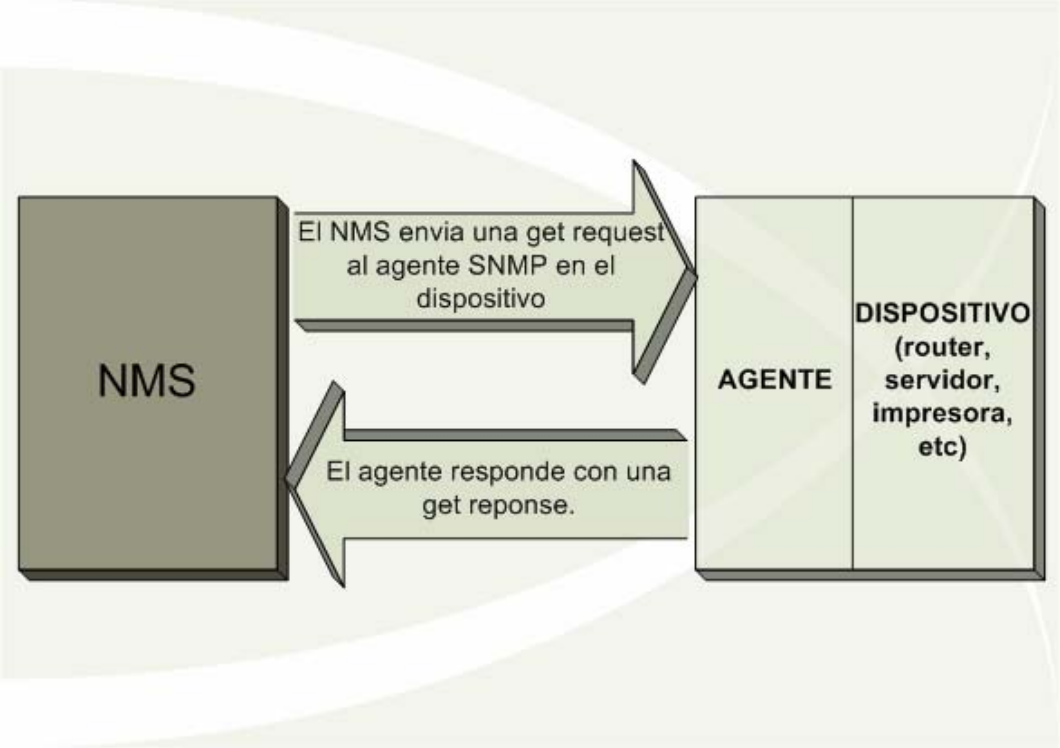


Figura 2.21 Secuencia get request

Ahora la pregunta es, como el agente conoce la información que le esta solicitando el NMS?. Una de las partes que conforman el formato de una *get request* es una variable ligada ("*variable binding*"), denominada "*varbind*", es una lista que hace referencia a los objetos de la MIB que permite a un recipiente de peticiones observar lo que el emisor desea dar a conocer.

Las variables ligadas pueden considerarse como una pareja de OID y valor que permite fácilmente al emisor (en este caso el NMS) recolectar la información que requiere cuando el receptor (agente) completa la petición y envía una respuesta de regreso.

El comando utilizado para recuperar la información en un agente es "*snmpget*" ,y la forma de utilizarlo es la siguiente:

```
$snmpget –v 2c –c public  localhost  system.sysLocation.0

SNMPv2-MIB::sysLocation.0 = STRING: UNAM, Dep. de Servidores
```

El comando se esta ejecutando sobre un equipo con sistema operativo Unix, el cual deberá implementar un agente SNMP para poder responder satisfactoriamente la petición, su principal objetivo es facilitar la recopilación de la información de administración del dispositivo utilizando una “get request”.

Se puede observar que se han proporcionado 4 parámetros, los cuales son: la versión SNMP utilizada, la comunidad, nombre del host que se desea cuestionar y el objeto que se desea recuperar (estas opciones son iguales para la versión 1 de SNMP pero difieren para la versión 3). En este caso se esta recopilando el objeto “*sysLocation*” que es una variable que hace referencia la ubicación física del dispositivo, es una “*varbind*” utilizada con fines de administración que en este caso sirve para ubicar físicamente al dispositivo. El valor recuperado es la cadena “UNAM, Dep. Servidores”, lo cual indica que el administrador puso este valor para describir que el dispositivo pertenece a la UNAM y esta siendo administrado por gente de esta institución.

El objeto “*sysLocation*” hace referencia a un objeto que ha sido definido en la MIB, esta asociado a un nodo único en la base de datos de administración, que como ya se ha mencionado, es denominado OID.

Esta operación es útil para recuperar un solo objeto de la MIB, sin embargo puede ser un problema si se pretende recuperar una gran cantidad de objetos de un dispositivo, ya que por su forma de operación resultaría en un consumo considerable de tiempo. Para ello se ejemplifica la operación “*get next request*”, la cual permite recuperar más de un objeto de una MIB.

2.2.3.2 Get Next Request (realizar la siguiente petición)

Es una petición del administrador al agente SNMP para que envíe los valores del MIB que se refieren al siguiente objeto del especificado anteriormente. Este tipo de operación permite emitir una secuencia de comandos desde la estación administradora para recuperar un grupo de valores de la MIB del dispositivo

administrado. En otras palabras, por cada objeto de la MIB que se desee recuperar una petición "*get next*" y una petición "*get response*" son generadas. Es así como la operación "*get next*" tiene la capacidad de recorrer toda una rama de la estructura jerárquica de la MIB, debido a que un OID es una secuencia de enteros que representa un nodo único en la MIB, es relativamente sencillo para un agente comenzar desde el nodo raíz de la MIB hasta ubicar en la SMI de la MIB el objeto especificado por el administrador. Cuando el NMS recibe una respuesta del agente a una petición "*get next*" , esto desencadena la emisión de otra operación "*get next*". Esta situación se seguirá repitiendo hasta que el agente retorne un error, situación que significará que el fin de la MIB ha sido alcanzado y no hay más objetos que recuperar.

El comando utilizado para realizar una petición "*get next*" es *snmpwalk,* veamos un ejemplo de su sintaxis:

```
$ snmpwalk –v 2c –c public  localhost  system

SNMPv2-MIB::sysDescr.0 = STRING: Linux gypsy 2.6.9-5.EL #1 Wed Jan 5 19:22:18 EST 2005 i686
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (43894206) 5 days, 1:55:42.06
SNMPv2-MIB::sysContact.0 = STRING: aaguilera@servidores.unam.mx
SNMPv2-MIB::sysName.0 = STRING: gypsy
SNMPv2-MIB::sysLocation.0 = STRING: UNAM, Dep. de Servidores
SNMPv2-MIB::sysORLastChange.0 = Timeticks: (0) 0:00:00.00
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.
```

Los parámetros del comando no difieren a los utilizados en *snmpget*, se vuelve especificar la versión, la comunidad, el nombre del dispositivo a cuestionar y el objeto que se desea recuperar. Pero en esta ocasión se hace referencia al objeto sistema ("*system*"), el cual es toda una rama del árbol de la MIB, rama que define una lista de objetos que pertenecen a la operación del sistema. Debido a esto el comando *snmpwalk* recuperó todos los objetos que se encuentran debajo de la rama de sistema como se puede ver en el resultado de la petición "*get next*".

2.2.3.3 Get BulK (recuperación de una gran cantidad de información)

Actualmente existen 3 versiones del protocolo SNMP, pero fue desde la snmpv2 que se definió la operación "*get bulk*", que significa obtener un volumen de datos o la obtención de información a granel (figura 2.22). Esta operación permite a las aplicaciones de administración recuperar una larga sección de una tabla de datos como si fuera una. El funcionamiento de la operación "get" puede intentar recuperar más de un objeto de la MIB a la vez, pero el tamaño del mensaje esta limitado por las capacidades del agente. Si el agente no puede retornar todas las peticiones de respuesta ("*get request*"), éste mandará un mensaje de error con ningún dato.

Por otro lado, la operación "*get bulk*" le dice al agente que envíe tantas respuestas de regreso como se lo permita sus propias capacidades. Esto significa que las respuestas incompletas son posibles. Dos campos deben ser colocados cuando se esta emitiendo una operación "*get bulk*" no repeticiones ("*nonrepeaters*") y máximo de repeticiones ("*max-repetitions*"). El campo "*nonrepeaters*" dice al comando "*get bulk*" que los primeros N objetos puede ser recuperados con una simple operación "*get next*". En cambio el campo "*max- repetitions*" le dice al

comando "*get bulk*" intentar por medio de M numero de operaciones "*get next*" recuperar el restante numero objetos.

El número total de variables que se pueden obtener esta dado por la siguiente formula N + (M * R) donde N es el número de "*nonrepeaters*", M es el número de "*max-repetitions*" y R es el número de objetos no escalares.

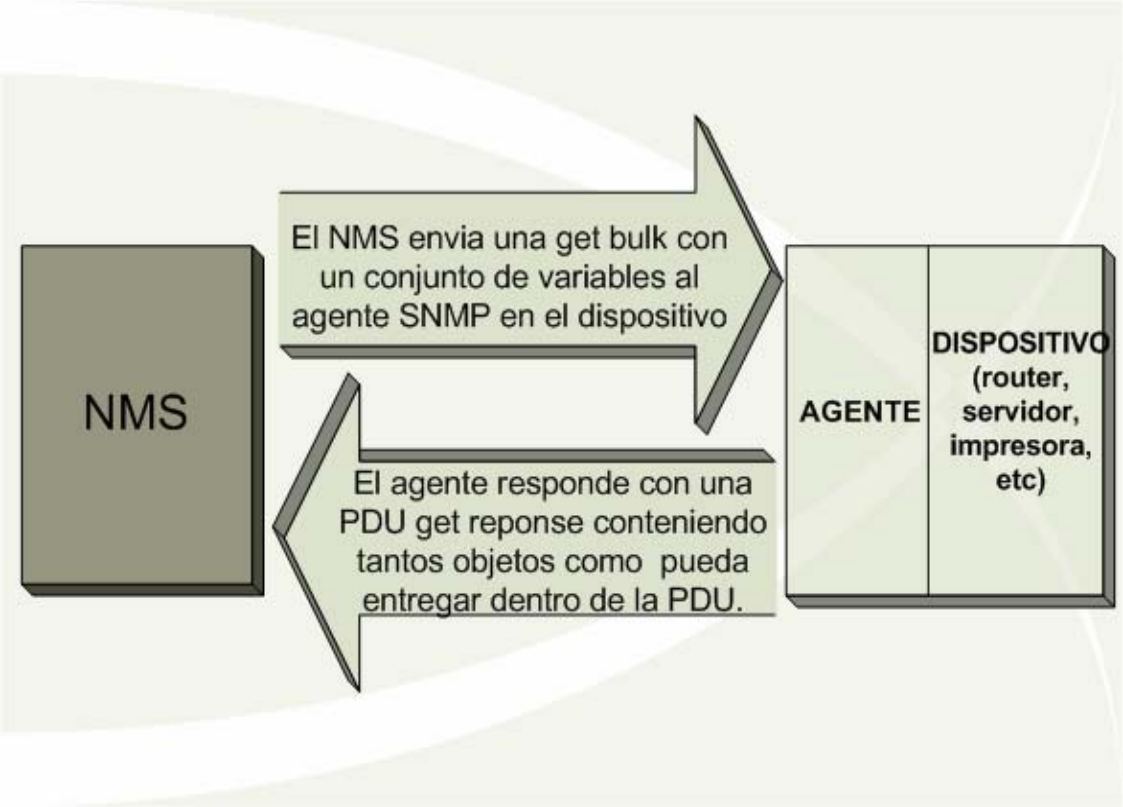


Figura 2.22 Secuencia get bulk

2.2.3.4 Get Response (realizar una respuesta)

Es la respuesta del agente SNMP a la petición del administrador. Como ya se ha visto en las anteriores operaciones, una "*get response*" es la respuesta del agente a los comandos *snmpget, snmpwalk y snmpbulk.*

2.2.3.5 Set Request (realizar una modificación)

Es una petición del administrador al agente SNMP para que éste realice la modificación de algún valor contenido en el MIB que se refiera a un objeto determinado (figura 2.23). Esta operación se podrá llevar a cabo siempre y cuando los objetos en la MIB estén definidos para escritura. Es posible para un NMS cambiar el valor de más de un objeto en un instante.

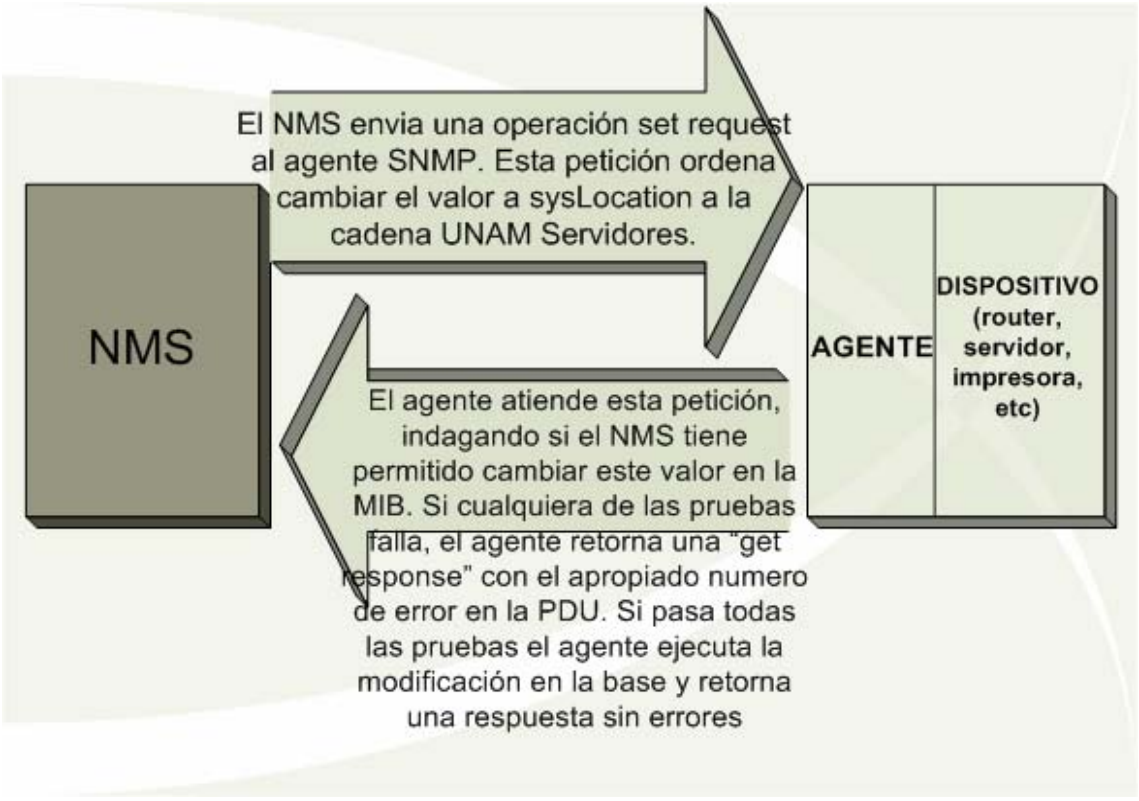


Figura 2.23 Secuencia set request

El comando utilizado para realizar una petición "*set request*", es *snmpset,* aquí se tiene un ejemplo de su sintaxis:

Primero obtenemos el valor actual del objeto:

```
$ snmpget –v 2c –c public  localhost  system.sysLocation.0
SNMPv2-MIB::sysLocation.0 = STRING: UNAM, Dep. de Servidores
```

Asignamos el nuevo valor del objeto:

```
$ snmpset –v 2c –c public  localhost  system.sysLocation.0 s “UNAM Servidores”
SNMPv2-MIB::sysLocation.0 = STRING: UNAM Servidores
```

Observamos como el valor del objeto fue modificado:

```
$ snmpget –v 2c –c public  localhost  system.sysLocation.0
SNMPv2-MIB::sysLocation.0 = STRING: UNAM Servidores
```

Se puede observar que el comando no varía mucho en su sintaxis con respecto a los ya vistos, sin embargo en esta ocasión se especifica el objeto que se desea modificar (“*sysLocation*”), listando el tipo de dato en el que se guarda este objeto, que es de tipo cadena (“*string*”).

2.2.3.6 Trap (alerta)

Es un mensaje espontáneo del agente SNMP al administrador. Esta situación se presenta cuando se detecta una condición anómala en el dispositivo administrado y el NMS es notificado por medio de una alerta (“*trap*”). La figura 2.24 muestra la secuencia de la generación de una “*trap*”.

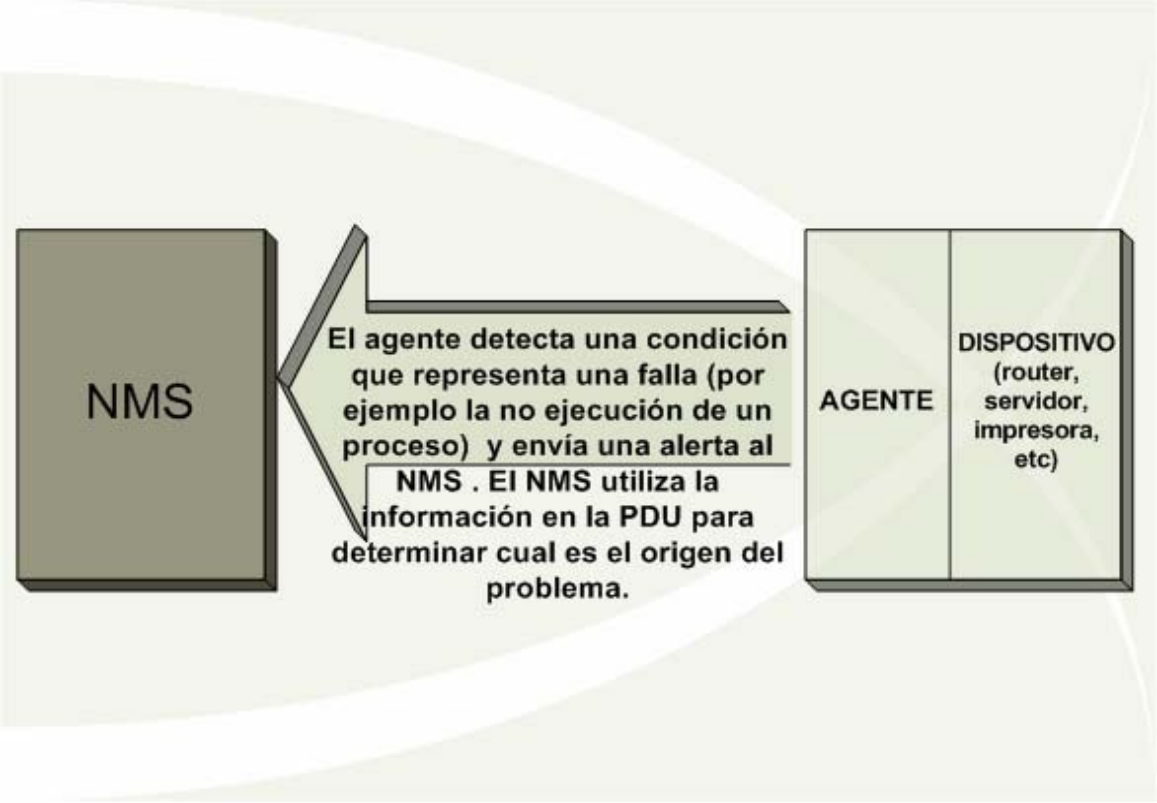


Figura 2.24 Secuencia de la generación de una Trap

La alerta ("*trap*") se origina del agente hacia el NMS, esta configuración se encuentra dentro del mismo agente y no en el NMS. El destino de la alerta es usualmente una dirección ip que suele ser la del NMS.

Un mensaje de alerta puede ser explícitamente enviado por medio del comando llamado "*snmptrap*", observar el siguiente ejemplo:

```
snmptrap -v 2c -c comuTRP 192.168.10.9 5 prErrorFlag prErrMessage s "Error en  monitoreo de procesos"
```

Al igual que los anteriores comandos se especifica la versión, la comunidad y la dirección ip del servidor que será capaz de recibir y procesar el mensaje de la alerta. Pero también se requiere indicar que tipo de alerta se esta enviando así como un mensaje descriptivo. En este caso se esta enviando una alerta de tipo *prErrorFlag*, que es un objeto definido en la MIB para indicar si existe algún

problema en el monitoreo de procesos específicos del servidor, también un mensaje descriptivo a través de la variable *prErrMessage* que esta definido en la MIB para éste propósito.

Sin embargo no es muy útil estar enviando alertas de forma explícita con el anterior comando, puesto que estas tienen que ser disparadas de forma automática por algún tipo de auto - monitoreo en el servidor. En el apartado siguiente se tratará más a fondo el tema de las alertas de SNMP.

2.2.3.7 Inform Request

Mensaje de administrador a administrador donde se describe el MIB local. La versión 2 de SNMP proporciona este mecanismo de información, el cual permite la comunicación entre administradores. Este tipo de operación puede ser útil cuando se requiere tener más de una estación NMS en la red. Cuando un informe es enviado de un administrador a otro, el receptor del mensaje envía una respuesta en cuanto obtiene la información, de esta manera el emisor tiene conocimiento de que el otro NMS ha recibido el informe. Es importante mencionar que un informe SNMP puede ser utilizado para enviar alertas ("*traps*") hacia el NMS. Si se utilizan los informes para este propósito, se resolverá el problema de las alertas en relación a que el agente que origina la alerta desconoce si el mensaje fue recibido por el NMS, los informes permitirán que el agente sea notificado cuando el NMS reciba la información de la alerta.

Es así como SNMP está orientado a monitorear y configurar el equipamiento físico de una red (puentes, ruteadores, concentradores, estaciones de trabajo, sevidores, etc). Cabe mencionar que el SNMP no sólo sirve para monitorear y/o configurar variables de los equipamientos de la red física sino también para mostrar el tráfico o cualquier otro dato en forma gráfica (con ayuda de otras plataformas de administración).

SNMP fue diseñado específicamente para ser un protocolo de transporte independiente. Los que significa que las solicitudes de SNMP sobre los agentes

pueden hacerse utilizando cualquier protocolo de transporte como TCP/IP o cualquier otro.

SNMP al pertenecer a una arquitectura tipo TCP/IP no es abierto y por lo tanto es mucho menos vulnerable.

2.2.4 Traps de SNMP

En el apartado anterior, referente a las operaciones del protocolo SNMP, se describieron brevemente el uso de las *traps*, ahora se entrará más a detalle en el uso de esta operación.

Las *traps* son una vía por medio de la cual, los agentes SNMP pueden enviar a una estación de monitoreo, de manera asíncrona, información acerca de una condición que la estación de monitoreo debiera conocer. Las *traps* que un agente puede generar son definidas por las MIB's, el número de *traps* puede ser del rango de cero a cientos. Para saber que tipo de *traps* están definidas en algún archivo MIB, se debe buscar el término "TRAP-TYPE" (para la SMIv1) o "NOTIFICATION-TYPE" (para la SMIv2) en el archivo MIB. Este tipo de búsqueda brindará rápidamente una lista de las posibles traps soportadas.

Sin embargo la simple generación de *traps* hacia un NMS podría ser poco interesante así como poco útil. Por ello es posible configurar la respuesta del NMS para diferentes tipos de *traps* que sean generadas, la respuesta podría ser desde no tomar ninguna acción así como ejecutar algún programa diseñado para una eventualidad en particular.

2.2.4.1 Concepto de trap

Una *trap*, es básicamente una notificación asíncrona enviada de un agente SNMP hacia una estación de administración. Como todo en SNMP, las *traps* son enviadas usando UDP a través del puerto 162, razón por la cual el proceso se

vuelve inestable. Esto significa que el emisor no puede asumir que la *trap* realmente llega a su destino, ni tampoco el destino puede asumir que esta logrando captar todas las *traps* que le son enviadas ya que no tiene forma de saber si algún agente le ha enviado alguna notificación. Seguramente en una red ideal, capaz de soportar un alto tráfico de datos, la gran mayoría de las *traps* generadas no deberán tener ningún problema en alcanzar su destino, sin embargo si todas las redes tuvieran siempre un comportamiento tan eficiente y sin ningún problema de tráfico seguramente no se necesitarían de herramientas como SNMP.

Entrando más a detalle, una *trap* es como un conjunto de datos que están definidos por una MIB. Las *traps* caen en dos categorías, genéricas (*generic*) y especificas de empresa (*enterprise - specific*). Existen siete números (0-6) que definen a las *traps* genéricas, las cuales son descritas en la siguiente tabla:

Trap Genérica, nombre y numero	**Definición**
coldStart (0)	Indica que el agente ha sido reiniciado. Esto causará que todas las variables de administración sean reiniciadas. Una cosa útil acerca de esta la trap coldStart es que esta nos puede ayudar a determinar cuando un hadware nuevo ha sido agregado a la red, o cuando un dispositivo acaba de ser prendido ya que este enviará este tipo de trap cuando esta situación se presenta.
warmStart (1)	Indica que el agente se ha reiniciado a si mismo. Ninguna de las variables de administración será reiniciada.
linkDown (2)	Es enviada cuando una interfaz de red de algún dispositivo se ha caído. La primera variable enviada identifica cual de las interfaces esta abajo.
linkUp (3)	Es enviada cuando una interfaz de red de algún dispositivo ha vuelto a funcionar. La primera variable enviada identifica cual de las interfaces regreso a funcionar.
authenthicationFailure (4)	Indica que alguien han intentado preguntar a un agente con un nombre de comunidad incorrecta, esta trap es útil para identificar si alguien esta intentando obtener acceso a alguno de los dispositivos gestionados.
egpNeighborLoss (5)	Indica que un protocolo de pasarela exterior (EGP) se ha caído.
enterpriseSpecific (6)	Indica que la trap es específica de empresa. Los vendedores que implementan SNMP en sus equipos así como los usuarios son libres de definir sus propias traps bajo la rama denominada privada para empresas del árbol de los objetos SMI. Para procesar esta trap de forma apropiada, el NMS tiene que descifrar el número de la trap específica que es parte del mensaje SNMP.

Tabla de los tipos de traps genéricas

Sin embargo hay que mencionar que lo que realmente hace a este mecanismo de *traps* poderoso e interesante, es la definición de *traps* específicas de empresa (enterprise - specific). Cualquiera que cuente con un número de empresa puede definir una *trap* específica para cualquier condición que se considere importante para el funcionamiento del equipo. Una *trap* especifica de empresa es identificada por 2 piezas de información; el ID de empresa de la organización que definió la *trap* y un número específico de *trap* asignado por esa organización. La noción de una *trap* específica de empresa es extremadamente flexible, las organizaciones tienen permitido subdividir sus empresas de la manera en que ellas deseen. Por ejemplo si el numero de una empresa es 2789, el ID de empresa sería .1.3.6.1.4.1.2789. Pero esto se puede subdividir aun más, definiendo *traps* con ID's de empresas como .1.3.6.1.4.1.2789.6000, .1.3.6.1.4.1.2789.6001, etc.

El hecho de poder recibir una *trap* y por consiguiente conseguir información como el número de *trap* genérica, el ID de empresa o el número de *trap* específica de empresa, es a menudo toda la información necesaria que se requiere para diagnosticar un problema en el dispositivo administrado. Pero las *traps* también traen consigo información adicional. En el caso de las *traps* genéricas (0 – 5) la información específica es predefinida y almacenada dentro del NMS. Cuando se recibe una *trap* genérica, el NMS sabe como interpretar la información que esta contiene y será capaz de desplegar así como manejar los datos de forma apropiada. En contraste la información acarreada por una *trap* específica de empresa es completamente del dominio de la persona quien definió la *trap*. Una *trap* específica de empresa puede contener cualquier número de variables ligadas o pares de objetos – valor de MIB. Cuando alguien define sus propias *traps* puede decidir que información es la apropiada para acarrear a su destino. Los objetos contenidos en una *trap* pueden ser objetos estandarizados de una MIB, objetos específicos de fabricante, u objetos de propia invención.

2.2.4.2 Variantes de las traps

En el apartado referente a las diferentes versiones del protocolo SNMP, ya se habían mencionado el formato de las *traps* haciendo hincapié en que la versión

SNMPv2 del protocolo define las *traps* de una manera ligeramente diferente a la primera versión, pero que en esencia tienen el mismo objetivo. En la MIB versión 1 las *traps* fueron definidas mediante la palabra reservada “TRAP-TYPE”, mientras que en la versión 2 las *traps* son definidas como “NOTIFICATION – TYPE”. SNMPv2 anula la noción de las *traps* genéricas, también define muchas *traps* específicas (llamadas propiamente, notificaciones) en MIB’s públicas. Las *traps* SNMPv3 son simplemente *traps* SNMPv2 pero con el agregado de capacidades de autenticación y privacidad.

2.2.4.3 Recepción de traps

Ahora se discutirá la forma en que se tratan a las *traps* entrantes. Ya se había mencionado que el encargado de manejar a las *traps* que están siendo generadas por los agentes SNMP, es responsabilidad del NMS. Algunos de los NMS solo hacen un pequeño proceso para desplegar la información de las *traps* entrantes a la salida estándar (llamada comúnmente “output”). Sin embargo, un servidor NMS típicamente tiene la habilidad para reaccionar a las *traps* SNMP que esta recibiendo. Por ejemplo cuando un NMS recibe una *trap* del tipo “*linkDown*” proveniente de un ruteador, este podría responder al evento informando por alguna vía a alguna persona de contacto, mostrando un mensaje de alerta en una consola de administración o reenviando el evento a otro servidor NMS.

El procedimiento de configuración y forma en que el NMS puede entender e interactuar con las *traps* provenientes de los agentes SNMP, son propias del software que se este implementando para este propósito, existe gran variedad de software propietario para lograr que el NMS pueda lograr este objetivo, sin embargo existen también opciones de software de libre distribución para alcanzar esta tarea.

2.2.4.3.1 Servidor de traps

En el mercado existen diversas herramientas que nos permiten manipular e interpretar las *traps* entrantes a un NMS, de hecho estas forman parte de un sistema completo de administración de red con diversas e interesantes características, sin embargo, como se expuso en el capitulo 1, el presente trabajo tiene como finalidad alcanzar sus objetivos mediante herramientas de libre distribución debido a que el departamento no cuenta con recursos suficientes para adquirir un software propietario. Por ello la solución utilizada para llevar a cabo el manejo de las *traps* entrantes fue el uso del servidor de *traps* ("trap receiver") que es parte de la distribución Net - SNMP utilizada en el desarrollo e implementación del presente trabajo.

Básicamente existen 2 herramientas que tienen que ver con el manejo de las traps, por un lado se tiene al programa "*snmptrap"* que permite enviar mensajes SNMP de tipo trap a las bitácoras de sistema o a la salida estándar, mismo que fue expuesto en el apartado referente a las operaciones SNMP. La segunda herramienta es la concerniente a la recepción de las *traps*, es un demonio de *traps*, el cual es iniciado mediante un programa llamado "*snmptrapd"*, este permite mediante una configuración adecuada, recibir las *traps* entrantes y poderlas diferenciar dependiendo del tipo de información que contenga. Así las *traps* son almacenadas en una bitácora, con lo cual resulta sencillo implementar una serie de programas que puedan basarse en dichos logs, con la finalidad de vigilar eventos importantes y reaccionar de manera adecuada ante ellos. Es así como fácilmente se puede implementar un sistema de monitoreo flexible y poderoso sin necesidad de hacer una gran inversión en otro software propietario con características similares (figura 2.25).

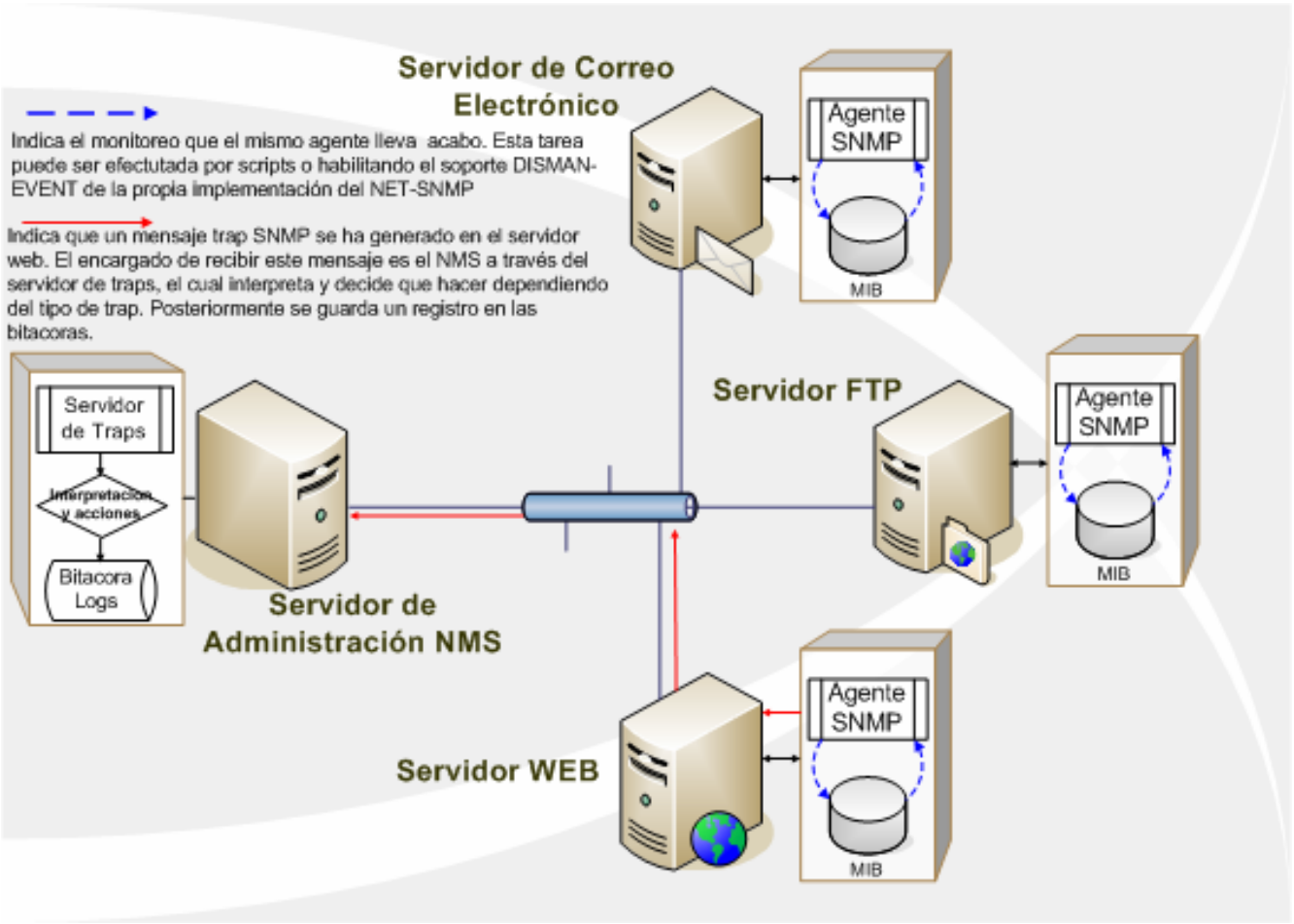


FIGURA 2.25 Esquema de un sistema de Traps

2.2.4.4 Envío de las traps

El envío de *traps* es el mecanismo contrario a la recepción. Existen diferentes herramientas que nos permiten el envío de *traps*, así como el desarrollo de propias *traps* que permitan ajustarse a las necesidades de un ambiente en particular. Casi todas estas utilerías de generación de *traps* están basadas en la línea de comandos. Esto permitirá que el comando requerido sea ejecutado desde cualquier programa, así se cuenta con una forma muy flexible de generar *traps* si tomamos en cuenta que se pueden desarrollar programas propios para monitorear aspectos que se crean importantes para un esquema de administración.

A pesar de que existen muchos tipos de programas para generar y enviar las *traps*, fundamentalmente todas son similares. En particular la sintaxis utilizada en la línea de comandos puede variar, pero en un aspecto global se requieren los mismos argumentos:

- **PUERTO**

Es el puerto UDP por el cual se envía la *trap*. El puerto predeterminado es el 162

- **Versión SNMP**

Es la versión SNMP apropiada por medio de la cual se desea enviar la *trap*.

- **Nombre de Host o dirección IP del NMS**

El nombre del Host o la dirección IP del NMS hace referencia al equipo destino a donde deberán llegar las *traps*.

- **Nombre de comunidad**

El nombre de la comunidad debe enviarse junto con la *trap*. La mayoría de las estaciones de administración pueden ser configuradas para ignorar las *traps* que no tienen una apropiada comunidad.

- **OID de empresa**

Es la ruta completa del identificador de objeto para la *trap* que se desee enviar, es todo el OID de la *trap* desde el nodo .1 hasta el número de empresa incluyendo cualquier subárbol dentro de la empresa. Por ejemplo, si un número de empresa es el 2789, el cual se ha extendido y subdivido para dar lugar a grupos de *traps* numeradas hasta el 5000, y se desea enviar la *trap* específica 1240, entonces el OID de empresa que se deberá enviar es .1.3.6.1.4.1.2789.5000.

- **Nombre de Host o dirección IP del emisor**

Es la dirección IP del agente que esta enviando la *trap*. A pesar de que

esto pueda parecer innecesario, esto puede ser importante si existe un servidor Proxy entre el agente y el NMS. Este parámetro permite grabar un histórico de las direcciones IP de los agentes que generan las alertas. Si no se especifica este parámetro, siempre se enviará predeterminadamente la dirección de la máquina que esta enviando la trap.

- **Numero de trap genérica**

Es un número en el rango de 0 a 6. Las *traps* genéricas tienen un número válido del 0 al 5, si se esta enviando una *trap* específica de empresa, el número de deberá ser enviado es el 6.

- **Numero de trap específica**

Es un número indicando la *trap* específica que se desee enviar. Si se esta enviando una *trap* genérica, éste parámetro es ignorado. Por ejemplo si esta enviando una *trap* con en OID .1.3.6.1.4.1.2700.3004.0, 3004 es el número de la *trap* específica.

- **Estampa de tiempo**

Es el tiempo transcurrido entra la última inicialización de la entidad de red y la generación de la *trap*.

- **OID_1 TIPO_1, VALOR_1**

Son los datos que se ligan para ser enviados por medio de la *trap*. Cada variable ligada en la *trap* consiste en un OID junto con un tipo de dato, seguido por el valor que se pretenda enviar. La mayoría de los programas dejan incluir cualquier número variables con datos ligados en la *trap*.

En el apartado referente a las operaciones SNMP, se describió el comando *snmptrap*, herramienta que forma parte del paquete “NET – SNMP” y la utilizada para el envío de las *traps*. Este comando utiliza los argumentos descritos anteriormente (remítase al ejemplo del apartado 2.2.3.6 *Trap*).

2.2.4.5 Administrador de Eventos DISMAN-EVENT

Se ha mencionado a través de este apartado, que existen varias y diferentes herramientas que permiten generar *traps* en el dispositivo administrado, algunas con muchas e interesantes características, casi todas ellas pertenecientes al software propietario. Otra alternativa es usar el "*snmptrap"* del propio paquete NET – SNMP, herramienta que permite enviar mensajes SNMP de tipo trap. Combinando esta herramienta junto con otras como "*snmpget***"** y **"***snmpwalk"*, comandos que permiten recuperar valores de la MIB del agente, resultaría fácil, por ejemplo en un entorno UNIX, realizar un script que tenga la tarea de cuestionar al propio agente, recuperando información de administración que se considere importante, y lanzar una *trap* al NMS en caso de que exista un valor que refleje alguna anomalía en el dispositivo gestionado.

Sin embargo el propio paquete NET – SNMP cuenta con soporte para monitoreo de eventos, lo que significa que si se instala para habilitar este comportamiento, se tendrá un agente SNMP capaz de monitorearse a si mismo, bajo ciertas reglas y parámetros que son configurados en el mismo agente (remitirse a la figura 2.25). Cabe mencionar que en la mayoría de los dispositivos de los fabricantes, tales como ruteadores, switches, impresoras y demás equipos de red, el auto – monitoreo es un comportamiento que tiene el dispositivo por defecto, por ejemplo algunos de los ruteadores de la empresa Cisco generan las *traps* de manera automática en caso de encontrar algún problema en los parámetros que este monitoreando el agente de dicho dispositivo, solo se tiene que configurar a que servidor se deben enviar las *traps* así como los parámetros que se deseen monitorear, pero la generación de las *traps* es en forma automática.

El caso de la instalación del agente SNMP en servidores es diferente, el paquete NET – SNMP requiere compilarse con un módulo llamado *"disman/event-mib",* este módulo habilita el soporte para algo denominado DISMAN-EVENT-MIB, situación que le permitirá al agente revisar su propia MIB a intervalos regulares de tiempo, y enviará *traps* en caso de que se presenten ciertas condiciones, las cuales son configurables en el mismo agente.

Las *traps* son enviadas en base a condiciones de cierto o falso, cuando en algún intervalo de tiempo se encuentra que algunas de las condiciones configuradas es falsa, la *trap* es enviada al servidor NMS. La *trap* es enviada solamente una vez, y solo se vuelve a enviar otra *trap* hasta que se vuelva a repetir la falla. Esto significa que se requiere que en alguno de los siguientes intervalos de monitoreo la condición vuelva a ser cierta, y que posteriormente se vuelva a presentar una condición falsa para que la *trap* sea enviada nuevamente al NMS. EL tipo de *traps* que son enviadas, están documentadas y definidas en la MIB DISMAN-EVENT-MIB.txt, que es instalada cuando se habilita el soporte de auto – monitoreo.

2.2.5 La seguridad en SNMP

Se ha expuesto durante el capitulo 2 que un sistema gestión de red basado en el protocolo SNMP está compuesto por elementos como: varios agentes, o nodos gestionados y al menos una estación de gestión (*manager*), así como un cierto volumen de información relativa a los dispositivos gestionados y un protocolo para la transmisión de dicha información entre los agentes y las estaciones de gestión.

Es así como los mecanismos utilizados para definir la información relativa a los dispositivos gestionados apenas han sufrido modificaciones desde su aparición a finales de los años 80. Uno de los objetivos principales de su diseño fue la flexibilidad, de modo que la información definida pudiese seguir siendo utilizada posteriormente por protocolos diferentes, o incluso por distintas versiones del mismo protocolo.

Por el contrario el requisito fundamental del diseño del protocolo fue la sencillez, lo que si bien facilitó su expansión en perjuicio de protocolos más complejos, como por ejemplo CMIP[27], ha hecho necesarias varias revisiones para adaptar el protocolo a las necesidades actuales, entre las que cabe destacar las exigencias en cuanto a la seguridad del sistema.

[27] CMIP: *Common Management Information Protocol* (Protocolo de Información de Administración Común)

2.2.5.1 Amenazas a la seguridad

El protocolo SNMP proporciona mecanismos para el acceso a un almacén de información jerárquica compuesta por un conjunto de variables, a lo que se ha denominado MIB. Se distinguen dos tipos distintos de acceso a dicha información: un acceso para lectura que permite consultar los valores asociados a cada una de las variables y un acceso para escritura que permite modificar dichos valores.

Los mensajes de la primera versión del protocolo incluyen una cadena de caracteres denominada nombre de comunidad que se utiliza como un sencillo mecanismo de control de acceso a la información. Los agentes que implementan dicha versión del protocolo disponen generalmente de dos comunidades, o conjuntos de variables (no necesariamente disjuntos), identificadas por un nombre de comunidad configurable por el administrador del sistema. Una de dichas comunidades recibe el nombre de comunidad pública, y sus variables pueden ser accedidas sólo para lectura. Por el contrario los valores asociados a las variables que componen la otra comunidad, denominada comunidad privada, pueden ser modificados.

Toda la seguridad proporcionada por el sistema se basa en el hecho de que es necesario conocer el nombre asignado a una comunidad para conseguir el acceso a la información proporcionada por sus variables. El nivel de protección ofrecido por la versión original del protocolo es, por tanto, muy débil. Más aún si se tiene en cuenta que los nombres de comunidad incluidos en los mensajes del protocolo SNMP viajan por la red en texto plano y por consiguiente pueden ser obtenidos como resultado de ataques pasivos (escuchas malintencionadas).

Además, y sobre todo en el caso de la comunidad pública, está muy extendido el uso del nombre de comunidad configurado por defecto (*public*) por lo que un usuario ajeno al sistema puede obtener gran cantidad de información acerca del mismo utilizando el protocolo SNMP.

Con el fin de aumentar la seguridad del protocolo es necesario realizar cambios en su modelo administrativo para introducir los conceptos de

autentificación, integridad y privacidad así como para mejorar el control de acceso a la información.

En primer lugar se identifican las posibles amenazas a las que dicho protocolo se encuentra sometido. Las más importantes son las siguientes: modificación de los mensajes en tránsito o de su orden, suplantación y ataques pasivos (escuchas). En el caso concreto del protocolo SNMP no se consideran relevantes las amenazas de los tipos negación de servicio y análisis de tráfico.

Una versión segura del protocolo debería impedir en la medida de lo posible ataques de los tipos mencionados. El apartado siguiente describe la evolución que ha sufrido el protocolo a través de sus distintas versiones así como las principales mejoras aportadas por cada una de dichas versiones en relación a la seguridad.

2.2.5.2 Evolución de la Seguridad

Cronológicamente hablando, el primer intento serio de dotar al protocolo SNMP de un cierto grado de seguridad se corresponde con la versión denominada SNMPsec, cuyos fundamentos se definen en los RFC 1351 y 1352. Los elementos introducidos en dicha versión para mejorar la seguridad del protocolo forman la base de todas las versiones posteriores y se siguen utilizando en la actualidad.

Las principales innovaciones propuestas en la versión SNMPsec son la identificación unívoca de las entidades que participan en las comunicaciones SNMP, lo que permitirá grandes mejoras y mayor flexibilidad en cuanto al control de acceso, así como la utilización de mecanismos criptográficos para conseguir autentificación, integridad de los mensajes y privacidad.

Dicha versión introduce los siguientes conceptos:

- ***Party* SNMP**. Es un contexto virtual de ejecución cuyas operaciones se pueden encontrar restringidas a un subconjunto del conjunto total de operaciones permitidas por el protocolo. Un *party* involucra un identificador, una localización en la red utilizando un protocolo de transporte

determinado, una vista MIB sobre la que opera, un protocolo de autentificación y un protocolo de privacidad.

- **Vista sub-árbol y vista MIB**. Una vista sub-árbol es un conjunto de variables de un MIB (*Management Information Base*) que tienen como prefijo un identificador de objeto común. Una vista MIB no es más que un conjunto de vistas sub-árbol.

- **Política de control de acceso**. Es el conjunto de clases de comunicación autorizadas entre dos *parties* SNMP o lo que es lo mismo el conjunto de mensajes del protocolo SNMP cuyo uso se permite entre dos elementos participantes en una comunicación de gestión.

- **Protocolo de autentificación**. Sirve al mismo tiempo para autentificar los mensajes y para poder comprobar su integridad. Se suele utilizar un mecanismo de firmas digitales, como por ejemplo el algoritmo MD5 que calcula un *digest* del mensaje. El valor obtenido se incluye entre los datos transmitidos a la hora de llevar a cabo una comunicación.

- **Protocolo de privacidad**. Sirve para proteger las comunicaciones contra escuchas malintencionadas. Se utiliza, por ejemplo, el algoritmo simétrico de encriptación DES (*Data Encryption Standard*).

La versión SNMPsec se adopta inicialmente con la introducción de la versión 2 del protocolo SNMP y pasa a denominarse SNMPv2p (*Party-based* SNMPv2).

Posteriormente el marco de trabajo SNMPv2, cuya definición no contiene ningún estándar en cuanto a seguridad, se asocia con otros modelos administrativos referentes a seguridad, y aparecen tres nuevas versiones del protocolo: SNMPv2c, SNMPv2u y SNMPv2*.

La versión SNMPv2c (*Community-based SNMPv2*) utiliza el mismo modelo administrativo que la primera versión del protocolo SNMP, y como tal no incluye mecanismos de seguridad. Las únicas mejoras introducidas en la nueva versión consisten en una mayor flexibilidad de los mecanismos de control de acceso, ya que se permite la definición de políticas de acceso consistentes en asociar un nombre de

comunidad con un perfil de comunidad formado por una vista MIB y unos derechos de acceso a dicha vista (*read-only* o *read-write*).

La versión SNMPv2* proporciona niveles de seguridad adecuados, pero no alcanzó el necesario nivel de estandarización y aceptación por el IETF.

Por último, la versión denominada SNMPv2u (*User-based SNMPv2*) reutiliza los conceptos introducidos en la versión SNMPsec, introduciendo la noción de usuario. En este caso, las comunicaciones se llevan a cabo bajo la identidad de usuarios en lugar de utilizar el concepto de *party* existente en las versiones precedentes. Un mismo usuario puede estar definido en varias entidades SNMP diferentes.

2.2.5.3 La seguridad en la versión 3 del protocolo

La principal novedad introducida en la versión 3 del protocolo SNMP es la modularidad. En dicha versión una entidad SNMP se considera compuesta por un motor y unas aplicaciones. A su vez el motor se divide en cuatro módulos: distribuidor, subsistema de proceso de mensajes, subsistema de seguridad y subsistema de control de acceso.

Se observa, por tanto, que en la versión SNMPv3 se independizan los mecanismos utilizados para la seguridad (autentificación y privacidad) y para el control de acceso. De este modo, una misma entidad puede utilizar diferentes modelos de seguridad y control de acceso simultáneamente, lo que incrementa notablemente la flexibilidad y la interoperabilidad.

Se define un modelo estándar para seguridad basada en usuarios, USM (*User Security Model*) y otro para control de acceso basado en vistas, VACM (*View-based Access Control Model*). Se aprovechan los conceptos definidos en las versiones previas y al mismo tiempo la característica monolítica del protocolo permite la introducción de futuros modelos independientes de los actuales.

2.3 Conclusiones

El nivel de seguridad proporcionado por la versión original del protocolo SNMP no es adecuado para las necesidades actuales. En caso de utilizar una gestión basada en dicha versión es imprescindible estar informado de los riesgos involucrados, debido a la sensibilidad de la información accesible a través de dicho protocolo. Además, se debe intentar en la medida de lo posible disminuir la influencia de dichos riesgos utilizando, por ejemplo, nombres de comunidad complejos. Incluso se puede restringir el acceso utilizando mecanismos externos al protocolo, como por ejemplo listas de control de acceso implementadas sobre *firewalls*.

Versiones posteriores del protocolo han hecho uso de mecanismos criptográficos y de listas de control de acceso para proporcionar mayor seguridad en las comunicaciones. La utilización de dichas versiones trae consigo una mayor dificultad en la configuración del sistema pero el nivel de seguridad obtenido es mucho mayor. Debido a estas situaciones se ha decidido utilizar para el presente proyecto de tesis la versión SNMPv3, la cual incorpora toda la funcionalidad y características de las versiones anteriores pero con los beneficios de la autenticación y encriptación de datos antes mencionados, asegurando todavía más la confidencialidad de los datos al utilizar listas de control de acceso por medio del firewall del dispositivo gestionado, agregando un mecanismo externo de seguridad al protocolo SNMP.

Capitulo 3
Diseño de la Solución

3.1 Contexto de la implementación

El presente capitulo tiene como objetivo describir el análisis y diseño de la solución, para ello es importante plantear el contexto general en el que se adecuó la implementación. La idea general propuso reforzar el anterior esquema de monitoreo mediante la incorporación de herramientas que enriquezcan esta tarea, por un lado se tuvo la necesidad de la generación de gráficas, donde el monitoreo se encuentra implícito debido a la representación de los recursos y servicios de los servidores, y por el otro un esquema que permitiera ampliar las alertas con las que contaba el departamento. Ante esta situación se observó que se requería dividir los esfuerzos en dos caminos:

- **Generación de reportes gráficos**. Esta parte de la solución conllevó un análisis acerca de las diferentes herramientas existentes de graficación, su forma de operación y requerimientos que permitirán generar para cada uno de los servidores una serie de gráficas en relación al monitoreo de recursos y servicios del propio servidor. De esta forma se planteó contar con una serie de herramientas, que juntas, permitan tener un sistema centralizado de gráficas, teniendo como parte importante de su funcionamiento al protocolo de administración de red SNMP.

- **Esquema de alertas.** El objetivo de esta parte de la implementación fue complementar el sistema de monitoreo mediante la incorporación de un sistema de alertas en cada uno de los servidores, tarea que esta estrechamente ligada con el protocolo SNMP.

Puede parecer que se estaban planteando dos soluciones totalmente diferentes, sin embargo ambas implementaciones tuvieron un objetivo común, complementar y reforzar el monitoreo que se lleva actualmente en el Departamento de Administración de Servidores. Para dejar más en claro las actividades realizadas durante esta etapa, se hará uso de un cuadro cronológico (tabla 3.0) en el cual se podrá observar paso a paso todo el desarrollo de la tesis y ubicar perfectamente en donde tuvo lugar esta fase de análisis y diseño que será expuesto a lo largo de este capitulo.

	Abr - 06				May – 06				Jun – 06				Jul – 06				Ago – 06				Sep - 06			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Planteamiento del problema																								
Propuestas de Solución																								
Análisis y Diseño																								
Pruebas y Validación																								
Implementación																								

Tabla 3.0 Cuadro cronológico del desarrollo de tesis

A continuación se describirán a detalle cada una de las soluciones a las que se llegó en base al análisis y diseño de las diferentes propuestas.

3.2 Diseño de la Graficación

La sola implementación del protocolo SNMP no permitía alcanzar el objetivo de la generación de gráficas, era necesaria la colaboración de otras herramientas informáticas, que interactuando con las características de SNMP, facilitaran una sistema donde fueran generadas las gráficas de todos los servidores del departamento. Esta tarea debía tener como premisa, ser una solución totalmente diseñada con software

libre, por lo que el análisis solo abarco herramientas que cumplieran con este requerimiento.

Esta tarea implicó el análisis de varias herramientas que permitieran generar gráficas como las pretendidas por el presente proyecto de tesis. El tipo de herramienta a utilizar debía cumplir principalmente con 3 características que se planteaban en ese momento:

- La herramienta debía ser capaz de recabar y acumular información de los servidores desde un solo equipo, para posteriormente ser representada mediante gráficas periódicas. De esta manera se tendrán los datos de todos los servidores consultando desde una sola Terminal de trabajo.
- EL software debía utilizar al protocolo SNMP para recopilar la información de interés de cada uno de los servidores.
- La herramienta no debía limitarse a solo desplegar las gráficas, debe ofrecer un esquema de administración que permita fácilmente tener un control acerca de la creación y visualización de las mismas.

3.2.1 Análisis de herramientas de graficación

Existen varios tipos de programas para llevar a cabo la graficación de sistemas, dentro de estas opciones solo se seleccionaron estudiar aquellas que cumplieran de alguna manera con las características anteriormente planteadas. Si bien muchas de estas opciones comparten una forma similar de operación, también es cierto que existen determinadas circunstancias que dieron pie a la decisión de la opción mas adecuada. A continuación se presenta la tabla 3.1 donde se listan las opciones y posteriormente se analizarán sus ventajas y desventajas en base a sus características.

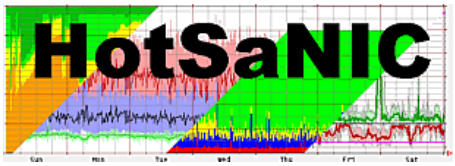
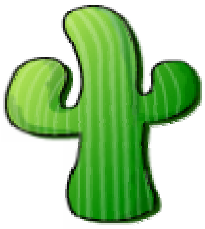
Herramienta	Logo	URL de Consulta
Cricket	**Cricket**	http://cricket.sourceforge.net/
HotSANiC		http://hotsanic.sourceforge.net/ http://merlin.com.ua/doc/rrd/rrdworld/hotsanic.html http://www.bernisys.prima.de/linux/HotSaNIC/
Percival	SOURCEFORGE™ .net	http://percival.sourceforge.net/
Nagios	MONITORED BY Nagios NETWORK MONITOR	http://nagios.org http://www.nagios.org
MRTG	MRTG® MULTI ROUTER TRAFFIC GRAPHER	http://www.mrtg.com/
CACTI		http://www.cacti.net

Tabla 3.1 Herramientas de graficación

3.2.1.1 Cricket

Esta herramienta es un sistema muy flexible para monitorear grandes series de datos. Cricket fue expresamente desarrollado para ayudar a los administradores de red a tener una visión mas global que facilitara la detección de los problemas de tráfico de sus redes de datos, aun que esta solución también puede ser usada para otro tipo de monitoreos.

Cricket tiene dos componentes principales, un colector y un graficador. El colector corre por medio de una tarea programada para cada 5 minutos (o diferente periodo si así se desea) y almacena los datos dentro de una estructura controlada por una herramienta llamada RRDtool. Posteriormente, cuando se desea consultar los datos que han sido recolectados puede ser utilizada una interfaz web que se encargará de desplegar las gráficas.

Cricket esta desarrollado totalmente en lenguaje de programación Perl y es distribuido bajo GNU[28]. A pesar de que fue desarrollado para trabajar bajo sistemas operativos Solaris junto con un servidor web Apache, es conocido que ha sido implementado con éxito otras variantes de sistema operativo UNIX, tal como Linux, HP-UX, etc. Algunos los han implementado exitosamente bajo sistemas Windows NT de Microsoft, sin embargo la documentación acerca de esto es muy escasa.

Ventajas

- Permite tener desde un solo equipo contar con una Terminal de monitoreo, donde las gráficas pueden ser consultadas desde una interfaz web de la misma aplicación.
- La recopilación de información puede ser obtenida a través de cualquier aplicación externa a cricket, sin embargo la forma más común de hacerlo es por medio del protocolo SNMP, por lo que aplicación tiene soporte para trabajar conjuntamente con el protocolo.
- Comparado con otras aplicaciones de graficación, no se requiere instalar gran cantidad de software para que realice su operación.

[28] GNU: Licencia Publica General

Desventajas

- La configuración de la aplicación es complicada, más si tomamos en cuenta que la propia pagina oficial de la herramienta no cuenta con una documentación suficiente para configurar y poner en marcha la herramienta.
- Aunque cuenta con soporte para SNMP, los programas solo aceptan el nombre de la comunidad para comunicarse con agentes SNMP, de manera que la comunicación con entidades SNMPv3 no esta soportada.
- La interfaz web no dispone de un módulo que permita ningún tipo de administración, por lo que la tareas como la creación de gráficas están fuera del alcance de la interfaz.

3.2.1.2 HotSANiC

Las siglas de esta herramienta significan Vista Global HTML para Sistemas y Centros de Información de Red. HotSANiC consiste en un conjunto de programas realizados en Perl para generar un sistema de graficación, actualmente soportado solo para sistemas operativos Linux y BSD.

Ventajas

- Los requerimientos para poner en funcionamiento la aplicación son mínimos, en comparación con otras herramientas.
- La configuración y creación de graficas es poco complicada.
- Cuenta con una interfaz web de manera que los datos pueden ser consultados desde cualquier equipo en red.

Desventajas

- No utiliza al protocolo SNMP para la recopilación de la información, esta se realiza por medio de programas escritos en lenguaje Perl.
- La aplicación dispone de una interfaz web muy simple, no posee otra característica más que las de mostrar las graficas de los datos.

3.2.1.3 Percival

Esta herramienta es un sistema para el monitoreo de redes, cuya interfaz web ofrece grandes funcionalidades. Este software fue desarrollado en base a otras herramientas de graficación, mejorando algunas características de estos sistemas.

Percival fue desarrollado como respuesta al excesivo costo de las herramientas comerciales de graficación, por lo que reúne características similares a tales herramientas como la creación de perfiles de usuario, un modelo simple de configuración, gran rendimiento, generación de reportes, etc.

Ventajas

- Al instalar la aplicación en el equipo, se cuenta con una Terminal de graficación, donde la información puede ser consultada a través de una funcional interfaz web.
- La aplicación no solo utiliza al protocolo SNMP para recabar la información de los servidores, también cuenta con soporte para diferentes bases de datos de administración MIB 2 para dispositivos de red para diferentes fabricantes.

- Esta desarrollado para ser implementado solo en sistemas operativos Linux y Solaris.
- La interfaz web brinda un esquema de administración, donde se pueden crear perfiles de usuario y otras configuraciones para el control y visualización de las gráficas.

Desventajas

- Es considerable la cantidad de software a instalar y configurar en comparación con herramientas de similares características.
- Es necesario invertir considerable tiempo para aprender el uso de las características de la misma herramienta.
- A pesar de que el software fue desarrollado a partir de herramientas de libre distribución, hay que pagar una mínima licencia por el uso de la aplicación.

3.2.1.4 Nagios

Nagios es una aplicación para el monitoreo de sistemas y redes. Verifica equipos y servicios que le son especificados, teniendo la capacidad de generar alertas en caso de que las cosas no funcionen de acuerdo a lo esperado.

Nagios fue originalmente diseñado para correr en el sistema operativo Linux, pero también se ha probado que puede trabajar en la mayoría de los UNIX. Nagios es un software que no solo tiene la capacidad de generar gráficas, brinda todo un soporte para el monitoreo de varios servicios fundamentales en cualquier área de cómputo.

Ventajas

- No requiere de gran instalación de software para su funcionamiento.
- Proporciona una interfaz web para ver el estado actual de la red, notificaciones, historial de problemas, archivo log, etc.
- Ofrece gran cantidad de monitores para servicios de red tales como: SMTP, POP3, HTTP, NNTP, PING,etc.
- La interfaz web brinda un esquema de administración y soporte que permite de manera sencilla la generación de gráficas para el monitoreo de los servidores(carga en el procesador, uso de disco duro, etc)
- Efectúa notificaciones a contactos cuando un servicio o equipo presenta problemas y necesita resolverse (via email, page o método definido por el usuario).
- Habilidad de definir manejadores de eventos para ser ejecutados cuando un servicio o equipo presente eventos para que pueda ser preactiva la resolución del problema.

Desventajas

- Es considerable la cantidad de tiempo a invertir en el aprendizaje de la herramienta. Al ser una aplicación con tantas características, conlleva involucrarse en el manejo de varios conceptos para lograr una configuración que permita operar a la aplicación.
- Aun que cuenta con soporte para SNMP, la forma de operar de la aplicación esta basado en programas escritos en Perl o mediante programas CGI.

3.2.1.5 MRTG

MRTG es una herramienta, escrita por Tobias Oetiker y Dave Rand, para monitorizar la carga de tráfico sobre determinados nodos de una red. MRTG genera páginas HTML que incluyen representaciones gráficas, en formato GIF[29], del tráfico registrado en un determinado nodo de la red.

MRTG consiste en un script en Perl que utiliza SNMP para obtener información de gestión sobre los nodos de la red y un programa en C para generar los registros de tráfico (logs) y crear representaciones gráficas de los datos recopilados. Estos gráficos se integran dentro de un documento en formato HTML.

Mediante MRTG es posible monitorizar cualquier variable SNMP que se quiera, de manera que se puede configurar para monitorizar la carga de un sistema, las sesiones abiertas por los usuarios de un determinado equipo, disponibilidad de modems. MRTG permite generar gráficas con cuatro niveles de detalle por cada interfaz: tráfico registrado en las últimas 24 horas, la última semana, el último mes y gráfica anual. Además de generar una primera página con la representación del tráfico registrado diariamente a través de cada uno de los posibles interfaces de un router.

Ventajas

- Su instalación y configuración no requiere de grandes esfuerzos en comparación a otras aplicaciones.
- Las graficas de los servidores pueden ser consultadas via web a través de un solo equipo.
- Utiliza al protocolo SNMP para recabar y acumular la información a graficar.

[29] GIF: *Graphics Interchange Format* (Formato de Intercambio de Gráficos)

Desventajas

- La aplicación solo genera páginas web simples donde son mostradas las graficas, no posee características de administración o mantenimiento.
- La generación de graficas es totalmente manual, por lo que entre más sea la cantidad de graficas a generar, mas complicado se vuelve esta tarea en la aplicación.

3.2.1.6 CACTI

Cacti es una herramienta de Software Libre desarrollada en SourceForge.net. El desarrollador original fue Ian Berry, respaldado por las aportaciones constantes realizadas por numerosas personas de la comunidad del Software Libre.

Cacti es un front-end[30] totalmente basado en la web para trabajar en conjunto a una aplicación llamada RRDtool, la cual es una aplicación estándar industrial para registro de datos y creación de gráficos que puede utilizarse para escribir scripts personalizados de monitorización y aplicaciones completas.

Cacti es una completa solución de graficado en red, diseñada para aprovechar el poder de almacenamiento y la funcionalidad de graficación que poseen las RRDtool. Esta herramienta, desarrollada en PHP, provee una interesante y funcional interfaz, que hacen que esta aplicación resulte muy conveniente para redes del tamaño de una LAN, así como también para redes complejas con cientos de dispositivos.

[30] *front-end:* Denominado así al principio de un proceso, es la parte del software que interactúa con el usuario.

Ventajas

- La aplicación no requiere de instalar gran número de software para realizar su operación, tomando en cuenta las características que ofrece.
- La forma fundamental de adquisición de los datos es a través del protocolo SNMP, aun que también tiene la flexibilidad de recopilar estos datos a través de scripts personalizados.
- A través de la interfaz web, cacti permite crear prácticamente cualquier gráfica, utilizando todos los estándares de tipos de gráficas de RDDtool, además hay varias formas de mostrarlas. Junto con una "lista de vistas" estándar y una "vista preliminar", también existe una "vista en árbol", la cual permite colocar gráficos un árbol jerárquico, para propósitos organizacionales.
- Incluye la generación de plantillas gráficas avanzadas, lo que permite la creación de una única plantilla de gráficos o fuente de datos, la cual define cualquier gráfico o fuente de datos asociada. De esta manera, no se tiene que volver a crear algún determinado tipo de gráfica para un nuevo host, solo bastara crear la plantilla con los datos que se deseen graficar y aplicarla a cuantos hosts sea requerido.
- Cuenta con la funcionalidad de manejo de usuarios, asi que es posible agregar un usuario y darle permisos a ciertas áreas de Cacti. Esto permite tener usuarios que puedan cambiar parámetros de un gráfico, mientras que otros sólo pueden ver los gráficos. Asimismo, cada usuario mantiene su propia configuración de vista de gráficos.

Desventajas

- Al ser una herramienta que ofrece bastantes características, se tiene que manejar una serie de conceptos para poder operar y aprovechar el potencial de la herramienta, por lo que el tiempo de aprendizaje puede ser considerable.

3.2.1.7 Elección de la herramienta

La decisión de la herramienta tuvo que apegarse lo más posible a las características planteadas al principio de este apartado, por lo que después del análisis de varias opciones se llegó a una serie de conclusiones acerca del camino a seguir.

La primera aplicación que se descartó fue HotSANiC, la principal desventaja es que los datos no son recopilados mediante SNMP aunado a que la interfaz web que ofrece es muy sencilla. Por otro lado teníamos a las herramientas Cricket y MRTG, ambas trabajan en conjunto con SNMP para recabar la información de los servidores, Cricket tiene una configuración más compleja comparada con MRTG, sin embargo ambas herramientas ofrecen características similares, limitándose a desplegar la gráficas por medio de una interfaz web. A pesar de que ambas cumplían con el objetivo de la generación de graficas, se buscaba una aplicación que tenga beneficios adicionales, algo con características administrativas que permitan generar y mantener fácilmente las gráficas. En este sentido las restantes aplicaciones analizadas, Percival, Nagios y Cacti brindaban este tipo de utilerías, contando con una interfaz más funcional.

Nagios resulto ser una herramienta muy completa para el monitoreo de servicios de red y sistemas, cuenta con soporte para la revisión de la mayoría de los servicios de red fundamentales en un área informática. Sin embargo no se consideró la opción más viable debido a que la capacidad de la aplicación rebasaba las pretensiones de la solución buscada, resultaba costoso el tiempo a invertir en el aprendizaje y configuración de la propia herramienta solo para utilizar la generación de graficas. Debido a sus características resultaría la opción mas conveniente en caso que el departamento no contara con ningún sistema de monitoreo, sin embargo el monitoreo actual realiza muchas de las tareas para las cuales Nagios fue desarrollado.

Al final del análisis, fueron dos las soluciones más viables para hacer el desarrollo de nuestra propuesta, tanto Percival como Cacti, son herramientas enfocadas a la generación de graficas, ambas utilizan al protocolo SNMP para recopilar los datos, del mismo modo brindan una interfaz web desde la cual se pueden crear perfiles de usuario y otras configuraciones para el control y visualización de las gráficas. Haciendo una comparación final entre estas dos aplicaciones, podemos mencionar lo siguiente:

- Percival requiere una instalación y configuración de software más complicada en comparación a Cacti.
- Cacti ofrece algunas ventajas con respecto al mantenimiento, creación y forma de visualización de las gráficas.
- Existe una documentación más detallada acerca de la instalación y mantenimiento de Cacti en comparación a Percival, hay más información de que Cacti ha sido implementado en muchas empresas, tanto privadas como públicas, con muy buenos resultados en comparación con Percival.
- Aun que Percival fue desarrollado a partir de software de libre distribución, los desarrolladores de la aplicación hacen el cobro de una licencia para su uso.

Debido a estas circunstancias Cacti fue elegido como la herramienta más viable para llevar a cabo la solución de graficación para el presente proyecto de tesis. Es una útil y flexible herramienta para llevar a cabo la graficación de los recursos y servicios del área, cuyas características cumplen con las deseadas por el diseño de la solución propuesta. En la tabla 3.2 podemos apreciar las características de las diferentes herramientas que fueron analizadas, en donde se puede observar de una manera mas clara todo lo comentado a lo largo de este apartado, en cuanto a la elección de la herramienta de graficación.

Herramienta	Cricket	HotSANiC	Percival	Nagios	MRTG	CACTI
Libre Distribución	SI	SI	NO	SI	SI	**SI**
Soporte SNMP	SI	NO	SI	NO	SI	**SI**
Centralización de Datos	SI	Si	SI	SI	SI	**SI**
Requerimientos de Software	BAJA	BAJA	ALTA	BAJA	BAJA	**MEDIA**
Complejidad de Instalación	BAJA	BAJA	ALTA	MEDIA	BAJA	**ALTA**
Características de Administración	NO	NO	SI	SI	NO	**SI**
Complejidad de Manejo	ALTA	BAJA	ALTA	ALTA	MEDIA	**ALTA**

Tabla 3.2 Análisis de las Herramientas de Graficación

3.2.2 Módulos que integran la solución

La solución es una completa herramienta de graficado en red, esta fue desarrollada para trabajar en un ambiente web, por medio del cual se aprovecha el poder de almacenamiento y la funcionalidad de graficar que poseen las RRDtool. Esta aplicación, programada en lenguaje PHP, provee una interfaz web eficiente donde se pueden crear plantillas de gráficos avanzadas, múltiples métodos para la recopilación de datos, funciones de administración de usuarios y la capacidad de sondear rápidamente cientos de dispositivos en una red.

Sin embargo Cacti es solo una aplicación Terminal que se auxilia de otras herramientas para llevar a cabo todas las funcionalidades comentadas anteriormente, de esta manera el usuario de la aplicación se abstrae de los procesos internos que

conlleva la recopilación y graficación de los datos, puesto que se ofrece una manera rápida y fácil de crear y administrar las gráficas. Estas utilerías o módulos que son requeridos por Cacti necesitan ser configurados para trabajar juntos de manera que se pueda presentar la información en una interfaz gráfica bajo un entorno web. Estos módulos comprenden un servidor web, un manejador de base de datos, un lenguaje de programación de lado servidor y una herramienta capaz de generar gráficos. Todos estos módulos serán descritos en los siguientes apartados.

3.2.2.1 Protocolo SNMP

La instalación de SNMP permitirá a Cacti recopilar los datos de los servidores para posteriormente llevar a cabo las gráficas. A pesar de que Cacti dispone de múltiples métodos de adquisición de datos, como pueden ser la creación de scripts personalizados para reunir datos que no se puedan obtener vía SNMP, la aplicación tiene como principal vía de recopilación de información al protocolo SNMP, contando de forma predeterminada con plantillas de gráficos que utilizan a SNMP como recolector de datos.

La implementación del protocolo SNMP sobre el equipo le otorgará los programas necesarios para que la aplicación pueda establecer comunicación con los hosts que se desee monitorear, es así como el servidor de graficación, denominando así al equipo donde se implementará la aplicación de Cacti, se encargará de recopilar la información de todos los servidores. Desde el punto de vista de los componentes de SNMP este servidor será el gestor o estación administradora, la cual se encargará de cuestionar y almacenar la información de todos y cada uno de los agentes SNMP.

No basta instalar el protocolo SNMP en el servidor de graficación, para que este esquema sea funcional es necesario instalar y configurar adecuadamente el protocolo SNMP en cada uno de los servidores que se tenga planeado el monitoreo y graficación

de recursos, de otra manera el servidor de graficación no podrá establecer comunicación.

3.2.2.2 RRDtool

Cacti se auxilia de una herramienta para poder generar los gráficos sobre una página html, esta es denominada RRDtool. Esta aplicación es un estándar industrial para registro de datos y creación de gráficos. Es un programa que fue escrito por Tobias Oetiker con la colaboración de muchas personas de diversas partes del mundo. Las siglas significan herramienta de base de datos round robin, esta técnica implica un número fijo de datos, y un apuntador al elemento más reciente. Este concepto es como un círculo en el cual se han dibujado una serie de puntos alrededor de su borde, estos puntos representan el lugar donde se almacenarán los datos. Ahora visualicemos al apuntador que leerá y escribirá los datos como a una flecha desde el centro del círculo a uno de los puntos. Cuando se lee o escribe el dato actualmente apuntado, la flecha se mueve al próximo elemento. Al tener una estructura como un círculo, no hay ni principio ni fin, por lo que se podrá guardar y leer la información por siempre. Es claro anticipar que al cabo de un tiempo ya se habrán usado todas las posiciones disponibles y el proceso empieza a reutilizar las antiguas, por ello la aplicación solo guarda un historial de 12 meses. De esta forma, la base de datos no crece en tamaño y, por lo tanto, no requiere ningún mantenimiento. RRDtool trabaja con estas bases de datos tipo round robin, guardando y recuperando datos de ellas.

RRDtool es una evolución de la herramienta MRTG. En un principio MRTG funcionaba a base de un pequeño script que proporcionaba los datos para graficar el uso de una conexión a la Internet. Luego evolucionó, permitiendo graficar otras fuentes de datos, como temperatura, velocidad, voltajes, cantidad de páginas impresas, etc. RRDtool te permite crear una base de datos, guardar los datos en ella, recuperarlos y crear gráficos en formato GIF o PNG para posteriormente mostrarlos en un navegador web. Las imágenes que se crean dependen de los datos que hayan alamcenado, por lo que pueden representar cualquier tipo de fenómeno que se pretenda estudiar, como

por ejemplo un sumario del promedio de uso de la red, los picos de tráfico, o incluso para fines de investigación como mostrar el nivel de las mareas, la radiación solar, el consumo de electricidad, los niveles de ruido en la periferia de un aeropuerto, la temperatura de alguna región, etc. Debido a estas características no habrá ningún inconveniente para generar cualquier tipo de gráfica que se desee, siempre y cuando se disponga de una fuente de datos y la capacidad de alimentar esa información a la base RRDtool.

3.2.2.3 Manejador de base de datos Mysql

Cacti necesita de una base de datos para poder llevar a cabo su funcionamiento. Esta base no tiene nada que ver con la utilizada para los datos de graficación, esta es requerida para manejar otro tipo de información referente a la operación interna de la herramienta. Cacti trabaja en conjunto con un manejador de base datos llamado Mysql, el cual es una solución de libre distribución para manejar bases de datos relacionales.

3.2.2.4 Servidor web Apache

Cacti requiere de la instalación de un servidor web, de esta manera el servidor de graficación podrá ser consultado desde cualquier equipo en la red. Puesto que la implementación es sobre el sistema operativo Linux, el servidor web nativo para este ambiente es Apache, por lo que Cacti trabaja bajo este esquema sin ningún problema.

3.2.2.5 Lenguaje PHP

Hasta ahora hemos visto todos los módulos de los que la aplicación se auxilia para llevar a cabo su funcionamiento, sin embargo no hay que pasar por alto que como tal la aplicación de Cacti esta desarrollada en lenguaje PHP, un lenguaje de programación de lado servidor, mediante el cual se basa la operación e interacción con

los módulos que conforman la aplicación. Debido a esto es necesario compilar e instalar PHP para que el servidor Apache sea capaz de interpretar el código de la aplicación.

3.2.3 Relación entre los módulos

A través de este apartado se describirá de forma general los procesos que se llevan a cabo para la generación de gráficas. Estos procesos involucran la interacción de todos los módulos vistos anteriormente, por lo que se espera que la siguiente información brinde un panorama claro de la forma de operación de Cacti.

Para explicar el proceso global que conlleva la graficación, es necesario dividirlo en 2 etapas, la primera de ellas esta relacionada con la recolección de los datos, denominada “poller”, mientras que la segunda etapa se encarga de entregar esos datos para que se lleve a cabo la construcción de las gráficas. A través de estos procesos entra la colaboración conjunta de todos los módulos vistos en el apartado anterior, logrando de esta manera una herramienta de graficación muy flexible y poderosa.

3.2.3.1 Recolección de datos

Tanto la adquisición de los datos para la graficación se realiza de forma periódica cada determinado tiempo, se considera que cada 5 minutos es un periodo adecuado. La recolecta de la información se realiza a través del protocolo SNMP, estableciendo comunicación con cada unos de los servidores que han sido preestablecidos, sin embargo ocurren una serie de procesos antes de llegar a esta etapa. A continuación se describirán una serie de fases por las que pasa el proceso de recolección de datos.

Fase 1

Al iniciar la recolección de los datos, la aplicación requiere 2 cosas, la primera es la información de los servidores que se deseen monitorear y la segunda es que tipo de datos se quieren obtener de dichos servidores. Para ello se ejecuta un programa en PHP que se encargará de consultar a la base de datos Mysql, esta consulta se realiza principalmente sobre 2 tablas de la base *host* y *poller_graph*. En la tabla host se tiene los datos que permitirán la comunicación con los servidores, información como el nombre descriptivo, la dirección IP, la versión de SNMP utilizada, la comunidad SNMP o el usuario y contraseña en caso de utilizar la versión 3 de SNMP, así como el puerto de comunicación. En la segunda tabla se tienen que datos de la MIB que se quieren obtener en cada servidor (remitirse a la figura 3.3).

La información contenida en ambas tablas de la base de datos es capturada mediante la aplicación web, por medio de páginas que contienen código PHP. La tabla host es llenada por medio de una página para dar de alta los dispositivos que se deseen monitorear, mientras que la segunda tabla poller_graph contiene los datos de la MIB asociados a cada servidor como producto de las graficas vinculadas a cada servidor.

Fase 2

Una vez que se han obtenido todos los datos necesarios para llevar a cabo la comunicación, el programa inicia un proceso por cada servidor a cuestionar. De esta manera la aplicación comienza a obtener los datos mediante el protocolo SNMP invocando al programa “snmpwalk”, a través del cual es consultado el agente SNMP obteniendo los datos requeridos de su MIB. Como resultado de este proceso la información recolectada es almacenada temporalmente dentro de la base de datos Mysql (figura 3.3).

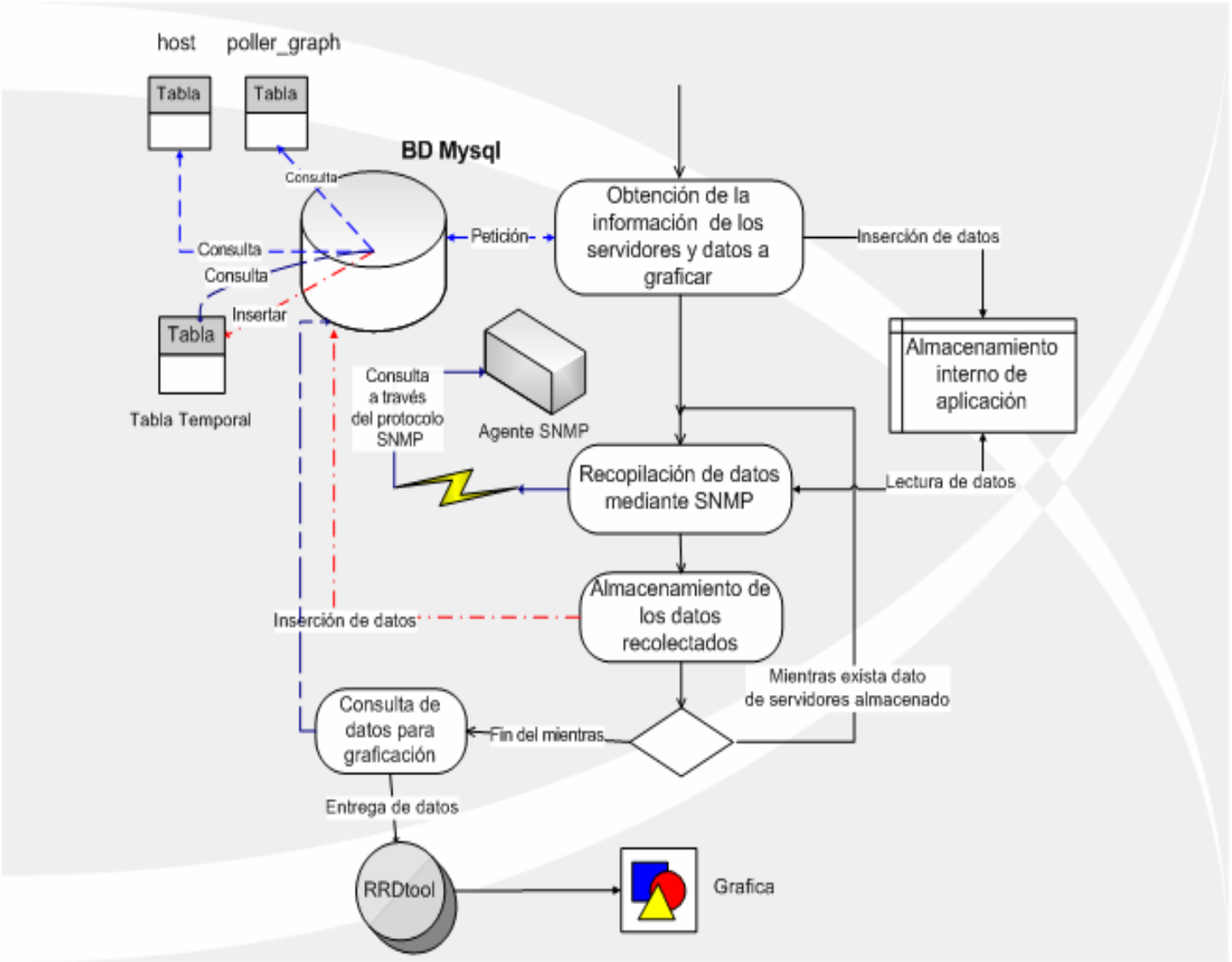


Figura 3.3 Interacción entre los módulos de Cacti

3.2.3.2 Graficación de datos

Una vez terminado el proceso de recolección comienza la fase de graficación. Debido a que la información ya ha sido capturada en la base de datos Mysql, solo resta iniciar el proceso que se encargará de entregar estos datos a la base de datos RRDtool para que se geste la conformación de las gráficas por cada servidor. Para que se lleve a cabo esta tarea se invoca a otro programa escrito en PHP que se encarga de recuperar los datos de la base y entregarlos a RRDtool para que se lleva a cabo la

construcción de la grafica, de forma predeterminada se realizan graficas con los datos de las últimas 24 horas, pero mediante las paginas interactivas escritas en PHP pueden verse rangos de horas, días, semanas, meses y de un año a la fecha.

3.3 Diseño del esquema de alertas

En el capítulo 2 de este trabajo se trató a fondo las características de la traps SNMP, exponiendo su concepto, funcionamiento, así como los tipos y variantes. Hablando respecto al servidor de traps, se había planteado que existen en el mercado gran variedad de software propietario para lograr que el NMS pudiera ser capaz de recibir y manejar las traps provenientes de los agentes SNMP. Sin embargo resultaría infructuoso hacer un análisis de estas opciones debido a que la filosofía a seguir es utilizar herramientas de libre distribución para evitar un gasto considerable en la compra de alguna de estas herramientas. Debido a esta situación la búsqueda de la solución se centro en software de libre distribución, aun que tampoco se decidió realizar un análisis exhaustivo ya que la opción más viable era utilizar el propio servidor de traps de la distribución SNMP. Lo anterior fue decidido en base a 2 razones, la primera es que el servidor de traps, de la propia distribución, no tendría problemas en entender el formato de la traps puesto que la misma herramienta es la utilizada para generar la traps, por lo tanto existe una compatibilidad implícita. La segunda razón es por que el manual de referencia de la misma distribución recomienda utilizar su propio servidor de traps para el esquema de alertas.

3.3.1 Partes que integran de la solución

Una vez que se decidió utilizar el propio servidor de traps del agente SNMP, es necesario contextualizar los componentes que integran el esquema de alertas, que principalmente esta integrado por dos partes:

- **Servidor de traps.** Es la parte encargada de recibir y manipular la trap, después de haber sido configurado para procesar y manejar la información

de las traps que se crean convenientes para fines de administración.

- **Agentes SNMP.** Serán los dispositivos que serán monitoreados, por lo que será necesario configurar el tipo de información que será consultada y establecer el tipo de monitoreo en cada dispositivo.

Por parte del servidor de traps, fue necesaria una configuración que permitiera determinar que traps serían procesadas y definir las acciones que se llevarían a cabo en respuesta a dichas alertas.

Del lado de los agentes SNMP fue necesario analizar el proceso que se genera cuando la trap es enviada desde un agente, ya que esta situación llevó al estudio de los métodos que los dispositivos tenían que implementar para llevar a cabo el monitoreo de su propia información de administración.

3.3.1.1 Auto monitoreo del agente SNMP

En el caso de muchos dispositivos de red, tales como switches, ruteadores, impresoras, etc, de diversos fabricantes, implementan un agente SNMP que es capaz de lanzar una trap en caso de presentarse una situación anómala en su funcionamiento. En dichos dispositivos solo es necesaria la configuración de una comunidad SNMP para la comunicación y de establecer cual es la dirección IP servidor que se encargará de recibir las traps. Las traps son generadas automáticamente y estas ya están definidas en la propia MIB del dispositivo, traps que son conocidas como específicas de empresa.

La implementación del agente SNMP sobre un servidor es diferente a lo anterior.

En este caso las alertas no son generadas de forma automática, por lo que hay que decidir el tipo de alternativa que permita al host contar con algún mecanismo de auto-monitoreo para generar las traps en caso de encontrar una condición anómala. A continuación se analizarán dos opciones con respecto a esta situación.

3.3.1.1.1 Monitoreo por Scripts

La instalación del paquete Net-SNMP, provee un agente SNMP extensible, esto significa que el servidor no esta limitado a comportarse como un agente SNMP, el cual solo es consultado desde un gestor. El agente extensible es una entidad SNMP que tiene la capacidad de recuperar información de otro agente así como ser consultado por otra entidad SNMP. Debido a esta situación el propio agente cuenta con las herramientas necesarias para consultar la MIB de otros dispositivos así como su propia información de administración. Teniendo en cuenta estos aspectos resulta viable la opción de desarrollar una serie de scripts que permitan recuperar los datos que se deseen monitorear de la propia MIB del servidor.

Esto implicaría el uso de programas como “snmpwalk” y “snmpget” con los cuales se consultaría de forma local las ramas de la MIB donde se encuentren los datos que se pretendan monitorear. Una vez desarrollados los scripts, solo sería cuestión de programar la ejecución de estos programas en forma periódica, lo que sería fácil de conseguir usando la herramienta de programación de tareas “crontab” del sistema operativo Linux.

3.3.1.1.2 Soporte de monitoreo DISMAN-EVENT

La propia distribución del Net-SNMP proporciona un modulo capaz de proporcionar un soporte para monitoreo de eventos. Para habilitar esta característica es necesario compilar el software agregando un modulo llamado “*disman/event-mib*”, situación que creará una base de datos llamada DISMAN-EVENT-MIB. De esta manera se tendrá un agente SNMP capaz de monitorearse a si mismo a intervalos regulares de tiempo, bajo ciertas reglas y parámetros que son configurados desde el mismo agente.

En el caso de encontrarse un parámetro que refleje una condición anómala en el dispositivo, el agente consultará las traps definidas en la base DISMAN-EVENT-MIB y enviará la trap definida para este tipo de problema.

De esta manera se cuenta con una forma estandarizada para monitorear periódicamente la información de administración del propio agente SNMP, evitando el desarrollo de scripts que podrían no funcionar en otras versiones de la propia distribución.

3.3.1.1.3 Diagrama del proceso de auto monitoreo

Cualquiera que sea la opción para que el agente SNMP lleve a cabo el monitoreo de su MIB, el proceso pasa por una serie de etapas que son descritas en el siguiente diagrama (figura 3.4).

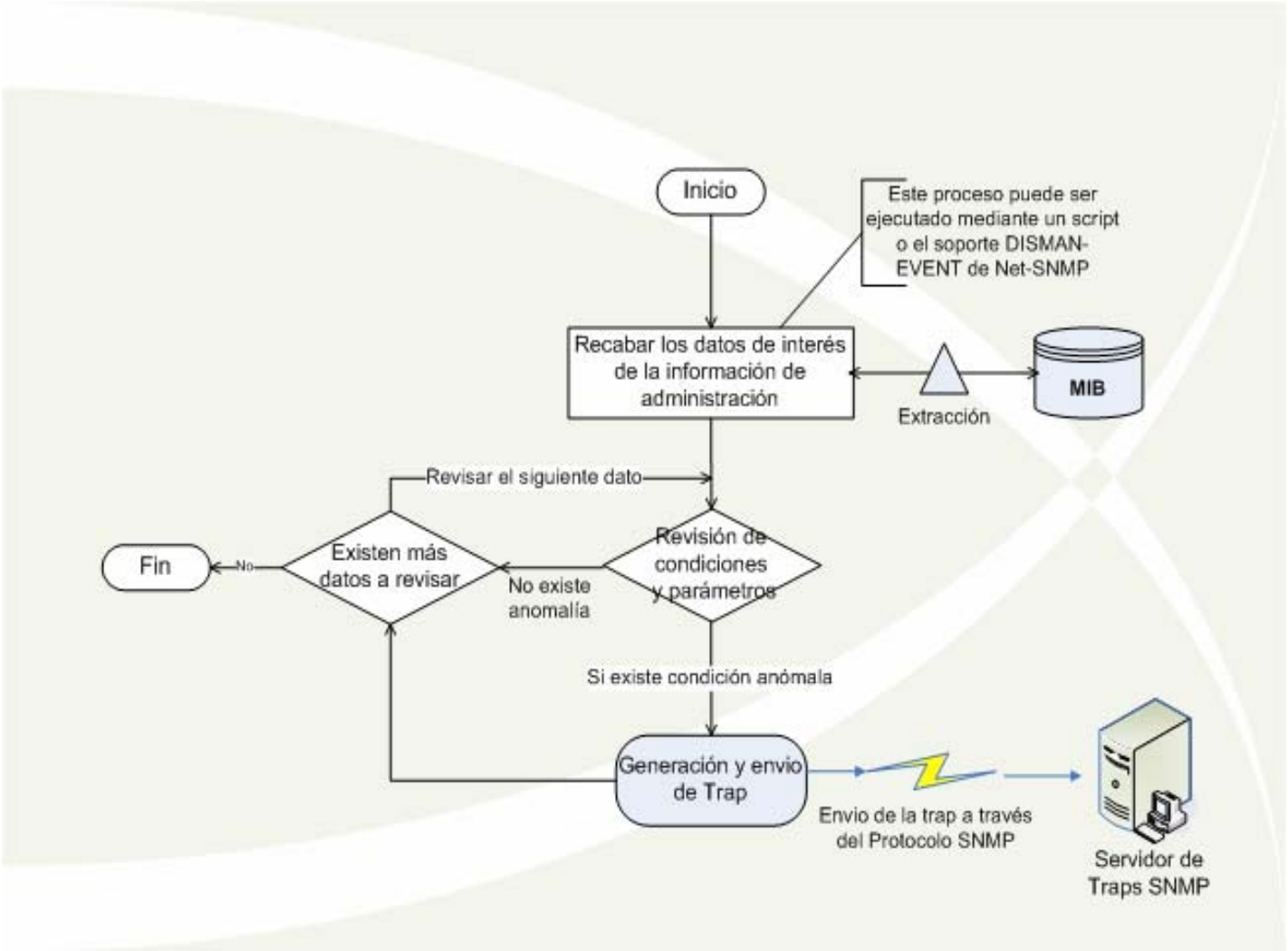


Figura 3.4 Diagrama del proceso de auto - monitoreo

3.3.1.2 Servidor de traps

Para que se realice el proceso de recepción e interpretación de las traps, es necesario implementar el servidor de alertas (*trap receiver*) en algún equipo que se destine para esta tarea. Comúnmente la opción a elegir es el servidor NMS, por lo que la mejor alternativa es implementarlo en el mismo servidor donde se llevará a cabo la graficación de los recursos.

Para poner en marcha el servidor de traps del Net-SNMP, basta con ejecutar un programa llamado "*SNMPTRAPD*", dicho programa funciona en base a una serie de argumentos proporcionados desde la misma línea de comandos, de los cuales los más importantes son; indicar el archivo de configuración, ruta donde registrará todas la alertas entrantes y si las traps serán leídas en su forma simbólica o numérica.

El archivo de configuración será el que determine el comportamiento del servidor de traps, en el sentido de elegir que traps serán permitidas y como serán interpretadas. Típicamente el archivo de configuración es llamado "*snmptrapd.conf*" y su sintaxis es bastante sencilla, solo son definidos los identificadores de objeto (OID) de las traps que serán permitidas. Por cada OID definida ésta será asociada a la ejecución de un programa que se ocupe de llevar a cabo cualquier tarea de administración en relación a esa trap.

El proceso que lleva a cabo el servidor de traps para procesar una alerta se puede describir en las siguientes etapas (figura 3.5):

- El demonio *snmptrapd* se recibe la trap entrante
- Se registra en la bitácora datos provenientes de la alerta
- Se carga la información de configuración, donde se encuentra las definiciones de las traps aceptadas y como serán interpretadas
- Se realiza el proceso de identificación basado en el archivo de configuración para determinar si la trap será procesada
- Si la trap es aceptada se ejecuta el programa externo asociado, el cual puede enviar una alerta o correo.

- fin

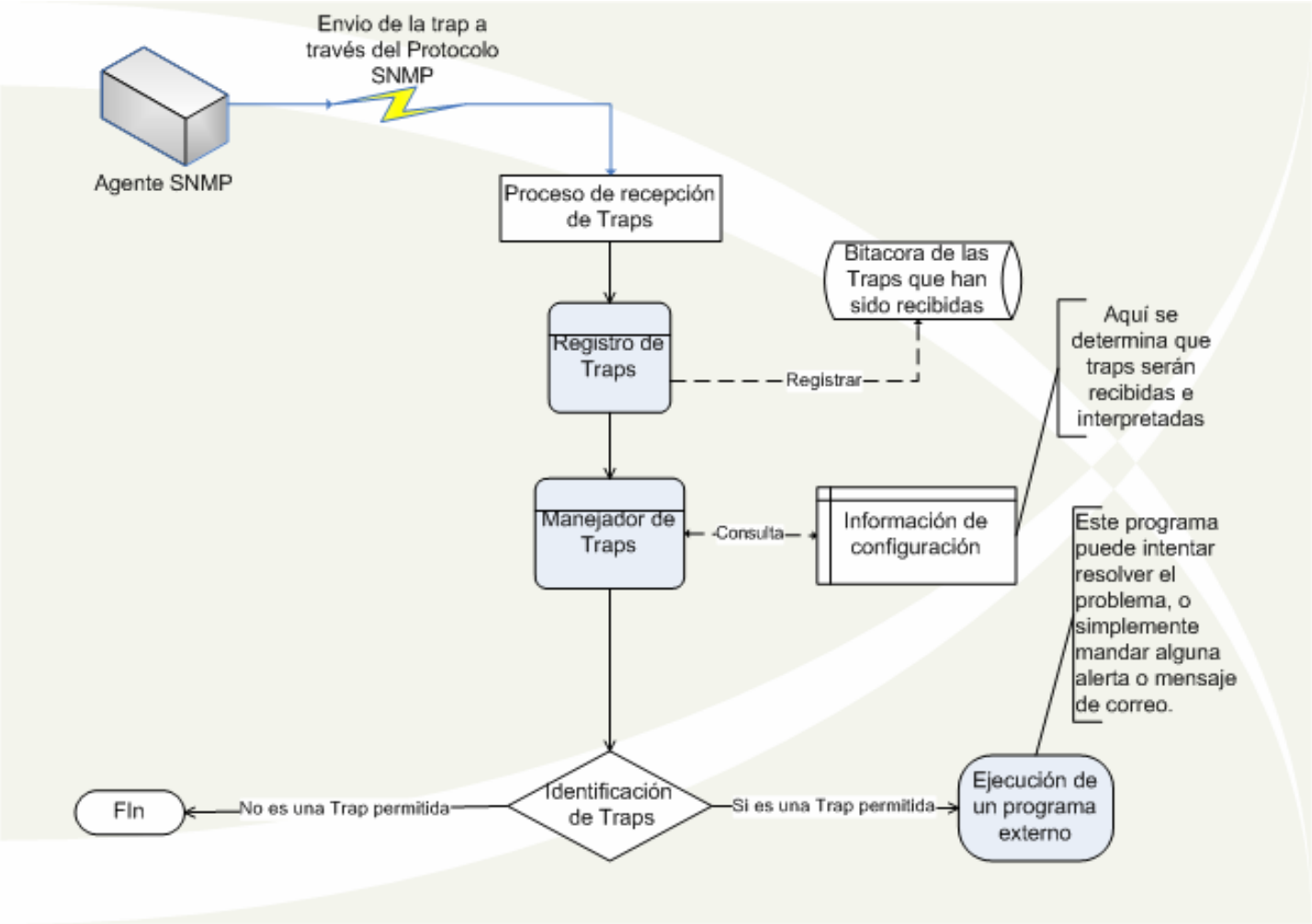


Figura 3.5 Funcionamiento de un servidor de alertas (trap receiver)

3.3.2 Requerimiento adicional en la implementación

Los procesos que se han descrito hasta el momento, parecen ser suficientes para contar con un sistema de alertas lo bastante flexible que permita diseñar un monitoreo de los recursos o servicios que se consideren más importantes en los servidores. Como se ha visto se cuenta con un agente SNMP capaz de revisar periódicamente su propia información de administración y generar una alerta hacia el servidor de traps en caso de encontrar un problema en su funcionamiento. Por el otro

lado las alertas son recibidas e interpretadas por el servidor de traps que tiene la capacidad de identificar el tipo de traps, determinar si estas están definidas y ejecutar los procesos particulares dependiendo del tipo de trap recibida.

Estos elementos nos proporcionaban las herramientas suficientes para diseñar un esquema de alertas y acciones a la medida de nuestras necesidades. El tipo de acción a tomar, según sea el tipo de alerta que se reciba, esta sujeto a la habilidad que se tenga para desarrollar un programa que pueda interpretar la información de la trap y actúe en consecuencia a esto.

Aparentemente teníamos los recursos necesarios para adecuarnos a cualquier necesidad, sin embargo surgió un requerimiento adicional a la implementación debido a los objetivos del presente trabajo de tesis. Desde un principio se planteo que el desarrollo de la presente solución, tanto en la parte de graficación así como en el esquema de alertas, era un complemento con el objetivo de reforzar el actual sistema de monitoreo del departamento. Debido a esta situación es necesario que lo que se haya implementado se adecue al sistema que esta funcionando actualmente. Puesto que el departamento no cuenta con ninguna herramienta para la graficación de los recursos de los servidores con las características de la solución que se pretende implementar, es obvio que esta parte no se tendrá que adecuar a nada existente. En cambio el esquema de alertas es el que tuvo que adecuarse al sistema de monitoreo de ese momento, específicamente en la parte de notificación.

Cabe recordar que el programa de monitoreo MON se encarga de enviar una notificación de alerta en caso de que alguno de los monitores detecte la falla de algún servicio. Esta notificación es enviada tanto a una lista de correo así como a un servicio de mensajes Skytel. La lista llamada staff@servidores.unam.mx incluye la dirección de correo electrónico de todos los miembros del departamento de servidores, mientras que son 2 los dispositivos portátiles con los que cuenta el área. De esta manera el departamento rápidamente se puede percatar de cualquier problema que suscite con alguno de los elementos que están siendo monitoreados. Ante este panorama fue necesario adecuar el esquema de alertas con el funcionamiento de notificaciones,

logrando que las traps que sean generadas por cualquier agente SNMP se notifiquen por medio del programa MON.

El requerimiento es que la información que sea generada por la trap sea notificada por medio de la herramienta MON y no directamente desde un proceso iniciado desde el servidor de traps. No existe ninguna barrera técnica que impida desarrollar un programa que se ocupe de realizar esta tarea desde un proceso iniciado desde el mismo servidor de traps, sin embargo el interés de que la información sea enviada desde el programa MON es por que esta herramienta genera una serie de bitácoras que registran la actividad y disponibilidad de los servicios del departamento, cuya información puede ser rápidamente consultada desde un portal web.

3.3.2.1 Problemática ante el requerimiento

El objetivo es lograr que la información proveniente de la trap sea entendida por el programa MON y este a su vez realice la notificación por medio de sus programas de alerta. Dicho de otra forma es necesario que SNMP y MON hablen en un mismo idioma que les permita entenderse y trabajar juntos. Para ello nos dimos a la tarea de realizar una investigación con respecto a una herramienta o alguna alternativa que se haya utilizado para solventar una situación parecida a la que nos enfrentabamos. La investigación dio a conocer que otras personas se habían enfrentado a esta misma situación y fue resuelto principalmente desarrollando un programa en Perl que se encarga de entregar la información de la trap de una manera que MON sea capaz de entenderla y actuar en consecuencia.

Debido a que MON es una solución de monitoreo totalmente desarrollada en Perl, el programa utiliza un modulo llamado "*Mon::Client*" que se encarga de establecer comunicación con el servidor MON. Una vez que se ha establecido una conexión el programa se encarga de entregarle la información en un formato que MON pueda entenderlo, para ello tiene que definir a que vigilante (*watchname*) y servicio (*service*) de MON tiene que mandar la información. Para que el proceso sea completado con

éxito es necesario que dicho vigilante y servicio sean definidos en el archivo de configuración de MON y sea reiniciado el proceso para que tome la nueva configuración.

Afortunadamente se encontró un programa en Perl que realiza la anterior tarea, la distribución es totalmente libre como una aportación de Eric Sorenson a la comunidad de Internet, el programa se llama "*snmptrap2mon*" y tiene una serie de argumentos que se encargan de definir a que vigilante y servicio de MON será enviada la información de la alerta.

De esta manera el proceso que describe la mecánica del servidor de traps no se modificaría y solo bastaría con ejecutar el programa "snmptrap2mon" desde otro proceso asociado a la traps definidas en el archivo de configuración del servidor de alertas.

De esta forma el flujo determina que por cada trap que sea recibida e identificada, se pasará el control a un programa asociado a esta trap para que efectúe las acciones en respuesta a la alerta, entre estas acciones se tendrá que ejecutar el programa "*snmptrap2mon*" para que sea notificado el servidor MON.

Lo anterior parecía ser la solución mas conveniente sin embargo existía un problema ante tal panorama, debido a que el programa "*snmptrap2mon*" tarda aproximadamente de 1 a 2 segundos para establecer una sesión con el servidor MON y pasarle los datos de la trap, existía la fuerte posibilidad de que al ser recibidas consecutivamente un numero mayor de 10 a 15 traps , las notificaciones fueran recibidas hasta de 1 a 3 minutos después. Esto implicaba que si en nuestro esquema de alertas fueran generadas un número de alertas más o menos grande de forma consecutiva, el proceso descrito tardaría más tiempo en notificar a MON en relación al número de alertas entrantes.

Si bien es cierto que no se esperaba que los servidores generaran un gran número de alertas, mucho menos que estas se presentaran consecutivamente, fue

necesario buscar una alternativa que permitiera manejar y enviar un gran número de alertas por segundo. Esto nos permitió tener un sistema que pudo enfrentarse a un escenario más extremo de lo que se había contemplado.

3.3.2.2 Herramienta SNMPTT

Fue necesario realizar nuevamente una investigación acerca de alguna herramienta que nos permitiera acelerar el proceso de comunicación con el servidor MON. Es así como nos encontramos con una alternativa llamada SNMPTT[31]. Esta herramienta es un manejador de traps desarrollado totalmente en Perl y funciona conjuntamente con el servidor de traps "*snmptrapd*" de la distribución Net-SNMP.

Básicamente la función de este manejador de alertas es traducir y dar formato a la información proveniente de las traps, de tal manera que se puedan crear bitácoras y mensajes más amigables al usuario. Esto es gracias a que podemos obtener fácilmente los datos por medio de una serie de variables de formato que la misma aplicación genera en base a la información .recibida, evitando de esta manera el tener que desarrollar un script que se ocupe de esta tarea.

SNMPTT en general nos evita trabajo de programación ya que de forma fácil no solo podemos dar formato rápido a la información si no que se pueden realizar configuraciones más complejas como las siguientes:

- Tiene la habilidad de aceptar o rechazar la trap basándose en el nombre del host, dirección IP, rango de red, o variables dentro de la información de la trap.
- Puede ejecutar programas externos asociados a una tipo de trap con el objetivo de enviar un tipo de alerta, correo o intentar resolver el problema.

[31] SNMPTT: Traductor de Traps SNMP

- Puede hacer búsquedas empleando expresiones regulares de tal forma que puede reemplazar en el mensaje cadenas de texto por otras.

SNMPTT levanta un demonio (un programa en Perl) el cual duerme durante 5 segundos y vuelve a revisar si existen traps en un directorio de encolamiento (*spool directory*). En caso de existir traps encoladas el programa ejecuta rápidamente el proceso que le permitirá determinar si la trap es procesada o no, así como ejecutar los programas externos asociados. Este funcionamiento permite que SNMPTT pueda manejar fácilmente un gran número de traps por minuto.

El funcionamiento de SNMPTT depende de las traps recibidas por "*snmptrapd*" ya que él no cuenta con la capacidad de recibir directamente las alertas del protocolo SNMP. Tal vez la característica más importante para nuestro caso es que el programa al estar escrito en Perl acelera considerablemente la ejecución del programa "*snmptrap2mon*", de manera que la comunicación con MON es mucho más rápida que ejecutándolo desde un shell script. El proceso de SNMPTT pasa por las siguientes etapas (figura 3.6):

1. Un programa llamado *snmptthandler* se encarga de leer la trap que le ha enviado en servidor de traps y escribir la información en un directorio de encolamiento
2. Se carga la información de configuración que contiene las definiciones de la traps a procesar
3. Comienzan a leerse las traps que se han escrito en el directorio de encolamiento (*spool directory*)
4. Se realiza el proceso de identificación basado en el archivo de configuración para determinar si la trap será procesada
5. Si la trap es aceptada se registra la trap en bitácora y se ejecuta el programa externo asociado, el cual puede enviar una alerta o correo. Si no es identificada se registra en otra bitácora de traps desconocidas.
6. El proceso duerme durante 5 segundos de forma predeterminada
7. El proceso inicia nuevamente para indagar si existen nuevas traps en el directorio de encolamiento

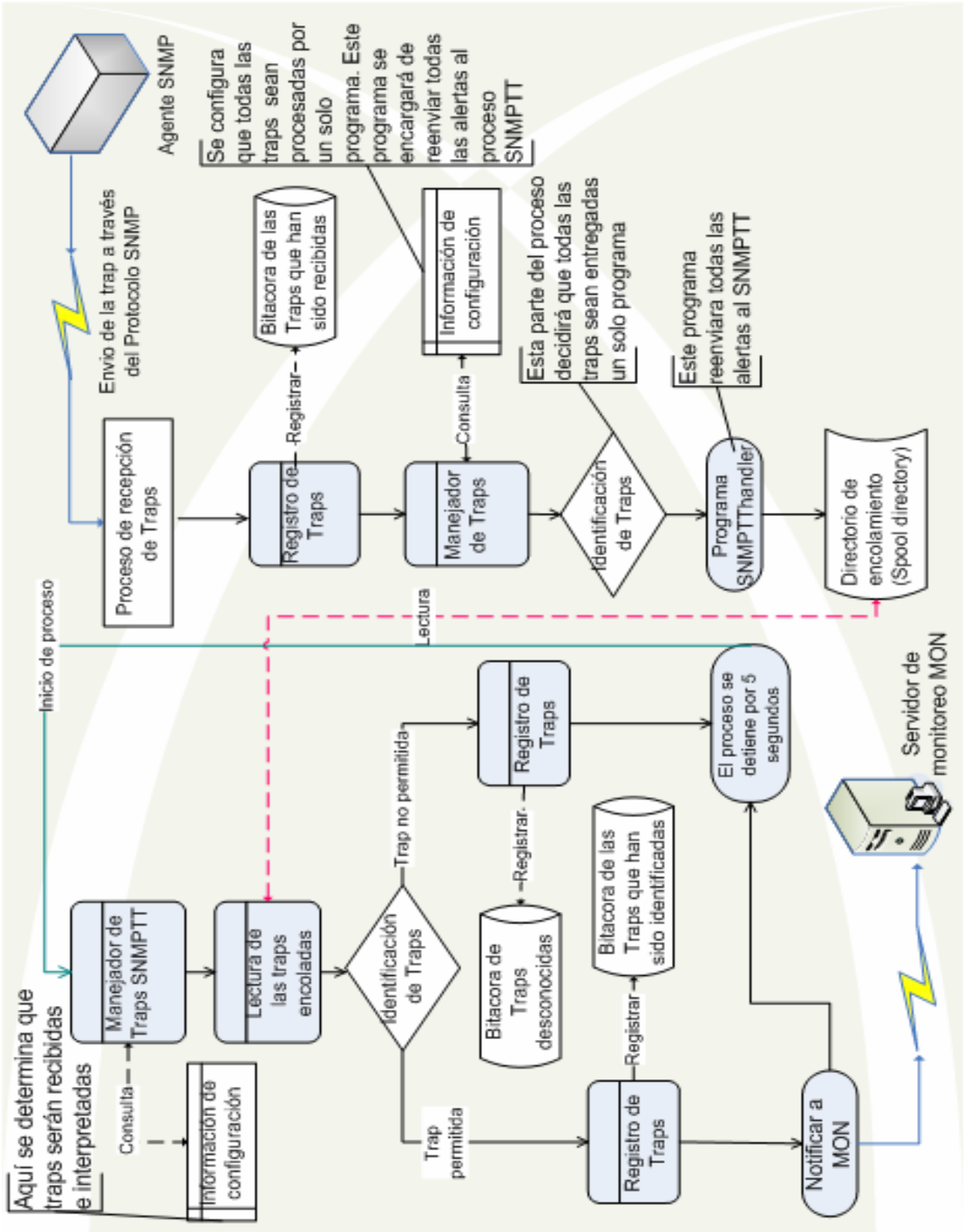


Figura 3.6 Interacción entre trap receiver y snmptt

3.4 Conclusiones

Todo lo expuesto en el presente capitulo permitió comprobar que existen varias herramientas de software libre tanto para la generación de gráficas así como para implementar esquemas de alerta. De esta forma se determinaron las características de cada una de las posibilidades eligiendo aquella que cumpliera con las necesidades planteadas en el presente trabajo. Se intentó integrar una solución que en su conjunto permitirá complementar el actual sistema de monitoreo, integrando reportes gráficos de cada uno de los servidores así como la incorporación de nuevas alertas para los servicios existentes, teniendo como base fundamental de su funcionamiento el uso del protocolo SNMP, que mediante la interacción con otras herramientas permitieron alcanzar los objetivos planteados a lo largo de este capitulo. La herramienta elegida así como el esquema de alertas planteado parecieron ser la solución ideal que permitió solventar los problemas existentes en el esquema de monitoreo, sin embargo todo este análisis, que se ha planteado desde el punto de vista teórico, fue probado en la etapa de desarrollo y pruebas donde los resultados comprobarían si todo lo expuesto funciono correctamente en la práctica, situación que se verá en el siguiente capitulo de este proyecto de tesis.

Capitulo 4
Desarrollo y Pruebas

A través de este capítulo se describirá la metodología que se llevó a cabo para la instalación y configuración de todos los elementos que integran la solución. Principalmente fueron 3 las etapas en las que se dividió el proceso global de implementación, en las cuales se describieron los detalles técnicos referentes a la implementación del software así como la descripción de la fase de pruebas en cada una de las etapas. Las 3 etapas en la que fue dividida la implementación fueron las siguientes:

Implementación de los agentes SNMP. Relacionado con la instalación y configuración del protocolo SNMP en cada uno de los servidores que se deseen monitorear.

Implementación del servidor de graficación. Se describirá la instalación y configuración de todos los elementos implicados en la solución de graficación, así como la explicación de algunos conceptos, en relación al manejo de la herramienta, para la creación de graficas.

Implementación del servidor de alertas. Todos los elementos que integran el manejo de alertas, serán expuestos a través de este apartado.

Para dejar más claro toda la serie de etapas que se han comentado en este apartado, las cuales abarcan el desarrollo, pruebas y validación, así como la implementación final de este proyecto de tesis, se expone la siguiente tabla.

	Jun – 06				Jul – 06				Ago – 06				Sep - 06			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Configuración de Agentes SNMP																
Pruebas y Validación de Agentes SNMP																
Configuración de Servidor de Graficación																
Pruebas y Validación de Graficas																
Configuración de Servidor de Alertas																
Pruebas y Validación de Alertas																
Implementación Final																

Tabla 4.0 Cuadro cronológico de las etapas de desarrollo, pruebas, validación e implementación

Al final de cada una de las etapas se tendrán una serie de resultados para determinar si el planteamiento teórico vertido en las etapas de análisis y diseño en los capítulos 2 y 3 tuvo un funcionamiento eficaz en la práctica. Se espera que a través de la información brindada en cada una de las etapas, puedan quedar más claros los conceptos vertidos en los anteriores capítulos.

4.1 Implementación de los agentes SNMP

Como se ha mencionado a lo largo de este trabajo, el servidor de administración será el encargado de la graficación así como la recepción y manejo de alertas SNMP. Para llevar a cabo estas dos tareas fue necesario establecer comunicación con cada uno de los agentes SNMP implementados en los servidores, comunicación que permitirá recuperar los datos y recibir las traps de los hosts administrados. Es por ello

que la primera etapa del camino a la implementación de la solución, fue instalar el protocolo SNMP en cada uno de los servidores que se contemplan monitorear. A través de este apartado se verá cada unos de los aspectos concernientes a la instalación de SNMP, tomando en cuenta los requerimientos necesarios, versiones de sistema operativo, configuración de los recursos a administrar, monitoreo autónomo del agente SNMP y la serie de pruebas realizadas antes de la implementación final.

4.1.1 Requerimientos

4.1.1.1 Procesamiento y memoria

La mayoría del software requiere de características mínimas de hardware para su óptimo funcionamiento, sin embargo en el caso de Net-SNMP los requerimientos necesarios para ser instalado son realmente menores. La propia distribución no documenta requerimientos mínimos en los aspectos de procesamiento y capacidad de memoria RAM puesto que la aplicación no requiere de grandes recursos para llevar a cabo su funcionamiento. Solo es necesaria una cantidad mínima de almacenamiento de 15 Megabytes libres en donde se pretenda instalar la aplicación. Debido a esta situación todos los servidores que se contemplan para el monitoreo y graficación son viables para llevar a cabo la implementación del protocolo SNMP.

4.1.1.2 Sistemas operativos compatibles

El Net-SNMP puede ser instalado sobre diversos sistemas operativos entre los cuales se tienen:

- Solaris
- AIX (4.1.5, 3.2.5)
- Ultrix (4.5 al 4.2)
- SunOS (4.1.4 al 4.1.2)

- NetBSD (1.5alpha al 1.0)
- FreeBSD (4.1 al 2.2)
- OpenBSD (2.8, 2.6)
- Irix (6.5 to 5.1)
- HP-UX (10.20 9.01 y 11.0)
- Linux
- Windows

Nuevamente no se tiene ningún inconveniente para que alguno de los servidores pueda ser monitoreado mediante SNMP puesto que los servidores del Área de Administración de Servidores están instalados bajo sistemas operativos Red Hat Enterprise 3, Red Hat Enterprise 4, Mandrake, todos sabores de Linux, y Solaris.

4.1.1.3 Software requerido

Para llevar a cabo la implementación del Net-SNMP se requiere que el sistema operativo tenga instalado el siguiente software:

- Compilador gcc. Es requerido para llevar a cabo la compilación del fuente de instalación.
- OpenSSL. Necesario para que la versión SNMPv3 del protocolo pueda operar con las características de autenticación y encriptación de datos.
- zlib. Librería utilizada para la compresión de datos, incluyendo funciones de revisión de integridad.

4.1.2 Instalación

En este apartado se describió el proceso de instalación del Net-SNMP sobre un servidor con sistema operativo Linux y otro con sistema operativo Solaris. Puesto que

estos son los principales sistemas operativos de los servidores del área es necesario detallar la forma de instalar la aplicación en cada uno de estos sistemas.

Para llevar a cabo la implementación del protocolo SNMP se decidió usar la versión Net-SNMP.5.1.4. Esta es una de las versiones más recientes del protocolo e incorpora las versiones SNMPv1, SNMPv2 y SNMPv3 así como el soporte DISMAN-EVENT-MIB para monitoreo autónomo. La distribución fuente puede ser compilada para instalarse en cualquier sistema derivado de Unix como Solaris, HP-UX, AIX así como para los sabores de Linux. Antes de comenzar con la parte de instalación fue necesario bajar el código fuente de la pagina http://www.net-snmp.org/download/ o en el sitio ftp://ftp.net-snmp.org/pub/sourceforge/net-snmp/.

Para llevar a cabo la instalación de Net-SNMP fue necesario que las siguientes instrucciones se ejecutaran como usuario administrador (root) del sistema.

4.1.2.1 Instalación sobre Linux

A continuación se describen una serie de pasos para llevar a cabo la instalación del Net-SNMP desde el código fuente de la distribución.

Deshabilitar SNMP instalados

Existe la posibilidad que durante la instalación del sistema operativo se haya instalado alguna versión de SNMP, situación que es muy normal en sistemas operativos como Red Hat y Mandrake. Es necesario que en todos los servidores este implementado la misma versión de SNMP, por lo que hay que deshabilitar cualquier agente SNMP previamente instalado.

Buscar si existe algún proceso de SNMP en el sistema.

```
[root@gypsy root]# ps -fea | grep snmp
```

SI se haya algún id de proceso como “snmpd“ hay que dar de baja ese proceso.

```
[root@gypsy root]# kill id_proceso
```

En el caso de sistemas operativos Linux, SNMP es instalado mediante RPM, por lo que se deberá buscar para desinstalarlo.

```
[root@gypsy root]# rpm -qa | grep snmp
```

En caso de encontrar algún RPM de SNMP se deberá desinstalar con el siguiente comando.

```
[root@gypsy root]# rpm -e <nombre_del_RPM>
```

Desempaquetar el paquete de la distribución

Una vez obtenido el código fuente del Net-SNMP es necesario descomprimirlo y desempaquetarlo. Esta tarea puede realizarse sobre cualquier ruta del sistema, pero para efectos de esta instalación se creará una carpeta llamada “SRC“ en el directorio /root del sistema.

```
[root@gypsy root]# mkdir SRC
[root@gypsy root]# mv net-snmp-5.1.4.tar.gz ./SRC
[root@gypsy root]# cd SRC
```

```
[root@gypsy SRC]# tar -zxvf  net-snmp-5.1.4.tar.gz
```

Configuración

Una vez que se ha descomprimido y desempaquetado la distribución hay que configurar las características con las que se desea compilar para su posterior instalación.

Primero se debe ubicar en el directorio donde se encuentran los programas de configuración e instalación.

```
[root@gypsy SRC]#cd  net-snmp-5.1.4
```

En el caso de nuestra implementación la configuración que se eligió fue la siguiente:

- No se eligió ninguna ruta en particular para la instalación, por lo que el directorio de instalación por defecto fue en "/usr/local/share/snmp/". Los programas ejecutables asi como las librerías fueron instaladas por defecto en las rutas "/usr/local/bin" y "/usr/local/lib" respectivamente.
- Se indicó compilar con el modulo "disman/event-mib", que permite habilitar el soporte para monitoreo y notificación autónoma en caso de cualquier situación errónea en los recursos administrados.

Para indicar la anterior configuración se deberá ejecutar el programa con los siguientes argumentos.

```
[root@gypsy net-snmp-5.1.4]# ./configure --with-mib-modules=disman/event-mib
```

Una vez que se ha iniciado el proceso de configuración se requiere que se introduzcan una serie de datos para que el agente sea configurado correctamente:

1.Default version of SNMP to use (3): 3. Hace referencia a la versión del protocolo a instalar por defecto, en nuestro caso se trabajará con la versión 3.

2.System Contact Information (root@): aaguilera@servidores.unam.mx . Es un dato que brinda la dirección de correo electrónico de la persona encargada de administrar ese agente SNMP.

3.System Location (Unknown): UNAM Depto. De Servidores. Información que da a conocer la localización del dispositivo.

4.Location to write logfile (/var/log/snmpd.log): /var/log/snmpd.log . Se refiere a la bitacora donde SNMP alojará mensajes de error e información acerca de su funcionamiento.

5.Location to write persistent information (/var/net-snmp): /var/net-snmp. Ruta del directorio para las librerias SNMP, donde se almacenará información acerca de la configuración del protocolo.

Al final del proceso se muestra una tabla resumiendo las características de configuración de SNMP.

```
---------------------------------------------------------
        Net-SNMP configuration summary:
---------------------------------------------------------

  Net-SNMP Version:          5.1.4
  Building for:              linux
  Network transport support:  Callback Unix TCP UDP
  SNMPv3 Security Modules:    usm
 Agent MIB code:             mibII ucd_snmp snmpv3mibs notification target agent_mibs agentx utilities host disman/event-mib
  SNMP Perl modules:         disabled
  Embedded perl support:     disabled
```

Authentication support: MD5 SHA1
Encryption support: DES AES

Iniciar el proceso de compilación

Para empezar a compilar el fuente teclear el siguiente comando:

```
[root@gypsy net-snmp-5.1.4]#  make
```

Se deberá observar si existen errores de compilación. Es posible que se reciban mensajes de advertencia, sin embargo esto es un comportamiento normal.

Realizar pruebas

Es posible realizar una serie de pruebas antes de realizar la instalación, de esta forma se puede observar el comportamiento de alguna prueba en especial. En la mayoría de los casos este paso puede ser omitido.

```
[root@gypsy net-snmp-5.1.4]# make test
```

Instalación

Para realizar la instalación se teclea el siguiente comando:

```
[root@gypsy net-snmp-5.1.4]# make install
```

Esto concluye el proceso de instalación, la configuración del agente se verá en el siguiente apartado.

4.1.2.2 Instalación sobre Solaris

A continuación se describen una serie de pasos para llevar a cabo la instalación del Net-SNMP desde el código fuente de la distribución.

Deshabilitar SNMP instalados

Es común que en el sistema operativo Solaris se encuentre instalado SNMP, puesto que este implementa su propia versión del protocolo para características muy específicas de su propio hardware. Es necesario que en todos los servidores este implementado la misma versión de SNMP, por lo que hay que deshabilitar cualquier agente SNMP previamente instalado.

Buscar si existe algún proceso de SNMP en el sistema.

```
[root@fenix root]# ps -fea | grep snmp
root 643 1 0 Jan 16 ? 5:49 /usr/local/sbin/snmpd
```

Si se observa algún proceso como el anterior, se deberá dar de baja el proceso especificando el id del proceso

```
[root@fenix root]# kill 643
```

Buscar otro proceso relacionado con el snmp nativo de los sistemas operativos solaris

```
[root@fenix root]# ps -fea | grep mibi
root 21371 1 0 Feb 07 ? 0:52 mibiisa -r -p 41178
```

Si se observa algún proceso como el anterior, se deberá dar de baja el proceso especificando el id del proceso

```
[root@fenix root]# kill 21371
```

En el caso de sistemas operativos Solaris podemos deshabilitar el SNMP instalado de 2 formas, una es desinstalando el Package, mientras que la otra es renombrando los scripts de inicio.

Desinstalando el package

```
[root@fenix root]# pkginfo -l | grep snmp
```

En caso de encontrar algún Package de SNMP se deberá desinstalar con el siguiente comando.

```
[root@fenix root]# pkgrm <nombre_del_Package>
```

Renombrando scripts de inicio

Esto dejara instalado el SNMP del sistema, sin embargo impedirá que se ejecute cada vez que la maquina sea reiniciada.

```
[root@fenix root]# cd /etc/rc3.d
[root@fenix root]#mv S76snmpdx s76snmpdx
[root@fenix root]#mv S77dmi s77dmi
```

Desempaquetar el paquete de la distribución

Una vez obtenido el código fuente del Net-SNMP es necesario descomprimirlo y desempaquetarlo. Esta tarea puede realizarse sobre cualquier ruta del sistema, pero para efectos de esta instalación se creará una carpeta llamada "SRC" en el directorio /root del sistema.

```
[root@fenix root]# mkdir SRC
[root@fenix root]# mv net-snmp-5.1.4.tar.gz ./SRC
[root@fenix root]# cd  SRC
[root@fenix SRC]#gzip  net-snmp-5.1.4.tar.gz
[root@fenix SRC]# tar -xvf  net-snmp-5.1.4.tar
```

Configuración

Una vez que se ha descomprimido y desempaquetado la distribución hay que configurar las características con las que se desea compilar el fuente para su posterior instalación.

Primero se debe ubicar en el directorio donde se hayan los programas de configuración e instalación.

```
[root@fenix SRC]#cd  net-snmp-5.1.4
```

Una vez ubicados en el directorio, es necesario colocar en la variable PATH la ruta del gcc para llevar a ca/mnt/floppy/bo la configuración.

```
[root@fenix net-snmp-5.1.4]# PATH=/usr/sbin:/usr/local/bin:/usr/ccs/bin:/usr/bin
[root@fenix net-snmp-5.1.4]# export PATH
```

En el caso de nuestra implementación la configuración que se eligió fue la siguiente:

- No se eligió ninguna ruta en particular para la instalación, por lo que el directorio de instalación por defecto fue en “/usr/local/share/snmp/“. Los programas ejecutables asi como las librerías fueron instaladas por defecto en las rutas “/usr/local/bin“ y “/usr/local/lib“ respectivamente.
- Se indicó compilar con el módulo “host“, el cual permite recabar información acerca de los recursos del equipo.
- Se indicó compilar con el modulo “disman/event-mib“, que permite habilitar el soporte para monitoreo y notificación autónoma en caso de cualquier situación errónea en los recursos administrados.

Para indicar la anterior configuración se deberá ejecutar el programa con los siguientes argumentos.

```
[root@fenix net-snmp-5.1.4]# ./configure --with-mib-modules="host disman/event-mib"
```

Una vez que se ha iniciado el proceso de configuración se requiere que se introduzcan una serie de datos para que el agente sea configurado correctamente:

1.Default version of SNMP to use (3): 3. Hace referencia a la versión del protocolo a instalar por defecto, en nuestro se trabajará con la versión 3.

2.System Contact Information (root@): aaguilera@servidores.unam.mx . Es un dato que brinda la dirección de correo electrónico de la persona encargada de administrar ese agente SNMP.

3.System Location (Unknown): UNAM Depto. De Servidores. Información que da a conocer la localización del dispositivo.

4.Location to write logfile (/var/log/snmpd.log): /var/log/snmpd.log . Se refiere a la bitácora donde SNMP alojará mensajes de error e información acerca de su funcionamiento.

5.Location to write persistent information (/var/net-snmp): /var/net-snmp . Ruta del directorio para las librerías SNMP, donde se almacenará información acerca de la configuración del protocolo.

Al final del proceso se muestra una tabla resumiendo las características de configuración de SNMP.

```
---------------------------------------------------------
      Net-SNMP configuration summary:
---------------------------------------------------------

  Net-SNMP Version:          5.1.4
  Building for:              solaris9
  Network transport support:  Callback Unix TCP UDP
  SNMPv3 Security Modules:    usm
 Agent MIB code:             mibII ucd_snmp snmpv3mibs notification target agent_mibs agentx utilities host disman/event-mib
  SNMP Perl modules:         disabled
  Embedded perl support:     disabled
```

Authentication support: MD5 SHA1
Encryption support: DES AES

Iniciar el proceso de compilación

Para empezar a compilar el fuente teclear el siguiente comando:

```
[root@fenix net-snmp-5.1.4]#  make
```

Se deberá observar si existen errores de compilación. Es posible que se reciban mensajes de advertencia, sin embargo esto es un comportamiento normal.

Realizar pruebas

Es posible realizar una serie de pruebas antes de realizar la instalación, de esta forma se puede observar el comportamiento de alguna prueba en especial. Aunque este paso en la mayoría de los casos puede ser omitido.

```
[root@fenix net-snmp-5.1.4]# make test
```

Instalación

Para realizar la instalación se teclea el siguiente comando:

```
[root@fenix net-snmp-5.1.4]#  make install
```

Esto concluye el proceso de instalación, la configuración del agente se verá en el siguiente apartado.

4.1.3 Configuración del agente SNMP

Este apartado trata sobre la forma en que será configurado el agente SNMP para llevar a cabo su operación junto con el gestor SNMP así como en su funcionamiento interno. La configuración del dispositivo esta dividida en los 3 siguientes aspectos:

- Método de comunicación
- Directivas de monitoreo
- Habilitación de monitoreo autónomo

A continuación se detallarán cada uno de los anteriores puntos.

4.1.3.1 Método de comunicación

Para establecer comunicación ya sea con un agente o un gestor SNMP es necesario configurar el control de acceso a dicho dispositivo. Existe un estándar llamado Modelo de Control de Acceso Basado en Vistas (VACM) definido en el RFC 2575. VACM permite administrar de manera muy flexible y poderosa, mediante grupos, vistas y nombres de seguridad, la forma en que se podrá tener acceso al agente, determinando los siguientes aspectos:

- Quien podrá consultarlo
- Determinar desde que Red podrá ser consultado
- Acceso total o parcial a la base de administración (MIB)
- Los permisos de lectura o de escritura sobre la información de administración.

Para llevar a cabo las anteriores tareas se aplican una serie de directivas que deben estar contenidas en el archivo de configuración del SNMP. Dicho archivo es llamado "*snmpd.conf*" y en el caso de nuestra implementación se encuentra en la ruta "*/usr/local/share/snmp/snmpd.conf*" de todos los agentes SNMP que serán configurados. A continuación se ofrece una descripción de las directivas relacionadas con este método de acceso.

4.1.3.1.1 Directivas de Control de Acceso

rouser USER [noauth|auth|priv [OID]]
rwuser USER [noauth|auth|priv [OID]]

Esto permite establecer un usuario SNMPv3 USM para ser configurado en las tablas de acceso de VACM. Esta directiva envuelve la funcionalidad de otras directivas como group, acces y view, sin embargo la administración es mucho más flexible y poderosa usando estas últimas que las directivas rouser y rwuser.

Sus opciones pueden establecer que el usuario se autentique o no (auth | noauth) asi como si debe establecer una comunicación privada (priv). El argumento OID restringe el acceso a la base de datos de administración (MIB), estableciendo que solo se puede consultar lo que se encuentre por debajo del OID especificado.

La diferencia entre el rouser y rwuser es que el primero solo tiene permisos de lectura (read – only user) y el segundo de lectura y escritura (read and write user).

rocommunity COMMUNITY [SOURCE [OID]]
rwcommunity COMMUNITY [SOURCE [OID]]

Estas directivas crean comunidades de solo lectura y lectura - escritura sobre la MIB respectivamente. Nuevamente estas directivas envuelven la funcionalidad de directivas como com2sec, group, access, y view, que aunque es mucho más flexible y poderosa la administración con estas últimas, el uso de rocommunity y rwcommunity es mucho más simple.

El argumento SOURCE establece que ip tendrá acceso a consultar el agente , este puede ser el nombre o ip de un equipo, o una sub red espeficando IP/MASK.

El argumento OID establece lo mismo que fue descrito en las directivas rouser y rwuser.

com2sec NAME SOURCE COMMUNITY

Esta directiva asocia un nombre de comunidad (COMMUNITY) y su fuente (SOURCE) a un concepto llamado nombre seguro (NAME). El nombre seguro sera usado en otras directivas como group o access para establecer el modelo y nivel de seguridad de comunicación asi como los permisos de lectura y escritura obre la MIB.

El argumento SOURCE tiene la misma funcionalidad que el descrito en las directivas rocommunity y rwcommunity**.**

group NAME MODEL SECURITY

Esta directiva permite administrar a una serie de comunidades por medio de su nombre seguro, de tal forma que asocia un modelo se seguridad y nombre seguro a un

grupo de administración. De esta forma podemos tener grupo con ciertos permisos de acceso que sean aplicados a todas las comunidades de su pertenencia.

access NAME CONTEXT MODEL LEVEL PREFX READ WRITE NOTIFY

Esta directiva implica el acceso que será brindado a un grupo y modelo de seguridad a una vista. Modelo (MODEL) puede ser any (cualquiera), v1 (versión 1 del protocolo), v2c (versión 2 del protocolo) o usm (modelo se seguridad de usuario). El argumento nivel (LEVEL) puede ser el de noauth (no autenticado), auth (autenticado) o priv (privado). Los argumentos READ, WRITE y NOTIFY especifican el nombre de la vista que será usada para estos accesos. Cabe aclarar que para el acceso de v1 o v2c el nivel (LEVEL) debe ser no autenticado (noauth) y el contexto (CONTEXT) deberá estar vacio.

view NAME TYPE SUBTREE [MASK]

Esta directiva define el nombre de una vista. Una vista no es mas que la definición de una parte de la MIB. Cabe recordar que la MIB es una base de datos jerárquica por lo que especificando una SUBTREE (una rama de la base de datos) podemos asociar una parte de la MIB a una vista. Por medio de la vista se puede brindar o denegar el acceso, esto es posible con el argumento TYPE especificando included (inclusión) o excluded (exclusión).

El argumento MASK (máscara) permite controlar el acceso sobre la sub rama especificada. Debido a que la base de datos es jerarquica, es posible que debajo de esa rama se encuentren más sub ramas con información que posiblemente no quiera que sea accesada, por lo que la mascara permite controlar el acceso a la sub rama de la vista.

4.1.3.2 Directivas de Monitoreo

Es posible mediante SNMP establecer el monitoreo de cualquier recurso o servicio que se desee mediante un script de propia invención. Sin embargo esta versión de SNMP posibilita el monitoreo de una serie de recursos simplemente poniendo una serie de fáciles directivas en el archivo de configuración “snmpd.conf“.

Existe una MIB denominada UCD-SNMP-MIB, la cual es incorporada por defecto en la instalación de SNMP. En esta base son almacenados ciertos datos que nos ayudarán a monitorear aspectos del agente SNMP, entre los cuales tenemos:

1. Procesos
2. Sistema de archivos
3. Carga de sistema
4. Tamaño de archivos
5. Memoria de intercambio Swap

Existen otros recursos que pueden ser monitoreados, como el estado de las interfaces de red, carga de procesos de CPU, usuario y sistema, etc. Sin embargo para los objetivos del presente trabajo solo consideraremos los anteriores tipos de monitoreo.

Procesos

Es posible monitorear cualquier número de procesos que estén ejecutándose en el sistema operativo mediante la siguiente directiva:

proc NAME [MAX=0] [MIN=0]

Por medio del parámetro NAME (nombre) se puede especificar el nombre del proceso. El nombre que sea proporcionado debe coincidir con el del proceso de la aplicación que se desee monitorear (por ejemplo para un servidor apache o mysql el proceso especificado deberá ser “httpd“ o “mysql“ respectivamente).

Los argumentos MAX y MIN establecen un rango del numero de procesos de un determinado tipo que se este ejecutando en el sistema operativo.

MAX indica el numero máximo permitido para ese tipo de proceso, si este no se especifica por defecto será 0, lo cual significa que no habrá un limite. Si solo se especifica el máximo, el mínimo (MIN) será por defecto 1. MIN indica un mínimo para ese proceso.

Si la directiva especifica un rango con los argumentos MAX y MIN, SNMP se encargará de levantar una bandera de error en caso de que el número de procesos caiga o sobrepase el rango especificado. Dicha bandera es llamada “prErrorFlag“, la cual tendrá un valor de 0 en caso de no existir ningún problema y 1 en caso contrario. La OID de la base de datos que se encargará de almacenar la información de todos los procesos que este siendo monitoreados mediante esta directiva es “.1.3.6.1.4.1.2021.2“ o bien puede ser especificada en su forma simbólica “.iso.org.dod.internet.private.enterprises.ucdavis.prTable“ .

Sistema de Archivos

Por medio de SNMP pueden ser monitoreados los sistemas de ficheros. Puede revisarse el tamaño total, el usado y disponible. Esto es configurado a través de la siguiente directiva:

disk PATH [MINSPACE | MINPERCENT%]

Esta sentencia permite revisar los discos montados en el sistema para determinar la cantidad de espacio disponible. Mediante el argumento PATH se indica que sistema de archivos desea ser monitoreado.

El argumento MINSPACE debe ser sustituido con la cantidad mínima de espacio libre que el *file system* debe brindar. Este dato debe ser proporcionado en Kilobytes.

El argumento MINPERCENT% es básicamente lo mismo que el MINISPACE, solo que aquí se especifica el porcentaje de espacio libre que el sistema de ficheros debe tener. El SNMP se encarga de determinar la cantidad en base a la capacidad total del sistema de ficheros.

En caso de configurarse una cantidad mínima de espacio disponible, ya sea en Kilobytes o en porcentaje, SNMP levantará una bandera de error colocando un valor de 1, llamada “diskErrorFlag“, en caso de que la cantidad de espacio libre en el sistema de ficheros este por debajo de lo especificado en la directiva. La OID que se encarga de almacenar la información con respecto a los sistemas de ficheros es “.1.3.6.1.4.1.2021.9“ o bien puede ser especificada en su forma simbólica “.iso.org.dod.internet.private.enterprises.ucdavis.dskTable“ .

Carga de sistema

SNMP puede determinar el promedio de carga del sistema en los intervalos de 1, 5 y 15 minutos. Esto es a través de la siguiente directiva:

load [1MAX=MAXLOADAV] [5MAX=MAXLOADAV] [15MAX=MAXLOADAV]

Los argumentos 1MAX, 5MAX y 15MAX, sirven para determinar un máximo de carga en los intervalos de 1, 5 y 15 minutos respectivamente. Si en cualquiera de estos intervalos el promedio de carga es mayor al especificado, SNMP levantará una bandera de error llamada “loadaveErrorFlag“, poniendo su valor a 1. La OID que se encarga de almacenar la información con respecto al promedio de carga es “.1.3.6.1.4.1.2021.10“ o bien puede ser especificada en su forma simbólica “.iso.org.dod.internet.private.enterprises.ucdavis.loadTable“ .

Tamaño de archivos

SNMP pude monitorear el tamaño de hasta 20 archivos del sistema operativo. Este comportamiento puede habilitarse con la siguiente directiva:

file FILE [MAXSIZE]

El argumento FILE deberá ser sustituido con la ruta del archivo que se determine monitorear.

El argumento MAXSIZE deberá sustituirse con la cantidad, especificada en Kilobytes, maxima de tamaño que el archivo pueda alcanzar. Si este valor no esta definido, se asume que el tamaño del archivo puede ser ilimitado.

En caso de que el tamaño de alguno de los archivos monitoreados rebase su respectivo limite, SNMP levantará una bandera llamada “fileErrorFlag“, poniendo su valor a 1. La OID utilizada para almacenar la información con respecto a los archivos es “.1.3.6.1.4.1.2021.15“ o bien puede ser especificada en su forma simbólica “.iso.org.dod.internet.private.enterprises.ucdavis.fileTable“ .

Memoria de intercambio Swap

La siguiente directiva permite monitorear el tamaño, cantidad utilizada y disponible de la memoria de intercambio del sistema operativo.

swap [MIN]

El argumento MIN debe ser sustituido por una cantidad, especificada en Kilobytes, que indicara el espacio mínimo disponible de memoria swap. En caso de que SNMP encuentre que el espacio disponible este por debajo del especificado, se levantará una bandera llamada “memSwapError“, poniendo su valor a 1.

La OID utilizada para almacenar la información con respecto a la memoria de intercambio es “.1.3.6.1.4.1.2021.4“ o bien puede ser especificada en su forma simbólica “.iso.org.dod.internet.private.enterprises.ucdavis.memory“ .

4.1.3.3 Habilitación de monitoreo autónomo

Una vez que se han visto el tipo de recursos que pueden ser monitoreados a través de las directivas de configuración, estamos ante la posibilidad de poder habilitar el monitoreo autónomo del agente SNMP. Esta directivas de monitoreo autónomo nuevamente deberán estar definidas en el archivo de configuración “snmpd.conf“ y se basarán en el valor de la banderas de error que fueron descritas en el anterior apartado. A continuación se ofrece una breve descripción de las directivas utilizadas para configurar hacia donde serán enviadas las alertas así como para habilitar el monitoreo interno en el agente SNMP.

4.1.3.3.1 Directivas de configuración para el envío de traps

trapcommunity STRING

El argumento STRING será substituido con el nombre de la comunidad que por defecto será la utilizada para enviar las traps. Esta directiva debe ser la primera en definirse puesto que determina la comunidad que será utilizada para el envio de alertas.

trapsink HOST [COMMUNITY [PORT]]
trap2sink HOST [COMMUNITY [PORT]]
informsink HOST [COMMUNITY [PORT]]

Todas estas directivas tienen como finalidad definir a que servidor (HOST) serán enviadas la traps, argumentando la comunidad (COMMUNITY) y el puerto (PORT) por el que serán enviadas. Si la comunidad ni el puerto son especificados, la comunidad será la misma que la definida en la directiva trapcommunity y el puerto por defecto será el 162.

La diferencia de cada directiva radica en el tipo de trap que será enviada, trapsink enviara una trap SNMPv1, trap2sink enviara una trap SNMPv2 mientras que informsink enviara un informe de notificación.

4.1.3.3.2 Directivas de monitoreo autónomo

agentSecName NAME

El soporte DISMAN-EVENT-MIB para monitoreo autónomo requiere de un usuario SNMPv3 valido para poder consultar su propia base de administración (MIB).

Esta directiva permite definir un usuario valido (NAME) a un nombre de agente seguro que tendrá la capacidad de consultar los datos de su propia MIB.

monitor [OPTIONS] NAME EXPRESSION

Esta directiva le dice al agente como monitorearse así mismo, encontrando algún tipo de problema en base a una expresión (EXPRESSION) booleana. El argumento EXPRESSION deberá ser sustituido por una simple expresión basada sobre una OID, un operador de comparación (!=, ==, <, <=, >, >=) y un valor entero. Es en esta parte en donde se debe comparar el valor de las banderas de error descritas en el anterior apartado de directivas de monitoreo. Se le podrá dar un nombre (NAME) a manera de rotulo solo para distinguir el monitor que se esta configurando.

Las opciones (OPTIONS) que pueden ser configuradas con las siguientes:

-t

Usado para monitorear un rango de valores en vez de un monitor booleano. Esto significa que el valor consultado en la MIB deberá entregar valores que pueden caer o estar por encima de los especificados en la expresión.

-r FRECUENCY

Monitorea la expresión dada con la frecuencia (FRECUENCY) en segundos especificada. El valor por defecto es de 600 segundos (10 minutos).

-u SECNAME

Indica que nombre de agente de seguridad (SECNAME) debe ser usado para consultar su propia MIB de datos. Por defecto toma al definido en la directiva agentSecName.

-o OID

Utilizado para especificar que otros valores de la MIB pueden ser entregados al generarse la trap en caso de una condición falsa. Esto es útil si se desea enviar una trap más descriptiva acerca del problema que se presento en el agente.

4.1.4 Pruebas

4.1.4.1 Planeación

Fue necesario definir un plan para llevar a cabo la instalación y configuración de los agentes SNMP sobre los servidores. Se decidió dividir la etapa de pruebas en 2 fases, a través de las cuales se realizarían la configuración y pruebas de comunicación con los agentes SNMP. En ambas fases se tomaron a 2 equipos para realizar las pruebas pertinentes, uno con sistema operativo Linux y otro con Solaris. Se realizaron diversas pruebas de comunicación al agente SNMP así como la configuración para monitorear ciertos recursos del equipo. La principal diferencia entre las fases de prueba es que se planteó que fuera hasta la segunda etapa que las pruebas fueran realizadas sobre 2 servidores productivos, permitiendo una primera etapa experimental en donde los equipos que intervinieran no tuvieran mucha importancia en la prestación de servicios del departamento.

4.1.4.2 Primera fase de prueba

4.1.4.2.1 Descripción de los equipos

Las características de los equipos se exponen en la tabla 4.1.

Nombre	*Sistema Operativo*	*Hardware*	*Servicios*	*Dirección IP*
LENTINIUS	Solaris 9	Sun ultra enterprise 450, con 256 Mb de RAM y 4 procesadores UltraSPARC II a una velocidad de 256 MHz y Disco duro de 7 Gb	Utilizada para exportar los gráficos de los servidores Solaris	132.248.120.107
GYPSY	Linux Enterprise Red Hat 4	Dispone de un procesador Intel Pentium 4 (3.73 GHz), memoria RAM de 512 Mb, disco duro de 100 Gb.	Brinda una conexión VPN hacia el site, permitiendo establecer comunicación a través de una red privada.	132.248.120.102

Tabla 4.1 Equipos de prueba en la fase1

4.1.4.2.2 Configuración de comunicación y monitoreo

Se dispuso que fueran realizadas pruebas tanto para la versión SNMPv2 y SNMPV3, por lo que fue necesario realizar configuraciones de comunicación a través de comunidades y usuarios SNMPv3. Cabe destacar que las directivas de configuración en cuanto a la comunicación así como al monitoreo no difieren en relación al sistema operativo, en este sentido lo único que varia es la forma de instalación, por ello la configuración que se describe a continuación fue la misma en ambos equipos de prueba.

4.1.4.2.2.1 Configuración de comunidades

Todas las directivas para la creación de comunidades tienen que ser agregadas en el archivo de configuración "snmpd.conf". Para las características de nuestra implementación la ruta de nuestro archivo de configuración es "/usr/local/share/snmp/snmpd.conf", y puede tomarse como plantilla el archivo "EXAMPLE.conf" proporcionado por la propia distribución del Net-SNMP.

- Crear un archivo de configuración de snmp, tomando como plantilla el de ejemplo proporcionado por Net-SNMP.

```
[root@gypsy root]#cd /root/SRC/net-snmp.5.1.4
[root@gypsy root]#cp EXAMPLE.conf  /usr/local/share/snmp/snmpd.conf
```

- Editar el archivo para agregar las siguientes líneas (en el archivo de ejemplo ya se encuentran configuradas unas comunidades, por lo que deben borrarse o comentarse dichas líneas).

```
#          sec.name  source          community
com2sec local          localhost          COMPRUEBAL
com2sec mynetwork  132.248.10.45  COMSERGRAF

####
# Second, map the security names into group names:

#              sec.model  sec.name
group MyRWGroup v1         local
group MyRWGroup v2c        local
group MyRWGroup usm        local
```

```
group MyROGroup v1         mynetwork
group MyROGroup v2c        mynetwork
group MyROGroup usm        mynetwork
####
# Third, create a view for us to let the groups have rights to:

#           incl/excl subtree                          mask
view all    included  .1                               80
####
# Finally, grant the 2 groups access to the 1 view with different
# write permissions:

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

En las anteriores líneas, se crearon dos comunidades llamadas COMPRUEBAL y COMSERGRAF respectivamente. La primera de ellas solo puede ser usada para establecer comunicación desde el mismo host, mientras que le segunda solo le permite la comunicación a la dirección 132.248.10.45 que es la IP del servidor de graficación. Además la primera comunidad tiene permisos de lectura y escritura mientras que la segunda solo tiene permiso de lectura. Si existen dudas sobre las directivas utilizadas remitirse a la sección Directivas de Control de Acceso del apartado de Configuración del Agente SNMP.

4.1.4.2.2.2 Configuración de usuarios SNMPv3

Antes de crear un usuario versión 3 es necesario que el demonio snmpd no se este ejecutando. Una vez que se ha detenido el demonio snmp se utilizará el siguiente script para generar un usuario v3:

net-snmp-config --create-snmpv3-user [-ro] [-a authpass] [-x privpass] [-X DES]
[-A MD5|SHA] username

De donde:

-ro Usuario de solo lectura. Si no se especifica se crea de lectura – escritura.
-a Contraseña de autenticación
-x Contraseña para cifrado. Si no se especifica por defecto toma la de autenticación
-X Protocolo de cifrado a utilizar. Por defecto toma el protocolo DES
-A Protocolo de autenticación a utilizar. Por defecto toma el protocolo MD5
username Nombre del usuario a crear

- En base a lo anterior se creó un usuario “pruebaL” de solo lectura con contraseña prueba2007.

```
[root@gypsy root]#net-snmp-config --create-snmpv3-user -ro -a “prueba2007“ pruebaL

adding the following line to /var/net-snmp/snmpd.conf:
   createUser pruebaL MD5 "“prueba2007“" DES
adding the following line to /usr/local/share/snmp/snmpd.conf:
   rouser pruebaL
```

La ejecución del comando informa que ha agregado directivas a dos archivos de configuración. Solo restará levantar el demonio snmp para que el agente SNMP pueda responder las peticiones de comunicación.

4.1.4.2.2.3 Configuración de monitoreo

Todas las directivas de monitoreo que se describen a continuación fueron agregadas al archivo "snmpd.conf" de configuración. Se planteó monitorear solo los siguientes recursos y servicios de ambos equipos:

- Los procesos httpd (Servidor web Apache), sendmail (Agente de trasporte de correo).

```
proc httpd 100 4
proc sendmail 3 1
```

- Los sistemas de archivo raíz "/" y "/home"

```
disk / 10%
disk /home 10%
```

- Promedio de carga de sistema

```
load 2 3 3
```

4.1.4.2.2.4 Arrancar el agente SNMP

Para levantar al agente basta con ejecutar el binario snmpd ubicado en la ruta "/usr/local/sbin/snmpd", sin embargo se desarrollo el siguiente shell script para levantar y dar de baja el demonio snmpd:

```
#!/bin/bash
#
#directorio donde se ubica el ejecutable que levanta el demonio
direxe=/usr/local/sbin

if [ $# -eq 0 ]; then

 echo "Sintaxis incorrecta"
 echo $0 " [start|stop]"
 exit 1
 else

   proceso=`ps -fea | grep "/usr/local/sbin/snmpd" | grep -v grep | awk '{print $2}'`

    case "$1" in

    'start')
          if [ "$proceso" = "" ];then

            $direxe/snmpd
            else
            echo "SNMP se encuentra en ejecución"
          fi
          ;;

     'stop')
          if [ "$proceso" != "" ];then

          kill $proceso
```

```
        else

        echo " No existe un proceso SNMP"

        fi
      ;;
        *)
         $0
     ;;
   esac
fi
##FIN DE SHELL####
```

El anterior código se guardará en un archivo llamado "snmp" en la ruta "/etc/init.d".

- En base a lo anterior se levantará el agente SNMP por medio del script

```
[root@gypsy root]# /etc/init.d/snmp start
```

4.1.4.2.3 Pruebas de comunicación a través de comunidades

4.1.4.2.3.1 Comunicación local

- Petición de datos a la MIB en relación a los procesos. Dicha petición fue realizada en gypsy y lentinius.

```
[root@gypsy root]# snmpwalk -v 2c -c COMPRUEBAL localhost .1.3.6.1.4.1.2021.2

UCD-SNMP-MIB::prIndex.1 = INTEGER: 1
```

```
UCD-SNMP-MIB::prIndex.2 = INTEGER: 2
UCD-SNMP-MIB::prNames.1 = STRING: httpd
UCD-SNMP-MIB::prNames.2 = STRING: sendmail
UCD-SNMP-MIB::prMin.1 = INTEGER: 4
UCD-SNMP-MIB::prMin.2 = INTEGER: 1
UCD-SNMP-MIB::prMax.1 = INTEGER: 100
UCD-SNMP-MIB::prMax.2 = INTEGER: 3
UCD-SNMP-MIB::prCount.1 = INTEGER: 6
UCD-SNMP-MIB::prCount.2 = INTEGER: 2
UCD-SNMP-MIB::prErrorFlag.1 = INTEGER: 0
UCD-SNMP-MIB::prErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::prErrMessage.1 = STRING:
UCD-SNMP-MIB::prErrMessage.2 = STRING:
UCD-SNMP-MIB::prErrFix.1 = INTEGER: 0
UCD-SNMP-MIB::prErrFix.2 = INTEGER: 0
UCD-SNMP-MIB::prErrFixCmd.1 = STRING:
UCD-SNMP-MIB::prErrFixCmd.2 = STRING:
```

```
[root@lentinius root]# snmpwalk -v 2c -c COMPRUEBAL localhost .1.3.6.1.4.1.2021.2
UCD-SNMP-MIB::prIndex.1 = INTEGER: 1
UCD-SNMP-MIB::prIndex.2 = INTEGER: 2
UCD-SNMP-MIB::prNames.1 = STRING: httpd
UCD-SNMP-MIB::prNames.2 = STRING: sendmail
UCD-SNMP-MIB::prMin.1 = INTEGER: 4
UCD-SNMP-MIB::prMin.2 = INTEGER: 1
UCD-SNMP-MIB::prMax.1 = INTEGER: 100
UCD-SNMP-MIB::prMax.2 = INTEGER: 3
UCD-SNMP-MIB::prCount.1 = INTEGER: 0
UCD-SNMP-MIB::prCount.2 = INTEGER: 2
UCD-SNMP-MIB::prErrorFlag.1 = INTEGER: 1
UCD-SNMP-MIB::prErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::prErrMessage.1 = STRING: Too few httpd running (# = 0)
UCD-SNMP-MIB::prErrMessage.2 = STRING:
UCD-SNMP-MIB::prErrFix.1 = INTEGER: 0
```

```
UCD-SNMP-MIB::prErrFix.2 = INTEGER: 0
UCD-SNMP-MIB::prErrFixCmd.1 = STRING:
UCD-SNMP-MIB::prErrFixCmd.2 = STRING:
```

- Petición de datos en relación a los sistemas de archivo del sistema. La solicitud de información fue ejecutada en gypsy.

```
[root@gypsy root]# snmpwalk -v 2c -c COMPRUEBAL localhost .1.3.6.1.4.1.2021.9

UCD-SNMP-MIB::dskIndex.1 = INTEGER: 1
UCD-SNMP-MIB::dskIndex.2 = INTEGER: 2
UCD-SNMP-MIB::dskPath.1 = STRING: /
UCD-SNMP-MIB::dskPath.2 = STRING: /home
UCD-SNMP-MIB::dskDevice.1 = STRING: /dev/hda2
UCD-SNMP-MIB::dskDevice.2 = STRING: /dev/hda5
UCD-SNMP-MIB::dskMinimum.1 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.2 = INTEGER: -1
UCD-SNMP-MIB::dskMinPercent.1 = INTEGER: 10
UCD-SNMP-MIB::dskMinPercent.2 = INTEGER: 10
UCD-SNMP-MIB::dskTotal.1 = INTEGER: 6048352
UCD-SNMP-MIB::dskTotal.2 = INTEGER: 1565392
UCD-SNMP-MIB::dskAvail.1 = INTEGER: 2954976
UCD-SNMP-MIB::dskAvail.2 = INTEGER: 1323820
UCD-SNMP-MIB::dskUsed.1 = INTEGER: 2786136
UCD-SNMP-MIB::dskUsed.2 = INTEGER: 162052
UCD-SNMP-MIB::dskPercent.1 = INTEGER: 49
UCD-SNMP-MIB::dskPercent.2 = INTEGER: 11
UCD-SNMP-MIB::dskPercentNode.1 = INTEGER: 18
UCD-SNMP-MIB::dskPercentNode.2 = INTEGER: 1
UCD-SNMP-MIB::dskErrorFlag.1 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::dskErrorMsg.1 = STRING:
UCD-SNMP-MIB::dskErrorMsg.2 = STRING:
```

- Petición de datos a la MIB en relación a la carga de sistema. Proceso ejecutado en ambos equipos.

```
[root@gypsy root]# snmpwalk -v 2c -c COMPRUEBAL localhost .1.3.6.1.4.1.2021.10

UCD-SNMP-MIB::laIndex.1 = INTEGER: 1
UCD-SNMP-MIB::laIndex.2 = INTEGER: 2
UCD-SNMP-MIB::laIndex.3 = INTEGER: 3
UCD-SNMP-MIB::laNames.1 = STRING: Load-1
UCD-SNMP-MIB::laNames.2 = STRING: Load-5
UCD-SNMP-MIB::laNames.3 = STRING: Load-15
UCD-SNMP-MIB::laLoad.1 = STRING: 0.00
UCD-SNMP-MIB::laLoad.2 = STRING: 0.05
UCD-SNMP-MIB::laLoad.3 = STRING: 0.08
UCD-SNMP-MIB::laConfig.1 = STRING: 2.00
UCD-SNMP-MIB::laConfig.2 = STRING: 3.00
UCD-SNMP-MIB::laConfig.3 = STRING: 3.00
UCD-SNMP-MIB::laLoadInt.1 = INTEGER: 0
UCD-SNMP-MIB::laLoadInt.2 = INTEGER: 5
UCD-SNMP-MIB::laLoadInt.3 = INTEGER: 8
UCD-SNMP-MIB::laLoadFloat.1 = Opaque: Float: 0.000000
UCD-SNMP-MIB::laLoadFloat.2 = Opaque: Float: 0.050000
UCD-SNMP-MIB::laLoadFloat.3 = Opaque: Float: 0.080000
UCD-SNMP-MIB::laErrorFlag.1 = INTEGER: 0
UCD-SNMP-MIB::laErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::laErrorFlag.3 = INTEGER: 0
UCD-SNMP-MIB::laErrMessage.1 = STRING:
UCD-SNMP-MIB::laErrMessage.2 = STRING:
UCD-SNMP-MIB::laErrMessage.3 = STRING:
```

```
[root@lentinius root]# snmpwalk -v 2c -c COMPRUEBAL localhost .1.3.6.1.4.1.2021.10

UCD-SNMP-MIB::laIndex.1 = INTEGER: 1
```

```
UCD-SNMP-MIB::laIndex.2 = INTEGER: 2
UCD-SNMP-MIB::laIndex.3 = INTEGER: 3
UCD-SNMP-MIB::laNames.1 = STRING: Load-1
UCD-SNMP-MIB::laNames.2 = STRING: Load-5
UCD-SNMP-MIB::laNames.3 = STRING: Load-15
UCD-SNMP-MIB::laLoad.1 = STRING: 0.00
UCD-SNMP-MIB::laLoad.2 = STRING: 0.02
UCD-SNMP-MIB::laLoad.3 = STRING: 0.03
UCD-SNMP-MIB::laConfig.1 = STRING: 2.00
UCD-SNMP-MIB::laConfig.2 = STRING: 3.00
UCD-SNMP-MIB::laConfig.3 = STRING: 3.00
UCD-SNMP-MIB::laLoadInt.1 = INTEGER: 0
UCD-SNMP-MIB::laLoadInt.2 = INTEGER: 2
UCD-SNMP-MIB::laLoadInt.3 = INTEGER: 3
UCD-SNMP-MIB::laLoadFloat.1 = Opaque: Float: 0.000000
UCD-SNMP-MIB::laLoadFloat.2 = Opaque: Float: 0.020000
UCD-SNMP-MIB::laLoadFloat.3 = Opaque: Float: 0.030000
UCD-SNMP-MIB::laErrorFlag.1 = INTEGER: 0
UCD-SNMP-MIB::laErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::laErrorFlag.3 = INTEGER: 0
UCD-SNMP-MIB::laErrMessage.1 = STRING:
UCD-SNMP-MIB::laErrMessage.2 = STRING:
UCD-SNMP-MIB::laErrMessage.3 = STRING:
```

4.1.4.2.3.2 Comunicación remota

Para llevar a cabo las siguientes pruebas fue necesario que el firewall de ambos equipos permitiera la comunicación a la ip 132.248.10.45 (servidor de graficación) al protocolo UDP a través del puerto 161.

- Petición desde el servidor de graficación (132.248.10.45) respecto a la información de procesos. El cuestionamiento fue realizado a gypsy.

```
[root@aletia root]# snmpwalk -v 2c -c COMSERGRAF 132.248.120.102 .1.3.6.1.4.1.2021.2

UCD-SNMP-MIB::prIndex.1 = INTEGER: 1
UCD-SNMP-MIB::prIndex.2 = INTEGER: 2
UCD-SNMP-MIB::prNames.1 = STRING: httpd
UCD-SNMP-MIB::prNames.2 = STRING: sendmail
UCD-SNMP-MIB::prMin.1 = INTEGER: 4
UCD-SNMP-MIB::prMin.2 = INTEGER: 1
UCD-SNMP-MIB::prMax.1 = INTEGER: 100
UCD-SNMP-MIB::prMax.2 = INTEGER: 3
UCD-SNMP-MIB::prCount.1 = INTEGER: 6
UCD-SNMP-MIB::prCount.2 = INTEGER: 2
UCD-SNMP-MIB::prErrorFlag.1 = INTEGER: 0
UCD-SNMP-MIB::prErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::prErrMessage.1 = STRING:
UCD-SNMP-MIB::prErrMessage.2 = STRING:
UCD-SNMP-MIB::prErrFix.1 = INTEGER: 0
UCD-SNMP-MIB::prErrFix.2 = INTEGER: 0
UCD-SNMP-MIB::prErrFixCmd.1 = STRING:
UCD-SNMP-MIB::prErrFixCmd.2 = STRING:
```

- Petición desde el servidor de graficación (132.248.10.45) respecto a la información de sistema de archivos

```
[root@aletia root]# snmpwalk -v 2c -c COMSERGRAF 132.248.120.102 .1.3.6.1.4.1.2021.9

UCD-SNMP-MIB::dskIndex.1 = INTEGER: 1
UCD-SNMP-MIB::dskIndex.2 = INTEGER: 2
UCD-SNMP-MIB::dskPath.1 = STRING: /
UCD-SNMP-MIB::dskPath.2 = STRING: /home
UCD-SNMP-MIB::dskDevice.1 = STRING: /dev/hda2
UCD-SNMP-MIB::dskDevice.2 = STRING: /dev/hda5
```

```
UCD-SNMP-MIB::dskMinimum.1 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.2 = INTEGER: -1
UCD-SNMP-MIB::dskMinPercent.1 = INTEGER: 10
UCD-SNMP-MIB::dskMinPercent.2 = INTEGER: 10
UCD-SNMP-MIB::dskTotal.1 = INTEGER: 6048352
UCD-SNMP-MIB::dskTotal.2 = INTEGER: 1565392
UCD-SNMP-MIB::dskAvail.1 = INTEGER: 2954976
UCD-SNMP-MIB::dskAvail.2 = INTEGER: 1323820
UCD-SNMP-MIB::dskUsed.1 = INTEGER: 2786136
UCD-SNMP-MIB::dskUsed.2 = INTEGER: 162052
UCD-SNMP-MIB::dskPercent.1 = INTEGER: 49
UCD-SNMP-MIB::dskPercent.2 = INTEGER: 11
UCD-SNMP-MIB::dskPercentNode.1 = INTEGER: 18
UCD-SNMP-MIB::dskPercentNode.2 = INTEGER: 1
UCD-SNMP-MIB::dskErrorFlag.1 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::dskErrorMsg.1 = STRING:
UCD-SNMP-MIB::dskErrorMsg.2 = STRING:
```

- Petición desde el servidor de graficación (132.248.10.45) respecto a la información de carga del sistema. El requerimiento de información fue realizado sobre lentinius.

```
[root@aletia root]# snmpwalk -v 2c -c COMSERGRAF 132.248.120.107 .1.3.6.1.4.1.2021.10

UCD-SNMP-MIB::laIndex.1 = INTEGER: 1
UCD-SNMP-MIB::laIndex.2 = INTEGER: 2
UCD-SNMP-MIB::laIndex.3 = INTEGER: 3
UCD-SNMP-MIB::laNames.1 = STRING: Load-1
UCD-SNMP-MIB::laNames.2 = STRING: Load-5
UCD-SNMP-MIB::laNames.3 = STRING: Load-15
UCD-SNMP-MIB::laLoad.1 = STRING: 0.00
UCD-SNMP-MIB::laLoad.2 = STRING: 0.02
UCD-SNMP-MIB::laLoad.3 = STRING: 0.03
```

```
UCD-SNMP-MIB::laConfig.1 = STRING: 2.00
UCD-SNMP-MIB::laConfig.2 = STRING: 3.00
UCD-SNMP-MIB::laConfig.3 = STRING: 3.00
UCD-SNMP-MIB::laLoadInt.1 = INTEGER: 0
UCD-SNMP-MIB::laLoadInt.2 = INTEGER: 2
UCD-SNMP-MIB::laLoadInt.3 = INTEGER: 3
UCD-SNMP-MIB::laLoadFloat.1 = Opaque: Float: 0.000000
UCD-SNMP-MIB::laLoadFloat.2 = Opaque: Float: 0.020000
UCD-SNMP-MIB::laLoadFloat.3 = Opaque: Float: 0.030000
UCD-SNMP-MIB::laErrorFlag.1 = INTEGER: 0
UCD-SNMP-MIB::laErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::laErrorFlag.3 = INTEGER: 0
UCD-SNMP-MIB::laErrMessage.1 = STRING:
UCD-SNMP-MIB::laErrMessage.2 = STRING:
UCD-SNMP-MIB::laErrMessage.3 = STRING:
```

4.1.4.2.4 Pruebas de comunicación a través de usuarios SNMPv3

4.1.4.2.4.1 Comunicación local

- Información de procesos en gypsy. Los argumentos del programa habilitan que la comunicación sea autenticada y cifrada.

```
[root@gypsy root]# snmpwalk -v 3 -u pruebaL -l authPriv -A prueba2007 -X prueba2007 localhost .1.3.6.1.4.1.2021.2

UCD-SNMP-MIB::prIndex.1 = INTEGER: 1
UCD-SNMP-MIB::prIndex.2 = INTEGER: 2
UCD-SNMP-MIB::prNames.1 = STRING: httpd
UCD-SNMP-MIB::prNames.2 = STRING: sendmail
UCD-SNMP-MIB::prMin.1 = INTEGER: 4
UCD-SNMP-MIB::prMin.2 = INTEGER: 1
```

```
UCD-SNMP-MIB::prMax.1 = INTEGER: 100
UCD-SNMP-MIB::prMax.2 = INTEGER: 3
UCD-SNMP-MIB::prCount.1 = INTEGER: 6
UCD-SNMP-MIB::prCount.2 = INTEGER: 2
UCD-SNMP-MIB::prErrorFlag.1 = INTEGER: 0
UCD-SNMP-MIB::prErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::prErrMessage.1 = STRING:
UCD-SNMP-MIB::prErrMessage.2 = STRING:
UCD-SNMP-MIB::prErrFix.1 = INTEGER: 0
UCD-SNMP-MIB::prErrFix.2 = INTEGER: 0
UCD-SNMP-MIB::prErrFixCmd.1 = STRING:
UCD-SNMP-MIB::prErrFixCmd.2 = STRING:
```

- Información de carga de sistema en lentinius. En esta ocasión la comunicación solo es autenticada y no cifrada.

```
[root@lentinius root]# snmpwalk -v 3 -u pruebaL -l authNoPriv -A prueba2007 localhost .1.3.6.1.4.1.2021.10

UCD-SNMP-MIB::laIndex.1 = INTEGER: 1
UCD-SNMP-MIB::laIndex.2 = INTEGER: 2
UCD-SNMP-MIB::laIndex.3 = INTEGER: 3
UCD-SNMP-MIB::laNames.1 = STRING: Load-1
UCD-SNMP-MIB::laNames.2 = STRING: Load-5
UCD-SNMP-MIB::laNames.3 = STRING: Load-15
UCD-SNMP-MIB::laLoad.1 = STRING: 0.00
UCD-SNMP-MIB::laLoad.2 = STRING: 0.02
UCD-SNMP-MIB::laLoad.3 = STRING: 0.03
UCD-SNMP-MIB::laConfig.1 = STRING: 2.00
UCD-SNMP-MIB::laConfig.2 = STRING: 3.00
UCD-SNMP-MIB::laConfig.3 = STRING: 3.00
UCD-SNMP-MIB::laLoadInt.1 = INTEGER: 0
UCD-SNMP-MIB::laLoadInt.2 = INTEGER: 2
UCD-SNMP-MIB::laLoadInt.3 = INTEGER: 3
```

```
UCD-SNMP-MIB::laLoadFloat.1 = Opaque: Float: 0.000000
UCD-SNMP-MIB::laLoadFloat.2 = Opaque: Float: 0.020000
UCD-SNMP-MIB::laLoadFloat.3 = Opaque: Float: 0.030000
UCD-SNMP-MIB::laErrorFlag.1 = INTEGER: 0
UCD-SNMP-MIB::laErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::laErrorFlag.3 = INTEGER: 0
UCD-SNMP-MIB::laErrMessage.1 = STRING:
UCD-SNMP-MIB::laErrMessage.2 = STRING:
UCD-SNMP-MIB::laErrMessage.3 = STRING:
```

- Información de los sistemas de ficheros en gypsy.

```
[root@gypsy root]# snmpwalk -v 3 -u pruebaL -l authPriv -A prueba2007 -X prueba2007 localhost .1.3.6.1.4.1.2021.9

UCD-SNMP-MIB::dskIndex.1 = INTEGER: 1
UCD-SNMP-MIB::dskIndex.2 = INTEGER: 2
UCD-SNMP-MIB::dskPath.1 = STRING: /
UCD-SNMP-MIB::dskPath.2 = STRING: /home
UCD-SNMP-MIB::dskDevice.1 = STRING: /dev/hda2
UCD-SNMP-MIB::dskDevice.2 = STRING: /dev/hda5
UCD-SNMP-MIB::dskMinimum.1 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.2 = INTEGER: -1
UCD-SNMP-MIB::dskMinPercent.1 = INTEGER: 10
UCD-SNMP-MIB::dskMinPercent.2 = INTEGER: 10
UCD-SNMP-MIB::dskTotal.1 = INTEGER: 6048352
UCD-SNMP-MIB::dskTotal.2 = INTEGER: 1565392
UCD-SNMP-MIB::dskAvail.1 = INTEGER: 2954932
UCD-SNMP-MIB::dskAvail.2 = INTEGER: 1323820
UCD-SNMP-MIB::dskUsed.1 = INTEGER: 2786180
UCD-SNMP-MIB::dskUsed.2 = INTEGER: 162052
UCD-SNMP-MIB::dskPercent.1 = INTEGER: 49
UCD-SNMP-MIB::dskPercent.2 = INTEGER: 11
UCD-SNMP-MIB::dskPercentNode.1 = INTEGER: 18
```

```
UCD-SNMP-MIB::dskPercentNode.2 = INTEGER: 1
UCD-SNMP-MIB::dskErrorFlag.1 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::dskErrorMsg.1 = STRING:
UCD-SNMP-MIB::dskErrorMsg.2 = STRING:
```

4.1.4.2.4.2 Comunicación remota

Al igual que las pruebas realizadas por medio de comunidades, es necesario tener la configuración correcta en el firewall de ambos equipos.

- Información de procesos en lentinius. La petición es ejecutada desde el servidor de graficación con características de autenticación y cifrado.

```
[root@aletia root]# snmpwalk -v 3 -u pruebaL -l authPriv -A prueba2007 -X prueba2007  132.248.120.107
.1.3.6.1.4.1.2021.2

UCD-SNMP-MIB::prIndex.1 = INTEGER: 1
UCD-SNMP-MIB::prIndex.2 = INTEGER: 2
UCD-SNMP-MIB::prNames.1 = STRING: httpd
UCD-SNMP-MIB::prNames.2 = STRING: sendmail
UCD-SNMP-MIB::prMin.1 = INTEGER: 4
UCD-SNMP-MIB::prMin.2 = INTEGER: 1
UCD-SNMP-MIB::prMax.1 = INTEGER: 100
UCD-SNMP-MIB::prMax.2 = INTEGER: 3
UCD-SNMP-MIB::prCount.1 = INTEGER: 0
UCD-SNMP-MIB::prCount.2 = INTEGER: 2
UCD-SNMP-MIB::prErrorFlag.1 = INTEGER: 1
UCD-SNMP-MIB::prErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::prErrMessage.1 = STRING: Too few httpd running (# = 0)
UCD-SNMP-MIB::prErrMessage.2 = STRING:
UCD-SNMP-MIB::prErrFix.1 = INTEGER: 0
UCD-SNMP-MIB::prErrFix.2 = INTEGER: 0
```

```
UCD-SNMP-MIB::prErrFixCmd.1 = STRING:
UCD-SNMP-MIB::prErrFixCmd.2 = STRING:
```

- Información de carga de sistema en gypsy. En esta ocasión la comunicación solo es autenticada y no cifrada.

```
[root@aletia root]# snmpwalk -v 3 -u pruebaL -l authNoPriv -A prueba2007  132.248.120.102 .1.3.6.1.4.1.2021.10

UCD-SNMP-MIB::laIndex.1 = INTEGER: 1
UCD-SNMP-MIB::laIndex.2 = INTEGER: 2
UCD-SNMP-MIB::laIndex.3 = INTEGER: 3
UCD-SNMP-MIB::laNames.1 = STRING: Load-1
UCD-SNMP-MIB::laNames.2 = STRING: Load-5
UCD-SNMP-MIB::laNames.3 = STRING: Load-15
UCD-SNMP-MIB::laLoad.1 = STRING: 0.05
UCD-SNMP-MIB::laLoad.2 = STRING: 0.08
UCD-SNMP-MIB::laLoad.3 = STRING: 0.02
UCD-SNMP-MIB::laConfig.1 = STRING: 2.00
UCD-SNMP-MIB::laConfig.2 = STRING: 3.00
UCD-SNMP-MIB::laConfig.3 = STRING: 3.00
UCD-SNMP-MIB::laLoadInt.1 = INTEGER: 5
UCD-SNMP-MIB::laLoadInt.2 = INTEGER: 8
UCD-SNMP-MIB::laLoadInt.3 = INTEGER: 2
UCD-SNMP-MIB::laLoadFloat.1 = Opaque: Float: 0.050000
UCD-SNMP-MIB::laLoadFloat.2 = Opaque: Float: 0.080000
UCD-SNMP-MIB::laLoadFloat.3 = Opaque: Float: 0.020000
UCD-SNMP-MIB::laErrorFlag.1 = INTEGER: 0
UCD-SNMP-MIB::laErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::laErrorFlag.3 = INTEGER: 0
UCD-SNMP-MIB::laErrMessage.1 = STRING:
UCD-SNMP-MIB::laErrMessage.2 = STRING:
UCD-SNMP-MIB::laErrMessage.3 = STRING:
```

- Información de los sistemas de ficheros en gypsy.

```
[root@aletia root]# snmpwalk -v 3 -u pruebaL -l authPriv -A prueba2007 -X prueba2007 132.248.120.102
.1.3.6.1.4.1.2021.9

UCD-SNMP-MIB::dskIndex.1 = INTEGER: 1
UCD-SNMP-MIB::dskIndex.2 = INTEGER: 2
UCD-SNMP-MIB::dskPath.1 = STRING: /
UCD-SNMP-MIB::dskPath.2 = STRING: /home
UCD-SNMP-MIB::dskDevice.1 = STRING: /dev/hda2
UCD-SNMP-MIB::dskDevice.2 = STRING: /dev/hda5
UCD-SNMP-MIB::dskMinimum.1 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.2 = INTEGER: -1
UCD-SNMP-MIB::dskMinPercent.1 = INTEGER: 10
UCD-SNMP-MIB::dskMinPercent.2 = INTEGER: 10
UCD-SNMP-MIB::dskTotal.1 = INTEGER: 6048352
UCD-SNMP-MIB::dskTotal.2 = INTEGER: 1565392
UCD-SNMP-MIB::dskAvail.1 = INTEGER: 2954932
UCD-SNMP-MIB::dskAvail.2 = INTEGER: 1323820
UCD-SNMP-MIB::dskUsed.1 = INTEGER: 2786180
UCD-SNMP-MIB::dskUsed.2 = INTEGER: 162052
UCD-SNMP-MIB::dskPercent.1 = INTEGER: 49
UCD-SNMP-MIB::dskPercent.2 = INTEGER: 11
UCD-SNMP-MIB::dskPercentNode.1 = INTEGER: 18
UCD-SNMP-MIB::dskPercentNode.2 = INTEGER: 1
UCD-SNMP-MIB::dskErrorFlag.1 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::dskErrorMsg.1 = STRING:
UCD-SNMP-MIB::dskErrorMsg.2 = STRING:
```

4.1.4.2.5 Resultados

Todos los resultados de las pruebas que fueron realizadas en los equipos gypsy y lentinius permitieron sacar una serie de conclusiones acerca del funcionamiento de SNMP. Las peticiones de información a ambos agentes SNMP permitieron concluir que la implementación del protocolo es totalmente funcional en ambos sistemas operativos, ya que todas las peticiones realizadas fueron procesadas de igual manera por ambos agentes.

La comunicación a través de comunidades así como usuarios SNMPv3 comprobó que Net-SNMP tiene el soporte para las versiones SNMPv1, SNMPv2 y SNMPv3 del protocolo.

Otro aspecto importante, fue observar la manera en que SNMP muestra la información acerca de los recursos que le fueron dictados a monitorear. En el caso de los procesos y sistema de archivos la información es almacenada bajo una sola rama (que podemos llamar tabla) que va generando un índice por cada elemento que esta siendo monitoreado, por lo que el numero de índices va aumentando conforme al numero de procesos o sistemas de archivos monitoreados. Por otra parte esta el monitoreo del promedio de carga del sistema, donde la información se guarda en una tabla con un numero de elementos fijos puesto que esta determinada a revisar la carga en los periodos de 1, 5 y 15 minutos. En toda la información de administración (MIB) almacenada por el agente SNMP se pueden observar detalles acerca de los elementos que están siendo monitoreados así como la presencia de banderas y mensajes de error en caso de presentarse una condición falsa, como es el caso del monitoreo de procesos del equipo lentinius donde no esta corriendo un servidor Apache (httpd).

En base a lo anterior se decidió continuar con la siguiente fase de pruebas, en donde se implementaría el protocolo SNMP sobre 2 servidores de vital importancia para el departamento. Solo que a diferencia de esta fase, solo serán configurados usuarios

SNMPv3 para asegurar la confidencialidad de los datos transmitidos, puesto que la comunicación establecida a través de comunidades no soporta ninguna característica de autenticación y encriptación de datos.

4.1.4.3 Segunda fase de prueba

4.1.4.3.1 Descripción de los equipos

Las características de los servidores se exponen en la tabla 4.2.

Nombre	***Sistema Operativo***	***Hardware***	***Servicios***	***Dirección IP***
Velvet	Solaris 9	Dominio del servidor Sun E6900 con 16 Gb de memoria RAM y 4 procesadores UltraSPARC IV + a una velocidad de 1200 MHz. Tiene una capacidad de almacenamiento de 700 Gb a través de un Sun Storage.	Proporciona el servicio de hosting y base de datos para más de 160 sitios web de la universidad	192.168.10.6
Servidor4	Linux Enterprise Red Hat 3	Dispone de 2 procesadores Intel Pentium 2, memoria RAM de 2 Gb y 4 discos duros de 160 Gb (2) y 140 Gb (2).	Brinda el servicio de directorio LDAP para autenticar a los usuarios de correo electrónico del dominio servidor.unam.mx	132.248.10.23

Tabla 4.2 Descripción de los equipos usados en Fase 2

4.1.4.3.2 Configuración de comunicación y monitoreo

Para esta fase de pruebas se determino que solo serían creados usuarios SNMPv3 para establecer la comunicación entre el gestor (servidor de graficación) y los agentes SNMP. Debido a la importancia de los servidores se consideró más seguro utilizar las características de autenticación y encriptación brindadas por la versión SNMPv3 del protocolo.

4.1.4.3.2.1 Configuración de usuarios SNMPv3

- Antes de llevar a cabo la creación de algún usuario SNMPv3 es importante recordar que debemos crear un archivo de configuración para el funcionamiento de snmp, por lo que se puede tomar como plantilla el archivo "EXAMPLE.conf" proporcionado por la propia distribución del Net-SNMP.

```
[root@velvet root]#cd /root/SRC/net-snmp.5.1.4
[root@velvet root]#cp EXAMPLE.conf  /usr/local/share/snmp/snmpd.conf
```

Cabe mencionar que si es utilizado el archivo "EXAMPLE.conf" como plantilla para crear el archivo de configuración "snmpd.conf" es necesario, por cuestiones de seguridad, comentar o borrar las líneas que tienen como objetivo ejemplificar la creación de comunidades.

- Antes de crear algún usuario, es necesario que el demonio de SNMP no se este ejecutando. Se puede buscar el proceso para darlo de baja o echar mano del shell script que se proporcionó en el apartado de 4.1.4.2.2 Configuración de comunicación y monitoreo, en la sección 4.1.4.2.2.4 Arrancar el agente SNMP de la primera fase de pruebas.

```
[root@velvet root]#/etc/init.d/snmp stop
```

- Crear los usuarios SNMPv3 tanto en el servidor velvet así como en servidor4. En ambos servidores el usuario será “prueba2L” y contraseña “prueba2L2007”.

```
[root@velvet root]#net-snmp-config --create-snmpv3-user -ro -a “prueba2L2007“ prueba2L

adding the following line to /var/net-snmp/snmpd.conf:
  createUser prueba2L MD5 "“prueba2L2007“" DES
adding the following line to /usr/local/share/snmp/snmpd.conf:
  rouser prueba2L
```

```
[root@servidor4 root]#net-snmp-config --create-snmpv3-user -ro -a “prueba2L2007“ prueba2L

adding the following line to /var/net-snmp/snmpd.conf:
  createUser prueba2L MD5 "“prueba2L2007“" DES
adding the following line to /usr/local/share/snmp/snmpd.conf:
  rouser prueba2L
```

Si se tienen dudas respecto al script utilizado para generar los usuarios SNMPv3, remitirse a la sección 4.1.4.2.2 Configuración de comunicación y monitoreo en el apartado 4.1.4.2.2.2 Configuración de usuarios SNMPv3, de la primera fase de prueba.

4.1.4.3.2.2 Configuración de monitoreo

Debido a que ambos servidores tienen diferentes características en cuanto a recursos y servicios informáticos, se decidió tener una configuración diferente en ambos servidores.

4.1.4.3.2.2.1 Configuración de velvet

- Debido a que velvet ofrece el servicio de hosting y base de datos, se decidió monitorear los procesos de los servidores web Apache, tomcat y los manejadores de base de datos sybase, postgres, mysql.

```
proc httpd 250 15
proc dataserver 2 1
proc postgres 80 1
proc mysqld_safe 2 1
proc java 3 1
```

- El número de sistemas de archivo en velvet es considerable debido a la capacidad de su Storage. Las siguientes directivas permiten monitorear todos sus puntos de montaje.

```
disk / 10%
disk /home/sybase 10%
disk /var/crash 10%
disk /usr/local 10%
disk /home/logs 10%
disk /var 10%
disk /home 20%
disk /home/users04 20%
disk  /home/users11 20%
disk  /home/users12 20%
disk  /home/users13 20%
disk  /home/users10 20%
disk  /home/users07 20%
disk  /home/users00 20%
disk  /home/users06 20%
disk  /home/users01 20%
```

```
disk  /home/users14 20%
disk  /home/users02 20%
disk  /home/users08 20%
disk  /home/users15 20%
disk  /home/users16 20%
disk  /home/users17 20%
disk  /home/users03 20%
disk  /home/users19 20%
disk  /home/users05 20%
disk  /home/users18 20%
disk  /home/users09 20%
```

- velvet cuenta con 40 Gb de memoria de intercambio (SWAP), por lo que se definió monitorear como mínimo disponible 500 Mb.

```
swap 512000
```

- Se definieron los siguientes parámetros para el promedio de carga del sistema.

```
load 8 10 10
```

4.1.4.3.2.2.2 Configuración de servidor4

- Debido a que servdor4 ofrece el servicio de directorio (LDAP) y correo electrónico, se decidió monitorear los procesos de LDAP y qmail, procesos para el servicio de directorio y correo respectivamente.

```
proc slapd 10 1
```

```
proc qmail-send 2 1
proc qmail-smtpd 2 1
```

- El número de sistemas de archivo en servidor4 fue monitoreado con las siguientes directivas.

```
disk / 10%
disk /var 10%
disk /home1 10%
disk /home2 10%
disk /usr 10%
disk /boot 10%
disk /spare 10%
```

- Servidor4 cuenta con 2 Gb de memoria de intercambio (SWAP), por lo que se definió monitorear como mínimo disponible 300 Mb.

```
swap 307200
```

- Se definieron los siguientes parámetros para el promedio de carga del sistema.

```
load 9 9 9
```

4.1.4.3.2.3 Arrancar el agente SNMP

- Para levantar al agente basta con ejecutar el binario snmpd ubicado en la ruta "/usr/local/sbin/snmpd", o utilizar el script proporcionado durante la primera fase de pruebas.

```
[root@velvet root]#/etc/init.d/snmp start
```

4.1.4.3.3 Pruebas de comunicación

Todas las pruebas de petición de datos fueron realizadas remotamente desde el gestor SNMP (servidor de graficación) cuya dirección ip es la 132.248.10.45. Los cuestionamientos hechos a la información de administración (MIB) de los agentes SNMP fueron utilizando las características de autenticación y encriptación de la versión 3 de SNMP.

4.1.4.3.3.1 Pruebas en velvet

- Petición de datos acerca de los procesos monitoreados en velvet.

```
[root@aletia root]# snmpwalk -v 3 -u prueba2L -l authPriv  -A prueba2L2007 -X prueba2L2007
192.168.10.6 .1.3.6.1.4.1.2021.2

UCD-SNMP-MIB::prIndex.1 = INTEGER: 1
UCD-SNMP-MIB::prIndex.2 = INTEGER: 2
UCD-SNMP-MIB::prIndex.3 = INTEGER: 3
UCD-SNMP-MIB::prIndex.4 = INTEGER: 4
UCD-SNMP-MIB::prIndex.5 = INTEGER: 5
UCD-SNMP-MIB::prNames.1 = STRING: httpd
```

```
UCD-SNMP-MIB::prNames.2 = STRING: dataserver
UCD-SNMP-MIB::prNames.3 = STRING: postgres
UCD-SNMP-MIB::prNames.4 = STRING: mysqld_safe
UCD-SNMP-MIB::prNames.5 = STRING: java
UCD-SNMP-MIB::prMin.1 = INTEGER: 15
UCD-SNMP-MIB::prMin.2 = INTEGER: 1
UCD-SNMP-MIB::prMin.3 = INTEGER: 1
UCD-SNMP-MIB::prMin.4 = INTEGER: 1
UCD-SNMP-MIB::prMin.5 = INTEGER: 1
UCD-SNMP-MIB::prMax.1 = INTEGER: 250
UCD-SNMP-MIB::prMax.2 = INTEGER: 10
UCD-SNMP-MIB::prMax.3 = INTEGER: 80
UCD-SNMP-MIB::prMax.4 = INTEGER: 10
UCD-SNMP-MIB::prMax.5 = INTEGER: 10
UCD-SNMP-MIB::prCount.1 = INTEGER: 200
UCD-SNMP-MIB::prCount.2 = INTEGER: 1
UCD-SNMP-MIB::prCount.3 = INTEGER: 27
UCD-SNMP-MIB::prCount.4 = INTEGER: 1
UCD-SNMP-MIB::prCount.5 = INTEGER: 3
UCD-SNMP-MIB::prErrorFlag.1 = INTEGER: 0
UCD-SNMP-MIB::prErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::prErrorFlag.3 = INTEGER: 0
UCD-SNMP-MIB::prErrorFlag.4 = INTEGER: 0
UCD-SNMP-MIB::prErrorFlag.5 = INTEGER: 0
UCD-SNMP-MIB::prErrMessage.1 = STRING:
UCD-SNMP-MIB::prErrMessage.2 = STRING:
UCD-SNMP-MIB::prErrMessage.3 = STRING:
UCD-SNMP-MIB::prErrMessage.4 = STRING:
UCD-SNMP-MIB::prErrMessage.5 = STRING:
UCD-SNMP-MIB::prErrFix.1 = INTEGER: 0
UCD-SNMP-MIB::prErrFix.2 = INTEGER: 0
UCD-SNMP-MIB::prErrFix.3 = INTEGER: 0
UCD-SNMP-MIB::prErrFix.4 = INTEGER: 0
UCD-SNMP-MIB::prErrFix.5 = INTEGER: 0
UCD-SNMP-MIB::prErrFixCmd.1 = STRING:
UCD-SNMP-MIB::prErrFixCmd.2 = STRING:
UCD-SNMP-MIB::prErrFixCmd.3 = STRING:
```

```
UCD-SNMP-MIB::prErrFixCmd.4 = STRING:
UCD-SNMP-MIB::prErrFixCmd.5 = STRING:
```

- Petición de datos acerca de los sistemas de archivo en velvet.

```
[root@aletia root]# snmpwalk -v 3 -u prueba2L -l authPriv -A prueba2L2007 -X prueba2L2007 192.168.10.6 .1.3.6.1.4.1.2021.9
UCD-SNMP-MIB::dskIndex.1 = INTEGER: 1
UCD-SNMP-MIB::dskIndex.2 = INTEGER: 2
UCD-SNMP-MIB::dskIndex.3 = INTEGER: 3
UCD-SNMP-MIB::dskIndex.4 = INTEGER: 4
UCD-SNMP-MIB::dskIndex.5 = INTEGER: 5
UCD-SNMP-MIB::dskIndex.6 = INTEGER: 6
UCD-SNMP-MIB::dskIndex.7 = INTEGER: 7
UCD-SNMP-MIB::dskIndex.8 = INTEGER: 8
UCD-SNMP-MIB::dskIndex.9 = INTEGER: 9
UCD-SNMP-MIB::dskIndex.10 = INTEGER: 10
UCD-SNMP-MIB::dskIndex.11 = INTEGER: 11
UCD-SNMP-MIB::dskIndex.12 = INTEGER: 12
UCD-SNMP-MIB::dskIndex.13 = INTEGER: 13
UCD-SNMP-MIB::dskIndex.14 = INTEGER: 14
UCD-SNMP-MIB::dskIndex.15 = INTEGER: 15
UCD-SNMP-MIB::dskIndex.16 = INTEGER: 16
UCD-SNMP-MIB::dskIndex.17 = INTEGER: 17
UCD-SNMP-MIB::dskIndex.18 = INTEGER: 18
UCD-SNMP-MIB::dskIndex.19 = INTEGER: 19
UCD-SNMP-MIB::dskIndex.20 = INTEGER: 20
UCD-SNMP-MIB::dskIndex.21 = INTEGER: 21
UCD-SNMP-MIB::dskIndex.22 = INTEGER: 22
UCD-SNMP-MIB::dskIndex.23 = INTEGER: 23
UCD-SNMP-MIB::dskIndex.24 = INTEGER: 24
UCD-SNMP-MIB::dskIndex.25 = INTEGER: 25
UCD-SNMP-MIB::dskIndex.26 = INTEGER: 26
UCD-SNMP-MIB::dskIndex.27 = INTEGER: 27
UCD-SNMP-MIB::dskPath.1 = STRING: /
UCD-SNMP-MIB::dskPath.2 = STRING: /home/sybase
UCD-SNMP-MIB::dskPath.3 = STRING: /var/crash
UCD-SNMP-MIB::dskPath.4 = STRING: /usr/local
UCD-SNMP-MIB::dskPath.5 = STRING: /home/logs
UCD-SNMP-MIB::dskPath.6 = STRING: /var
UCD-SNMP-MIB::dskPath.7 = STRING: /home
UCD-SNMP-MIB::dskPath.8 = STRING: /home/users04
UCD-SNMP-MIB::dskPath.9 = STRING: /home/users11
```

```
UCD-SNMP-MIB::dskPath.10 = STRING: /home/users12
UCD-SNMP-MIB::dskPath.11 = STRING: /home/users13
UCD-SNMP-MIB::dskPath.12 = STRING: /home/users10
UCD-SNMP-MIB::dskPath.13 = STRING: /home/users07
UCD-SNMP-MIB::dskPath.14 = STRING: /home/users00
UCD-SNMP-MIB::dskPath.15 = STRING: /home/users06
UCD-SNMP-MIB::dskPath.16 = STRING: /home/users01
UCD-SNMP-MIB::dskPath.17 = STRING: /home/users14
UCD-SNMP-MIB::dskPath.18 = STRING: /home/users02
UCD-SNMP-MIB::dskPath.19 = STRING: /home/users08
UCD-SNMP-MIB::dskPath.20 = STRING: /home/users15
UCD-SNMP-MIB::dskPath.21 = STRING: /home/users16
UCD-SNMP-MIB::dskPath.22 = STRING: /home/users17
UCD-SNMP-MIB::dskPath.23 = STRING: /home/users03
UCD-SNMP-MIB::dskPath.24 = STRING: /home/users19
UCD-SNMP-MIB::dskPath.25 = STRING: /home/users05
UCD-SNMP-MIB::dskPath.26 = STRING: /home/users18
UCD-SNMP-MIB::dskPath.27 = STRING: /home/users09
UCD-SNMP-MIB::dskDevice.1 = STRING: /dev/md/dsk/d0
UCD-SNMP-MIB::dskDevice.2 = STRING: /dev/dsk/c8t600C0FF00000000008776A51836BF509d0s0
UCD-SNMP-MIB::dskDevice.3 = STRING: /dev/md/dsk/d5
UCD-SNMP-MIB::dskDevice.4 = STRING: /dev/md/dsk/d100
UCD-SNMP-MIB::dskDevice.5 = STRING: /dev/md/dsk/d103
UCD-SNMP-MIB::dskDevice.6 = STRING: /dev/md/dsk/d4
UCD-SNMP-MIB::dskDevice.7 = STRING: /dev/md/dsk/d101
UCD-SNMP-MIB::dskDevice.8 = STRING: /dev/dsk/c8t600C0FF0000000000876DB04CFB9A804d0s0
UCD-SNMP-MIB::dskDevice.9 = STRING: /dev/dsk/c8t600C0FF00000000008776A51836BF500d0s0
UCD-SNMP-MIB::dskDevice.10 = STRING: /dev/dsk/c8t600C0FF00000000008776A51836BF501d0s0
UCD-SNMP-MIB::dskDevice.11 = STRING: /dev/dsk/c8t600C0FF00000000008776A51836BF502d0s0
UCD-SNMP-MIB::dskDevice.12 = STRING: /dev/dsk/c8t600C0FF00000000008776A51836BF50Ad0s0
UCD-SNMP-MIB::dskDevice.13 = STRING: /dev/dsk/c8t600C0FF0000000000876DB04CFB9A807d0s0
UCD-SNMP-MIB::dskDevice.14 = STRING: /dev/dsk/c8t600C0FF0000000000876DB04CFB9A800d0s0
UCD-SNMP-MIB::dskDevice.15 = STRING: /dev/dsk/c8t600C0FF0000000000876DB04CFB9A806d0s0
UCD-SNMP-MIB::dskDevice.16 = STRING: /dev/dsk/c8t600C0FF0000000000876DB04CFB9A801d0s0
UCD-SNMP-MIB::dskDevice.17 = STRING: /dev/dsk/c8t600C0FF00000000008776A51836BF503d0s0
UCD-SNMP-MIB::dskDevice.18 = STRING: /dev/dsk/c8t600C0FF0000000000876DB04CFB9A802d0s0
UCD-SNMP-MIB::dskDevice.19 = STRING: /dev/dsk/c8t600C0FF0000000000876DB04CFB9A808d0s0
UCD-SNMP-MIB::dskDevice.20 = STRING: /dev/dsk/c8t600C0FF00000000008776A51836BF504d0s0
UCD-SNMP-MIB::dskDevice.21 = STRING: /dev/dsk/c8t600C0FF00000000008776A51836BF505d0s0
UCD-SNMP-MIB::dskDevice.22 = STRING: /dev/dsk/c8t600C0FF00000000008776A51836BF506d0s0
UCD-SNMP-MIB::dskDevice.23 = STRING: /dev/dsk/c8t600C0FF0000000000876DB04CFB9A803d0s0
UCD-SNMP-MIB::dskDevice.24 = STRING: /dev/dsk/c8t600C0FF00000000008776A51836BF508d0s0
UCD-SNMP-MIB::dskDevice.25 = STRING: /dev/dsk/c8t600C0FF0000000000876DB04CFB9A805d0s0
UCD-SNMP-MIB::dskDevice.26 = STRING: /dev/dsk/c8t600C0FF00000000008776A51836BF507d0s0
UCD-SNMP-MIB::dskDevice.27 = STRING: /dev/dsk/c8t600C0FF0000000000876DB42A7160F00d0s0
UCD-SNMP-MIB::dskMinimum.1 = INTEGER: -1
```

```
UCD-SNMP-MIB::dskMinimum.2 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.3 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.4 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.5 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.6 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.7 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.8 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.9 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.10 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.11 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.12 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.13 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.14 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.15 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.16 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.17 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.18 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.19 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.20 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.21 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.22 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.23 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.24 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.25 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.26 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.27 = INTEGER: -1
UCD-SNMP-MIB::dskMinPercent.1 = INTEGER: 10
UCD-SNMP-MIB::dskMinPercent.2 = INTEGER: 10
UCD-SNMP-MIB::dskMinPercent.3 = INTEGER: 10
UCD-SNMP-MIB::dskMinPercent.4 = INTEGER: 10
UCD-SNMP-MIB::dskMinPercent.5 = INTEGER: 10
UCD-SNMP-MIB::dskMinPercent.6 = INTEGER: 10
UCD-SNMP-MIB::dskMinPercent.7 = INTEGER: 20
UCD-SNMP-MIB::dskMinPercent.8 = INTEGER: 20
UCD-SNMP-MIB::dskMinPercent.9 = INTEGER: 20
UCD-SNMP-MIB::dskMinPercent.10 = INTEGER: 20
UCD-SNMP-MIB::dskMinPercent.11 = INTEGER: 20
UCD-SNMP-MIB::dskMinPercent.12 = INTEGER: 20
UCD-SNMP-MIB::dskMinPercent.13 = INTEGER: 20
UCD-SNMP-MIB::dskMinPercent.14 = INTEGER: 20
UCD-SNMP-MIB::dskMinPercent.15 = INTEGER: 20
UCD-SNMP-MIB::dskMinPercent.16 = INTEGER: 20
UCD-SNMP-MIB::dskMinPercent.17 = INTEGER: 20
UCD-SNMP-MIB::dskMinPercent.18 = INTEGER: 20
UCD-SNMP-MIB::dskMinPercent.19 = INTEGER: 20
UCD-SNMP-MIB::dskMinPercent.20 = INTEGER: 20
```

```
UCD-SNMP-MIB::dskMinPercent.21 = INTEGER: 20
UCD-SNMP-MIB::dskMinPercent.22 = INTEGER: 20
UCD-SNMP-MIB::dskMinPercent.23 = INTEGER: 20
UCD-SNMP-MIB::dskMinPercent.24 = INTEGER: 20
UCD-SNMP-MIB::dskMinPercent.25 = INTEGER: 20
UCD-SNMP-MIB::dskMinPercent.26 = INTEGER: 20
UCD-SNMP-MIB::dskMinPercent.27 = INTEGER: 20
UCD-SNMP-MIB::dskTotal.1 = INTEGER: 5165838
UCD-SNMP-MIB::dskTotal.2 = INTEGER: 31956596
UCD-SNMP-MIB::dskTotal.3 = INTEGER: 10327372
UCD-SNMP-MIB::dskTotal.4 = INTEGER: 10327372
UCD-SNMP-MIB::dskTotal.5 = INTEGER: 39347300
UCD-SNMP-MIB::dskTotal.6 = INTEGER: 10327372
UCD-SNMP-MIB::dskTotal.7 = INTEGER: 20655528
UCD-SNMP-MIB::dskTotal.8 = INTEGER: 31245654
UCD-SNMP-MIB::dskTotal.9 = INTEGER: 31956596
UCD-SNMP-MIB::dskTotal.10 = INTEGER: 31956596
UCD-SNMP-MIB::dskTotal.11 = INTEGER: 31956596
UCD-SNMP-MIB::dskTotal.12 = INTEGER: 31956596
UCD-SNMP-MIB::dskTotal.13 = INTEGER: 31245654
UCD-SNMP-MIB::dskTotal.14 = INTEGER: 31245654
UCD-SNMP-MIB::dskTotal.15 = INTEGER: 31245654
UCD-SNMP-MIB::dskTotal.16 = INTEGER: 31245654
UCD-SNMP-MIB::dskTotal.17 = INTEGER: 31956596
UCD-SNMP-MIB::dskTotal.18 = INTEGER: 31245654
UCD-SNMP-MIB::dskTotal.19 = INTEGER: 31249750
UCD-SNMP-MIB::dskTotal.20 = INTEGER: 31956596
UCD-SNMP-MIB::dskTotal.21 = INTEGER: 31956596
UCD-SNMP-MIB::dskTotal.22 = INTEGER: 31956596
UCD-SNMP-MIB::dskTotal.23 = INTEGER: 31245654
UCD-SNMP-MIB::dskTotal.24 = INTEGER: 31956596
UCD-SNMP-MIB::dskTotal.25 = INTEGER: 31245654
UCD-SNMP-MIB::dskTotal.26 = INTEGER: 31956596
UCD-SNMP-MIB::dskTotal.27 = INTEGER: 31956596
UCD-SNMP-MIB::dskAvail.1 = INTEGER: 3484865
UCD-SNMP-MIB::dskAvail.2 = INTEGER: 25885948
UCD-SNMP-MIB::dskAvail.3 = INTEGER: 7835512
UCD-SNMP-MIB::dskAvail.4 = INTEGER: 4912142
UCD-SNMP-MIB::dskAvail.5 = INTEGER: 12592254
UCD-SNMP-MIB::dskAvail.6 = INTEGER: 9704476
UCD-SNMP-MIB::dskAvail.7 = INTEGER: 7750576
UCD-SNMP-MIB::dskAvail.8 = INTEGER: 18609814
UCD-SNMP-MIB::dskAvail.9 = INTEGER: 25442348
UCD-SNMP-MIB::dskAvail.10 = INTEGER: 26887600
UCD-SNMP-MIB::dskAvail.11 = INTEGER: 23676128
UCD-SNMP-MIB::dskAvail.12 = INTEGER: 14025767
```

```
UCD-SNMP-MIB::dskAvail.13 = INTEGER: 29238596
UCD-SNMP-MIB::dskAvail.14 = INTEGER: 20678552
UCD-SNMP-MIB::dskAvail.15 = INTEGER: 24244810
UCD-SNMP-MIB::dskAvail.16 = INTEGER: 27015224
UCD-SNMP-MIB::dskAvail.17 = INTEGER: 23899748
UCD-SNMP-MIB::dskAvail.18 = INTEGER: 29825532
UCD-SNMP-MIB::dskAvail.19 = INTEGER: 24003992
UCD-SNMP-MIB::dskAvail.20 = INTEGER: 24487228
UCD-SNMP-MIB::dskAvail.21 = INTEGER: 10240532
UCD-SNMP-MIB::dskAvail.22 = INTEGER: 10777890
UCD-SNMP-MIB::dskAvail.23 = INTEGER: 27975202
UCD-SNMP-MIB::dskAvail.24 = INTEGER: 12510201
UCD-SNMP-MIB::dskAvail.25 = INTEGER: 25716116
UCD-SNMP-MIB::dskAvail.26 = INTEGER: 23472988
UCD-SNMP-MIB::dskAvail.27 = INTEGER: 30470100
UCD-SNMP-MIB::dskUsed.1 = INTEGER: 1629315
UCD-SNMP-MIB::dskUsed.2 = INTEGER: 5751084
UCD-SNMP-MIB::dskUsed.3 = INTEGER: 2388587
UCD-SNMP-MIB::dskUsed.4 = INTEGER: 5311957
UCD-SNMP-MIB::dskUsed.5 = INTEGER: 26361572
UCD-SNMP-MIB::dskUsed.6 = INTEGER: 519623
UCD-SNMP-MIB::dskUsed.7 = INTEGER: 12698398
UCD-SNMP-MIB::dskUsed.8 = INTEGER: 12323384
UCD-SNMP-MIB::dskUsed.9 = INTEGER: 6194684
UCD-SNMP-MIB::dskUsed.10 = INTEGER: 4749431
UCD-SNMP-MIB::dskUsed.11 = INTEGER: 7960905
UCD-SNMP-MIB::dskUsed.12 = INTEGER: 17611264
UCD-SNMP-MIB::dskUsed.13 = INTEGER: 1694602
UCD-SNMP-MIB::dskUsed.14 = INTEGER: 10254647
UCD-SNMP-MIB::dskUsed.15 = INTEGER: 6688388
UCD-SNMP-MIB::dskUsed.16 = INTEGER: 3917973
UCD-SNMP-MIB::dskUsed.17 = INTEGER: 7737285
UCD-SNMP-MIB::dskUsed.18 = INTEGER: 1107667
UCD-SNMP-MIB::dskUsed.19 = INTEGER: 6933261
UCD-SNMP-MIB::dskUsed.20 = INTEGER: 7149804
UCD-SNMP-MIB::dskUsed.21 = INTEGER: 21396500
UCD-SNMP-MIB::dskUsed.22 = INTEGER: 20859142
UCD-SNMP-MIB::dskUsed.23 = INTEGER: 2957996
UCD-SNMP-MIB::dskUsed.24 = INTEGER: 19126832
UCD-SNMP-MIB::dskUsed.25 = INTEGER: 5217083
UCD-SNMP-MIB::dskUsed.26 = INTEGER: 8164044
UCD-SNMP-MIB::dskUsed.27 = INTEGER: 1166932
UCD-SNMP-MIB::dskPercent.1 = INTEGER: 32
UCD-SNMP-MIB::dskPercent.2 = INTEGER: 18
UCD-SNMP-MIB::dskPercent.3 = INTEGER: 23
UCD-SNMP-MIB::dskPercent.4 = INTEGER: 52
```

```
UCD-SNMP-MIB::dskPercent.5 = INTEGER: 68
UCD-SNMP-MIB::dskPercent.6 = INTEGER: 5
UCD-SNMP-MIB::dskPercent.7 = INTEGER: 62
UCD-SNMP-MIB::dskPercent.8 = INTEGER: 40
UCD-SNMP-MIB::dskPercent.9 = INTEGER: 20
UCD-SNMP-MIB::dskPercent.10 = INTEGER: 15
UCD-SNMP-MIB::dskPercent.11 = INTEGER: 25
UCD-SNMP-MIB::dskPercent.12 = INTEGER: 56
UCD-SNMP-MIB::dskPercent.13 = INTEGER: 5
UCD-SNMP-MIB::dskPercent.14 = INTEGER: 33
UCD-SNMP-MIB::dskPercent.15 = INTEGER: 22
UCD-SNMP-MIB::dskPercent.16 = INTEGER: 13
UCD-SNMP-MIB::dskPercent.17 = INTEGER: 24
UCD-SNMP-MIB::dskPercent.18 = INTEGER: 4
UCD-SNMP-MIB::dskPercent.19 = INTEGER: 22
UCD-SNMP-MIB::dskPercent.20 = INTEGER: 23
UCD-SNMP-MIB::dskPercent.21 = INTEGER: 68
UCD-SNMP-MIB::dskPercent.22 = INTEGER: 66
UCD-SNMP-MIB::dskPercent.23 = INTEGER: 10
UCD-SNMP-MIB::dskPercent.24 = INTEGER: 60
UCD-SNMP-MIB::dskPercent.25 = INTEGER: 17
UCD-SNMP-MIB::dskPercent.26 = INTEGER: 26
UCD-SNMP-MIB::dskPercent.27 = INTEGER: 4
UCD-SNMP-MIB::dskPercentNode.1 = INTEGER: 13
UCD-SNMP-MIB::dskPercentNode.2 = INTEGER: 0
UCD-SNMP-MIB::dskPercentNode.3 = INTEGER: 0
UCD-SNMP-MIB::dskPercentNode.4 = INTEGER: 3
UCD-SNMP-MIB::dskPercentNode.5 = INTEGER: 0
UCD-SNMP-MIB::dskPercentNode.6 = INTEGER: 1
UCD-SNMP-MIB::dskPercentNode.7 = INTEGER: 1
UCD-SNMP-MIB::dskPercentNode.8 = INTEGER: 10
UCD-SNMP-MIB::dskPercentNode.9 = INTEGER: 0
UCD-SNMP-MIB::dskPercentNode.10 = INTEGER: 0
UCD-SNMP-MIB::dskPercentNode.11 = INTEGER: 0
UCD-SNMP-MIB::dskPercentNode.12 = INTEGER: 3
UCD-SNMP-MIB::dskPercentNode.13 = INTEGER: 0
UCD-SNMP-MIB::dskPercentNode.14 = INTEGER: 4
UCD-SNMP-MIB::dskPercentNode.15 = INTEGER: 0
UCD-SNMP-MIB::dskPercentNode.16 = INTEGER: 1
UCD-SNMP-MIB::dskPercentNode.17 = INTEGER: 0
UCD-SNMP-MIB::dskPercentNode.18 = INTEGER: 0
UCD-SNMP-MIB::dskPercentNode.19 = INTEGER: 0
UCD-SNMP-MIB::dskPercentNode.20 = INTEGER: 0
UCD-SNMP-MIB::dskPercentNode.21 = INTEGER: 13
UCD-SNMP-MIB::dskPercentNode.22 = INTEGER: 8
UCD-SNMP-MIB::dskPercentNode.23 = INTEGER: 1
```

```
UCD-SNMP-MIB::dskPercentNode.24 = INTEGER: 0
UCD-SNMP-MIB::dskPercentNode.25 = INTEGER: 0
UCD-SNMP-MIB::dskPercentNode.26 = INTEGER: 0
UCD-SNMP-MIB::dskPercentNode.27 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.1 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.3 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.4 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.5 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.6 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.7 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.8 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.9 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.10 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.11 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.12 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.13 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.14 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.15 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.16 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.17 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.18 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.19 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.20 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.21 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.22 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.23 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.24 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.25 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.26 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.27 = INTEGER: 0
UCD-SNMP-MIB::dskErrorMsg.1 = STRING:
UCD-SNMP-MIB::dskErrorMsg.2 = STRING:
UCD-SNMP-MIB::dskErrorMsg.3 = STRING:
UCD-SNMP-MIB::dskErrorMsg.4 = STRING:
UCD-SNMP-MIB::dskErrorMsg.5 = STRING:
UCD-SNMP-MIB::dskErrorMsg.6 = STRING:
UCD-SNMP-MIB::dskErrorMsg.7 = STRING:
UCD-SNMP-MIB::dskErrorMsg.8 = STRING:
UCD-SNMP-MIB::dskErrorMsg.9 = STRING:
UCD-SNMP-MIB::dskErrorMsg.10 = STRING:
UCD-SNMP-MIB::dskErrorMsg.11 = STRING:
UCD-SNMP-MIB::dskErrorMsg.12 = STRING:
UCD-SNMP-MIB::dskErrorMsg.13 = STRING:
UCD-SNMP-MIB::dskErrorMsg.14 = STRING:
UCD-SNMP-MIB::dskErrorMsg.15 = STRING:
```

```
UCD-SNMP-MIB::dskErrorMsg.16 = STRING:
UCD-SNMP-MIB::dskErrorMsg.17 = STRING:
UCD-SNMP-MIB::dskErrorMsg.18 = STRING:
UCD-SNMP-MIB::dskErrorMsg.19 = STRING:
UCD-SNMP-MIB::dskErrorMsg.20 = STRING:
UCD-SNMP-MIB::dskErrorMsg.21 = STRING:
UCD-SNMP-MIB::dskErrorMsg.22 = STRING:
UCD-SNMP-MIB::dskErrorMsg.23 = STRING:
UCD-SNMP-MIB::dskErrorMsg.24 = STRING:
UCD-SNMP-MIB::dskErrorMsg.25 = STRING:
UCD-SNMP-MIB::dskErrorMsg.26 = STRING:
UCD-SNMP-MIB::dskErrorMsg.27 = STRING:
```

- Petición de información acerca del estado de la memoria SWAP del sistema en velvet

```
[root@aletia root]# snmpwalk -v 3 -u prueba2L -l authPriv -A prueba2L2007 -X prueba2L2007
192.168.10.6 .1.3.6.1.4.1.2021.4

UCD-SNMP-MIB::memIndex.0 = INTEGER: 0
UCD-SNMP-MIB::memErrorName.0 = STRING: swap
UCD-SNMP-MIB::memTotalSwap.0 = INTEGER: 29459512
UCD-SNMP-MIB::memAvailSwap.0 = INTEGER: 29459512
UCD-SNMP-MIB::memTotalReal.0 = INTEGER: 16777216
UCD-SNMP-MIB::memAvailReal.0 = INTEGER: 12114960
UCD-SNMP-MIB::memTotalFree.0 = INTEGER: 28788248
UCD-SNMP-MIB::memMinimumSwap.0 = INTEGER: 307200
UCD-SNMP-MIB::memSwapError.0 = INTEGER: 0
UCD-SNMP-MIB::memSwapErrorMsg.0 = STRING:
```

- Petición de datos acerca del promedio de carga del sistema en velvet

```
[root@aletia root]# snmpwalk -v 3 -u prueba2L -l authPriv -A prueba2L2007 -X prueba2L2007
192.168.10.6 .1.3.6.1.4.1.2021.10
```

```
UCD-SNMP-MIB::laIndex.1 = INTEGER: 1
UCD-SNMP-MIB::laIndex.2 = INTEGER: 2
UCD-SNMP-MIB::laIndex.3 = INTEGER: 3
UCD-SNMP-MIB::laNames.1 = STRING: Load-1
UCD-SNMP-MIB::laNames.2 = STRING: Load-5
UCD-SNMP-MIB::laNames.3 = STRING: Load-15
UCD-SNMP-MIB::laLoad.1 = STRING: 0.88
UCD-SNMP-MIB::laLoad.2 = STRING: 0.90
UCD-SNMP-MIB::laLoad.3 = STRING: 0.88
UCD-SNMP-MIB::laConfig.1 = STRING: 8.00
UCD-SNMP-MIB::laConfig.2 = STRING: 10.00
UCD-SNMP-MIB::laConfig.3 = STRING: 10.00
UCD-SNMP-MIB::laLoadInt.1 = INTEGER: 88
UCD-SNMP-MIB::laLoadInt.2 = INTEGER: 90
UCD-SNMP-MIB::laLoadInt.3 = INTEGER: 88
UCD-SNMP-MIB::laLoadFloat.1 = Opaque: Float: 0.882812
UCD-SNMP-MIB::laLoadFloat.2 = Opaque: Float: 0.902344
UCD-SNMP-MIB::laLoadFloat.3 = Opaque: Float: 0.882812
UCD-SNMP-MIB::laErrorFlag.1 = INTEGER: 0
UCD-SNMP-MIB::laErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::laErrorFlag.3 = INTEGER: 0
UCD-SNMP-MIB::laErrMessage.1 = STRING:
UCD-SNMP-MIB::laErrMessage.2 = STRING:
UCD-SNMP-MIB::laErrMessage.3 = STRING:
```

4.1.4.3.3.2 Pruebas en servidor4

- Petición de datos acerca de los procesos monitoreados en servidor4

```
[root@aletia root]# snmpwalk -v 3 -u prueba2L -l authPriv -A prueba2L2007 -X prueba2L2007
132.248.10.23 .1.3.6.1.4.1.2021.2

UCD-SNMP-MIB::prIndex.1 = INTEGER: 1
```

```
UCD-SNMP-MIB::prIndex.2 = INTEGER: 2
UCD-SNMP-MIB::prIndex.3 = INTEGER: 3
UCD-SNMP-MIB::prNames.1 = STRING: slapd
UCD-SNMP-MIB::prNames.2 = STRING: qmail-send
UCD-SNMP-MIB::prNames.3 = STRING: qmail-smtpd
UCD-SNMP-MIB::prMin.1 = INTEGER: 1
UCD-SNMP-MIB::prMin.2 = INTEGER: 1
UCD-SNMP-MIB::prMin.3 = INTEGER: 1
UCD-SNMP-MIB::prMax.1 = INTEGER: 10
UCD-SNMP-MIB::prMax.2 = INTEGER: 2
UCD-SNMP-MIB::prMax.3 = INTEGER: 2
UCD-SNMP-MIB::prCount.1 = INTEGER: 1
UCD-SNMP-MIB::prCount.2 = INTEGER: 1
UCD-SNMP-MIB::prCount.3 = INTEGER: 1
UCD-SNMP-MIB::prErrorFlag.1 = INTEGER: 0
UCD-SNMP-MIB::prErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::prErrorFlag.3 = INTEGER: 0
UCD-SNMP-MIB::prErrMessage.1 = STRING:
UCD-SNMP-MIB::prErrMessage.2 = STRING:
UCD-SNMP-MIB::prErrMessage.3 = STRING:
UCD-SNMP-MIB::prErrFix.1 = INTEGER: 0
UCD-SNMP-MIB::prErrFix.2 = INTEGER: 0
UCD-SNMP-MIB::prErrFix.3 = INTEGER: 0
UCD-SNMP-MIB::prErrFixCmd.1 = STRING:
UCD-SNMP-MIB::prErrFixCmd.2 = STRING:
UCD-SNMP-MIB::prErrFixCmd.3 = STRING:
```

- Petición de datos acerca de los sistemas de archivo en servidor4

```
[root@aletia root]# snmpwalk -v 3 -u prueba2L -l authPriv -A prueba2L2007 -X prueba2L2007 132.248.10.23
.1.3.6.1.4.1.2021.9

UCD-SNMP-MIB::dskIndex.1 = INTEGER: 1
UCD-SNMP-MIB::dskIndex.2 = INTEGER: 2
UCD-SNMP-MIB::dskIndex.3 = INTEGER: 3
```

```
UCD-SNMP-MIB::dskIndex.4 = INTEGER: 4
UCD-SNMP-MIB::dskIndex.5 = INTEGER: 5
UCD-SNMP-MIB::dskIndex.6 = INTEGER: 6
UCD-SNMP-MIB::dskIndex.7 = INTEGER: 7
UCD-SNMP-MIB::dskPath.1 = STRING: /
UCD-SNMP-MIB::dskPath.2 = STRING: /var
UCD-SNMP-MIB::dskPath.3 = STRING: /home1
UCD-SNMP-MIB::dskPath.4 = STRING: /home2
UCD-SNMP-MIB::dskPath.5 = STRING: /usr
UCD-SNMP-MIB::dskPath.6 = STRING: /boot
UCD-SNMP-MIB::dskPath.7 = STRING: /spare
UCD-SNMP-MIB::dskDevice.1 = STRING: /dev/md1
UCD-SNMP-MIB::dskDevice.2 = STRING: /dev/md2
UCD-SNMP-MIB::dskDevice.3 = STRING: /dev/md5
UCD-SNMP-MIB::dskDevice.4 = STRING: /dev/md6
UCD-SNMP-MIB::dskDevice.5 = STRING: /dev/md3
UCD-SNMP-MIB::dskDevice.6 = STRING: /dev/md0
UCD-SNMP-MIB::dskDevice.7 = STRING: /dev/hda8
UCD-SNMP-MIB::dskMinimum.1 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.2 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.3 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.4 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.5 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.6 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.7 = INTEGER: -1
UCD-SNMP-MIB::dskMinPercent.1 = INTEGER: 10
UCD-SNMP-MIB::dskMinPercent.2 = INTEGER: 10
UCD-SNMP-MIB::dskMinPercent.3 = INTEGER: 10
UCD-SNMP-MIB::dskMinPercent.4 = INTEGER: 10
UCD-SNMP-MIB::dskMinPercent.5 = INTEGER: 10
UCD-SNMP-MIB::dskMinPercent.6 = INTEGER: 10
UCD-SNMP-MIB::dskMinPercent.7 = INTEGER: 10
UCD-SNMP-MIB::dskTotal.1 = INTEGER: 4032000
UCD-SNMP-MIB::dskTotal.2 = INTEGER: 45358448
UCD-SNMP-MIB::dskTotal.3 = INTEGER: 70556976
UCD-SNMP-MIB::dskTotal.4 = INTEGER: 70565040
UCD-SNMP-MIB::dskTotal.5 = INTEGER: 6048196
UCD-SNMP-MIB::dskTotal.6 = INTEGER: 295449
UCD-SNMP-MIB::dskTotal.7 = INTEGER: 40203564
UCD-SNMP-MIB::dskAvail.1 = INTEGER: 3343260
UCD-SNMP-MIB::dskAvail.2 = INTEGER: 42938388
```

```
UCD-SNMP-MIB::dskAvail.3 = INTEGER: 59700488
UCD-SNMP-MIB::dskAvail.4 = INTEGER: 66944524
UCD-SNMP-MIB::dskAvail.5 = INTEGER: 3785288
UCD-SNMP-MIB::dskAvail.6 = INTEGER: 261018
UCD-SNMP-MIB::dskAvail.7 = INTEGER: 38128476
UCD-SNMP-MIB::dskUsed.1 = INTEGER: 483920
UCD-SNMP-MIB::dskUsed.2 = INTEGER: 115944
UCD-SNMP-MIB::dskUsed.3 = INTEGER: 7272392
UCD-SNMP-MIB::dskUsed.4 = INTEGER: 36020
UCD-SNMP-MIB::dskUsed.5 = INTEGER: 1955676
UCD-SNMP-MIB::dskUsed.6 = INTEGER: 19177
UCD-SNMP-MIB::dskUsed.7 = INTEGER: 32828
UCD-SNMP-MIB::dskPercent.1 = INTEGER: 13
UCD-SNMP-MIB::dskPercent.2 = INTEGER: 0
UCD-SNMP-MIB::dskPercent.3 = INTEGER: 11
UCD-SNMP-MIB::dskPercent.4 = INTEGER: 0
UCD-SNMP-MIB::dskPercent.5 = INTEGER: 34
UCD-SNMP-MIB::dskPercent.6 = INTEGER: 7
UCD-SNMP-MIB::dskPercent.7 = INTEGER: 0
UCD-SNMP-MIB::dskPercentNode.1 = INTEGER: 7
UCD-SNMP-MIB::dskPercentNode.2 = INTEGER: 0
UCD-SNMP-MIB::dskPercentNode.3 = INTEGER: 2
UCD-SNMP-MIB::dskPercentNode.4 = INTEGER: 0
UCD-SNMP-MIB::dskPercentNode.5 = INTEGER: 11
UCD-SNMP-MIB::dskPercentNode.6 = INTEGER: 0
UCD-SNMP-MIB::dskPercentNode.7 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.1 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.3 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.4 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.5 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.6 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.7 = INTEGER: 0
UCD-SNMP-MIB::dskErrorMsg.1 = STRING:
UCD-SNMP-MIB::dskErrorMsg.2 = STRING:
UCD-SNMP-MIB::dskErrorMsg.3 = STRING:
UCD-SNMP-MIB::dskErrorMsg.4 = STRING:
UCD-SNMP-MIB::dskErrorMsg.5 = STRING:
UCD-SNMP-MIB::dskErrorMsg.6 = STRING:
UCD-SNMP-MIB::dskErrorMsg.7 = STRING:
```

- Petición de información acerca del estado de la memoria SWAP del sistema en servidor4

```
[root@aletia root]# snmpwalk -v 3 -u prueba2L -l authPriv -A prueba2L2007 -X prueba2L2007
132.248.10.23 .1.3.6.1.4.1.2021.4

UCD-SNMP-MIB::memIndex.0 = INTEGER: 0
UCD-SNMP-MIB::memErrorName.0 = STRING: swap
UCD-SNMP-MIB::memTotalSwap.0 = INTEGER: 2048248
UCD-SNMP-MIB::memAvailSwap.0 = INTEGER: 2048248
UCD-SNMP-MIB::memTotalReal.0 = INTEGER: 2061092
UCD-SNMP-MIB::memAvailReal.0 = INTEGER: 20424
UCD-SNMP-MIB::memTotalFree.0 = INTEGER: 2068672
UCD-SNMP-MIB::memMinimumSwap.0 = INTEGER: 307200
UCD-SNMP-MIB::memShared.0 = INTEGER: 0
UCD-SNMP-MIB::memBuffer.0 = INTEGER: 246992
UCD-SNMP-MIB::memCached.0 = INTEGER: 1571384
UCD-SNMP-MIB::memSwapError.0 = INTEGER: 0
UCD-SNMP-MIB::memSwapErrorMsg.0 = STRING:
```

- Petición de datos acerca del promedio de carga del sistema en servidor4

```
[root@aletia root]# snmpwalk -v 3 -u prueba2L -l authPriv -A prueba2L2007 -X prueba2L2007
132.248.10.23 .1.3.6.1.4.1.2021.10

UCD-SNMP-MIB::laIndex.1 = INTEGER: 1
UCD-SNMP-MIB::laIndex.2 = INTEGER: 2
UCD-SNMP-MIB::laIndex.3 = INTEGER: 3
UCD-SNMP-MIB::laNames.1 = STRING: Load-1
UCD-SNMP-MIB::laNames.2 = STRING: Load-5
UCD-SNMP-MIB::laNames.3 = STRING: Load-15
UCD-SNMP-MIB::laLoad.1 = STRING: 0.14
UCD-SNMP-MIB::laLoad.2 = STRING: 0.16
UCD-SNMP-MIB::laLoad.3 = STRING: 0.17
```

```
UCD-SNMP-MIB::laConfig.1 = STRING: 9.00
UCD-SNMP-MIB::laConfig.2 = STRING: 9.00
UCD-SNMP-MIB::laConfig.3 = STRING: 9.00
UCD-SNMP-MIB::laLoadInt.1 = INTEGER: 14
UCD-SNMP-MIB::laLoadInt.2 = INTEGER: 16
UCD-SNMP-MIB::laLoadInt.3 = INTEGER: 17
UCD-SNMP-MIB::laLoadFloat.1 = Opaque: Float: 0.140000
UCD-SNMP-MIB::laLoadFloat.2 = Opaque: Float: 0.160000
UCD-SNMP-MIB::laLoadFloat.3 = Opaque: Float: 0.170000
UCD-SNMP-MIB::laErrorFlag.1 = INTEGER: 0
UCD-SNMP-MIB::laErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::laErrorFlag.3 = INTEGER: 0
UCD-SNMP-MIB::laErrMessage.1 = STRING:
UCD-SNMP-MIB::laErrMessage.2 = STRING:
UCD-SNMP-MIB::laErrMessage.3 = STRING:
```

4.1.4.3.4 Resultados

Las pruebas realizadas en esta segunda fase permitieron comprobar que la operabilidad y características ofrecidas por Net-SNMP son totalmente funcionales en la práctica. La implementación del protocolo facilitó, tanto en la primera y segunda fase de pruebas, la definición del monitoreo específico de procesos, sistemas de archivos, promedio de carga de sistema y espacio de memoria de intercambio (swap) en todos los equipos donde fue instalado. Debido a estas circunstancias se decidió que el protocolo SNMP fuera implementado en todos aquellos servidores en donde se tuviera la necesidad de monitorear y graficar los recursos y servicios brindados por el propio servidor.

En las subsecuentes instalaciones de SNMP, solo se habilitó la comunicación entre el gestor y los agentes, a través de la versión 3 de SNMP, por lo que solo fueron creados usuarios SNMPv3, descartándose la definición de comunidades. A pesar de que se contaba con las características de autenticación y cifrado de la versión 3 del

protocolo, la seguridad fue incrementada utilizando el firewall de cada servidor donde sea configurado un agente (figura 4.1). El firewall solo permite que el gestor (servidor de graficación) establezca comunicación con el agente SNMP a través del protocolo UDP por el puerto 161.

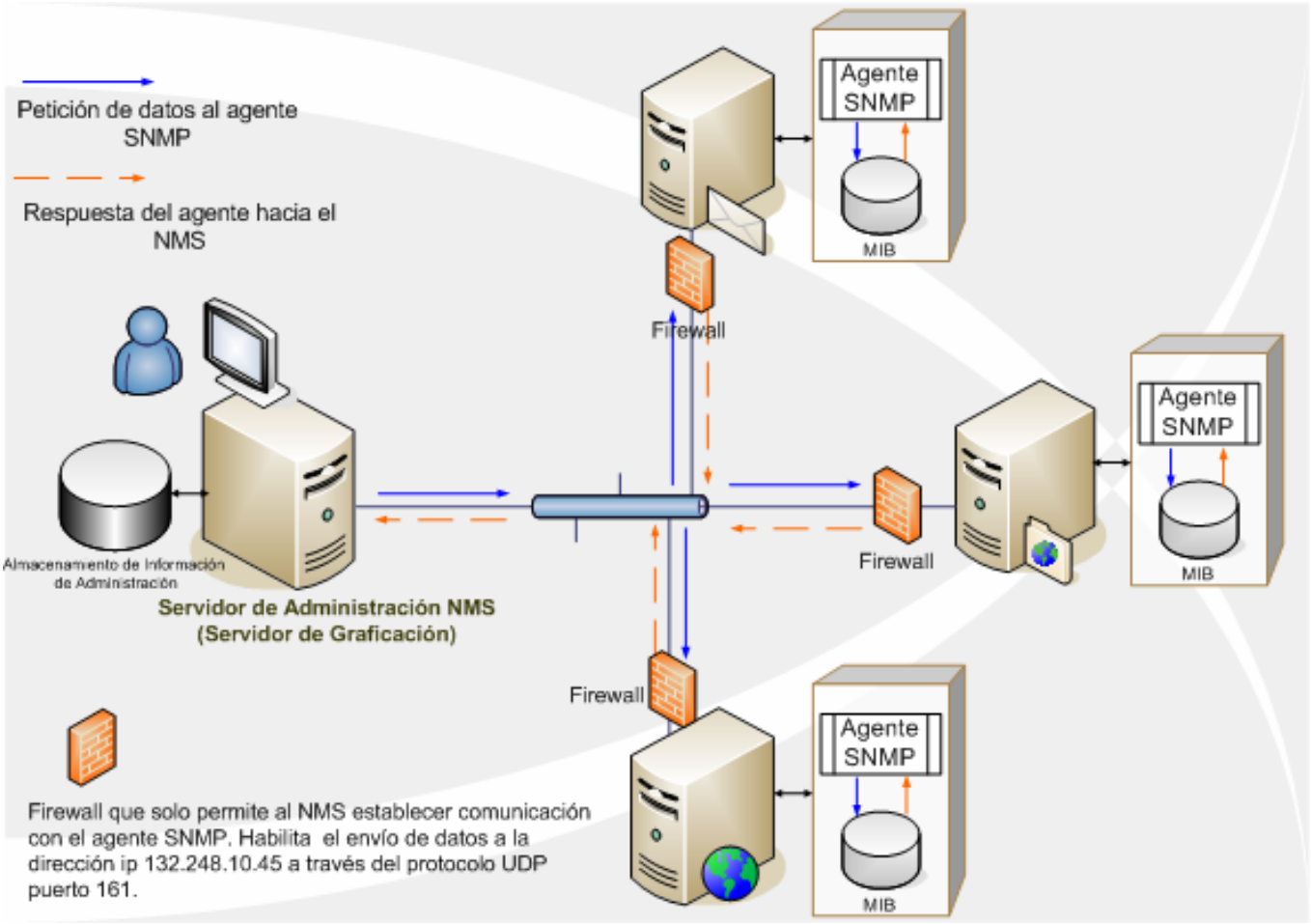


Figura 4.1 Esquema de monitoreo de agentes SNMP

4.2 Implementación del Servidor de Graficación

A través de este apartado se describirán los requerimientos necesarios para el servidor de graficación, así como los detalles de la instalación y configuración de todos los elementos que componen dicha implementación.

4.2.1 Requerimientos

4.2.1.1 Procesamiento y memoria

La documentación de Cacti no especifica un requerimiento mínimo de memoria RAM o de velocidad de procesamiento de CPU para que se lleve a cabo un óptimo rendimiento de la herramienta. Sin embargo, como se expuso en el capitulo 3, cacti incorpora el uso de varios elementos para efectuar sus tareas de graficación, por lo que el aspecto de procesamiento y cantidad de memoria RAM deben ser tomados en cuenta. Para efectos de nuestro trabajo de tesis se recomienda utilizar un equipo con por lo menos una velocidad de procesamiento superior a 500 Mhz y como mínimo 128 Megabytes de memoria RAM.

4.2.1.2 Sistema Operativo

Cacti es una solución de software libre desarrollada en SourceForge.net. Debido a esta situación la herramienta ha sido diseñada para instalarse sobre un sistema operativo Linux, por lo que se deberá implementar sobre uno de los siguientes sistemas operativos:

- Red Hat
- Mandrake

- Fedora
- Debian
- Otros sabores de Linux

4.2.1.3 Software requerido

Cacti es una herramienta terminal que involucra la interacción de otros elementos para realizar la creación de las gráficas, por lo que es necesario que el siguiente software se encuentre instalado:

- net-snmp.5.1.4
- RRDTool 1.0.49 or 1.2.x o superior
- MySQL 3.23 o superior, 4.0.20d o superior, altamente recomendado para características avanzadas.
- Apache, httpd-2.0.52 o superior
- PHP 4.1 o superior, 4.3.6 o superior altamente recomendado para características avanzadas.

4.2.2 Instalación y configuración

A través de este apartado se describirán los pasos que tuvieron que efectuarse para la instalación y configuración de todos los componentes necesarios para la solución de graficación. Para efectos del presente trabajo, todo el software que se instalará se alojará en una carpeta llamada "SRC" en el directorio /root del sistema.

4.2.2.1 Instalación de Mysql

Para ejecutar los pasos que describen la instalación será necesario estar como root del sistema.

- Ubicarse en el directorio donde se encuentre el fuente

```
cd /root/SRC
```

- Descomprimir y desempaquetar el fuente

```
tar -zxvf mysql-x.x.x.tar.gz
```

- Ubicarse en el directorio del fuente

```
cd mysql-x.x.x
```

- Configurar la compilación, de manera que se instale en el directorio "/usr/local"

```
./configure --prefix=/usr/local/mysql
```

- Compilar el fuente

```
make
```

- Si ha compilado exitosamente, realizar la instalación

```
make install
```

- Crear un grupo llamado “mysql” en el sistema operativo

```
groupadd mysql
```

- Crear el usuario “mysql” , de manera que pertenezca al grupo “mysql”

```
useradd -g mysql mysql
```

- Crear el diccionario de datos del manejador mediante el siguiente script

```
scripts/mysql_install_db
```

- Asegurarse que el súper - usuario (root) sea el dueño de la instalación de mysql

```
chown -R root /usr/local/mysql
```

- Cambiar a la propiedad de el usuario “mysql” el directorio de las bases de datos

```
chown -R mysql /usr/local/mysql/var
```

- Cambiar al grupo “mysql” el directorio de instalación

```
chgrp -R mysql /usr/local/mysql
```

- Asegurarse de borrar cualquier archivo de configuración existente de otra instalación de mysql

```
rm -f /etc/my.cnf
```

- Copiar el archivo de configuración que los archivos de instalación nos brinda

```
cp support-files/my-medium.cnf  /etc/my.cnf
```

- Agregar la ruta de las librerías

```
echo /usr/local/mysql/lib/mysql >> /etc/ld.so.conf
echo /usr/local/lib >> /etc/ld.so.conf
ldconfig –v
```

- Copiar el script de inicialización de mysql que la propia distribución nos brinda

```
cp support-files/mysql.server /etc/init.d/mysql
```

- Cambiar los permisos de manera que pueda ser ejecutado

```
chmod 750 /etc/init.d/mysql
```

- Levantar el demonio de mysql para permitir que el manejador comience a funcionar (también puede ser levantado como usuario mysql mediante el script "/etc/init.d/mysql" con el parámetro start)

```
/usr/local/mysql/bin/mysqld_safe --user=mysql &
```

- Configurar que cada vez que se reinicie la maquina se levante Mysql. Para ello crearemos ligas al script de inicio en los niveles de ejecución 3 y 5

```
cd /etc/rc3.d/
ln -s ../init.d/mysql S85mysql
ln -s ../init.d/mysql K85mysql
cd /etc/rc5.d/
ln -s ../init.d/mysql S85mysql
ln -s ../init.d/mysql K85mysql
```

4.2.2.2 Instalación del servidor web Apache

Para ejecutar los pasos que describen la instalación será necesario estar como root del sistema.

- Ubicarse en el directorio donde se ubica el fuente

```
cd /root/SRC
```

- Descomprimir y desempaquetar el fuente

```
tar -zxvf httpd-x.x.x.tar.gz
```

- Ubicarse dentro del directorio del fuente

```
cd httpd-x.x.x
```

- Configurar que sea instalado en la ruta “/usr/local/www”. También es necesario habilitar los objetos compartidos para posteriormente cargar a PHP como módulo de Apache

```
./configure --prefix=/www –enable-so
```

- Compilar el fuente

```
make
```

- Si ha compilado exitosamente iniciar la instalación

```
make install
```

- Ubicarse en el directorio de los binarios de Apache

```
cd /usr/local/www/bin
```

- Copiar el script de inicialización de Apache a la ruta “/etc/init.d”

```
cp apachectl  /etc/init.d/httpd
```

- Configurar que cada vez que se reinicie la maquina se levante Apache. Para ello crearemos ligas al script de inicio en los niveles de ejecución 3 y 5

```
cd /etc/rc3.d/
ln -s ../init.d/httpd S85httpd
ln -s ../init.d/httpd K85httpd
cd /etc/rc5.d/
ln -s ../init.d/httpd S85httpd
ln -s ../init.d/httpd K85httpd
```

- Mediante el script se procede a levantar el servidor web Apache

```
/etc/init.d/httpd start
```

4.2.2.3 Instalación de PHP

Para ejecutar los pasos que describen la instalación será necesario estar como root del sistema.

- Ubicarse en el directorio donde se ubica el fuente

```
cd /root/SRC
```

- Descomprimir y desempaquetar el fuente

```
tar -zxvf  php-x.x.x.tar.gz
```

- Ubicarse dentro del directorio del fuente

```
cd  php-x.x.x
```

- Configurar la instalación con las siguientes opciones

```
./configure --prefix=/usr/local/www/php --with-apxs2=/usr/local/www/bin/apxs --with-config-filepath=/usr/local /www/php --enable-sockets --with-mysql=/usr/local/mysql --with-zlibdir=/usr/include –with-gd
```

- Compilar el fuente

```
make
```

- Si la compilación ha sido exitosa será necesario detener el servidor Apache antes de iniciar la instalación.

```
/etc/init.d/httpd stop
```

- Llevar a cabo la instalación

```
make install
```

- Copiar el archivo de configuración que nos brinda la distribución a la siguiente ruta

```
cp php.ini-dist /usr/local/www/php/php.ini
```

- Agregar la siguientes líneas al archivo de configuración del servidor Apache

```
vi /usr/local/www/conf/httpd.conf
AddType application/x-tar .tgz
AddType application/x-httpd-php .php
AddType image/x-icon .ico
DirectoryIndex index.php index.html index.html.var
LoadModule phpX_module libexec/libphpX.so
```

Nota: La X deberá ser sustituida con la versión de PHP que se instale

- Volver a iniciar el servidor Apache

```
/etc/init.d/httpd start
```

4.2.2.4 Configuración de Mysql

No basta con instalar el manejador de base de datos Mysql, hay que crear la base de datos que la aplicación requiere para llevar a cabo su funcionamiento. Los pasos que se describen a continuación deberán ser ejecutados como super – usuario (root) del sistema.

- Crear un grupo de sistema llamado “cacti”

```
groupadd cacti
```

- Crear un usuario de sistema llamado “cactiuser”

```
useradd -g cacti cactiuser
```

- Conectarse a Mysql

```
/usr/local/mysql/bin/mysql
```

- Establecer una contraseña de administrador para Mysql

```
mysql> set password for root@localhost=password('rootpassword');
```

- Crear una base de datos llamada “cactidb”

```
mysql> create database cactidb;
```

- Conceder todos los permisos al usuario root sobre la base de datos “cactidb”

```
mysql> grant all on cactidb.* to root;
mysql> grant all on cactidb.* to root@localhost;
```

- Crear un usuario llamado “cactiuser” con todos los permisos sobre la base “cactidb”

```
mysql> grant all on cactidb.* to cactiuser;
mysql> grant all on cactidb.* to cactiuser@localhost;
```

- Establecer una contraseña al usuario “cactiuser”

```
mysql> set password for cactiuser@localhost=password('cactipw');
```

- Salir de mysql

```
mysql> exit
```

4.2.2.5 Instalación de RRDTool

Para ejecutar los pasos que describen la instalación será necesario estar como root del sistema.

- Ubicarse en el directorio donde se ubica el fuente

```
cd /root/SRC
```

- Descomprimir y desempaquetar el fuente

```
tar -zxvf  rrdtool-x.x.x.tar.gz
```

- Ubicarse dentro del directorio del fuente

```
cd  rrdtool-x.x.x
```

- Configurar la instalación para que sea instalado en la siguiente ruta del sistema

```
./configure --prefix=/usr/local/rrdtool
```

- Realizar la compilación

```
make
```

- Si la compilación ha sido exitosa, llevar a cabo la instalación

```
make install
```

4.2.2.6 Instalación de Net-SNMP

Para realizar la instalación de Net-SNMP remitirse al aparato 4.1 Implementación de los agentes SNMP en la parte de instalación sobre un sistema operativo Linux. Cabe mencionar que solo es necesario instalar sin llevar a cabo ningún tipo de configuración de comunicación o monitoreo del SNMP, debido a que Cacti solo requiere

de los programas del Net-SNMP para recabar los datos de los agentes SNMP. Sin embargo si se desea graficar y monitorear al mismo servidor de graficación, entonces si será necesaria la configuración descrita en el apartado 4.1.3 Configuración del agente SNMP.

4.2.2.7 Instalación de Cacti

Para completar la implementación de Cacti, será necesario seguir los siguientes pasos. Esta parte comprende la instalación del software de Cacti, herramienta que basa su funcionamiento en todo el software que se ha descrito en su instalación. Como en todos los casos anteriores, será necesario ejecutar los procesos como usuario administrador (root).

- Ubicarse en el directorio donde se ubica el fuente

```
cd /root/SRC
```

- Copiar el fuente a la carpeta de documentos de Apache

```
cp cacti-x.x.x.tar.gz  /usr/local/www/htdocs
```

- Ubicarse en el directorio de documentos de Apache

```
cd  /usr/local/www/htdocs
```

- Descomprimir y desempaquetar el fuente

```
tar -zxvf  cacti-x.x.x.tar.gz
```

- Renombrar la carpeta a solo “cacti”

```
mv  cacti-x.x.x  cacti
```

- Ubicarse dentro del directorio

```
cd  cacti
```

- Ejecutar un script con sentecias SQL para crear la estructura de la base de datos utilizada por cacti. Este script viene dentro del directorio fuente.

```
/usr/local/mysql/bin/mysql --user=root --password=rootpassword cactidb < cacti.sql
```

- Cambiar de propietario al usuario “cactiuser” los siguientes directorios

```
chown -R cactiuser rra/ log/
```

- Editar el archivo de configuración de manera que se encuentre la siguiente información

```
vi  /www/htdocs/cacti/include/config.php
$database_defaut = “cactidb”;
$database_hostname = “localhost”;
$database_username = “cactiuser”;
$database_password = “cactipw”;
```

- Es necesario establecer el periodo de tiempo en que cacti realizará la recopilación de datos para llevar a cabo las graficas. Se considera que un

muestreo de cada 5 minutos sería suficiente. Para configurar esta tarea es necesario editar el crontab del usuario "cactiuser" de la manera siguiente

```
su – cactiuser
crontab –e

*/5 * * * * /usr/local/www/php/bin/php /usr/local/www/htdocs/cacti/poller.php > /dev/null 2>&1
```

Nota: guardar los cambios para que el crontab ejecute el poller cada 5 minutos.

4.2.2.8 Configuración de Cacti

Una vez que se ha seguido el proceso de instalación y configuración de todo el software implicado en el funcionamiento de Cacti, estamos ante la posibilidad de realizar la configuración final de la interfaz. Para iniciar este proceso se requiere un navegador web para acceder a la herramienta Cacti, por lo que se pondrá cualquiera de estas 2 direcciones URL:

Desde el mismo equipo donde se encuentra instalado Cacti
http://localhost/cacti/

Desde un equipo remoto
http://direccion-IP/cacti/

Donde direccion-ip es la ip del servidor donde fue instalado Cacti.

La primera pantalla que se desplegará a través del navegador, será una serie de preguntas para llevar a cabo la configuración final. Básicamente serán 7 pasos que permitirán finalizar el proceso de configuración del servidor.

1. Dar clic en Next
2. Elegir la opción New Install (nueva instalación) y dar clic en Next
3. Proporcionar la ruta de los archivos ejecutables tanto de RRDTool y PHP

RRDTool=/usr/local/rrdtool/bin/rrdtool

PHP=/usr/local/www/php/bin/php

4. Dar click en Finish (Terminar)
5. Entrar al sistema Cacti con el siguiente usuario y contraseña

Login: admin

Password: admin.

6. Por seguridad se recomienda cambiar el password de admin.
7. Dar click en Save (guardar)

4.2.3 Creación de Graficas

Concluida la etapa de instalación y configuración de Cacti, es momento de involucrarse con el tema de creación de gráficas, por lo que a lo largo de esta sección se detallarán los pasos básicos que la interfaz requiere para llevar a cabo la creación de las gráficas.

4.2.3.1 Creación de un dispositivo

El primer paso para comenzar la creación de graficas, es decidir que dispositivos serán cuestionados para llevar a cabo la graficación de sus recursos. Los dispositivos pueden ser ruteadores, switches, impresoras, pero para propósitos de nuestro trabajo serán básicamente servidores.

La administración de los dispositivos en Cacti se encuentra en la opción Devices (dispositivos) en el menú de administración, el cual se encuentra en la parte izquierda de la interfaz (figura 4.3).

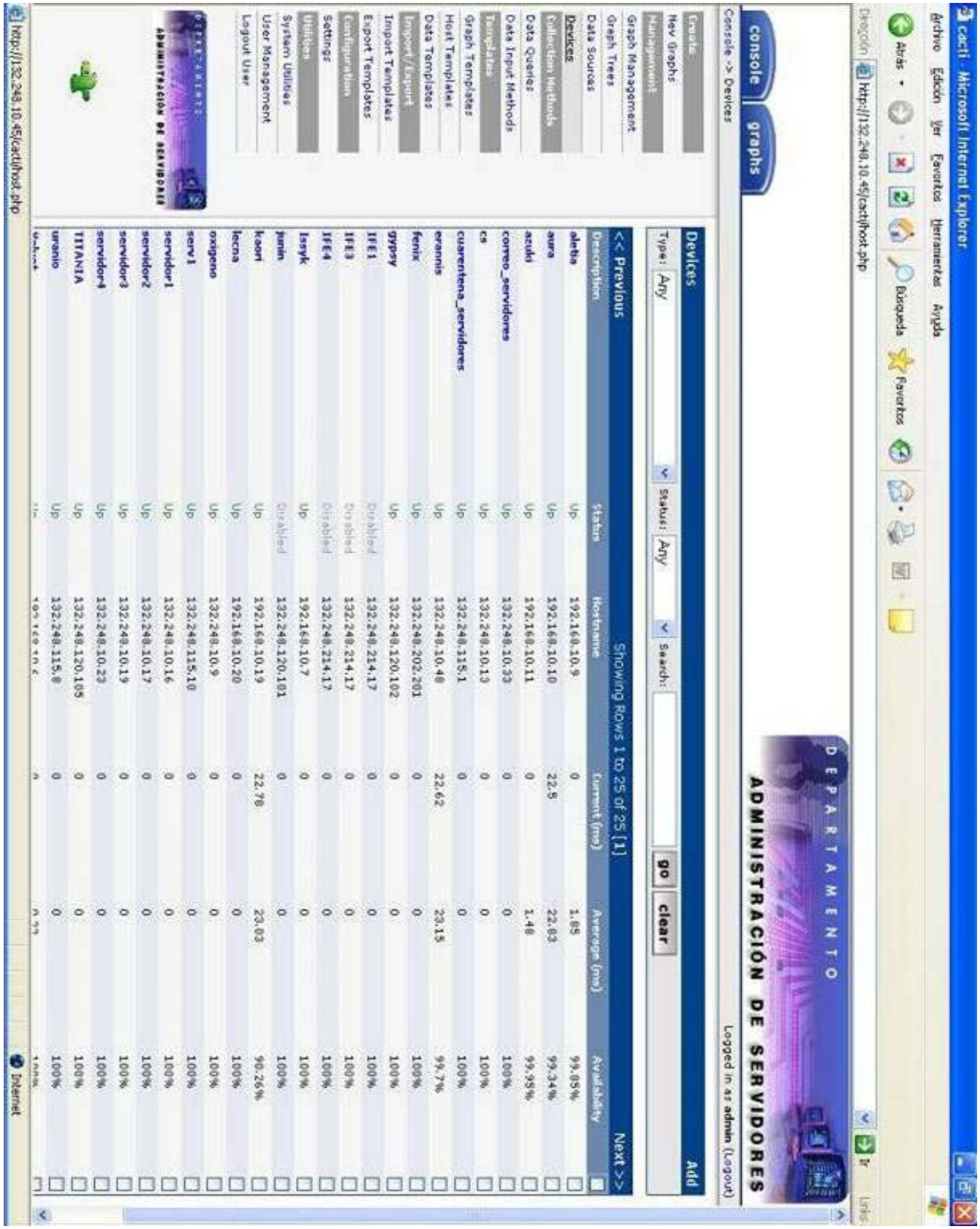


Figura 4.3 Menú de dispositivos

Una vez que se ha dado click en la opción se muestra una pantalla con información de los dispositivos que se han dado de alta, que en el caso de una primera instalación se encontrará vacía. Para agregar un dispositivo será necesario dar clic en la opción Add (agregar) , que se encuentra en la parte superior izquierda de la pantalla.

Una vez que se ha elegido la opción de agregar un nuevo dispositivo se mostrará una pantalla (figura 4.4) en donde se deben especificar detalles muy importantes acerca de un dispositivo, tal como el nombre de host o dirección IP, parámetros SNMP y tipo de host.

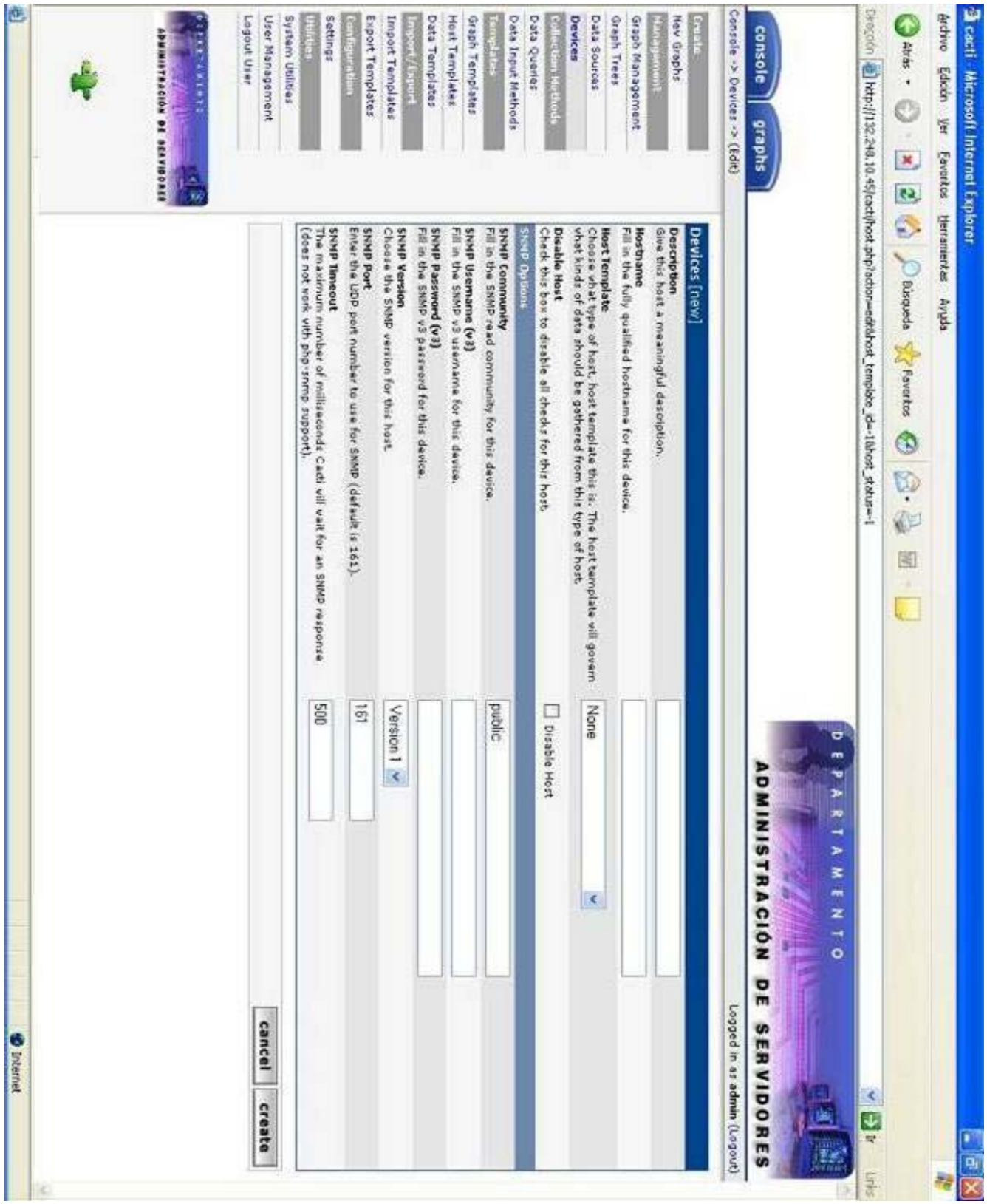


Figura 4.4 Alta de dispositivo

Los dos primeros campos deberán ser la descripción y el nombre de host o dirección IP del dispositivo que se desea graficar, por lo que se supone no habrá problema para llenar tal información. El tercer campo (Host Template) define que tipo de host se cuestionará, para efectos de nuestro trabajo la opción que deberá ser elegida es “ucd/net-SNMP Host” puesto que todos los servidores que serán graficados tendrán implementados el agente Net-SNMP, con todas las características descritas en el capitulo 2.

Las opciones SNMP tienen que ver con la manera en que Cacti puede establecer comunicación con el agente SNMP. La forma en que un gestor puede cuestionar a un agente SNMP fue descrita en el apartado 4.1.3 Configuración del Agente SNMP en la sección 4.1.3.1 Método de comunicación de este mismo capitulo. En estas opciones se puede especificar la comunidad SNMP o el usuario y contraseña de un usuario SNMPv3, el tipo de versión de SNMP así como el puerto de comunicación y el tiempo de espera una solicitud de datos. En el caso de esta implementación solo fueron usadas las opciones para el usuario SNMPv3, especificando el usuario y contraseña así como la versión 3 del protocolo. El puerto seguirá siendo 161 y el valor de 500 en el tiempo de espera no será modificado. Estos usuarios serán los que se crearon en cada agente SNMP de cada servidor, tal como fue realizado en la sección 4.1.4 Pruebas del apartado 4.1.3 Configuración del Agente SNMP.

Una vez que se han definido todos los datos necesarios para dar de alta un nuevo dispositivo se oprimirá el botón create (crear), tras el cual después de un momento de espera se deberán desplegar información del dispositivo en la parte superior de la pantalla. Esto datos hacen referencia al nombre de host, localización del sistema (sysLocation), dirección de contacto (sysContact) y otra información que fue proporcionada durante la instalación de ese agente SNMP. Es imprescindible que no se presente ningún error en la creación del dispositivo, ya que si esta situación se presenta, querrá decir que los datos proporcionados no fueron los correctos para comunicarse con el dispositivo y por lo tanto no podrán ser creadas las graficas correspondientes.

4.2.3.2 Creación de gráficas

Una vez que se han creado algunos dispositivos es posible la configuración de las gráficas. Cabe recordar que en el capitulo 3 en la sección de análisis de herramientas de graficación se había expuesto algunas características de Cacti, en las cuales se mencionó que la herramienta disponía de una serie de plantillas predeterminadas para la graficación de ciertos recursos. Estas plantillas recopilan los datos en base a los estándares de Net-SNMP por lo que son perfectamente compatibles con la forma en que los agentes SNMP almacenan la información de administración. Gracias a esta situación es posible crear fácilmente graficas para un dispositivo en aspectos como; uso de CPU, Promedio de carga, uso de memoria, usuarios en sistema, cantidad de procesos, tráfico de interfaces así como monitoreo de sistema fe ficheros.

Para comenzar el proceso de creación de graficas se deberá elegir la opción de New Graphs (nuevas gráficas) en el menú de administración (figura 4.5). Posteriormente se desplegará una pantalla que muestra las plantillas que Cacti proporciona para su asociación a cualquier dispositivo. Antes de elegir alguna plantilla es necesario especificar sobre que host se crearán dichas graficas, por lo que se deberá desplegar el menú Create new graphs for the following host (Crear graficas para el siguiente dispositivo) para seleccionar el host deseado. Una vez definido el host se muestran una serie de plantillas que pueden ser elegidas para ser asociadas al dispositivo, bastará con elegirlas y dar clic en botón create (crear).

Después de 5 0 6 minutos se podrán ver las graficas del dispositivo a través de la pestaña graphs en la parte superior izquierda de la pantalla.

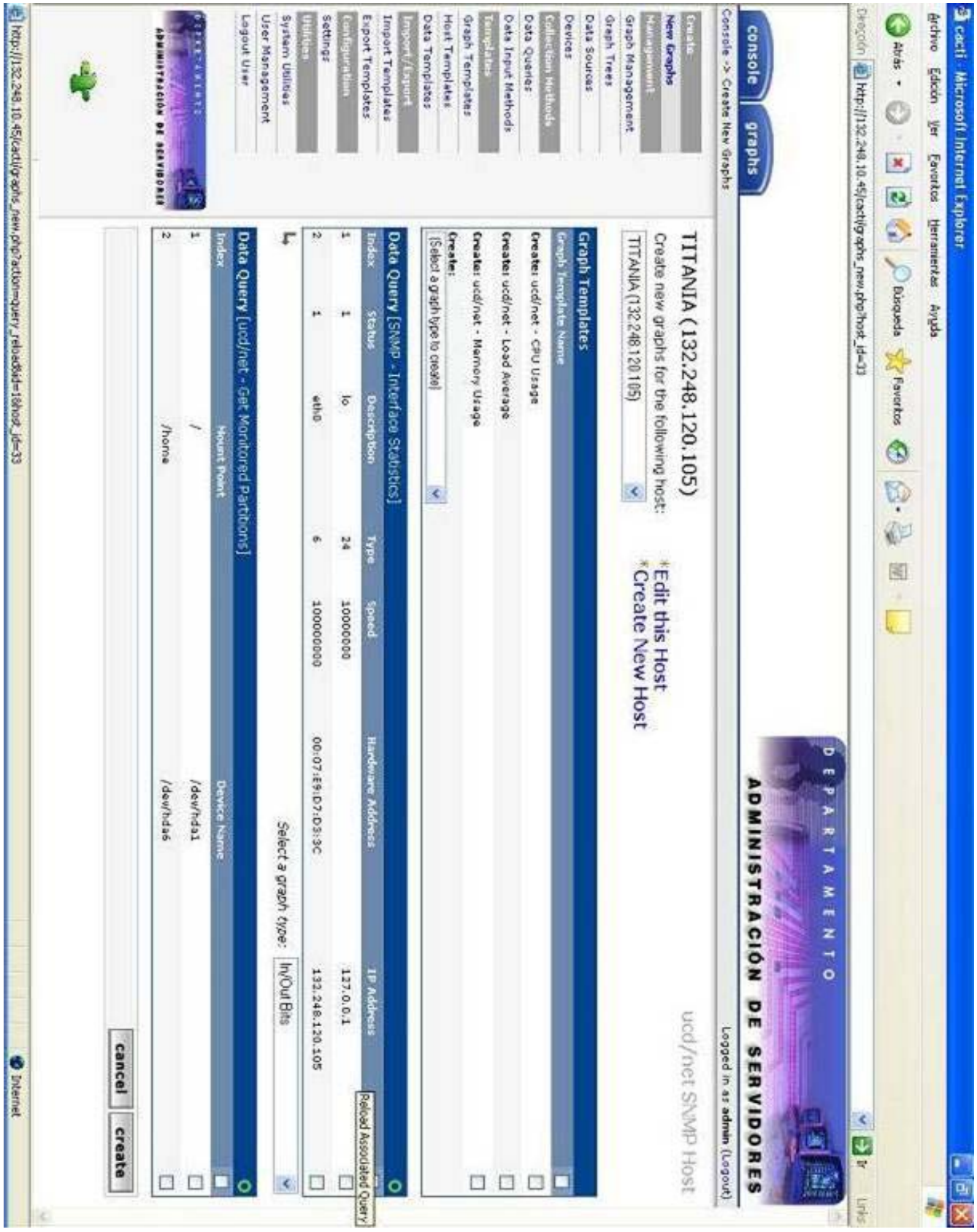


Figura 4.5 Creación de graficas

4.2.3.3 Vista de árbol

Las graficas que han sido creadas pueden ser vistas dando clic en la pestaña graphs, ubicada en la parte superior izquierda de la pantalla, donde se muestra un filtro mediante el cual se pueden mostrar las graficas de un determinado host o recurso. Sin embargo existe una manera más funcional de mostrar las graficas que han sido creadas en Cacti, esto es posible gracias a que la herramienta permite desplegar las gráficas en forma de árbol jerárquico, de manera que pueden ser organizadas de acuerdo a las necesidades que se presenten.

Para crear un árbol se debe dar clic en la opción Graph Trees (árbol de graficas) en el menú de administración (figura 4.6). Posteriormente se deberá clic en la opción Add (agregar) para agregar un nuevo árbol. Se mostrará una pantalla que pedirá el nombre para identificar el árbol de graficas, en nuestro caso se elegirá el nombre del servidor a crear (figura 4.7). Una vez que se ha proporcionado el nombre del árbol se deberá dar clic en el botón create (crear) tras lo cual se desplegará una pantalla que permitirá crear mas nodos sobre el nuevo árbol.

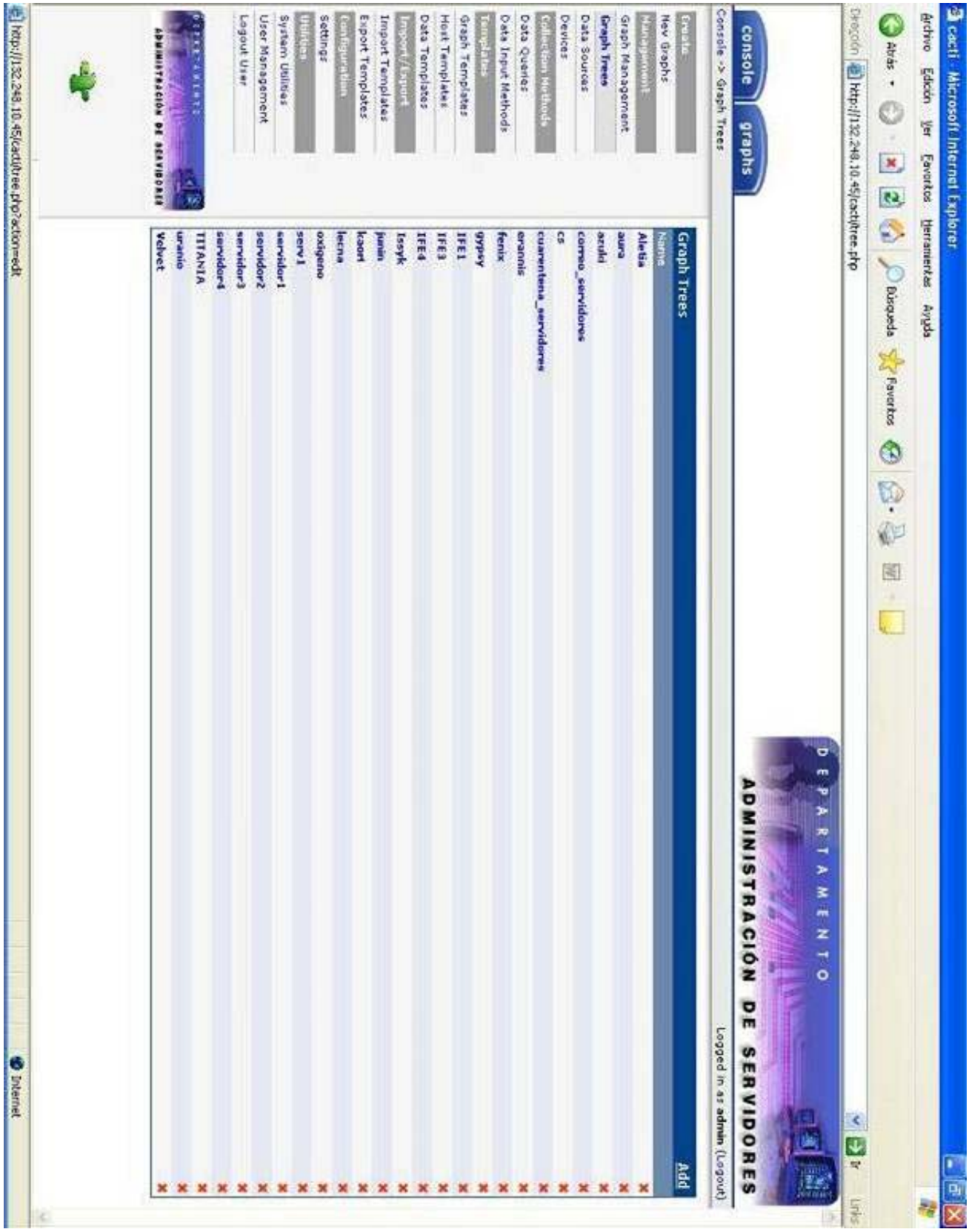


Figura 4.6 Menú para vistas de árbol

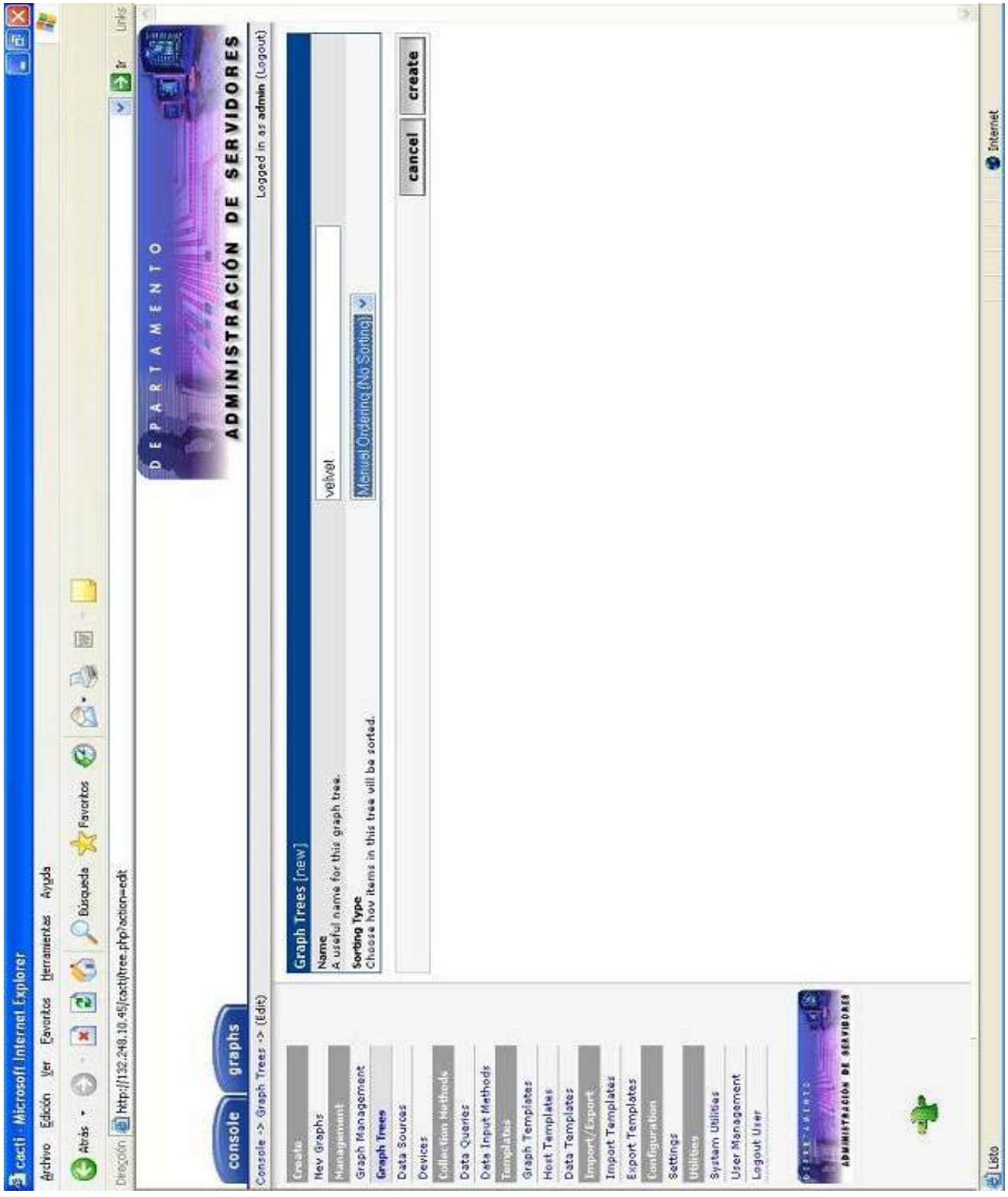


Figura 4.7 Creación de un árbol

Para comenzar a agregar elementos al árbol se deberá dar clic en la opción Add (agregar) del recuadro Tree Ítems (elementos del árbol). Existen 3 tipos de elementos que pueden agregarse en un árbol: header (cabecera), graph (gráfico), o host (dispositivo). Esta situación se puede apreciar en la figura 4.8.

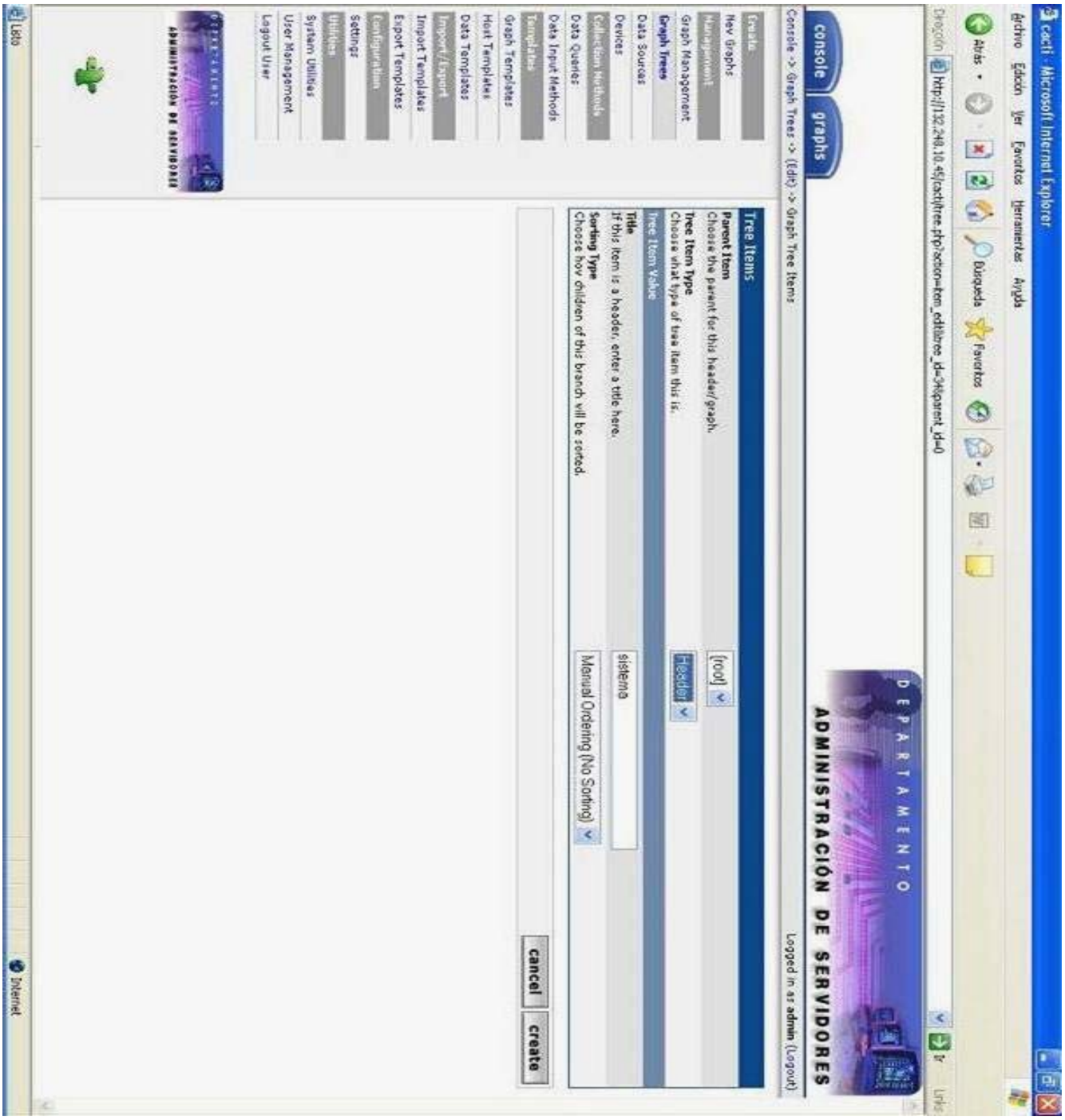


Figura 4.8 Tipos de elementos para un árbol

4.2.4 Pruebas

4.2.4.1 Planeación

Debido a la complejidad de la instalación y configuración de todos los elementos que incorpora la solución, se optó que antes de iniciar la implementación fuera determinado el equipo donde definitivamente se instalaría la herramienta de graficación. Esta tarea tuvo como objetivo llevar a cabo la instalación y configuración de Cacti solo una vez, descartando la posibilidad de tener fases de prueba.

La instalación de todos los elementos que conforman la herramienta de graficación fue exactamente como se describió en la sección 4.2.2 Instalación y configuración del apartado 4.2 Implementación del Servidor de Graficación. En la siguiente tabla se muestran las versiones de software que fueron instaladas:

Software	Versión
Mysql	mysql-5.0.21.tar.gz
Apache	httpd-1.3.35.tar.gz
Php	php-4.4.2.tar.gz
RRDtool	rrdtool-1.2.11.tar.gz
Net-SNMP	net-snmp-5.1.4.tar.gz

Versiones de Software para la herramienta de graficación

Una vez que se concluyó la instalación se definió que el servidor velvet sería el primer host sobre el cual se generarían las primeras graficas de prueba. A continuación solo se describirán las características del servidor donde fue instalado Cacti y de forma breve los pasos que se siguieron para llevar a cabo las gráficas.

4.2.4.2 Descripción del equipo

El equipo utilizado como servidor de graficación supera con mucho los requerimientos de todos los elementos que incorpora Cacti. En la siguiente tabla se resumen las características de este equipo.

Nombre	***Sistema Operativo***	***Hardware***	***Dirección IP***
aletia	Red Hat Enterprise Linux 3	Sun FIRE V20z con 2 procesadores AMD Opteron, 4 Gb de memoria RAM y 2 discos SCSI (Ultra 320) de 200 Gb cada uno.	132.248.10.45

Tabla de las Características del Servidor de Graficación

4.2.4.3 Primeras gráficas

En esta sección serán descritos de forma muy breve los pasos que se siguieron para la creación de gráficas en el servidor velvet. Si se tiene alguna duda acerca de algún procedimiento visto en esta sección remitirse al apartado 4.2.3 Creación de Graficas de este mismo capítulo, donde fueron descritos más a detalle.

- Crear el dispositivo para el host velvet eligiendo la opción Devices en el menú de administración. Posteriormente desde la pantalla de Devices dar clic en la opción Add donde se desplegará una pantalla donde se deberán introducir los datos para el alta del dispositivo (figura 4.9).

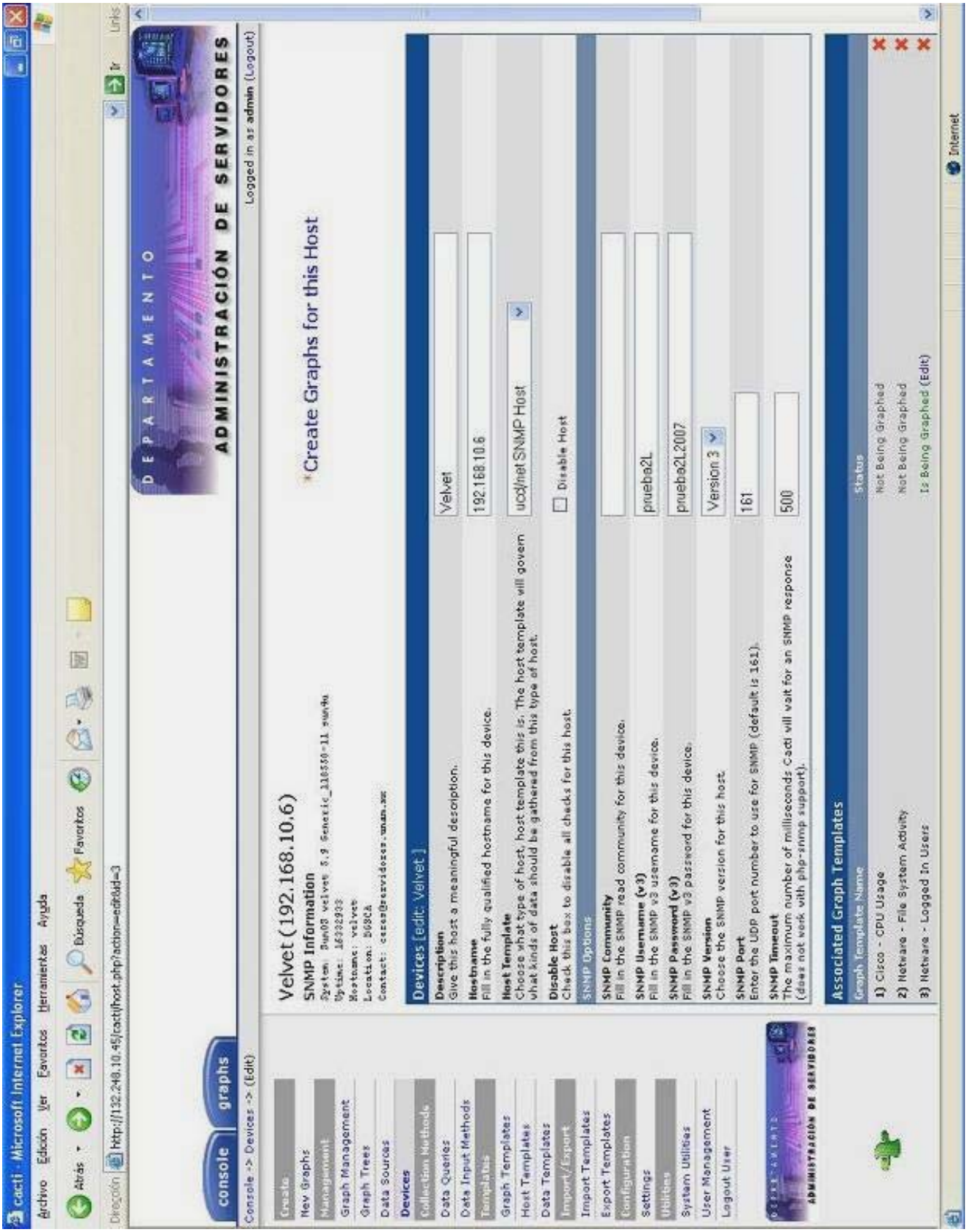


Figura 4.9 Alta del dispositivo velvet

- Para comenzar la creación de graficas se debe elegir la opción New Graphs en el menú de administración. Una vez que se haya desplegado la pantalla para la generación de graficas, se debe elegir de la lista Create new graphs for the following host, el servidor sobre el cual se quieren generar las gráficas.

- Sobre la pantalla de creación de graficas, se pueden observar 3 secciones de diferentes tipos de plantillas (figura 4.10). En el caso del servidor velvet se decidió generar gráficas de Logged In Users (usuarios conectados al sistema), CPU Usage (uso de procesador), Load Average (promedio de carga de sistema) y Memory Usage (uso de memoria) del recuadro Graph Templates (Plantillas de grafica). También se crearon graficas para el monitoreo de todos los sistemas de fichero del servidor eligiendo todas las particiones mostradas en el recuadro Data Query ucd/net – Get Monitored Partitions. Es importante mencionar que para poder generar graficas de los sistemas de fichero es necesario configurar el monitoreo de las particiones mediante la directiva “disk” descrita en la sección 4.1.3.2 Directivas de monitoreo del apartado 4.1.3 Configuración del agente SNMP (para poder desplegar las particiones desde cacti se debe dar clic en el circulo verde ubicado en la parte derecha del recuadro Data Query). Después de haberse creado las graficas, las plantillas utilizadas se mostrarán en una tonalidad gris (figura 4.10).

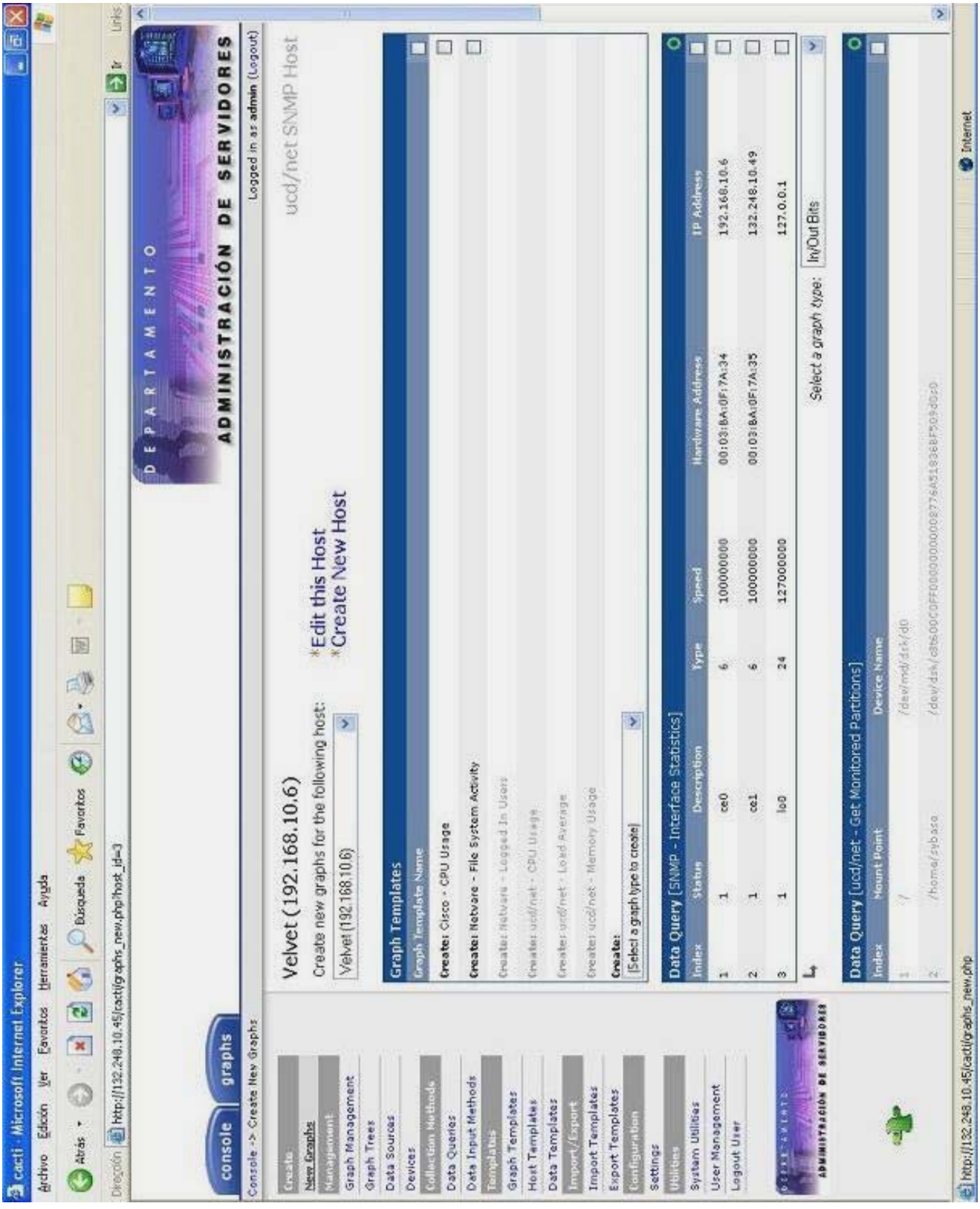


Figura 4.10 Plantillas para la generación de graficas

- Se dispuso que las graficas fueran vistas en disposición de árbol jerárquico, por lo que cada servidor seria representado por un nuevo árbol. Dentro de cada árbol se creó un elemento de tipo Header (cabecera) llamado Sistema dentro del cual se crearon otras dos cabeceras llamadas File System y Recursos para organizar y distinguir las graficas creadas.

4.2.4.3.1 Visualización de gráficas

A través de esta sección se mostrarán las gráficas que fueron creadas en la anterior sección.

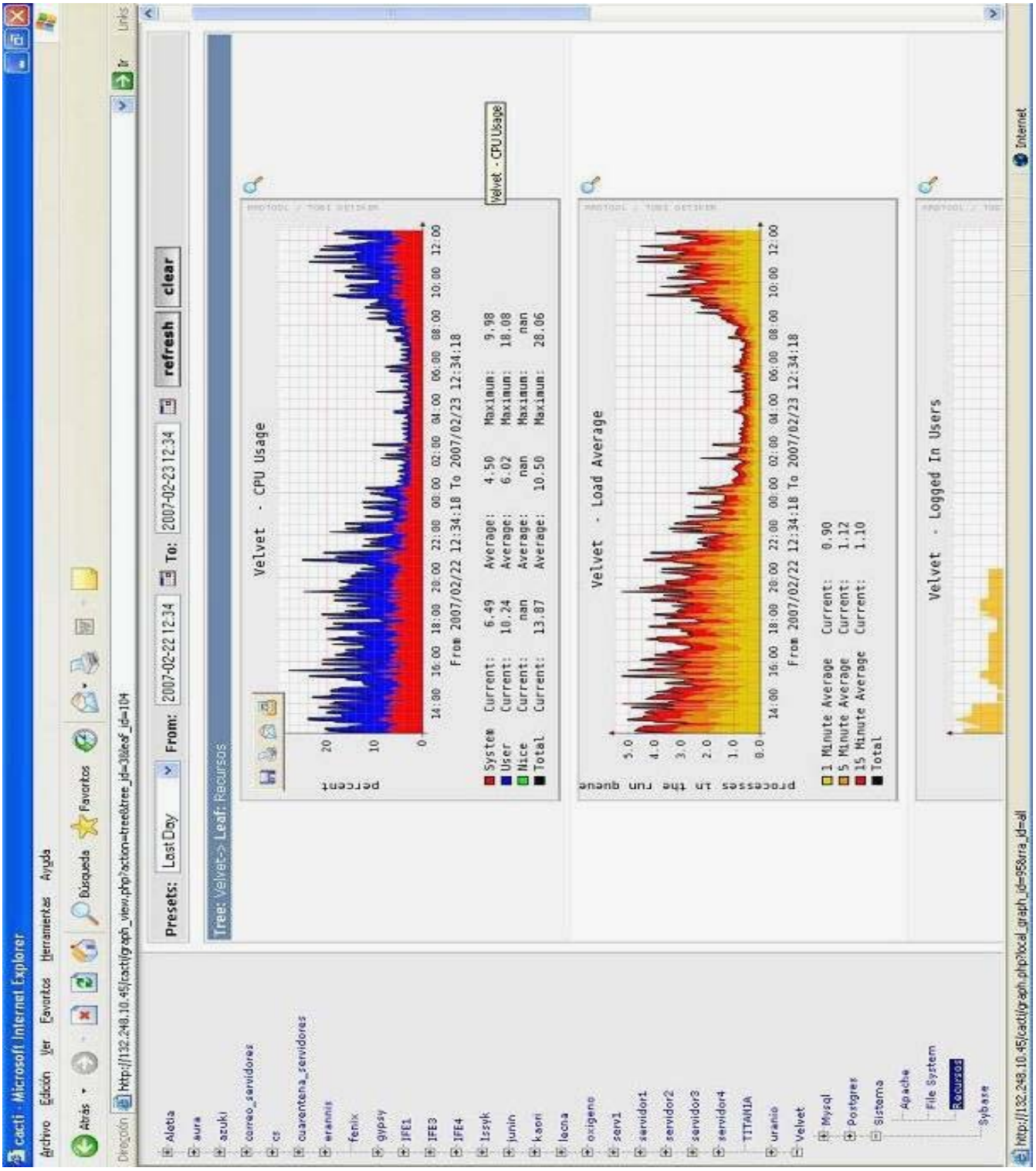


Figura 4.11 Graficas de los recursos del sistema (CPU, Carga, Usuarios conectados, etc).

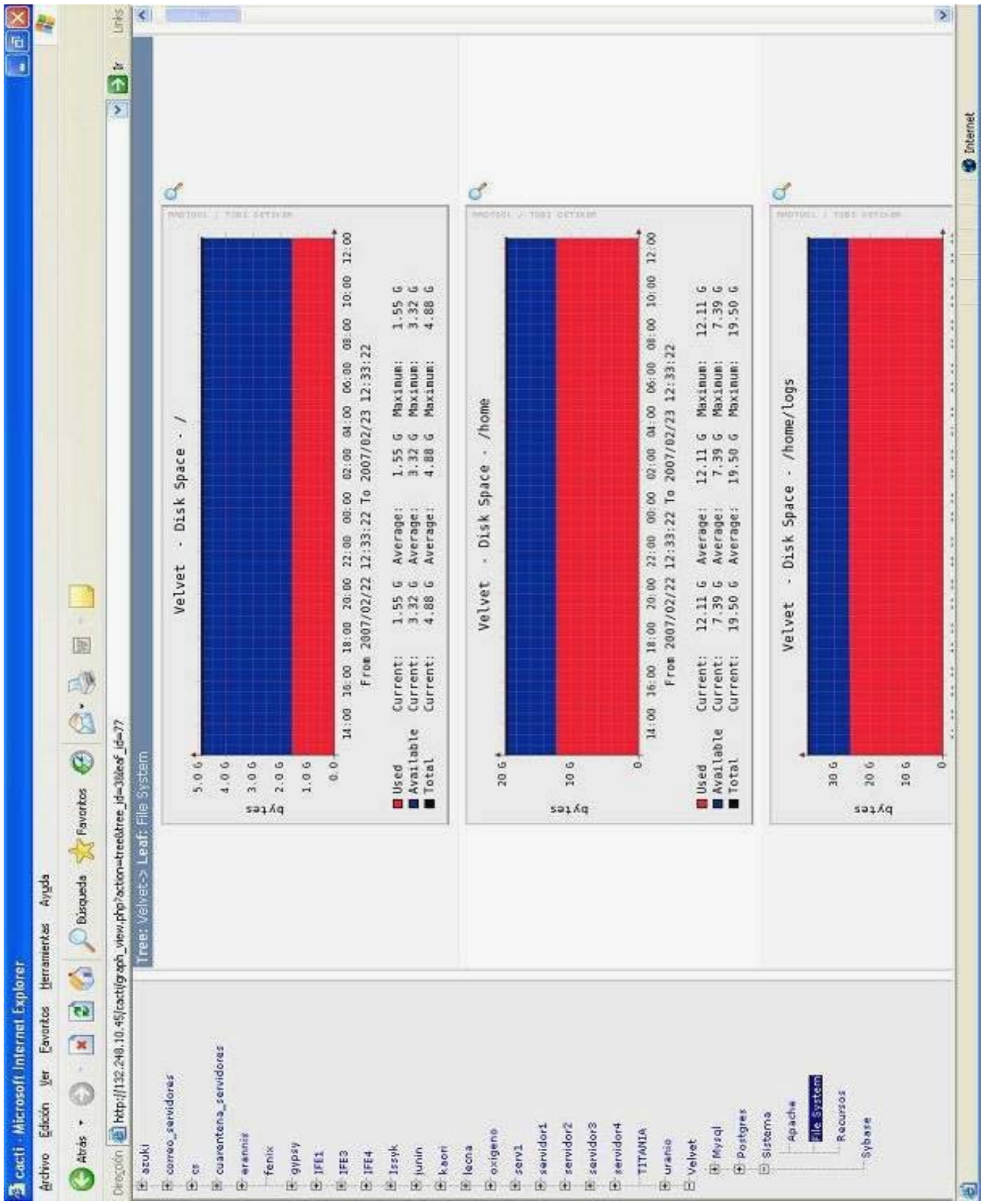


Figura 4.12 Graficas de los sistemas de fichero

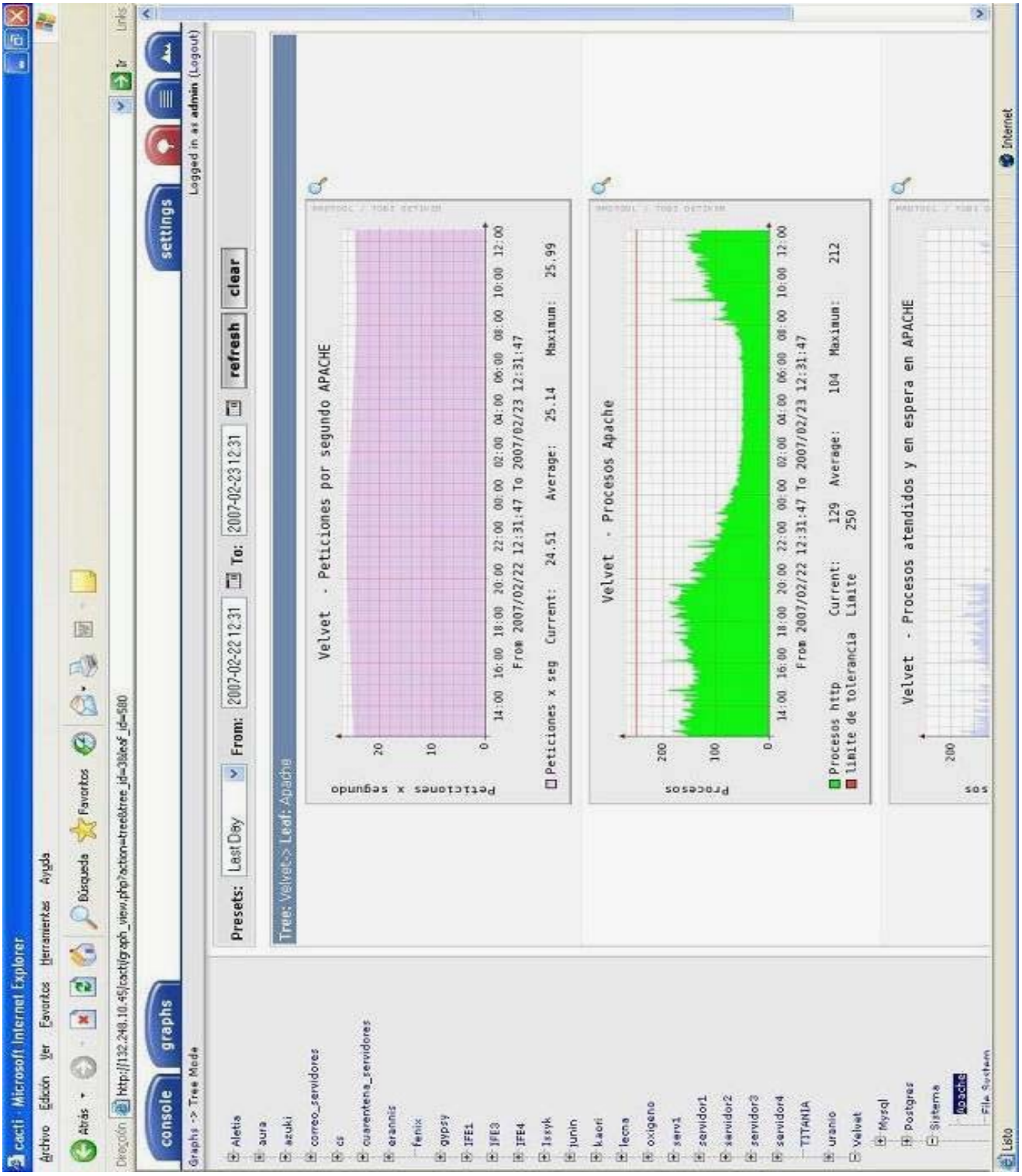


Figura 4.13 Graficas del comportamiento del servidor web Apache

Las gráficas de la figura 4.13 ejemplifican la capacidad de SNMP para ejecutar un programa externo mediante el cual se pueden recuperar distintos tipos de datos. En este caso se recupera información que nos muestra diversos aspectos del servidor web Apache. También se puede apreciar que Cacti permite generar nuestras propias graficas y no solo generar gráficos en base a sus propias plantillas.

4.2.4.4 Resultados

Las pruebas realizadas permitieron concluir que la herramienta de graficación es totalmente compatible con la distribución Net-SNMP, lo que posibilita fácilmente la creación de graficas en diversos aspectos de un servidor que incorpora un agente SNMP. Así mismo se comprobó que pueden ser creadas plantillas personalizadas para crear gráficos acerca de recursos y servicios muy específicos en los servidores. Otra de las características es que se tiene un esquema completo de administración de usuarios permitiendo que cualquier cantidad de usuarios con diversos perfiles de acceso puedan consultar la información gráfica acerca de los recursos y servicios ofrecidos por el departamento.

De esta manera se finalizó exitosamente el diseño de graficación que incluyó las etapas de implementación de los agentes SNMP e implementación del servidor de graficación.

4.3 Implementación del Servidor de Alertas

Esta es la última etapa de desarrollo y pruebas que será descrita a través de este capitulo, con el cual se concluye el proceso global de la implementación de graficación y monitoreo de esta propuesta de tesis. A través de este apartado se describirán los

requerimientos, instalación y configuración necesarios para llevar a cabo la recepción y tratamiento de las alertas SNMP.

4.3.1 Requerimientos

Los requerimientos en los aspectos de procesamiento y memoria así como de sistema operativo son los mismos que los descritos en el apartado 4.1 Implementación de los agentes SNMP, debido a que el servidor de alertas que se utilizará es proporcionado por la misma distribución Net- SNMP.

4.3.1.1 Software requerido

Cabe recordar que en el capitulo 3 se mencionó durante el aparatado 3.3 Diseño del esquema de alertas en la sección 3.3.2 Requerimiento adicional en la implementación, que además de utilizar al servidor de alertas era necesario la utilización de otro software que fuera capaz de traducir dichas alertas y acelerar el proceso de comunicación con la herramienta de monitoreo MON, por lo que la lista de software requerido para completar exitosamente la implementación del servidor de traps será el siguiente:

- net-snmp.5.1.4
- SNMP Trap Translator v1.1
- Perl 5.6.1 o superior con los siguientes módulos:
    1. Text::ParseWords
    2. Text::ParseWords
    3. Posix
    4. Config::IniFiles
    5. Time::HiRes

4.3.2 Instalación

La instalación será dividida en dos partes, una que solo trate la referente al servidor de alertas Net-SNMP y la otra al servidor de traducción SNMPTT. Para llevar a cabo ambos procesos será necesario ejecutar los procesos como usuario administrador (root) del sistema.

4.3.2.1 Instalación del servidor de traps

La instalación para poner en marcha al servidor de traps es muy similar a la expuesta en el apartado de implementación de los agentes SNMP, por lo que queda ampliamente recomendado remitirse a dicha sección para llevar a cabo este proceso. A continuación solo se describen brevemente los pasos para llevar a cabo la instalación de Net-SNMP sobre un sistema operativo Linux ya que esta será la propuesta manejada en la etapa de pruebas al final de este apartado.

Una vez obtenido el código fuente del Net-SNMP es necesario descomprimirlo y desempaquetarlo. Esta tarea puede realizarse sobre cualquier ruta del sistema, pero para efectos de esta instalación se creará una carpeta llamada "SRC" en el directorio /root del sistema.

Desempaquetar el paquete de la distribución

```
[root@gypsy root]#  mkdir SRC
[root@gypsy root]#  mv net-snmp-5.1.4.tar.gz ./SRC
[root@gypsy root]#  cd  SRC
[root@gypsy SRC]# tar -zxvf  net-snmp-5.1.4.tar.gz
```

Configuración

Una vez que se ha descomprimido y desempaquetado la distribución hay que configurar las características con las que se desea compilar para su posterior instalación.

Primero se debe ubicar en el directorio donde se haya los programas de configuración e instalación.

```
[root@gypsy SRC]#cd  net-snmp-5.1.4
```

En el caso de nuestra implementación la configuración que se eligió fue la siguiente:

- No se eligió ninguna ruta en particular para la instalación, por lo que el directorio de instalación por defecto fue en "/usr/local/share/snmp/". Los programas ejecutables asi como las librerías fueron instaladas por defecto en las rutas "/usr/local/bin" y "/usr/local/lib" respectivamente.

Es opcional si se desea que el servidor de alertas tenga capacidad de auto monitoreo, por lo que si se desea tener esta capacidad se debe habilitar el modulo "disman/event-mib".

Para indicar la anterior configuración se deberá ejecutar el programa de la siguiente manera:

```
[root@gypsy net-snmp-5.1.4]# ./configure
```

Una vez que se ha iniciado el proceso de configuración se requiere que se introduzcan una serie de datos para que el agente sea configurado correctamente:

1.Default version of SNMP to use (3): 3. Hace referencia a la versión del protocolo a instalar por defecto, en nuestro se trabajará con la versión 3.

2.System Contact Information (root@): aaguilera@servidores.unam.mx . Es un dato que brinda la dirección de correo electrónico de la persona encargada de administrar ese agente SNMP.

3.System Location (Unknown): UNAM Depto. De Servidores. Información que da a conocer la localización del dispositivo.

4.Location to write logfile (/var/log/snmpd.log): /var/log/snmpd.log . Se refiere a la bitacora donde SNMP alojará mensajes de error e información acerca de su funcionamiento.

5.Location to write persistent information (/var/net-snmp): /var/net-snmp . Ruta del directorio para las librerias SNMP, donde se almacenará información acerca de la configuración del protocolo.

Iniciar el proceso de compilación

Para empezar a compilar el fuente teclear el siguiente comando:

```
[root@gypsy net-snmp-5.1.4]#  make
```

Se deberá observar si existen errores de compilación. Es posible que se reciban mensajes de advertencia, sin embargo esto es un comportamiento normal.

Instalación

Para realizar la instalación se teclea el siguiente comando:

```
[root@gypsy net-snmp-5.1.4]# make install
```

4.3.2.2 Instalación del servidor SNMPTT

Una vez que se ha instalado el Net-SNMP se tendrán que seguir los siguientes pasos para la instalación de SNMPTT.

- Instalar perl superior a la versión 5.6.8(recomendado instalar la versión 5.8.0 o superior). Es muy posible que en un sistema operativo Linux este software ya se encuentre instalado.

- Instalar el modulo de perl del agente SNMP

```
$cd /root/SRC/net-snmp.5.4.1
$cd perl
$perl Makefile.PL
$make
$make install
```

- Proceder a instalar la siguiente lista de módulos de perl (para efectos de esta guía los módulos serán instalados mediante el programa “cpan”)

```
$perl –MCPAN –e shell

cpan>install Net-SNMP
cpan>install Text::ParseWords
cpan>install Getopt::Long
cpan>install Config::IniFiles
cpan>install Time::HiRes
```

- Descomprimir y extraer el contenido de los archivos de snmptt. Posteriormente ubicarse dentro del directorio.

```
$ cd /root/SRC
$tar –zxvf net-snmptt1.1.tar.gz
$cd net-snmptt1.1
```

- Copiar los siguientes programas a la ruta "/usr/sbin" y garantizar permisos de ejecución.

```
$cp snmptt snmptthandler /usr/sbin
$chmod ug+x  /usr/sbin/snmptt*
```

- Copiar el archivo "snmptt.ini" a la ruta "/etc/snmp". Este archivo establece la configuración bajo la que trabajará SNMPTT, por lo que será explicado más a detalle en el siguiente apartado.

```
$cp snmptt.ini /etc/snmp
```

Si no existe el directorio, crearlo (mkdir /etc/snmp).

- Crear el directorio de encolamiento (spool directory) así como los archivos para registrar los logs.

```
$mkdir /var/spool/snmptt
$touch /var/log/snmptt.log
$touch /var/log/snmpttunknown.log
```

- Copiar el archivo “snmptt.conf” a la ruta “/etc/snmp**”.** Este archivo contiene la lista de las traps definidas para su identificación y procesamiento.

```
$cp snmptt..conf /etc/snmp
```

4.3.3 Configuración

Una vez que se han seguido los pasos para la instalación, es necesario llevar a cabo la configuración correcta para poner a funcionar al servidor de traps junto con SNMPTT.

4.3.3.1 Configuración del servidor de alertas

Para poner a funcionar el servidor de alertas simplemente se necesita ejecutar un programa llamado “snmptrapd” con ciertos argumentos que son explicados a continuación.

snmptrapd -c *archivo_configuración* –o *bitácora_de_traps* -On

-c

Hace referencia al archivo de configuración de snmptrapd. La información contenida en el archivo determina el comportamiento del servidor de alertas ante las traps recibidas.

-o

Determina el archivo donde se registrarán las traps recibidas por el servidor.

-On

Este argumento determinará que el servidor no traducirá a su forma simbólica las alertas, si no que serán manejadas en forma numérica. Existen otros argumentos para afectar la forma en que el servidor despliega la información de las traps recibidas, sin embargo para nuestro trabajo será necesario que las alertas sean manejadas por el número de su OID.

Conforme a lo anterior será necesario realizar los siguientes pasos para poner en funcionamiento el servidor de traps Net-SNMP:

- Crear el archivo de configuración de "snmptrapd", asegurando que tenga permisos de lectura.

```
$touch /usr/local/share/snmp/snmptrapd.conf
$chmod ug+r  /usr/local/share/snmp/snmptrapd.conf
```

- Configurar que todas las traps que sean recibidas por el servidor sean reenviadas hacia el manejador de alertas de SNMPTT. Este manejador las escribirá en el directorio de encolamiento para su posterior tratamiento.

```
$vi  /usr/local/share/snmp/snmptrapd.conf
traphandle default /usr/sbin/snmptthandler
```

- Ejecutar el demonio snmptrapd para que el servidor de alertas empiece a recibir las alertas entrantes.

```
$snmptrapd –c /usr/local/share/snmp/snmptrapd.conf  -o /root/snmp/traps.log –On
```

Se determinó la ruta “/root/snmp/” para guardar las traps en una bitácora llamada traps.log.

Hasta este momento solo se tiene funcionando un servidor de traps que guarda toda la información de las alertas entrantes en una bitácora, reenviando todos esos datos a un programa llamado “snmptthandler” que se encargará de entregar las alertas a SNMPTT. A continuación se describen los pasos para habilitar a SNMPTT la lectura y procesamiento de las traps recibidas.

4.3.3.2 Configuración de SNMPTT

Esta parte de la configuración posibilita la definición de acciones en respuesta a las alertas recibidas. Los siguientes pasos describirán la forma en que trabajará SNMPTT bajo las condiciones que fueron planteadas para nuestra implementación.

4.3.3.2.1 Directivas de SNMPTT

Cabe mencionar que existen múltiples directivas que modifican el comportamiento del servidor de traducción de traps SNMPTT, sin embargo en los siguientes pasos solo se describirán las que fueron utilizadas para la configuración de esta implementación. El archivo que contiene las directivas de funcionamiento esta ubicado en la ruta “/etc/snmp” bajo el nombre de “snmptt.ini”, el cual será modificado siguiendo los siguientes pasos:

- Es necesario habilitar que SNMPTT trabaje en forma de demonio, lo que permitirá que maneje gran cantidad de traps por minuto.

```
mode = daemon
```

- Habilitar que múltiples definiciones de traps puedan ser ejecutadas por la misma trap. Si se deshabilita (0) solo será ejecutada la primera definición encontrada.

```
multiple_event = 1
```

- Habilitar a SNMPTT para que pueda ejecutar programas externos en respuesta a alguna trap definida.

```
exec_enable = 1
```

- Asegurarse que la ruta del directorio de encolamiento corresponda con el creado en la etapa de instalación.

```
spool_directory = /var/spool/snmptt
```

- Se deberá habilitar la creación de bitácoras para registrar las alertas conocidas y desconocidas.

```
log_enable = 1
log_file = /var/log/snmptt.log
unknown_trap_log_file = /var/log/snmpttunknown.log
```

4.3.3.2.2 Definición de traps

La definición de las alertas que serán procesadas se realiza en el archivo ubicado en “/etc/snmp” bajo el nombre de “snmptt.conf”. El formato para definir una alerta es el siguiente:

EVENT nombre_de_evento OID_de_evento "categoria" severidad
FORMAT cadena_de_formato
[**EXEC** cadena_de_comando]
[**NODES** lista_de_fuentes]
[**MATCH [MODE=[or | and]] | [$n:[!][( ) | n | n-n | > n | < n | x.x.x.x |**
x.x.x.x-x.x.x.x | x.x.x.x/x]]
[**REGEX ( )( )[i][g][e]**]
[**SDESC**]
[**EDESC**]

De los cuales solo serán explicados los que fueron usados en la implementación.

EVENT nombre_de_evento OID_de_evento "categoria" severidad

Usado para distinguir de manera única a una trap.

nombre_de_evento: Es una etiqueta para distinguir la trap. Esta no debe contener espacios en blanco. Esta etiqueta debería coincidir con el nombre de la TRAP-TYPE o NOTIFICATION-TYPE aun que no es necesario.

OID_de_evento: Es la cadena numérica que representa el identificador de objeto de la trap que se esta definiendo. Este dato es el que determina que trap será procesada por SNMPTT, por lo que se deberá conocer el OID de la trap que se desee procesar. Para propósitos de nuestra implementación la única trap que será agregada es mteTriggerFired definida en la MIB UCD-DISMAN-EVENT-MIB cuyo OID es el ".1.3.6.1.2.1.88.2.0.1". Esta trap es la que se lanza en caso de que el monitoreo autónomo detecte alguna situación anómala en el agente SNMP en base a los parámetros establecidos.

"categoria": Cadena que define que tipo de acción será llevada a Cabo. Existen las posibilidades de “IGNORE”, ”LOGONLY” y “Status Events”. Para la implementación será utilizada “Status Events”, debido a que “IGNORE” no realiza ninguna acción en caso de recibir la trap, mientras que ”LOGONLY” solo ejecuta la sentencia FORMAT pero no ejecuta programas externos al ignorar la sentencia EXEC.

severidad: Cadena de texto que define la severidad de la alerta. Puede ser cualquiera de estas posibilidades: Minor, Major, Normal, Critical, Warning. Solo se usa para registrarlo en las bitácoras.

FORMAT cadena_de_formato

cadena_de_formato: Esta sentencia es utilizada para generar textos más amigables para el administrador, el cual será guardado en las bitácoras de las traps conocidas. SNMPTT permite la sustitución de variables que permiten fácilmente generar logs mucho más entendibles que si se tratará con la información real de la trap. Las variables más comúnmente usadas serán

$A Nombre de Host del servidor que generó la alerta
$aR Dirección ip del Host que generó la alerta
$c Categoría
$N Nombre del evento definido en la sentencia EVENT
$X Hora en que fue encolada la trap

$x Fecha en que fue encolada la trap

$* Variables de la trap

$Fn Carácter de nueva linea

EXEC cadena_de_comando

cadena_de_comando: Esta sentencia permite ejecutar un comando, shell script o cualquier programa cuando la trap es identificada. La sustitución de variables es la misma que en la sentencia FORMAT, por lo que estas pueden ser entregadas como argumentos del programa que se pretenda utilizar.

SDESC
cadena_de_descripción
EDESC

cadena_de_descripción: Cantidad de texto que se considere necesario para definir que tipo de trap se esta definiendo y que acciones se ejecutarán en caso que se presente alguna trap de este tipo. El texto se deberá ubicar entre la sentencia de apertura de comentario SDESC y la de cierre EDESC.

En base a la anterior información se deberán seguir los siguientes pasos para configurar las traps que serán procesadas por el servidor SNMPTT:

- Modificar el archivo “snmptt.conf” para tener las siguientes lineas:

```
EVENT coldStart .1.3.6.1.6.3.1.1.5.1 "Status Events" Normal
FORMAT  Servidor Reiniciado (coldStart) HOST: $A IP: [$aR] TIME: $x $X
EXEC /usr/sbin/snmptrap2mon -w SNMP_C -s REINICIO -h $A -n $N -t "$x" "Servidor
Reiniciado HOST: $A IP: [$aR] TIME: $x $X (coldStart)"
SDESC
```

```
A coldStart trap signifies that the SNMPv2 entity, acting
in an agent role, is reinitializing itself and that its
configuration may have been altered.
EDESC
#
#
#

EVENT NotifyShutdown .1.3.6.1.4.1.8072.4.0.2 "Status Events" Normal
FORMAT  Shutdown sobre Servidor (Shutdown) HOST: $A IP: [$aR] TIME: $x $X
EXEC /usr/sbin/snmptrap2mon -w SNMP_S -s SHUTDOWN -h $A -n $N -t "$x" "Servidor
Apagado HOST: $A IP: [$aR] TIME: $x $X (Shutdown)"
SDESC
A coldStart trap signifies that the SNMPv2 entity, acting
in an agent role, is reinitializing itself and that its
configuration may have been altered.
EDESC
#
#
#
EVENT mteTriggerFired .1.3.6.1.2.1.88.2.0.1  "Status Events" Normal
FORMAT  Error en Monitoreo de RECURSOS(mteTriggerFired)  HOST: $A IP: [$aR] TIME
: $x $X Variables: $*
EXEC /usr/sbin/snmptrap2mon -w SNMP -s SNMPTRAPS -h $A -n $N -t "$x" "Monitoreo
de RECURSOS  HOST: $A IP: [$aR] TIME: $x $X  $Fn Variables: $*"
SDESC
Trap que especifica algun error en el monitoreo mediante DISMAN-EVENT configurad
os en el Agente SNMP
Estos procesos son configurados con la directiva monitor
EDESC
#
```

Los dos primeros EVENT hacen referencia a dos traps genéricas que fueron explicadas en el capitulo 2, estas traps ya se encuentran definidas en el archivo que se

copio de la distribución SNMPTT, solo que se modificaron el FORMAT y EXEC. El tercer EVENT define la trap que es lanzada por el monitoreo autónomo que se habilitó en todos los agentes SNMP instalados.

- Crear el programa que se encargará de establecer comunicación con MON y enviarle la información de la trap SNMP. Todos los EVENT que se definieron anteriormente ejecutan dicho programa para notificar a MON. El archivo se llamará “snmptrap2mon” y se ubicará en la ruta “/usr/sbin”. Dicho archivo deberá tener el siguiente código.

```
$vi  /usr/sbin/snmptrap2mon

use Carp;
use Mon::Client;
use Getopt::Std;
getopts("w:s:h:n:t:u");
my $detail = join(" ",@ARGV);
my $monhost = 'monhost.domain.com';
my $monport = 2583;
my $watch = $opt_w or 'default';
my $service = $opt_s or 'default';
my $status = $opt_u ? 'ok' : 'fail' ;
croak "Need all of -h, -n and -t" unless (defined $opt_h and
   defined $opt_n and defined $opt_t);
my $alerthost = $opt_h;
my $alertname = $opt_n;
my $alerttime = $opt_t;
my $summary = "$alertname trap from $alerthost at $alerttime";
$mon = new Mon::Client( host => $monhost, port => $monport );
croak "Couldn't make a new Mon::Client to $monhost on $monport"
 unless $mon;
$t = $mon->send_trap(group => $watch,
                service => $service,
                retval => 1,
```

```
                opstatus => $status,
                summary => $summary,
                detail => $detail,
                );
exit ( $t );
```

De este código se deberá modificar las variables $monhost y $monport (resltadas en letra negrita) que deberán hacer referencia a los del servidor MON que será notificado. Es importante mencionar que para que las traps puedan ser recibidas por MON, se deberán crear los watch y service a los que hacen referencia los argumentos del programa "/usr/sbin/snmptrap2mon" (en las opciones w y s).

- Dar permisos de ejecución al programa

```
$chmod ug+x  /usr/sbin/snmptrap2mon
```

4.3.3.2.3 Levantar SNMPTT

Una vez que se tiene funcionando el servidor de traps Net-SNMP así como la configuración de SNMPTT y la definición de las alertas que será procesadas, solo resta levantar el servidor SNMPTT. Existen dos formas de levantar el servicio en modo demonio:

```
$/usr/sbin/snmptt –daemon
```

o utilizar el script proporcionado por la distribución

```
$cp snmptt.init.d /etc/init.d/snmptt

$/etc/init.d/snmptt start
```

4.3.4 Pruebas

4.3.4.1 Planeación

A pesar de que la instalación y configuración del servidor de traps es menos compleja en comparación al servidor de graficación, se decidió que no habrían fases de prueba en cuanto a la instalación del servidor de alertas, de manera que todo el proceso de implementación se realizaría solamente una vez. Por ello lo primero fue elegir el servidor que tendría la tarea de recibir y procesar las alertas generadas por los agentes SNMP. Una vez determinado el servidor, se llevo a cabo la instalación y configuración tal y como fue descrito en las secciones 4.3.2 Instalación y 4.3.3 Configuración del apartado 4.3 Implementación del Servidor de Alertas. También se decidió tomar como servidor prueba a gypsy en donde se realizó la configuración necesaria en el agente SNMP para que este realizará de manera autónoma el monitoreo de su propia información de administración (MIB), con el objetivo de determinar la existencia de errores de acuerdo a las directivas de monitoreo del propio agente.

En las siguientes secciones se describirán las características del equipo utilizado como servidor de alertas, asi como la configuración del soporte de monitoreo DISMAN-EVENT-MIB del agente SNMP en gypsy.

4.3.4.2 Descripción del equipo

Las características del equipo utilizado como servidor de traps resultan más que suficientes para el rendimiento óptimo de la solución. La descripción se detalla en la siguiente tabla.

Nombre	***Sistema Operativo***	***Hardware***	***Dirección IP***
aura	Red Hat Enterprise Linux 3	Sun FIRE V20z con 2 procesadores AMD Opteron, 4 Gb de memoria RAM y 2 discos SCSI (Ultra 320) de 200 Gb cada uno.	132.248.10.47

Tabla de las Características del Servidor de Alertas

4.3.4.3 Configuración de auto monitoreo

Al igual que el monitoreo de recursos, todas las directivas de monitoreo autónomo deben ser agregadas en el archivo de configuración “snmpd.conf” del propio agente SNMP. Estas pruebas fueron realizadas en el servidor gypsy cuya configuración e instalación fue descrita en la sección 4.1.4.2.2.3 Configuración de monitoreo, del apartado 4.1.4.2 Primera fase de prueba de la implementación de los agentes SNMP.

Todos los pasos referentes al monitoreo autónomo se basan en la directiva “monitor”, la cual se ocupa de revisar la propia información de administración (MIB) del agente SNMP. La detección de alguna condición anómala se determina gracias a una comparación booleana entre un número entero y un valor de la MIB. Para saber si se ha presentado alguna condición falsa la mejor alternativa es revisar el valor de las banderas de error descritas en la sección 4.1.3.2 Directivas de Monitoreo del apartado 4.1.3 Configuración del agente SNMP, en donde un valor diferente de 0 (usualmente 1)

significa la presencia de algún error en el monitoreo de un recurso o servicio. Para configurar exitosamente el auto monitoreo se siguieron los siguientes pasos:

- Definir un nombre de comunidad para el envío de las traps. El nombre de esta comunidad debe corresponder con la configurada en el servidor de traps. Si en el servidor de traps no se configuró ninguna comunidad (esto permite que el servidor de traps reciba alertas de cualquier agente SNMP) el nombre de esta comunidad puede ser cualquier palabra. En nuestro caso la comunidad fue nombrada comuTRAPS.

```
trapcommunity  comuTRAPS
```

- Definir que la dirección ip del servidor aura será el encargado de recibir y procesar las traps generadas por el monitoreo autónomo.

```
trap2sink 132.248.10.47 comuTRAPS 162
```

- Asociar un nombre de agente seguro con un usuario que tenga permisos de explorar la MIB del propio agente.

```
agentSecName pruebaL
```

- Monitorear cada 5 minutos si existe algún error en el monitoreo de procesos.

```
monitor -u pruebaL -r 300 -o prNames -o prErrMessage "Procesos" prErrorFlag != 0
```

- Monitorear cada 5 minutos si existe algún error en el monitoreo de la memoria de intercambio SWAP.

```
monitor -u pruebaL -r 300 -o memErrorName -o memSwapErrorMsg "Memoria SWAP" memSwapError != 0
```

- Monitorear cada 5 minutos si existe algún error en el monitoreo de los sistemas de archivos.

```
monitor -u pruebaL -r 300 -o dskPath -o dskErrorMsg "Discos" dskErrorFlag != 0
```

- Monitorear cada 5 minutos si existe algún error en el monitoreo del promedio de carga del sistema.

```
monitor -u pruebaL -r 300 -o laNames -o laErrMessage  "Promedio Carga" laErrorFlag != 0
```

- Se deberá reiniciar el agente SNMP para que tome la nueva configuración.

```
[root@gypsy root]# /etc/init.d/snmp stop
[root@ gypsy root]# /etc/init.d/snmp start
```

Si existe alguna duda con respecto a las directivas utilizadas en los anteriores pasos, remitirse al apartado 4.1.3.3 Habilitación de monitoreo autónomo.

4.3.4.4 Pruebas del soporte DISMAN-EVENT-MIB

Para comprobar que las alertas son generadas de forma automática en caso de presentarse alguna bandera de error (usualmente con valor 1), se debió provocar el error o cambiar los parámetros de monitoreo del agente SNMP. En las siguientes secciones se describe como fueron provocados algunos errores o modificados algunos parámetros para que el agente generará las alertas correspondientes.

4.3.4.4.1 Pruebas con las traps genéricas coldStart y nsNotifyShutdown

Antes de comenzar con la pruebas para el monitoreo autónomo es necesario comprobar que el agente puede recibir al menos las traps coldStart y nsNotifyShutdown. Estas traps no necesitan ser configuradas mediante directivas, puesto que son generadas en forma automática en cuanto el agente SNMP tiene definido un servidor de traps, a quien notificará las alertas generadas (usando la directiva trap2sink).

Se recuerda que la trap coldStart es generada cuando el agente SNMP es activado, lo cual puede significar que el servidor ha sido iniciado, mientras que la trap nsNotifyShutdown es enviada cuando el agente SNMP ha sido detenido, lo cual puede implicar que el servidor ha sido dado de baja. A continuación se describen los pasos que nos permitirán probar si el servidor de traps puede recibir las citadas alertas.

- Dar de baja el demonio de snmpd

```
[root@gypsy root]# /etc/init.d/snmp stop
```

- Revisar si la alerta nsNotifyShutdown ha sido recibida en el servidor

```
[root@aletia root]# tail -f /root/snmp/traps.log

2007-02-25 20:35:51 gypsy [132.248.120.102]:
.1.3.6.1.2.1.1.3.0 = Timeticks: (178959) 0:29:49.59 .1.3.6.1.6.3.1.1.4.1.0 = OID: .1.3.6.1.4.1.8072.4.0.2
```

Se puede observar que la trap genérica "NET-SNMP-AGENT-MIB::nsNotifyShutdown" (cuya OID numérica es: .1.3.6.1.4.1.8072.4.0.2) ha sido recibida exitosamente por el servidor de traps.

- Dar de alta el demonio snmpd

```
[root@gypsy root]# /etc/init.d/snmp start
```

- Revisar si la alerta coldStart ha sido recibida en el servidor

```
[root@aletia root]# tail -f /root/snmp/traps.log

2007-02-25 20:45:04 gypsy [132.248.120.102]:
.1.3.6.1.2.1.1.3.0 = Timeticks: (6) 0:00:00.06 .1.3.6.1.6.3.1.1.4.1.0 = OID: .1.3.6.1.6.3.1.1.5.1
.1.3.6.1.6.3.1.1.4.3.0 = OID: .1.3.6.1.4.1.8072.3.2.10
```

Se puede observar que la trap genérica SNMPv2-MIB::coldStart (cuya OID numérica es: .1.3.6.1.4.1.8072.3.2.10) ha sido recibida exitosamente por el servidor de traps.

4.3.4.4.2 Pruebas con el monitoreo de procesos

- Antes que nada recuperamos los datos acerca del monitoreo de procesos del agente SNMP, en los cuales se dictó observar a los procesos de httpd y sendmail (recordar que se tiene como base la configuración realizada durante la primera fase de pruebas del apartado Implementación de los agentes SNMP).

```
[root@gypsy root]# snmpwalk -v 3 -u pruebaL -l authPriv -A prueba2007 -X prueba2007 localhost .1.3.6.1.4.1.2021.2

UCD-SNMP-MIB::prIndex.1 = INTEGER: 1
UCD-SNMP-MIB::prIndex.2 = INTEGER: 2
UCD-SNMP-MIB::prNames.1 = STRING: httpd
UCD-SNMP-MIB::prNames.2 = STRING: sendmail
UCD-SNMP-MIB::prMin.1 = INTEGER: 4
UCD-SNMP-MIB::prMin.2 = INTEGER: 1
UCD-SNMP-MIB::prMax.1 = INTEGER: 100
UCD-SNMP-MIB::prMax.2 = INTEGER: 3
UCD-SNMP-MIB::prCount.1 = INTEGER: 6
UCD-SNMP-MIB::prCount.2 = INTEGER: 2
UCD-SNMP-MIB::prErrorFlag.1 = INTEGER: 0
UCD-SNMP-MIB::prErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::prErrMessage.1 = STRING:
UCD-SNMP-MIB::prErrMessage.2 = STRING:
UCD-SNMP-MIB::prErrFix.1 = INTEGER: 0
UCD-SNMP-MIB::prErrFix.2 = INTEGER: 0
UCD-SNMP-MIB::prErrFixCmd.1 = STRING:
UCD-SNMP-MIB::prErrFixCmd.2 = STRING:
```

Se puede observar que el número de procesos esta dentro los rangos establecidos tanto para httpd y sendmail, por lo tanto no se presenta ninguna bandera ni mensaje de error.

- Debido a que no existe ningún problema con el monitoreo de esos dos procesos, fue necesario provocar el error para observar si el soporte de monitoreo autónomo era capaz de generar y enviar las alertas al servidor de traps. Para esta tarea se dio de baja tanto al servidor web como al agente de trasporte de correo sendmail.

```
[root@gypsy root]# /etc/init.d/sendmail stop
Apagando sendmail:                         [  OK  ]
Desactivación de sm-client:                [  OK  ]


[root@gypsy root]# /usr/local/httpd/bin/apachectl stop
```

- Volvemos a consultar la información de la MIB

```
[root@gypsy root]# snmpwalk -v 3 -u pruebaL -l authPriv -A prueba2007 -X prueba2007 localhost
.1.3.6.1.4.1.2021.2

UCD-SNMP-MIB::prIndex.1 = INTEGER: 1
UCD-SNMP-MIB::prIndex.2 = INTEGER: 2
UCD-SNMP-MIB::prNames.1 = STRING: httpd
UCD-SNMP-MIB::prNames.2 = STRING: sendmail
UCD-SNMP-MIB::prMin.1 = INTEGER: 4
UCD-SNMP-MIB::prMin.2 = INTEGER: 1
UCD-SNMP-MIB::prMax.1 = INTEGER: 100
UCD-SNMP-MIB::prMax.2 = INTEGER: 3
UCD-SNMP-MIB::prCount.1 = INTEGER: 0
UCD-SNMP-MIB::prCount.2 = INTEGER: 0
UCD-SNMP-MIB::prErrorFlag.1 = INTEGER: 1
UCD-SNMP-MIB::prErrorFlag.2 = INTEGER: 1
UCD-SNMP-MIB::prErrMessage.1 = STRING: Too few httpd running (# = 0)
UCD-SNMP-MIB::prErrMessage.2 = STRING: Too few sendmail running (# = 0)
UCD-SNMP-MIB::prErrFix.1 = INTEGER: 0
```

```
UCD-SNMP-MIB::prErrFix.2 = INTEGER: 0
UCD-SNMP-MIB::prErrFixCmd.1 = STRING:
UCD-SNMP-MIB::prErrFixCmd.2 = STRING:
```

Aquí se puede observar claramente (en letras negritas) como la información de administración (MIB) del agente nos indica que el número de procesos para ambos servicios es de 0, por lo que se han levantado 2 banderas de error (prErrorFlag) junto con dos mensajes (prErrMessage) que describen la anomalía encontrada.

- Como se configuró que el monitor fuera aplicado cada 5 minutos (300 segundos) se tendrá que esperar a que este tiempo transcurra para observar si en la bitácora de traps del servidor de alertas se han agregado las alertas correspondientes a ese agente SNMP .

```
[root@aletia root]# tail -f /root/snmp/traps.log

2007-02-25 18:36:32 gypsy [132.248.120.102]:
.1.3.6.1.2.1.1.3.0 = Timeticks: (12007) 0:02:00.07 .1.3.6.1.6.3.1.1.4.1.0 = OID: .1.3.6.1.2.1.88.2.0.1
.1.3.6.1.2.1.88.2.1.1 = STRING: Procesos .1.3.6.1.2.1.88.2.1.2 = STRING: .1.3.6.1.2.1.88.2.1.3 = STRING:
.1.3.6.1.2.1.88.2.1.4 = OID: .1.3.6.1.4.1.2021.2.1.100.1 .1.3.6.1.2.1.88.2.1.5 = INTEGER: 1
.1.3.6.1.4.1.2021.2.1.2.1 = STRING: httpd .1.3.6.1.4.1.2021.2.1.101.1 = STRING: Too few httpd running
(# = 0)


2007-02-25 18:36:32 gypsy [132.248.120.102]:
.1.3.6.1.2.1.1.3.0 = Timeticks: (12008) 0:02:00.08 .1.3.6.1.6.3.1.1.4.1.0 = OID: .1.3.6.1.2.1.88.2.0.1
.1.3.6.1.2.1.88.2.1.1 = STRING: Procesos .1.3.6.1.2.1.88.2.1.2 = STRING: .1.3.6.1.2.1.88.2.1.3 =
STRING: .1.3.6.1.2.1.88.2.1.4 = OID: .1.3.6.1.4.1.2021.2.1.100.2 .1.3.6.1.2.1.88.2.1.5 = INTEGER:
1 .1.3.6.1.4.1.2021.2.1.2.2 = STRING: sendmail .1.3.6.1.4.1.2021.2.1.101.2 = STRING: Too few
sendmail running (# = 0)
```

Aquí se puede observar (en letras negras) como exitosamente han llegado 2 mensajes trap SNMP al servidor de alertas. Ambas traps hacen referencia a los procesos httpd y sendmail indicando la hora, nombre de host, IP del servidor agente SNMP que envió la alerta así como el nombre del proceso y mensaje de error. Los datos que hacen referencia a los errores son como los que se pudieron observar cuando consultamos la información de la MIB del agente.

4.3.4.4.3 Pruebas con el monitoreo de memoria swap

- Se recuperan los datos del agente en cuanto a la información del monitoreo de la memoria de intercambio.

```
[root@gypsy root]# snmpwalk -v 3 -u pruebaL -l authPriv -A prueba2007 -X prueba2007 localhost .1.3.6.1.4.1.2021.4

UCD-SNMP-MIB::memIndex.0 = INTEGER: 0
UCD-SNMP-MIB::memErrorName.0 = STRING: swap
UCD-SNMP-MIB::memTotalSwap.0 = INTEGER: 409648
UCD-SNMP-MIB::memAvailSwap.0 = INTEGER: 407540
UCD-SNMP-MIB::memTotalReal.0 = INTEGER: 255308
UCD-SNMP-MIB::memAvailReal.0 = INTEGER: 5104
UCD-SNMP-MIB::memTotalFree.0 = INTEGER: 412644
UCD-SNMP-MIB::memMinimumSwap.0 = INTEGER: 102400
UCD-SNMP-MIB::memShared.0 = INTEGER: 0
UCD-SNMP-MIB::memBuffer.0 = INTEGER: 60716
UCD-SNMP-MIB::memCached.0 = INTEGER: 124592
UCD-SNMP-MIB::memSwapError.0 = INTEGER: 0
UCD-SNMP-MIB::memSwapErrorMsg.0 = STRING:
```

Se observa (en letras negras) que no existe la presencia de error puesto que fue definido un mínimo de memoria SWAP libre de 102400Kb (100 Mb) de los 409648Kb (400 Mb) totales, y la información nos indica que hay 407540Kb libres.

- Es necesario configurar un nuevo parámetro de monitoreo en el archivo de configuración "snmpd.conf" para que el agente SNMP pueda lanzar una trap. Por ello se decidió configurar un valor absurdo con el fin de que se comprobar si el agente sería capaz de generar y lanzar la alerta. Así que se definió como mínimo casi la totalidad de la memoria swap.

```
swap 409000
```

- Para que se tome la nueva configuración es necesario parar y volver a iniciar al agente SNMP.

```
[root@gypsy root]# /etc/init.d/snmp stop
[root@ gypsy root]# /etc/init.d/snmp start
```

- Consultar nuevamente la MIB del agente SNMP.

```
[root@gypsy root]# snmpwalk -v 3 -u pruebaL -l authPriv -A prueba2007 -X prueba2007 localhost .1.3.6.1.4.1.2021.4

UCD-SNMP-MIB::memIndex.0 = INTEGER: 0
UCD-SNMP-MIB::memErrorName.0 = STRING: swap
UCD-SNMP-MIB::memTotalSwap.0 = INTEGER: 409648
UCD-SNMP-MIB::memAvailSwap.0 = INTEGER: 407476
UCD-SNMP-MIB::memTotalReal.0 = INTEGER: 255308
UCD-SNMP-MIB::memAvailReal.0 = INTEGER: 3680
UCD-SNMP-MIB::memTotalFree.0 = INTEGER: 411156
UCD-SNMP-MIB::memMinimumSwap.0 = INTEGER: 409000
UCD-SNMP-MIB::memShared.0 = INTEGER: 0
```

```
UCD-SNMP-MIB::memBuffer.0 = INTEGER: 62172
UCD-SNMP-MIB::memCached.0 = INTEGER: 124972
UCD-SNMP-MIB::memSwapError.0 = INTEGER: 1
UCD-SNMP-MIB::memSwapErrorMsg.0 = STRING: Running out of swap space (407476)
```

Aquí se puede apreciar (en letras negras) como al agente SNMP registra la bandera de error y genera un mensaje descriptivo aludiendo a la cantidad de memoria de intercambio disponible.

- Como se configuró que el monitor fuera aplicado cada 5 minutos (300 segundos) se tendrá que esperar a que este tiempo transcurra para observar si en la bitácora de traps del servidor de alertas se han agregado las alertas correspondientes a ese agente SNMP .

```
[root@aletia root]# tail -f /root/snmp/traps.log

2007-02-25 19:30:49 gypsy [132.248.120.102]:
.1.3.6.1.2.1.1.3.0= Timeticks: (12008) 0:02:00.08  .1.3.6.1.6.3.1.1.4.1.0 = OID: .1.3.6.1.2.1.88.2.0.1
.1.3.6.1.2.1.88.2.1.1 = STRING: Memoria SWAP  .1.3.6.1.2.1.88.2.1.2 = STRING:
.1.3.6.1.2.1.88.2.1.3 = STRING:  .1.3.6.1.2.1.88.2.1.4 = OID: .1.3.6.1.4.1.2021.4.100.0
.1.3.6.1.2.1.88.2.1.5 = INTEGER: 1  .1.3.6.1.4.1.2021.4.2.0 = STRING: swap  .1.3.6.1.4.1.2021.4.101.0 =
STRING: Running out of swap space (407468)
```

Se puede observar (en letras negras) que la alerta ha sido recibida de manera exitosa, mostrando toda la información referente al error que se presentó en el monitoreo de la memoria de intercambio.

4.3.4.4.4 Pruebas con el monitoreo de los sistemas de archivo

- Se recuperan los datos del agente en cuanto a la información del monitoreo de los sistemas de archivo.

```
[root@gypsy root]# snmpwalk -v 3 -u pruebaL -l authPriv -A prueba2007 -X prueba2007 localhost .1.3.6.1.4.1.2021.9

UCD-SNMP-MIB::dskIndex.1 = INTEGER: 1
UCD-SNMP-MIB::dskIndex.2 = INTEGER: 2
UCD-SNMP-MIB::dskPath.1 = STRING: /
UCD-SNMP-MIB::dskPath.2 = STRING: /home
UCD-SNMP-MIB::dskDevice.1 = STRING: /dev/hda2
UCD-SNMP-MIB::dskDevice.2 = STRING: /dev/hda5
UCD-SNMP-MIB::dskMinimum.1 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.2 = INTEGER: -1
UCD-SNMP-MIB::dskMinPercent.1 = INTEGER: 10
UCD-SNMP-MIB::dskMinPercent.2 = INTEGER: 10
UCD-SNMP-MIB::dskTotal.1 = INTEGER: 6048352
UCD-SNMP-MIB::dskTotal.2 = INTEGER: 1565392
UCD-SNMP-MIB::dskAvail.1 = INTEGER: 2953764
UCD-SNMP-MIB::dskAvail.2 = INTEGER: 1323560
UCD-SNMP-MIB::dskUsed.1 = INTEGER: 2787348
UCD-SNMP-MIB::dskUsed.2 = INTEGER: 162312
UCD-SNMP-MIB::dskPercent.1 = INTEGER: 49
UCD-SNMP-MIB::dskPercent.2 = INTEGER: 11
UCD-SNMP-MIB::dskPercentNode.1 = INTEGER: 18
UCD-SNMP-MIB::dskPercentNode.2 = INTEGER: 1
UCD-SNMP-MIB::dskErrorFlag.1 = INTEGER: 0
UCD-SNMP-MIB::dskErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::dskErrorMsg.1 = STRING:
UCD-SNMP-MIB::dskErrorMsg.2 = STRING:
```

Se observa (en letras negras) que no existe la presencia de ningún error ya que los dos sistemas de archivo monitoreados, "/" y "/home" , están por encima del 10% de capacidad de almacenamiento disponible.

- Es necesario configurar un nuevo parámetro de monitoreo en el archivo de configuración "snmpd.conf", para que el agente SNMP pueda lanzar una trap. Por ello se decidió configurar nuevamente un valor absurdo para observar si el agente podría detectar y lanzar la alerta. El parámetro fue que la partición de root (/) tuviera al menos el 60% de espacio mínimo disponible.

```
disk  /  60%
```

- Para que se tome la nueva configuración es necesario parar y volver a iniciar al agente SNMP.

```
[root@gypsy root]# /etc/init.d/snmp stop
[root@ gypsy root]# /etc/init.d/snmp start
```

- Consular nuevamente la MIB del agente SNMP.

```
[root@gypsy root]# snmpwalk -v 3 -u pruebaL -l authPriv -A prueba2007 -X prueba2007 localhost
.1.3.6.1.4.1.2021.9

UCD-SNMP-MIB::dskIndex.1 = INTEGER: 1
UCD-SNMP-MIB::dskIndex.2 = INTEGER: 2
UCD-SNMP-MIB::dskPath.1 = STRING: /
UCD-SNMP-MIB::dskPath.2 = STRING: /home
UCD-SNMP-MIB::dskDevice.1 = STRING: /dev/hda2
UCD-SNMP-MIB::dskDevice.2 = STRING: /dev/hda5
UCD-SNMP-MIB::dskMinimum.1 = INTEGER: -1
UCD-SNMP-MIB::dskMinimum.2 = INTEGER: -1
UCD-SNMP-MIB::dskMinPercent.1 = INTEGER: 60
```

```
UCD-SNMP-MIB::dskMinPercent.2 = INTEGER: 10
UCD-SNMP-MIB::dskTotal.1 = INTEGER: 6048352
UCD-SNMP-MIB::dskTotal.2 = INTEGER: 1565392
UCD-SNMP-MIB::dskAvail.1 = INTEGER: 2953752
UCD-SNMP-MIB::dskAvail.2 = INTEGER: 1323560
UCD-SNMP-MIB::dskUsed.1 = INTEGER: 2787360
UCD-SNMP-MIB::dskUsed.2 = INTEGER: 162312
UCD-SNMP-MIB::dskPercent.1 = INTEGER: 49
UCD-SNMP-MIB::dskPercent.2 = INTEGER: 11
UCD-SNMP-MIB::dskPercentNode.1 = INTEGER: 18
UCD-SNMP-MIB::dskPercentNode.2 = INTEGER: 1
UCD-SNMP-MIB::dskErrorFlag.1 = INTEGER: 1
UCD-SNMP-MIB::dskErrorFlag.2 = INTEGER: 0
UCD-SNMP-MIB::dskErrorMsg.1 = STRING: /: less than 60% free (= 49%)
UCD-SNMP-MIB::dskErrorMsg.2 = STRING:
```

Se aprecia (en letras negras) que el agente SNMP detecta el error debido a que la partición root (/) tiene menos del 60% de espacio disponible.

- Como se configuró que el monitor fuera aplicado cada 5 minutos (300 segundos) se tendrá que esperar a que este tiempo transcurra para observar si en la bitácora de traps del servidor de alertas se han agregado las alertas correspondientes a ese agente SNMP .

```
[root@aletia root]# tail -f /root/snmp/traps.log

2007-02-25 20:08:02 gypsy [132.248.120.102]:
.1.3.6.1.2.1.1.3.0 = Timeticks: (12008) 0:02:00.08  .1.3.6.1.6.3.1.1.4.1.0 = OID: .1.3.6.1.2.1.88.2.0.1
.1.3.6.1.2.1.88.2.1.1 = STRING: Discos  .1.3.6.1.2.1.88.2.1.2 = STRING:  .1.3.6.1.2.1.88.2.1.3 =
STRING:  .1.3.6.1.2.1.88.2.1.4 = OID: .1.3.6.1.4.1.2021.9.1.100.1  .1.3.6.1.2.1.88.2.1.5 = INTEGER:
```

1 .1.3.6.1.4.1.2021.9.1.2.1 = STRING: / .1.3.6.1.4.1.2021.9.1.101.1 = STRING: **/: less than 60% free (= 49%)**

Nuevamente la alerta es recibida describiendo (en letras negras) el error suscitado en el agente SNMP con respecto al sistema de archivo root (/) del sistema.

4.3.4.5 Resultados

Las pruebas realizadas permitieron comprobar que el esquema de traps y soporte DISMAN-EVENT-MIB proporcionado por la distribución Net-SNMP permite tener una solución de notificación de alertas altamente flexible y funcional. Esta característica logró, de manera fácil y rápida, configurar un monitoreo autónomo de todos los recursos y servicios que SNMP puede revisar por defecto, permitiendo generar traps con la suficiente información para saber en que tipo de recurso o servicio se suscito un problema y en que servidor se dio la problemática.

Es importante mencionar que aunque en el presente trabajo no fueron documentadas las pruebas acerca del funcionamiento del servidor de traducción de traps SNMPTT, dicha implementación probó tener una funcionalidad total al generar y mandar alertas mucho más entendibles y amigables de acuerdo a la configuración descrita en este mismo trabajo. A su vez el programa utilizado para el envió de alertas SNMP al servidor MON tuvo pruebas exitosas al lograr que MON pudiera mandar, tanto por correo electrónico y el servicio de mensajes de texto skytel, las alertas generadas por los agentes SNMP.

De esta manera se concluyó exitosamente la implementación del esquema de alertas con las características y objetivos planteados en el presente trabajo de tesis.

Conclusiones

Después de toda la investigación que se realizó durante este proyecto de tesis, se pudieron sacar una serie de conclusiones acerca del uso de software libre para el desarrollo de sistemas. Sin duda para el caso del presente trabajo la mejor opción fue decidir que se llevara cabo la solución a través del uso de software de libre distribución, ya que esto permitió ahorrar un gasto considerable en la compra de algún software propietario y además nos ofreció la flexibilidad de adecuar el sistema a nuestras necesidades.

La compra de algún sistema de graficación y monitoreo no era una opción desde que se planteo la necesidad de reforzar estas actividades en el departamento, sin embargo muchos de estos productos no se adecuaban a nuestras necesidades, situación que fue resuelta por todas las características que incorporaban muchas herramientas de graficación y monitoreo de libre distribución, que junto con la colaboración del protocolo SNMP permitió conjuntar una solución altamente flexible y poderosa para adecuarla perfectamente a nuestros objetivos. Fue muy importante estudiar a fondo las características y capacidades del protocolo de gestión de redes SNMP, esto aporto un gran conocimiento acerca de la simplicidad y poder de esta herramienta para la gestión de dispositivos de red, lo que permitió aprovechar su potencial para alcanzar los objetivos de nuestro desarrollo.

Un punto importante a destacar, es que el conocimiento a fondo acerca de las características y operación del protocolo SNMP fue más importante para llevar a cabo la solución de nuestro problema que el hecho de contar con una metodología adecuada para la implantación del sistema.

El desarrollo de la solución descrito en el presente trabajo de tesis, permitió concluir que la incorporación de software libre para llevar a cabo el sistema resultó ser una buena opción para diseñar las mejoras en los anteriores esquemas de monitoreo y graficación, lo que permite pensar que el uso de software de libre distribución es una buena alternativa para emprender la solución de cualquier problema que se pretenda resolver bajo un contexto informático.

Por otro lado la implementación de esta solución fue un verdadero reto personal, debido a que se involucraron una cantidad considerable de conocimiento teóricos y técnicos de los cuales solo conocía algunos aspectos. Esta situación permitió poner a prueba mi capacidad y aplicar los conocimientos adquiridos durante la carrera, de tal manera que pude comprobar que con esfuerzo, dedicación y un alto compromiso se puede llegar a cualquier meta, aunque en un principio no se tenga claro el camino hacia el objetivo.

Bibliografía

W. Satllings, SNMP, SNMPv2, SNMPv3 and RMON1
Addison Wesley Tercera Edición, 1999, Parte IV y V.

Douglas Mauro, Kevin Schmidt
Essential SNMP
O'Reilly, 2001

Documentos Electrónicos

Teoria y configuración de las funciones de SNMP
http://www.net-snmp.org/

La seguridad en la familia de protocolos SNMP
http://www.rediris.es/rediris/boletin/50-51/ponencia16.html

Protocolo Básico de Administración de Red (SNMP) Versión 3.
http://neutron.ing.ucv.ve/revista-e/No6/Brice%C3%B1o%20Maria/SNMPv3.html

Mecanismos de gestión de Redes
http://ditec.um.es/laso/docs/tut-tcpip/3376c414.html#smi

Manual de gestión de Redes
www.lazaizarweb.com

Panorama global del protocolo SNMP
http://es.tldp.org/Manuales-LuCAS/GSAL/gsal-19991128-htm/snmp.htm

Una alternativa simple, SNMP

http://www.coit.es/publicac/publbit/bit102/quees.htm

Formato de mensajes en la versión 2 de SNMP

http://www.juniper.net/techpubs/software/junos/junos74/swconfig74-net-mgmt/html/chassis-mib16.html

SNMP en linux

http://es.tldp.org/Articulos-periodisticos/jfs/snmp/snmp.html

Documentos de la IETF

http://tools.ietf.org/html/

Guía de la pila de protocolos TCP/IP

http://tcpipguide.com/free/

Herramienta de Graficación Cricket

http://cricket.sourceforge.net/

Herramienta de Graficación HotSANIC

http://hotsanic.sourceforge.net/

Herramienta de Graficación Percival

http://percival.sourceforge.net/

Herramienta de Graficación Nagios

http://www.nagios.org

Herramienta de Graficación MRTG

http://www.mrtg.com/

Instalación y configuración de cacti

http://www.cacti.net/

Traductor de Traps SNMP V1.1

http://www.snmptt.org

Comunicación entre MON y SNMP

http://snmptt.sf.net