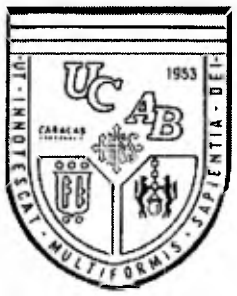


# UNIVERSIDAD CATÓLICA ANDRÉS BELLO
## FACULTAD DE INGENIERÍA
## ESCUELA DE INGENIERÍA DE TELECOMUNICACIONES

*Diseño e Implementación de un Sistema para el Control, Supervisión y Gestión de Redes basadas en el protocolo IPv4.*
*(Apéndice)*

**TRABAJO ESPECIAL DE GRADO**

presentado ante la

**UNIVERSIDAD CATÓLICA ANDRÉS BELLO**

Como parte de los requisitos para optar al título de

**INGENIERO EN TELECOMUNICACIONES**

REALIZADO POR *Miguel Ángel Sabal Matheus*

PROFESOR GUÍA *Ing. José Gregorio Cotua*

FECHA *Caracas, 13 de Octubre de 2006*

# APÉNDICE I

## ESTUDIO DE MERCADO

*SISTEMA DE CONTROL, SUPERVISIÓN Y GESTIÓN DE REDES BASADAS EN IPV4.*

# APÉNDICE I

## LISTA DE HERRAMIENTAS QUE OFRECE EL MERCADO

En la siguiente tabla se encuentra una lista de todas las herramientas encontradas que permiten la gestión y supervisión de redes basadas en *IPv4* con su respectivo enlace electrónico.

| Sistema | Fuente |
|---|---|
| AdventNet Web NMS | http://manageengine.adventnet.com/products/opmanager/ |
| AIRWAVE | http://www.airwave.com/ |
| SPECTRUM | http://www.aprisma.com/index2.html |
| IBM Netview/AIX | http://www.ibm.com/us/ |
| Intellipool Network Monitor | http://www.intellipool.se/ |
| InterMapper | http://www.intermapper.com/ |
| Little:eye, | http://www.cbr.com.tr/eng/littleye.asp |
| NetCrunch | http://www.adremsoft.com/netcrunch/ |
| NetMechanica | http://www.netmechanica.com/ |
| SolarWinds Orion | http://www.solarwinds.net/ |
| Sun Solstice | http://www.sun.com/software/index.jsp?cat=Networking&tab=3/system.mgmt.html |
| StableNet PME | http://www.infosim.net/stablenet/stablenetPME |
| SwitchMonitor | http://www.netlatency.com/products/switchMonitor.asp |
| SysOrb | http://www.evalesco.com/ |
| SysUpTime | http://www.ireasoning.com/sysuptime.shtml |
| VitalNet | http://www.lucent.com/products/solution/0,,CTID+2020-STID+10439-SOID+1335-LOCL+1,00.html |
| Fidelia | http://www.fidelia.com/ |
| Castle Rock | http://www.castlerock.com/ |
| EM7 | http://www.sciencelogic.com/ |
| IpMonitor | http://www.ipmonitor.com/default.aspx |
| Just For Fun Network Monitoring System | http://www.jffnms.org/ |
| LANsurveyor | http://www.neon.com/LSwin.shtml |
| Netcool | http://www.micromuse.com/products_sols/ |
| LogisoftAR | http://www.logisoftar.com/ |
| N-vision | http://www.n-able.com/msp/technology/n-central/ |
| NimBUS | http://www.nimsoft.com/netman/index.shtml |
| OpenNMS | http://opennms.org/wiki// |
| HP OpenView | http://www.openview.hp.com/ |

# APÉNDICE II

## ANÁLISIS COMPARATIVO COMPILADORES C++

*Sistema de Control, Supervisión y Gestión de Redes basadas en IPv4.*

# APÉNDICE II

## ANÁLISIS COMPARATIVO COMPILADORES C++

*Análisis Comparativo Compiladores de C++ (Tabla 1/3 )*

| Características | DEV C++ | VISUAL C++ | BUILDER C++ |
|---|---|---|---|
| Vigencia | Compilador muy eficiente es accesible, gratuito totalmente compatible y multiplataforma<br>El código que use librerías para diferentes plataformas, puede ser recompilado en ellas sin ningún problema. | Es un compilador muy utilizado en el desarrollo de aplicaciones de alta potencia. El trabajo se basa en el concepto de "proyecto", y todas las características de este proyecto se pueden configurar de una manera bastante sencilla en "Project/Settings. | Es una herramienta muy poderosa y muy utilizada en entornos gráficos, además permite crear aplicaciones orientadas a Internet desde la versión 4. Permite escribir código mientras compila el programa, siendo esto una gran ventaja. |
| Facilidad | Es un ambiente recomendado para principiantes. Fácil de usar. La navegación entre clases, funciones y ficheros es muy fácil, muy intuitivo.<br>El depurador funciona estupendamente, e indica los posibles errores en el manejo de memoria. | Es un lenguaje complicado y muy laborioso.<br><br>Entre sus principales ventajas de uso se encuentran las MFC (las Microsoft Foundation Classes). Estas son clases construidas por Microsoft (aunque su código está disponible y es abierto). Estas ahorran mucho trabajo ya que pueden usarse directamente en lugar de tener que construir nuevas clases.<br><br>Otra de las ventajas de Visual C++ son los Wizard. Un wizard es lo que en castellano se suele llamar asistente: un programa que ayuda a realizar una tarea. Visual C++ incorpora muchos Wizards que, en este caso, permiten construir programas automáticamente. | Es una herramienta muy fácil de utilizar, que se puede aprovechar sin leer manuales pues es totalmente intuitiva.<br><br>Este compilador de C es bastante potente ya que permite gestionar con no mucha dificultad programas con otros usuarios en red, unir ficheros y demás<br>a sus librerías.<br><br>Es fácil de aprender y tiene una ayuda bastante buena. |

*Análisis Comparativo Compiladores de C++ (Tabla 2/3 )*

| Características | DEV C++ | VISUAL C++ | BUILDER C++ |
|---|---|---|---|
| Interfaz Gráfica | No permite programación gráfica. Para poder desarrollar aplicaciones de este tipo es necesario instalar librerías especializadas tales como la librería winbgim.h que tiene como objetivo emular la librería graphics.h de Borland C++ | Es un lenguaje complicado, y en el que el programador tiene que hacer demasiado código para gestionar las ventanas que desarrolla. Sin embargo, cuenta con la gran ventaja de la arquitectura documento/vista: los datos están en una clase, y pueden ser presentados de diferentes formas a través de vistas<br><br>Para programación gráfica se tiene que trabajar con la API de Windows, lo cual hace mucho más pesada la realización de interfaces amigables. | Añade a la potencia del C y de la programación orientada a objetos (todo ello C++), Presenta la facilidad, de crear interfaces gráficas, a diferencia del Visual C++, que no ayuda nada en ese aspecto.<br>Dicha facilidad permite que el programador se ocupe de la realización del programa sin invertir mucho tiempo detalles gráficos.<br><br>Con el Builder, se pueden realizar desde aplicaciones que gestionen graficos (con el paintbox o shape entre otros) ,aplicaciones multimedia, bases de datos puesto que permite el uso de sql(el lenguaje del modelo relacional de las bases de datos), uso del corba; en definitiva un sin fin de aplicaciones.<br><br>Y es que la gran ventaja de cualquiera de estos programas, es que combinan un avanzado y cómodo entorno de diseño, en el que se puede hacer las ventanitas de la aplicación al gusto del programador. |

*Análisis Comparativo Compiladores de C++ (Tabla 3/3 )*

| Características | DEV C++ | VISUAL C++ | BUILDER C++ |
|---|---|---|---|
| Robustez/Estabilidad | Tiene menos beneficios que otros compiladores, es menos rico en librerías.<br><br>Debido a que tiene pocas librerías (las estándar y poco más), a veces da fallos de más (generalmente es un único fallo que se copia varias veces, y suele ser el primero)<br><br>NO tiene modo de ejecución paso a paso lo que complica un poco a la hora de corregir errores en tiempo de ejecución. | Visual C++ tiende a compilar menos basura que otros, lo cual lo hace muy eficiente.<br><br>Trae muchos ejecutables externos para hacer y debuggear los programas en desarrollo.<br><br>Se puede acceder a toda la API de Windows 32, tiene soporte para openGl ( librerias de C), Glide y DirectX (de Microsoft con lo cual se complementa muy bien con el compilador) | Este compilador es, el mas estándar eficiente y portable.<br><br>A pesar de ser una herramienta "a base botones" lo que se llaman componentes, no por ello pierde funcionalidad puesto que posee todas las características de potencia como optimizador de compilador, etc. Otra ventaja es que ocupa menos espacio en el disco que otros compiladores y esto siempre es bueno a la hora de optimizar el sistema. |
| Recursos | Genera ejecutables muy livianos | Genera ejecutables livianos | Genera ejecutables un poco grandes Sin embargo tiene como ventaja que al desensamblarlos no se ven los strings (cadenas) como en otros ejecutables, con lo que será más difícil "modificarlo" |

# APÉNDICE III

## MANUAL DE USUARIO

*Sistema de Control, Supervisión y Gestión de Redes basadas en IPv4.*

# APÉNDICE III
# MANUAL DE USUARIO

## I.- Nociones Básicas:

El Sistema de Control, Supervisión y Gestión de Redes, a partir de ahora llamado SCSG, corre bajo ambiente Windows, sin embargo puede gestionar estaciones soportadas bajo cualquier Sistema Operativo que tenga un servicio SNMP activo.

## I.1.- Activando el servicio SNMP en Windows XP.

Se recomienda que para familiarizarse con la herramienta, se empiece haciendo pruebas con estaciones que presten servicios de la capa de aplicación, es decir, ordenadores personales que tengan instalado Windows XP. Activar el servicio SNMP, es decir, abrir los puertos 161 y 162 en este sistema operativo, es sumamente sencillo, sólo se necesita el CD de instalación de Windows XP y seguir las siguientes instrucciones.

> *Introduzca el CD de instalación de Windows XP, presione Inicio, Panel de Control, Agregar o Quitar Programas, Agregar o Quitar Componente de Windows. Seleccione la opción Herramienta de Supervisión y Administración, y finalmente servicio SNMP.*

Ya se encuentra instalado el servicio SNMP, sólo falta activarlo:

> *Presione Inicio, Panel de Control, Rendimiento y Mantenimiento, Herramientas Administrativas, Servicios. Se desplegará una lista de todos los servicios activos que tiene el sistema operativo, debe seleccionar SNMP.*

Una vez que aparezca la pantalla inicial, deberá completar los parámetros como se muestra en la *figura 1.* Se recomienda utilizar tipo de inicio **Automático.** Presione el Botón Iniciar y ya estará corriendo un Agente SNMP en su computadora.

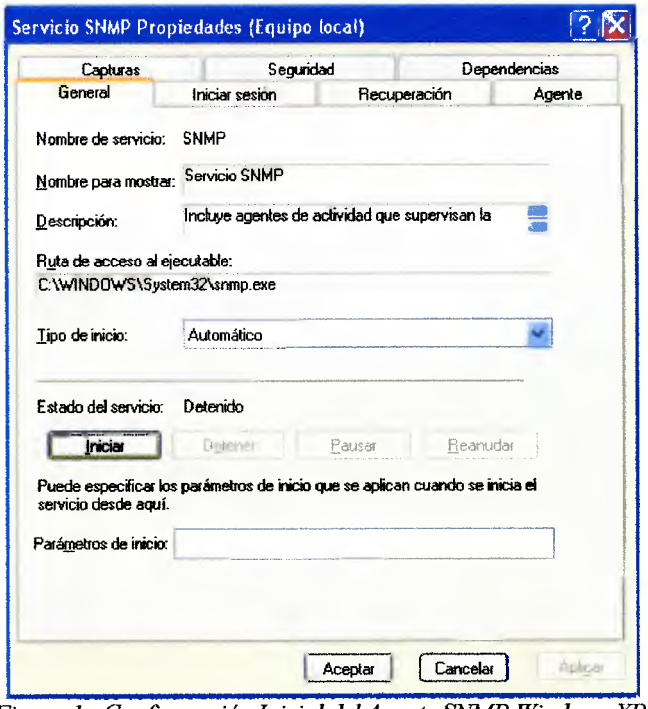



*Figura 1.- Configuración Inicial del Agente SNMP Windows XP.*
*Fuente: S.O. Windows XP.*

Sin embargo, el procedimiento no ha terminado. Ahora deberá configurar otras características para que SCSG pueda demostrar toda su potencialidad.

Seleccione la pestaña Agente y escriba el nombre de Contacto y la Ubicación del computador. El Sistema Operativo generalmente propone los datos que colocaron en el proceso de instalación del mismo. Si se desea modificarlos, se debe tomar en cuenta que así será visto por cualquier sistema de administración que trabaje con SNMP

El nombre de Contacto y Locación, son el contenido de las variables SysContact y SysLocation respectivamente del MIB de la estación en gestión.

Finalmente, tal y como se muestra en la *figura 2*, marque todas las opciones que se encuentran en el *frame de servicios*. De esta forma SCSG, podrá solicitar información de cualquier entorno de la estación (físico, aplicaciones, Internet, datos, entre otros)

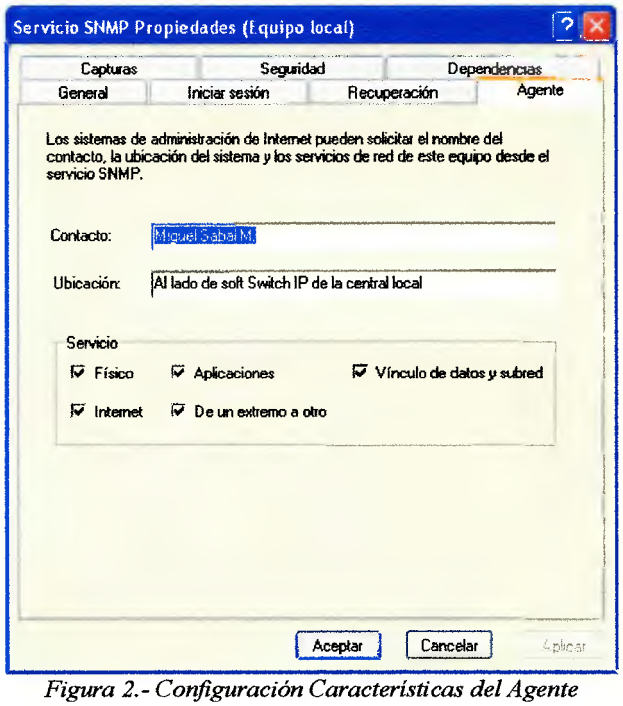



*Figura 2.- Configuración Características del Agente*
*Fuente: S.O. Windows XP.*

Para culminar con el proceso de instalación y activación del servicio SNMP, deberá seleccionar la pestaña CAPTURAS.

Las capturas son los conocidos TRAPS, o alertas que se generan en la estación y que se envían al sistema de supervisión cuando ocurre algún evento fuera de lo común. Es muy importante que se inicialicen estos valores para que SCSG escuche las alertas que se producen y tome las acciones necesarias. Para ello sólo deberá colocar el nombre de la comunidad (que por convención internacional se estila colocar public) y la dirección IP de la máquina sobre la cual está corriendo SCSG.

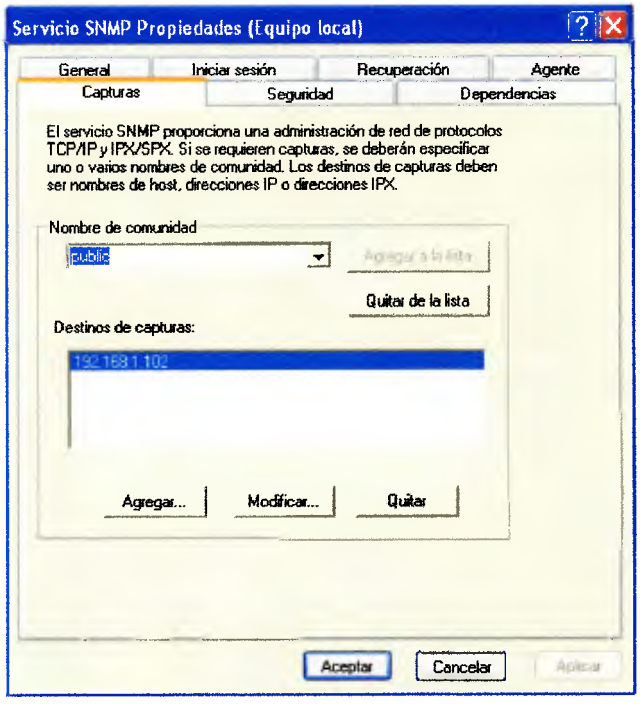



*Figura 3.- Configuración TRAPS del Agente*
*Fuente: S.O. Windows XP.*

Ya se encuentra activo el servicio SNMP con todos los parámetros inicializados correctamente, ahora deberá instalar SCSG para iniciar el proceso de Supervisión y Control de la Red.

## II.2.- Iniciando SCSG: Módulo de Sesión y Configuración

Una vez instalado SCSG, se presentará la primera pantalla del sistema. Es importante que inicialice los valores correctamente para que logre apreciar el funcionamiento de la herramienta. Una vez que se haya familiarizado, puede alterar los valores recomendados y empezar a disfrutar de las ventajas que ofrece SCSG

En la *figura 4*, se presenta la pantalla inicial. Se puede observar que el único botón activo es **COMENZAR;** antes de presionarlo, lea detenidamente la

descripción de la pantalla para conocer el significado general de las partes que la integran.

En la parte superior se encuentra un cuadro con la etiqueta **Eventos,** en la cual se irán reportando todas las acciones que se realicen durante el uso de la aplicación. Debajo de ésta se encuentran las pestañas de configuración de SCSG. La que se encuentra activa en la *figura 4* con la cual se carga la pantalla, es la relacionada con los nodos disponibles, es decir, aquellos que tienen un servicio SNMP activo y están conectados directamente al sistema. La otra pestaña, *Configurar Supervisión*, contiene los parámetros de los timer y el destino en el cual se almacenarán los archivos con los eventos ocurridos y finalmente la tercera pestaña, relacionada con el tipo de reporte que se desea recibir, si es Manual o Automático. En la columna derecha se encuentran los botones que activan los módulos de Control, Supervisión y Reportes de SCSG, que se detallarán más adelante.

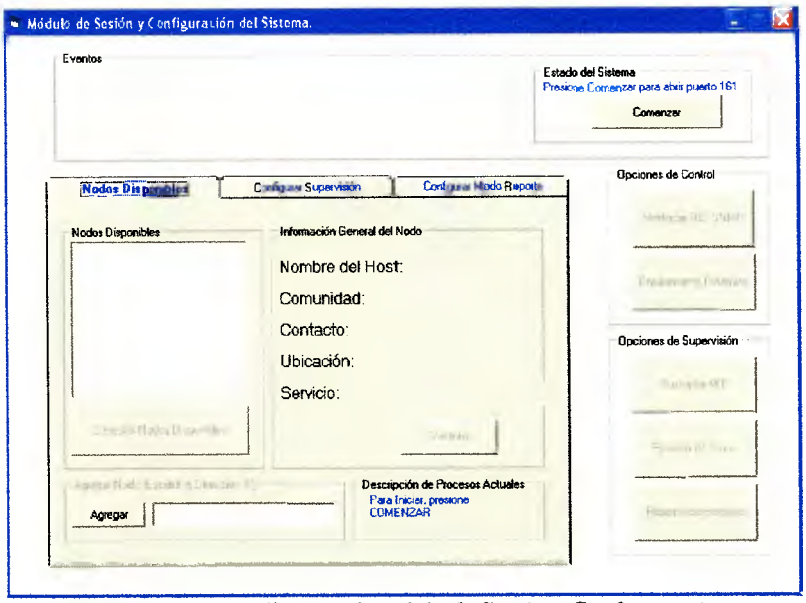



*Figura 4 .-Pantalla Inicial módulo de Sesión y Configuración*
*Fuente: SCSG*

## II.2.1- Módulo de Sesión y Configuración: Pestaña Nodos Activos.

Presione COMENZAR y en el cuadro de eventos observe el mensaje: *"Se ha iniciado el Sistema con éxito"*. Se habilitaron todos las opciones del Menu principal, además del botón actualizar lista.

Al presionar el botón ACTUALIZAR LISTA, aparecerán las direcciones IP de los nodos activos. Adicionalmente, se puede agregar de forma manual alguna estación, colocando en el cuadro de texto inferior la dirección de Internet, si es conocida.

En la Pestaña Nodos Activos, además de visualizar las direcciones IP de las estaciones, se presentan también las características generales de sistema de cada nodo en supervisión. Para ello sólo se debe seleccionar alguna de las estaciones de la lista y presionar el Botón VER DETALLE, como se muestra en la *Figura 5*.

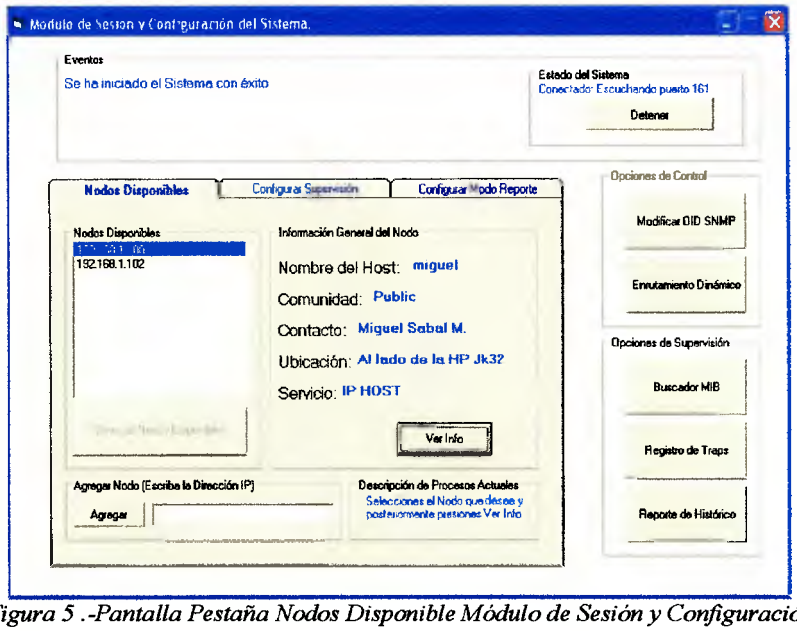



*Figura 5 .-Pantalla Pestaña Nodos Disponible Módulo de Sesión y Configuración*
*Fuente: SCSG*

## II.2.2- Módulo de Sesión y Configuración: Pestaña Configurar Supervisión.

En la presente pestaña se inicializan parámetros fundamentales para el proceso de supervisión. El primero de ellos es el *Intervalo de Tiempo* en el cual se va a realizar la encuesta secuencial de la información de cada nodo. El usuario puede seleccionar 10 seg, 30 seg, 1 min, 5min, 10 min y 30 min. Si por ejemplo, un usuario selecciona 10 segundos, entonces cada agente recibirá continuos mensajes SNMP de solicitud de información (GET Messages) cada 10 segundos.

Evidentemente colocar intervalos de tiempo tan pequeños va a generar una sobrecarga del sistema de gestión, más aún cuando se estén supervisando más de 10 nodos simultáneamente. Sin embargo, dependiendo de la situación específica y de la forma en que se quieran ver los reportes, se puede variar este parámetro. Para un comportamiento estable de la red, se recomienda mantener el TIMER en 10 minutos, es un lapso considerablemente confiable y permite conocer en términos generales el comportamiento de la red.

En la pestaña *Configurar Supervisión* también se selecciona el camino (Path) del archivo de texto que va a almacenar los reportes de las capturas o Traps que se generen durante el proceso de supervisión. Es importante recordar que los Traps o capturas, son aquellos mensajes SNMP que envían los agentes, sin previa solicitud del administrador, para notificar que algo fuera de lo común está ocurriendo.

En la *figura 6* se muestra la pestaña *Configurar Supervisión*, tal y como se describió anteriormente. Se aprecia en la parte inferior de la pantalla la ayuda interactiva que acompaña al usuario en el recorrido por la aplicación. En este caso, la ayuda le indica el status del timer seleccionado y le explica el significado de la acción.

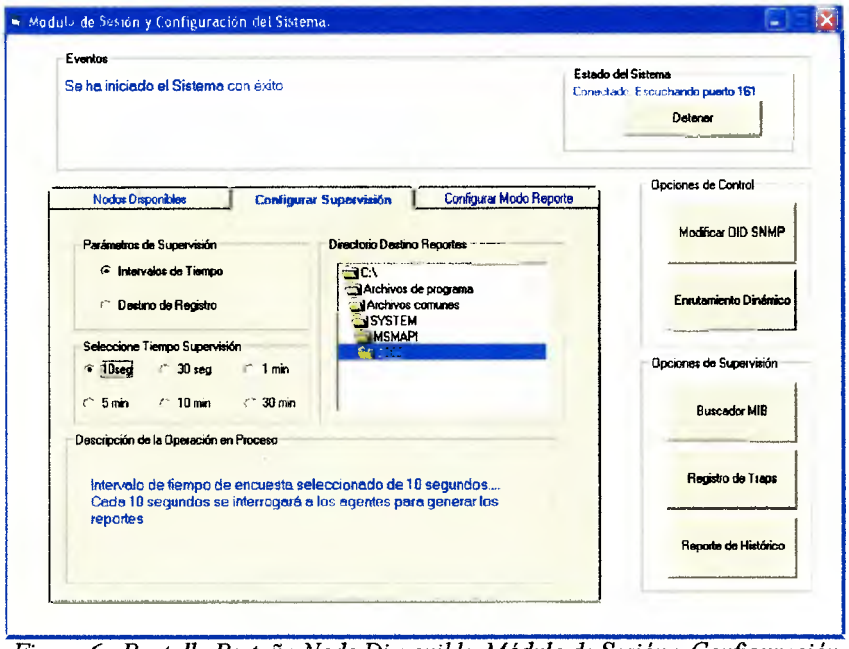



*Figura 6.- Pantalla Pestaña Nodo Disponible .Módulo de Sesión y Configuración*
*Fuente: SCSG*

## II.2.3- Módulo de Sesión y Configuración: Pestaña Configurar Supervisión.

El Módulo de Sesión y Configuración consta finalmente con la pestaña *Configurar Modo Reporte*, que permite que el usuario del sistema, en algunos casos el administrador de la red, seleccione si la supervisión se realizará de forma Manual (en frente del ordenador ) o automática.

Si selecciona AUTOMÁTICA, se habilita el cuadro de texto de inicialización de correo electrónico. El usuario tendrá la posibilidad de cambiar la dirección electrónica o mantenerla. En la etiqueta de status de la pantalla se escribirá entonces el resultado del proceso realizado. (el modo de reporte que se seleccionó, la dirección de correo destino a la que van a llegar los reportes TRAPS en caso de falla o congestión.)

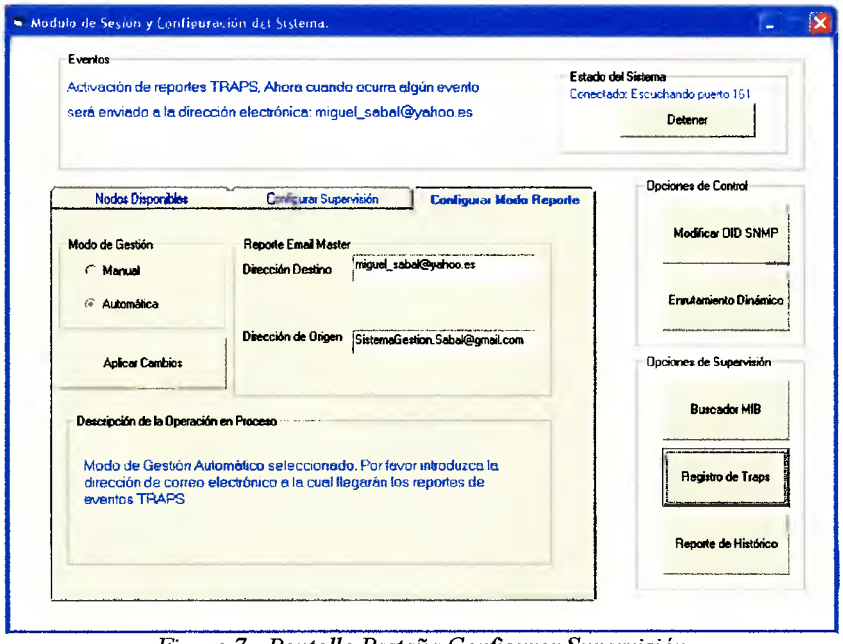



*Figura 7.- Pantalla Pestaña Configurar Supervisión*
*Fuente: SCSG*

Ya se completaron todos los pasos necesarios para la configuración inicial de parámetros de SCSG. Ahora, seleccionando cualquiera de los botones del menú que se encuentra en la columna derecha, se empezará a disfrutar de las ventajas que ofrece esta poderosa herramienta de Supervisión y Control de Redes.

Para iniciar con la navegación, haga clic en el botón MODIFICAR OID SNMP.

## II.3 Módulo de Control: Modificar Variable Remota

En la *figura 8* se muestra la pantalla estándar de la función Modificar Variable Remota, una de las más importantes del prototipo. En la esquina superior izquierda se encuentra el cuadro de autenticación del nodo, en otras palabras, el lugar donde se selecciona la dirección IP de la estación que se desea supervisar y el nombre de la comunidad. A su lado se encuentra la lista de las variables contenidas en el MIB de la estación seleccionada. Es una lista pre-compilada, que

se carga cuando aparece la pantalla. En la parte inferior la lista se encuentra una etiqueta que informa el tipo de acceso de cada variable (Lectura, escritura o ambas). Esta información de vital importancia, pues solo se pueden modificar las variables que sean de lectura y escritura simultáneamente. En la mitad de la pantalla se encuentra el cuadro de texto en que se escribe el nuevo valor de la variable.

La pantalla presenta adicionalmente la ayuda interactiva que le indica al usuario si el proceso se realizó con éxito o no. Generalmente, el mensaje es positivo si: La variable existe en la estación, el servicio SNMP permite modificar ese tipo de dato (porque pudiese no tenerse acceso a algun recurso) y evidentemente si esta variable es de escritura.


*Figura 8.- Pantalla Modificar OID SNMP*
*Fuente: SCSG*

**PASO 1:** Seleccione la estación a la que desea modificar una variable. Tal y como se muestra en la *figura 9,* se debe desplazar por la lista que contiene las direcciones IP de todas las estaciones que se encuentran activas. Si no recuerda específicamente la estación a través de su IP, se le recomienda volver a la pantalla inicial, perteneciente al módulo de sesión y configuración, pestaña nodos activos y buscar los datos generales del sistema, que le permitirán identificar fácilmente la estación que desea.

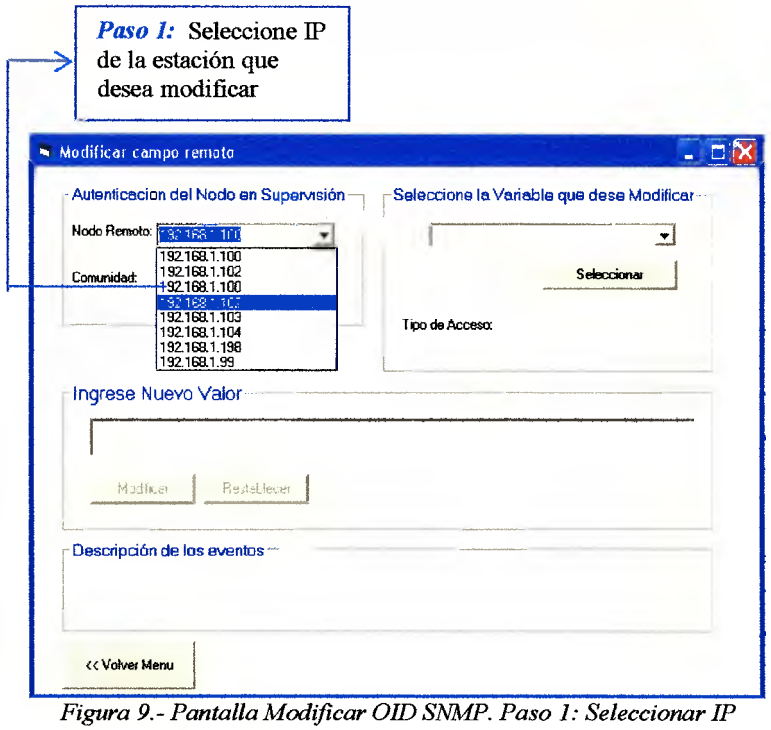



*Figura 9.- Pantalla Modificar OID SNMP. Paso 1: Seleccionar IP*
*Fuente: SCSG*

**PASO 2:** Coloque el nombre de la comunidad. Este valor, es por defecto "PUBLIC" a menos que por medidas de seguridad el administrador de la red decida modificarlo en todas las estaciones de la red. Puede haber valores de comunidad distintos para cada estación, pero no es recomendable porque es más complejo manejar una matriz de valores distintos.

**PASO 3:** Seleccione la variable que desea modificar. Recuerde estar alerta al mensaje que se presenta en la etiqueta ubicada en la parte inferior de la lista. Si la variable es de sólo lectura no se podrá alterar su contenido.

**PASO 4:** Presione el botón SELECCIONAR. Una vez que marque la variable en la lista, deberá pulsar el botón seleccionar. De otra forma no se agregará la variable al PDU del mensaje SNMP, por tanto, se enviará un SET REQUEST vacío. Una vez que realice este procedimiento, se activarán los botones MODIFICAR y RESTABLECER

**PASO 5:** Introduzca el nuevo valor de la variable seleccionada y presione MODIFICAR. Si desea recuperar el valor anterior, pues se equivocó durante el proceso de modificación, presione RESTABLECER.

*Nota: sólo podrá restablecer el valor anterior al cual realizó la operación pues éste se mantiene en una variable temporal, diseñada por cuestiones de seguridad.*

**PASO 6:** Verifique el status de su operación en la etiqueta descripción de los eventos que aparece en la base de la pantalla y que forma parte de la ayuda interactiva de SCSG.

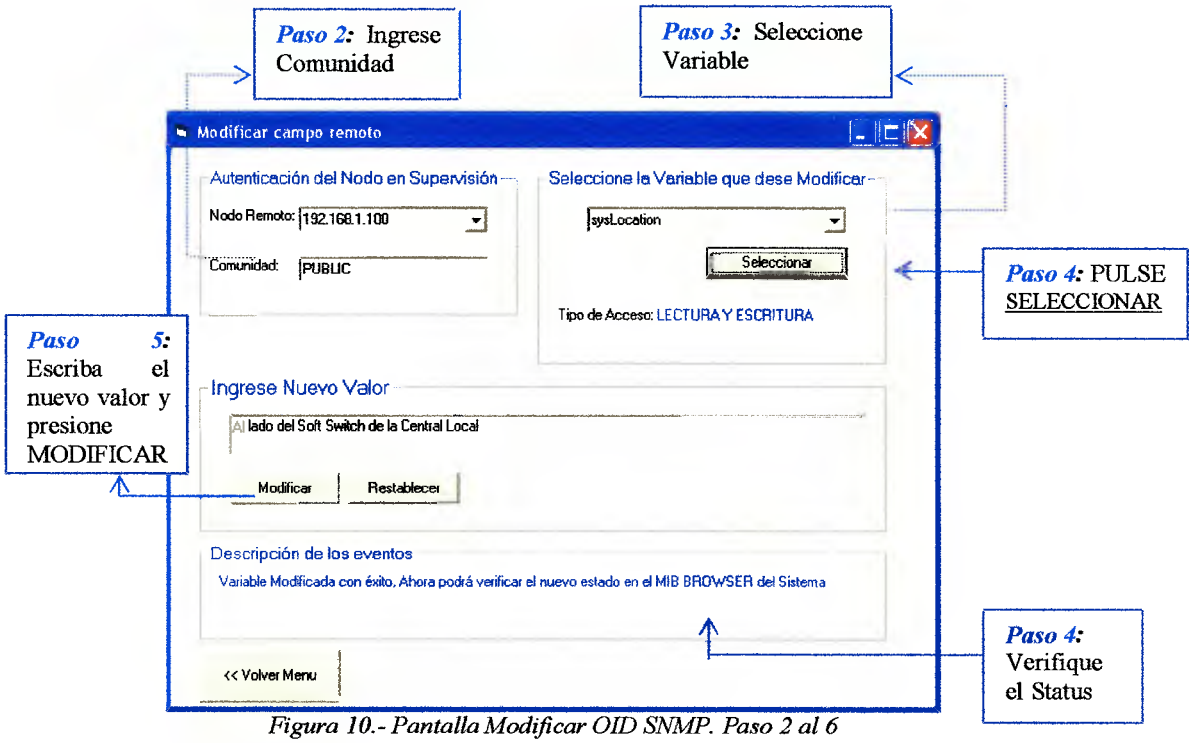



*Figura 10.- Pantalla Modificar OID SNMP. Paso 2 al 6*
*Fuente: SCSG*

## II.4 Módulo de Supervisión: Buscador MIB

La pantalla que se muestra en la *figura 10* representa la herramienta denominada "Buscador MIB" la cual permite verificar el valor de cualquier OID contenido en el MIB de determinada estación, pero adicionalmente, tiene la capacidad de desplegar por pantalla la explicación del significado de cada variable. Esta herramienta facilita considerablemente el proceso de supervisión pues se tiene acceso a la información contenida en más de 1003 variables.

En la *figura 10* se muestra la pantalla estándar del Buscador MIB, que se procederá a describir brevemente: En la esquina superior izquierda se ubica el cuadro de autenticación del nodo que se desea supervisar, con las mismas características de la pantalla descrita anteriormente (dirección IP y comunidad).

Debajo de este cuadro se encuentra la lista de variables que se desean visualizar junto a dos botones: Limpiar y Enviar. Del lado derecho se ubican las etiquetas que despliegan la información relacionada tanto del status del mensaje de solicitud (GETRequest) como la descripción teórica de la variable seleccionada. Adicionalmente, el botón VER DETALLE , activa la etiqueta localizada en la base de la pantalla y que representa la ayuda interactiva de este módulo. Finalmente, en el centro se ubican la lista de variables seleccionadas y las respuestas recibidas de la estación en supervisión, son los valores solicitados.

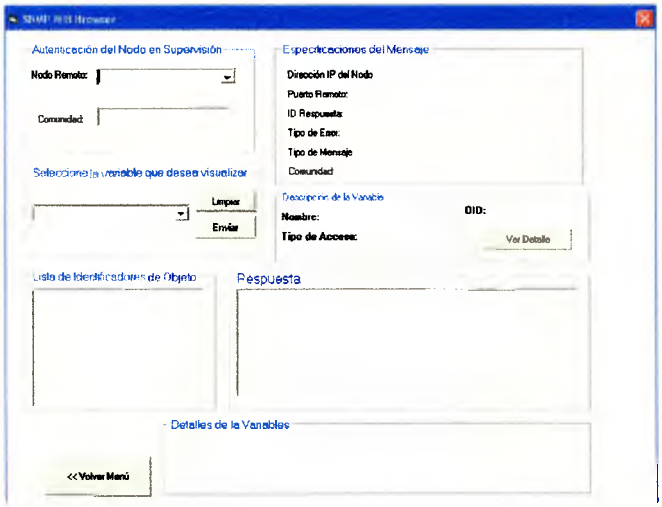



*Figura 11.- Pantalla Estándar Buscador MIB*
*Fuente: SCSG*

Una vez identificadas las partes fundamentales del Buscador MIB, se procederá a continuación a la explicación del procedimiento para solicitar valores a las estaciones en supervisión.

**PASO 1:** Tal y como se realizó en la pantalla de Control, en la lista de estaciones disponibles deberá seleccionar la dirección IP del nodo con el cual desea trabajar y escribir el nombre de la comunidad.

**PASO 2:** Seleccione la variable que desea supervisar. Al marcar cualquiera de las variables que se encuentran en la lista, éstas se incluirán en el cuadro de texto que se refiere a variables seleccionadas.

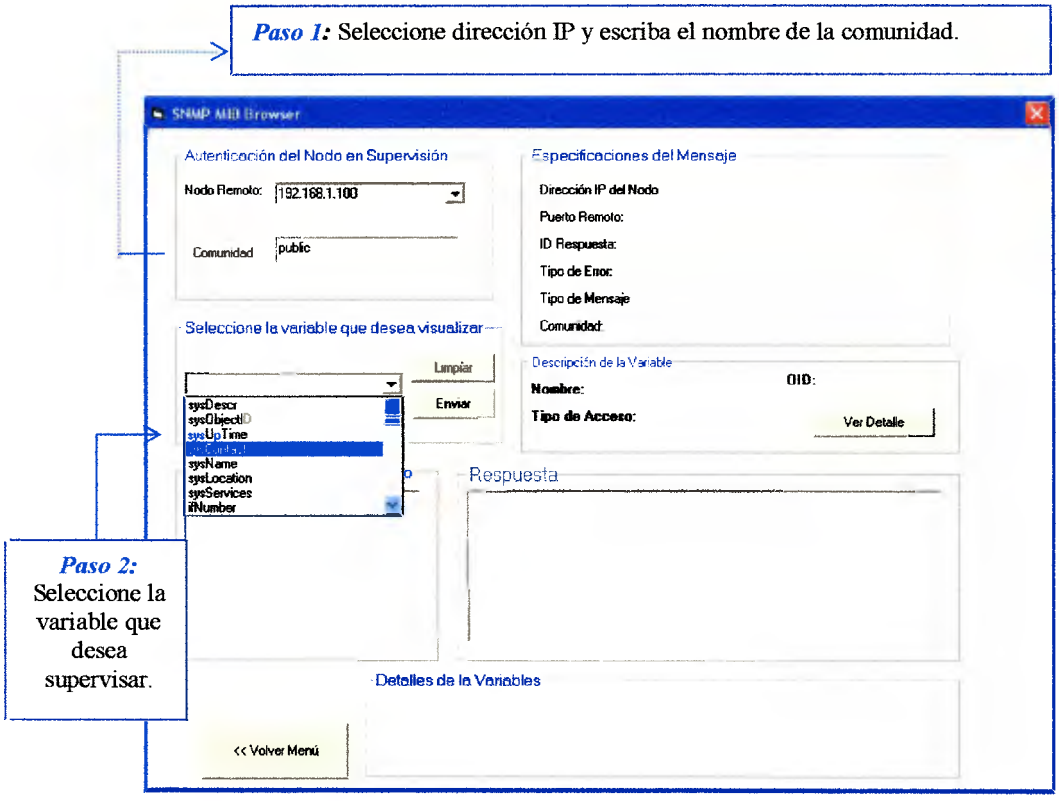



*Figura 12.- Buscador MIB. Pasos 1 y 2*
*Fuente: SCSG*

**PASO 3:** Presione ENVIAR y verifique en el cuadro de respuesta el valor de la variable seleccionada. Visualice también el estatus del mensaje ubicado en las etiquetas que se encuentra en la parte superior de la pantalla.

**PASO 4:** La función VER DETALLE se puede utilizar antes o después de enviar la solicitud de información a la estación, pues ésta se encarga de

presentar en la etiqueta de ayuda interactiva la explicación de la variable seleccionada.

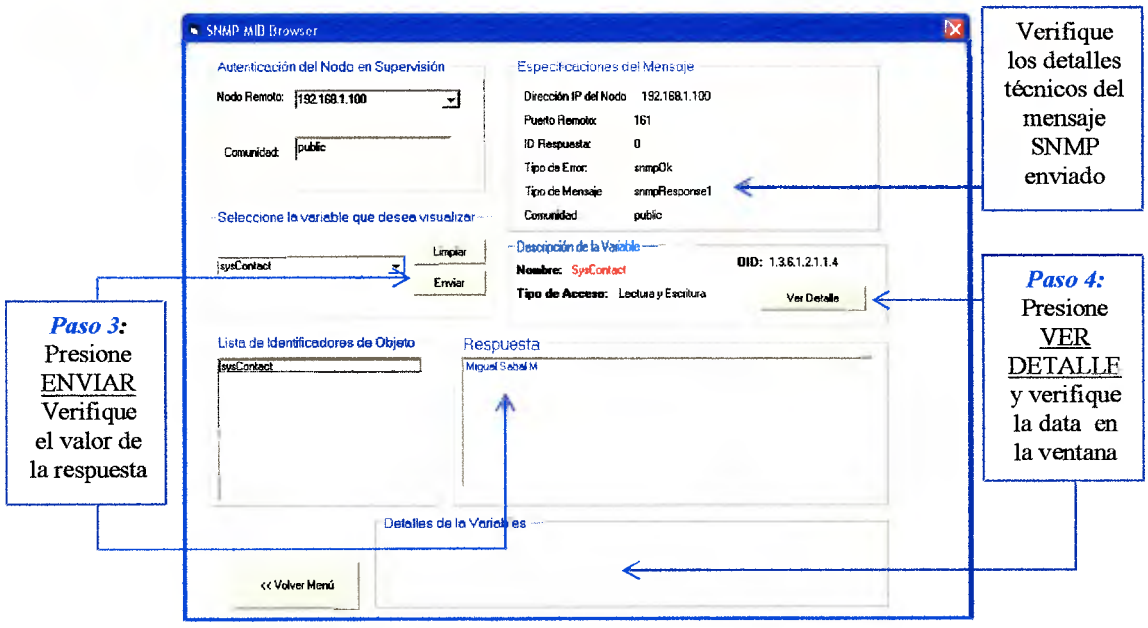



*Figura 13.- Buscador MIB. Pasos 3 y 4*
*Fuente: SCSG*

## II.5 Módulo de Reportes

La pantalla que se muestra en la *figura 13* es una de las más importantes de SCSG, ya que a través de ella se pueden visualizar todos los reportes obtenidos de los procesos de supervisión y control. En la columna izquierda se presenta el Menú principal con 5 opciones: Interface de Red, Memoria Física y Dispositivos, Carga del CPU y Memoria, Latencia de la Red y finalmente, Tabla MIB SNMP. Como en las demás pantallas descritas en las secciones anteriores del MANUAL DE USUARIO, en la base de la pantalla se presenta un cuadro que indica los pasos que se deben seguir durante la utilización de SCSG.


*Figura 14.- Pantalla de Reportes*
*Fuente: SCSG*

Como se muestra en la *figura 14*, debajo del Menú de Opciones se presenta un Sub-Menu específico para cada alternativa. En otras palabras, si selecciona la opción Interfaces de Red, se desplegará un Sub-Menu con los cuatros reportes específicos relacionados con el comportamiento de las interfaces de red. En este sentido, el módulo ofrece al usuario más de 20 reportes completos, presentados en forma de gráficos de barras, tabla de datos, listas, entre otros.


*Figura 15.- Pantalla de Reportes con Sub Menú*
*Fuente: SCSG*

A continuación se procederá a la explicación detallada del funcionamiento del módulo de reportes.

**PASO 1:** Seleccione la dirección IP de la cual desea visualizar el reporte. Verifique si es la estación correcta comparando con el nombre del sistema que se muestra en la cabecera de la pantalla.

**PASO 2:** Seleccione la Opción del Menú Principal, posteriormente verifique el Sub Menú que se despliega en la parte inferior de la pantalla. Seleccione la opción que requiera.

**PASO 3:** Visualice su reporte en la "Ventana de Reportes" ubicada en el centro de la pantalla. Cualquier inquietud la ayuda interactiva ubicada en la base le indicará el procedimiento que debe llevar a cabo.

En la *figura 15* que se muestra a continuación, se seleccionó la opción Interfaces de Red en el Menú Principal y la opción Ancho de Banda Consumido en el Sub-Menú de opciones.

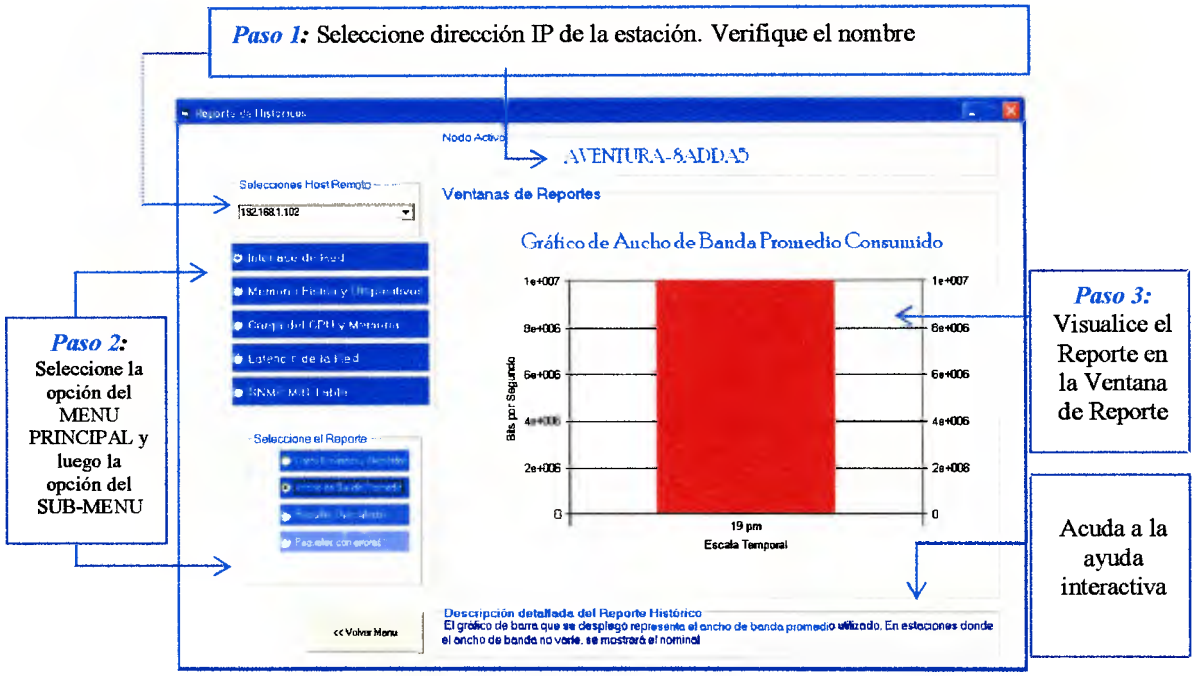



*Figura 16.- Pantalla de Reportes. Paso 1,2 y 3. Ancho de Banda Consumido*
*Fuente: SCSG*

Una de las grandes ventajas que presenta el módulo de reportes es que la escala temporal de los gráficos es dinámica, lo cual significa que si visualizo el reporte de una determinada estación a la hora 0, será completamente distinto que al visualizarlo a la hora 4. El proceso de supervisión de SCSG tiene una capacidad de generar reportes de hasta 48 horas continuas.

En la *figura 16* se muestra un reporte diferente al apreciado anteriormente. En este caso se seleccionó la opción Memoria y Dispositivos del Menú Principal y la alternativa Dispositivos Periféricos del Sub-Menú de opciones. En la pantalla se puede apreciar la lista de todos los dispositivos periféricos que están instalados en la estación en supervisión. Entiéndase por dispositivos periféricos todos aquellos que proveen de información al ser humano tales como impresoras, unidades de memoria, entre otros.


*Figura 17.- Pantalla de Reportes. Dispositivos Periféricos*
*Fuente: SCSG*

Por último, con el objetivo de mostrar otro de los reportes que ofrece SCSG, en la figura 17 presenta la tabla de MIB SNMP, referida específicamente a los valores del sistema. A través de esta opción, se pueden visualizar los 1003 valores que ofrece el MIB de una estación. Siendo esto de gran utilidad para los Administradores de red que dominan el lenguaje ASN 1.

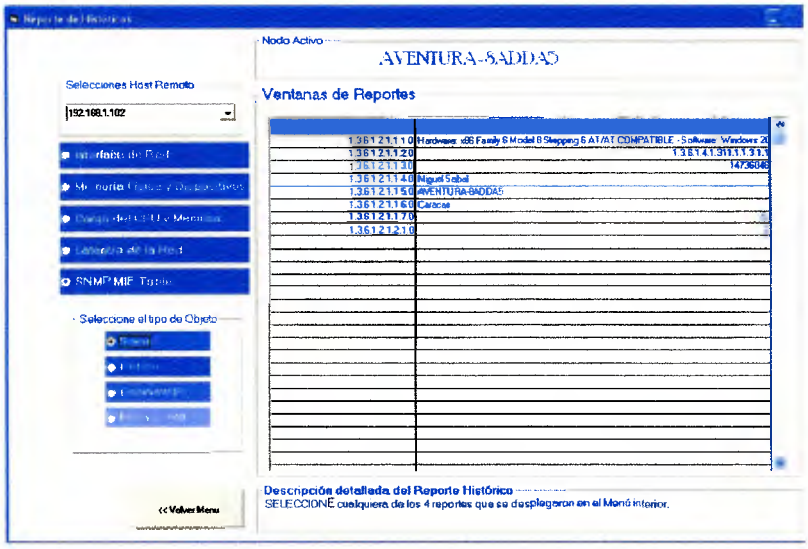



*Figura 18.- Pantalla de Reportes.SNMP MIB TABLE. Valores del Sistema.*
*Fuente: SCSG*

## II.5 Disposiciones Finales y Recomendaciones de Uso.

Después de haber realizado el recorrido por el funcionamiento de las 4 pantallas del Sistema de Control, Supervisión y gestión de Redes, se pudo apreciar que es una herramienta poderosa, capaz de generar reportes de históricos a través de una interfaz amigable y cuyo uso es bastante sencillo.

No se requiere de conocimientos muy técnicos para entender los resultados generados, lo cual hace de la herramienta un producto verdaderamente competitivo.

Se recomienda la utilización del presenta Manual de Usuario durante los primeros 3 días de aplicación y empezar la supervisión de estaciones en ordenadores personales con Sistema Operativo Windows, instalando el servicio SNMP, como se indicó en las nociones básicas del presente manual.

En caso de presentarse alguna inquietud o falla del sistema, puede consultar a través del correo electrónico miguel_sabal@yahoo.es

# APÉNDICE IV

## MANUAL DEL SISTEMA

*SISTEMA DE CONTROL, SUPERVISIÓN Y GESTIÓN DE REDES BASADAS EN IPV4.*

# APÉNDICE IV
# MANUAL DE SISTEMA

## I.- Requerimientos de Software para la Implementación del Prototipo.

- Windows 2000, XP, NT o superior.
- Lenguaje de Programación Visual Basic 6.0
- Librería Power SNMP.

## II.- Requerimientos de Hardware

- Procesador Pentium III o superior
- Memoria Ram mínimo 512 Megabytes
- Disco Duro mínimo 15 GygaBytes.
- Interfaz de red RJ45

## III.- De la Herramienta Power SNMP

La librería a través de la cual se realiza la comunicación vía SNMP es un desarrollo de la empresa Dart Communications, dedicada a la implementación de herramientas basadas en controles ActiveX.

Dart Communication da soporte a ambientes de desarrollo muy extendidos como Visual Studios, Delphi, Java, entre otros y su confiabilidad se ha consolidado por muchos años de amplia y reconocida experiencia.

La librería PowerSNMP, provee las instrucciones necesarias para la implementación de las 7 funciones del protocolo SNMP: Get, Set, GetNext, GetBulk, GetInform, GetResponse y Traps. Además provee de un objeto *Manager* que administra e invoca todas esas funciones.

La librería PowerSNMP tiene un costo de $19 por 365 días de uso, por tanto resulta sumamente económica y sus funciones facilitan considerablemente la implementación del protocolo de gestión y supervisión de redes SNMP.

Para cualquier información adicional de la librería se puede consultar la página www.dart.com.

## IV.- Nociones Generales del Diseño.

En la *Figura 1,* se muestra el diseño del sistema y la relación existente entre cada uno de los módulos funcionales, el Objeto Agentes, los métodos del ambiente de programación provenientes de la librería importada *PowerSNMP* y los elementos del entorno que permiten la interconexión a nivel de software de todos los factores que intervienen.

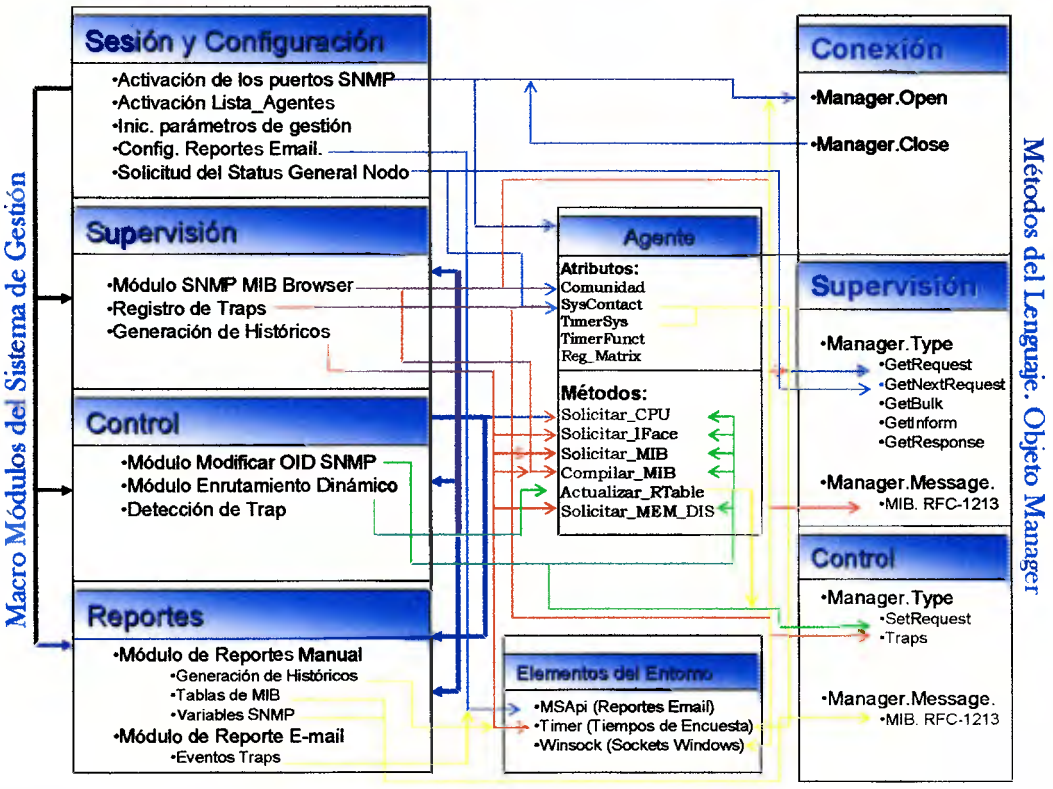



*Figura 1. Diseño General del Sistema de Control, Supervisión y Gestión*
*Fuente: Elaboración Propia*

El diseño general presentado, modela el funcionamiento completo del sistema. Los 4 módulos funcionales, Sesión, Supervisión, Control y Reportes, invocan los métodos del Objeto Agentes y éstos a su vez utilizan las funciones específicas del lenguaje de programación y de la librería *PowerSNMP*.

En el desarrollo del Trabajo Especial de Grado se explica detalladamente las funciones de cada uno de los módulos que forman parte del sistema. El objetivo del presente manual, es describir de la forma más detallada posible el funcionamiento a nivel de código de cada uno de los módulos que integran el prototipo.

# V. Módulo de Sesión y Configuración

El módulo de Sesión y Configuración se divide explícitamente (a través de pestañas en la interfaz), en tres fases de inicialización. *La Fase 1* se refiere a la actualización y despliegue de los nodos que se encuentran activos en la red. *La Fase 2* a la fijación de los parámetros de supervisión y la *Fase 3* configura el modo de reporte (Manual o automático). A continuación se describe el funcionamiento de cada una de las Fases del Módulo de Sesión y Configuración

## V.1. Fase 1: Actualización y despliegue de los nodos activos.

En la figura 2 y 3 se presenta el diagrama de colaboración y de secuencia respectivamente de la *Fase 1*, donde se puede apreciar que una vez que el usuario decide activar los puertos de comunicación SNMP (161 y 162) y se inicialice el sistema, se pueden llevar a cabo dos acciones, la primera es agregar una estación a la red de forma manual a través de la dirección IP de la misma y la segunda, solicitar al prototipo envíe un Broadcast, espere la respuesta de las estaciones activas y actualice la lista de nodos en gestión que se despliega por pantalla. La respuesta de cualquier estación, activa una función denominada *Manager_Response,* la cual se invoca siempre que exista un mensaje de tipo snmpGetResponse, por parte de los agentes en supervisión.

En la Fase 1, a través del uso de la función GetNextRequest del protocolo SNMP, se solicitan los valores generales del nodo seleccionado y se presentan por pantalla. Estos valores iniciales que se solicitan a cualquiera de las estaciones en gestión son:

- Nombre del Sistema
- Comunidad
- Contacto

- Ubicación
- Tipo de servicio

La *Fase 1* no está concebida dentro del módulo de supervisión, sin embargo se realiza la solicitud de las variables contenidas en el "ramal sistema" del MIB, para introducir al usuario las características esenciales de cada uno de los nodos que se encuentran activos y así identificar de forma rápida, oportuna y eficiente alguna estación en específico que se sospeche esté generando conflictos en la topología de la red.

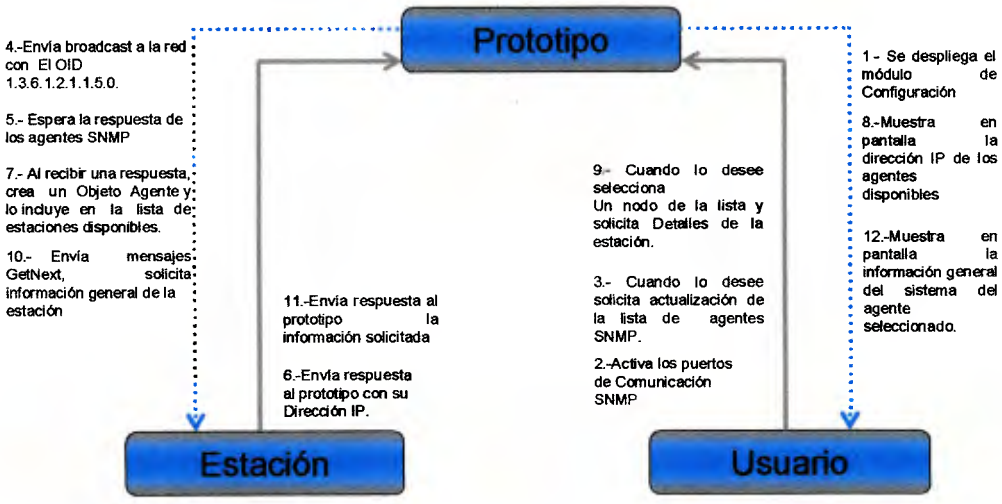



*Figura 2. Diagrama de Colaboración Fase 1 del Módulo de Sesión y ConfiguraciónFuente: Elaboración propia*

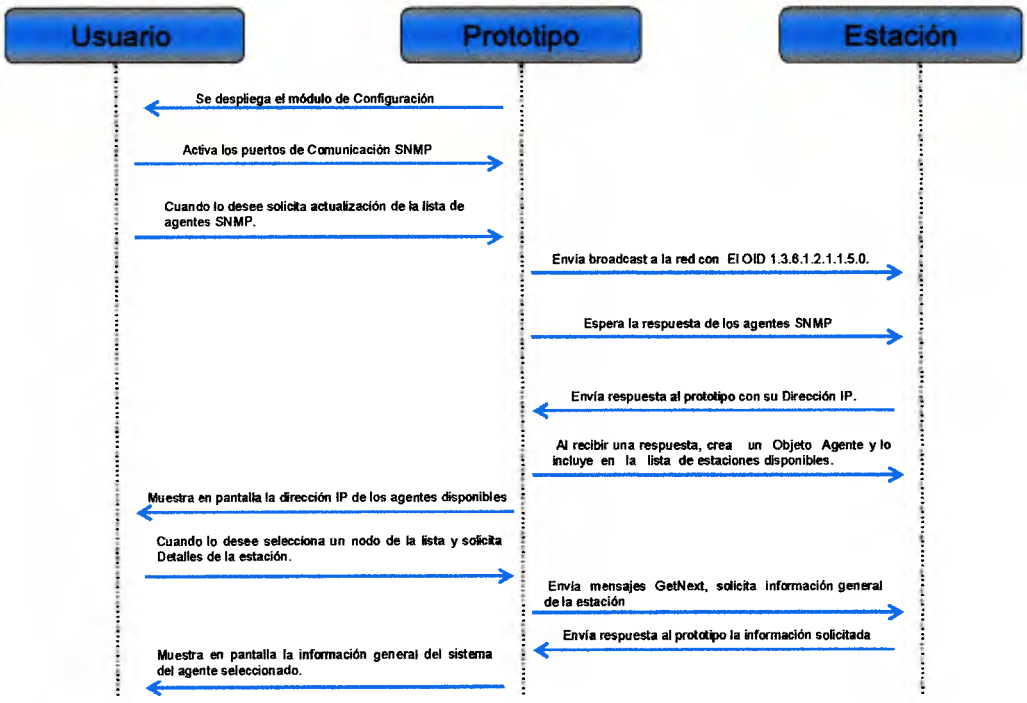



*Figura 3. Diagrama de Secuencia Fase 1 del Módulo de Sesión y Configuración*
*Fuente: Elaboración propia*

Antes de describir el funcionamiento de las siguientes fases, es importante complementar la definición del mensaje GetNextRequest utilizado en la fase 1 para la obtención de los valores del sistema, pero que sin duda será la función estrella en los procedimientos de supervisión que se explicarán más adelante.

Es frecuente que surja la duda de por qué utilizar una función que solicite el siguiente valor del que se le pase como parámetro, ¿por qué no pasarle directamente la variable que se desea obtener?. La respuesta es muy simple y de ella depende el entendimiento de las funciones de supervisión. La función GetRequest solamente puede obtener el valor de los nodos hijos de primera generación en un determinado árbol MIB. Eso se debe a que la notación de los MIB de cada estación tiende a variar y no existe un Identificador de Objeto (OID) específico para cada variable.

A través del árbol que provee la herramienta MG-SOFT LAB se puede visualizar más fácilmente la explicación que se está realizando.

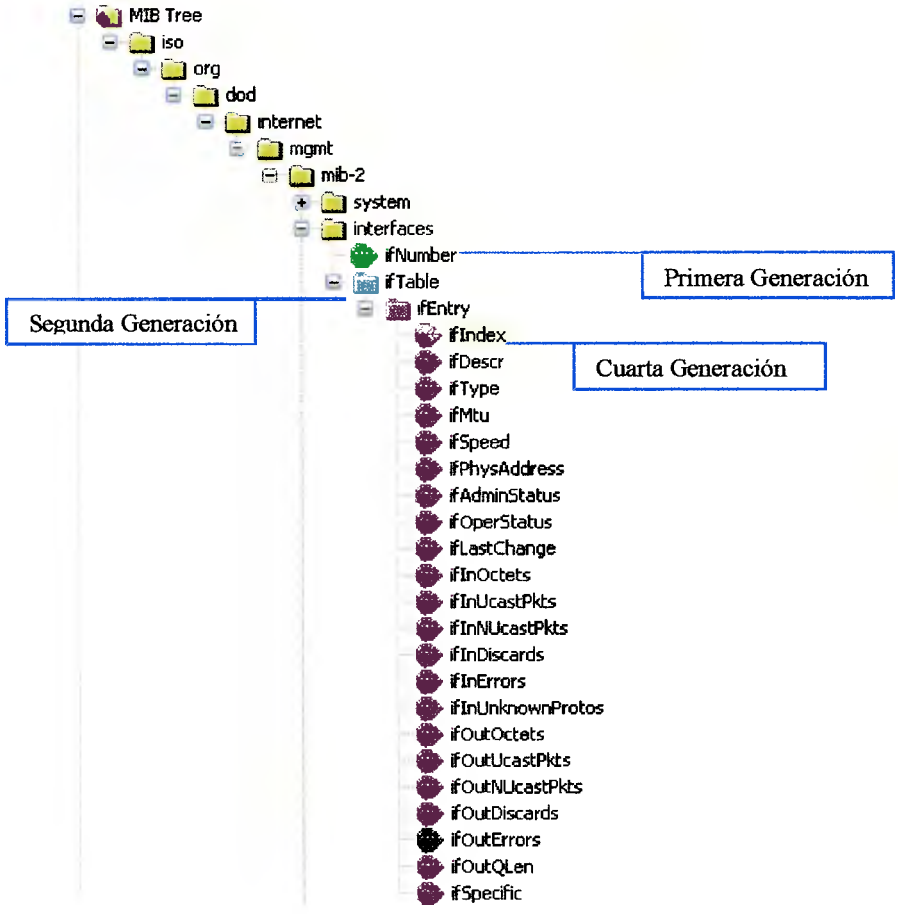



*Figura 4 . Árbol MIB de Identificadores de Objeto*
*Fuente: MG-SOFTLAB*

Como se puede apreciar en la *Figura 4*, el nodo de primera generación de la categoría Interfaces es IfNumber; el cual puede ser solicitado directamente haciendo uso de la función GetRequest. Sin embargo, para acceder por ejemplo a la variable IfIndex, que es un hijo de la cuarta generación, se requiere la utilización de la función GetNextRequest, a la cual se le pasa como parámetro el valor de IfNumber y se realiza un *loop* de tantas repeticiones como saltos existan en el árbol.

El manejo de la función GetNextRequest parece complejo, pero con la práctica y un poco de abstracción se convierte en un procedimiento de rutina.

Explicado el funcionamiento y significado de la función GetNextRequest se continuará la descripción técnica de las fases restantes del módulo de Sesión y Configuración.

## V.2. Fase 2: Configuración de Supervisión

Una vez actualizada o inicializada la lista de agentes disponibles, el prototipo habilita las demás pestañas pertenecientes al Módulo de Sesión y Configuración. La *Fase 2*, está representada en los diagramas de colaboración y secuencia que se muestran en las Figuras 5 y 6. Esta fase no interactúa con la estación, pues como se puede apreciar, en ésta sólo se definen los parámetros generales que enmarcarán el proceso de supervisión, descrito en el próximo módulo.

El primer valor que se inicializa es el tiempo asignado a cada *Objeto Agente* de la lista de estaciones disponibles para realizar su encuesta. Como se explicó en el capítulo IV del desarrollo del proyecto, cada *Objeto Agente* tiene un intervalo de tiempo para realizar distintas peticiones a la estación que representa dentro del modelo y a su vez, cada petición (GET) debe realizarse en otro intervalo de tiempo específico. Esa compleja asignación temporal que soporta al módulo de supervisión del prototipo, se define en esta fase 2 del Módulo de Sesión y Configuración.

Es importante destacar que la asignación de los tiempos de encuesta secuencial deben tomar en cuenta las características del reporte que se desea realizar, ya que depende de este elemento la cantidad de procesamiento que muestre la aplicación. En otras palabras, si se selecciona un tiempo de encuesta muy pequeño (cada 15 segundos), es necesario tener conciencia que la herramienta enviará al menos 20 peticiones por agente activo, cada 15 segundos. Si se está supervisando

una red de 60 estaciones, la herramienta pudiese monopolizar totalmente el CPU de la máquina y traer consecuencias negativas en el *performance* de la aplicación. Evidentemente que si el prototipo presenta la opción de tiempos de encuestas tan reducidos es porque existe alguna argumentación y ésta es, que ocurridos determinados eventos (como congestión en una interfaz de red que sirve de estación puente), pudiese ser necesario visualizar el comportamiento de la estación en tiempo real y las encuestas enviadas a alta frecuencia, ofrecen esa facilidad.

Finalmente, en la fase 2 se selecciona el *Path* o ruta de acceso del archivo de texto que almacena los eventos Traps que se generan durante el proceso de supervisión, como se muestra en las figuras 5 y 6.

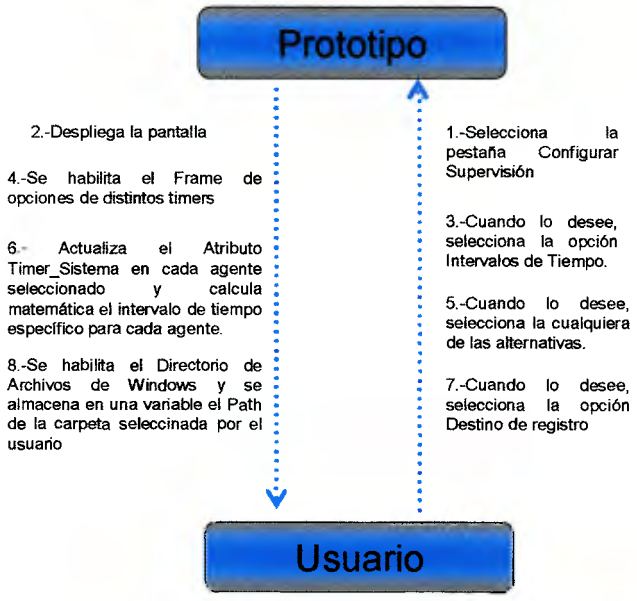



*Figura 5. Diagrama de Colaboración. Fase 2 Módulo de Sesión y Configuración*
*Fuente: Elaboración Propia.*


*Figura 6. Diagrama de Secuencia. Fase 2 Módulo de Sesión y Configuración*
*Fuente: Elaboración Propia.*

## V.3. Fase 3: Configuración Modo Reporte

La *Fase 3* del Módulo de Sesión y Configuración se encarga de inicializar el modo de reporte que genera el sistema: Manual o Automático. En la *figura 7* se muestra el diagrama de secuencia que sintetiza la iteración Usuario-Prototipo. Es una fase sencilla en la cual solo se valida el tipo de reporte. Si el usuario selecciona Manual, entonces se habilitarán todos los botones de la pantalla de reportes para poder visualizar los gráficos de históricos. Si selecciona Automático, significa que posiblemente el administrador de la red no se encontrará de forma presencial frente al Sistema de Control, Supervisión y Gestión, por tanto se inicializa el parámetro del correo electrónico destino, para recibir cualquier alarma de eventos que ocurran en alguna estación supervisada.

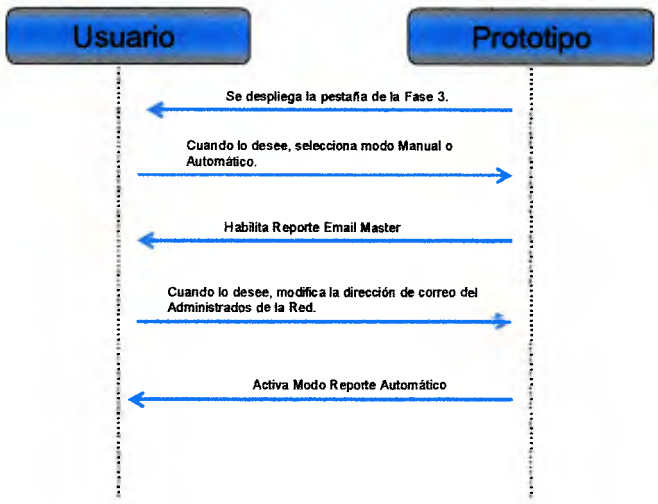



*Figura 7.- Diagrama de Secuencia Configuración Modo Reporte*
*Fuente: Elaboración Propia*

## VI. Módulo de Supervisión

El Módulo de Supervisión se representa en el prototipo a través de dos aplicaciones. La primera, facilita la obtención de diversas variables que se procesan y en conjunto con el módulo de reportes, generan gráficas amigables que permiten visualizar el comportamiento de la red. La segunda, consiste en un "Buscador MIB", que es una herramienta similar a la utilizada durante el desarrollo del proyecto (MG-SOFT LAB), que permite conocer el significado y el valor de los distintos identificadores de objetos (OID's) contenidos en el MIB de cualquier estación.

El diagrama de secuencia del módulo de supervisión se presenta en la *Figura 7.* En la cual se puede observar que una vez cargada la pantalla y actualizada la lista de agentes disponibles (la cual se inicializa al presionar el botón Buscador MIB, que se encuentra en el Menú Principal del módulo se Sesión y Configuración), el usuario deberá seleccionar el nodo que desea supervisar y colocar el nombre de la comunidad. Este procedimiento se denominó "Autenticación del Nodo en Gestión" y es usado en el resto de los módulos, por tanto sólo será descrito en este diagrama de secuencia.

El procedimiento de Autenticación del Nodo, es una especie de medida de seguridad que garantiza que quien está solicitando o inicializando valores a través de la herramienta es una persona autorizada, para ello se solicita el nombre de la comunidad, que por defecto siempre es "Public" pero puede ser modificada por el Administrador de la red. Cuando el mensaje SNMP con el GetRequest llegue al agente SNMP, éste no solo se encargará de decodificar el PDU y buscar en el MIB la variable seleccionada sino que también comparará el valor de la comunidad, para poder autorizar el procesamiento del mensaje.

Una vez alcanzados los pasos anteriores el agente busca en su archivo .MIB el identificador de objeto localizado en el mensaje SNMP recibido y lo asigna como parámetro a un mensaje Response que se envía de vuelta a la herramienta.

El sistema lee el mensaje invocando la función GetResponse, lo almacena en la lista de variables que posee el objeto Manager y se imprime por pantalla, tantos los valores solicitados, como el status del mensaje.

En este punto, sería conveniente detenerse y explicar con detalle dos posibles tipos de mensajes SNMP que puede recibir el sistema de gestión como respuesta de una estación. Uno de los más comunes es la frase NO_SUCH_NAME, que se refiere a que la variable solicitada no fue encontrada en la estación destino. Cuando ocurra ese error no se debe a una falla de programación de la herramienta ni un problema de comunicación. Simplemente el Buscador MIB, permite solicitar una lista de más de 1003 variables, sin embargo, es muy posible que los MIB de determinadas estaciones no estén actualizados o no sean los mismos de forma que no se encuentre un valor específico que se haya solicitado. Ese error ocurre frecuentemente con las variables relacionadas al Host_Status, es decir, la que indica los procesos y programas activos, dispositivos periféricos, unidades de almacenamiento etc, y la razón es porque el MIB que contiene esas variables está cargado en equipos fabricados del año 2004 en adelante.

En el diagrama de secuencia que se muestra en la figura 8 se sintetiza la explicación realizada anteriormente.

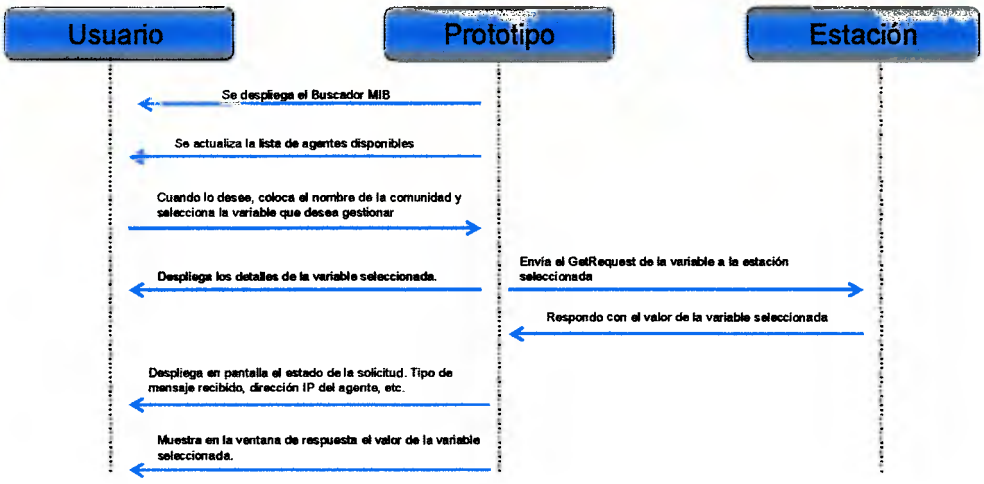



*Figura 13. Diagrama de Secuencia Módulo de Supervisión. Buscador MIB*
*Fuente: Elaboración Propia*

## VII. Módulo de control

El módulo de control también ofrece dos grandes funcionalidades: permite modificar el valor de cualquier variable de las estaciones activas, siempre que éstas sean de lectura y escritura. Y la segunda, que no se puede visualizar a través de ninguna pantalla, es la toma de decisiones que realiza el sistema en cuanto al enrutamiento del tráfico en caso de ocurrir cualquier falla o congestión de una estación, basado en la modificación de las rutas por defecto.

El diagrama de secuencia del módulo de control se presenta en la *Figura 9* La lógica computacional de este módulo es muy similar al descrito anteriormente con la diferencia que el mensaje SNMP utilizado es SetRequest y que el PDU de éste además de llevar el nombre de la variable lleva el nuevo valor que se desea asignar.

En este caso el agente no solo determina la autenticación del nodo (a través del nombre de la comunidad) sino que verifica si la variable que desea modificar el Sistema de Gestión es de escritura, de lo contrario en el mensaje de respuesta le indica que la operación no fue exitosa, pues no tiene acceso a ese tipo de datos.

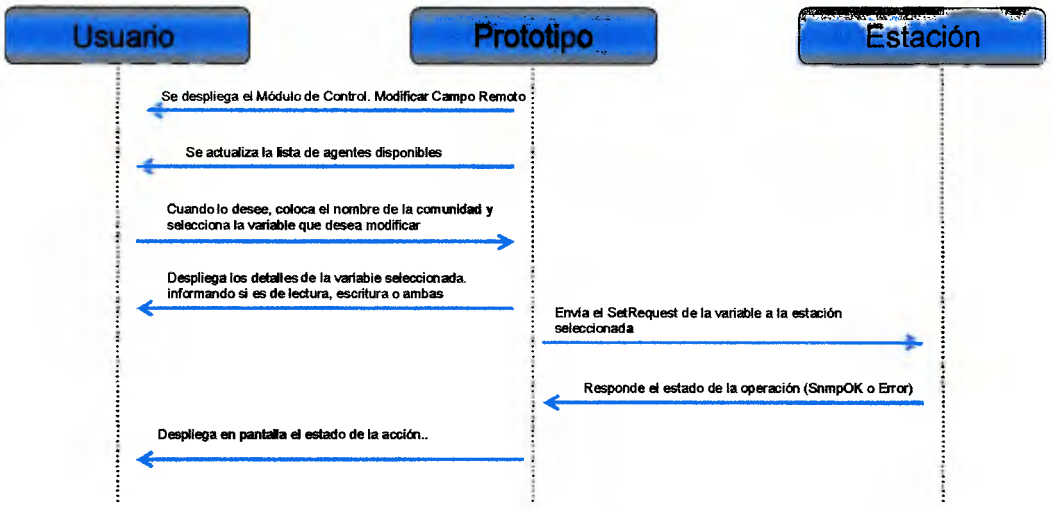



*Figura 9. Pantalla Estándar Módulo de Control. Modificar Campo Remoto*
*Fuente: Elaboración Propia*

## VIII. Módulo de Reportes

El módulo de reportes representa la interfaz más importante de todo el prototipo ya que a partir de éste se hace la demostración de todos los procesos de supervisión y control ejecutados por los módulos anteriores. Los reportes que se generan son: Interfaces de Red, Memoria Física y Dispositivos, Latencia, Consumo de CPU y el último que se refiera al SNMP MIB Table. El diagrama de secuencia del módulo de reportes se presenta en la *Figura 10.* Es un poco disímil a los diagramas de secuencia anteriores, aunque incluye algunos de los procesos descritos como autenticación del nodo. Lo más importante de la implementación del módulo de reportes es el proceso de encuesta secuencial que funciona junto con el módulo de supervisión y cuya asignación temporal fue explicada anteriormente.

Las encuestas se realizan, como se muestra en el diagrama de forma periódica y se van almacenando los valores obtenidos en una lista dinámica asignada a cada variable, esto es:

El Agente A, en el intervalo de tiempo asignado, solicita la información relacionada a un campo y almacena el valor obtenido en una lista asociada a ese campo, luego solicita la información de la siguiente variable y la almacena en otra lista asociada a esa variable. En total son 9 solicitudes por agente. Con los datos que se encuentran almacenados en la lista se calculan los promedios y se grafican los resultados.

Todo el procedimiento de encuesta secuencial descrito en el parráfo anterior se repite para cada agente según el intervalo de tiempo que se asigne en el módulo de Sesión y Configuración, Fase 2 explicado anteriormente.

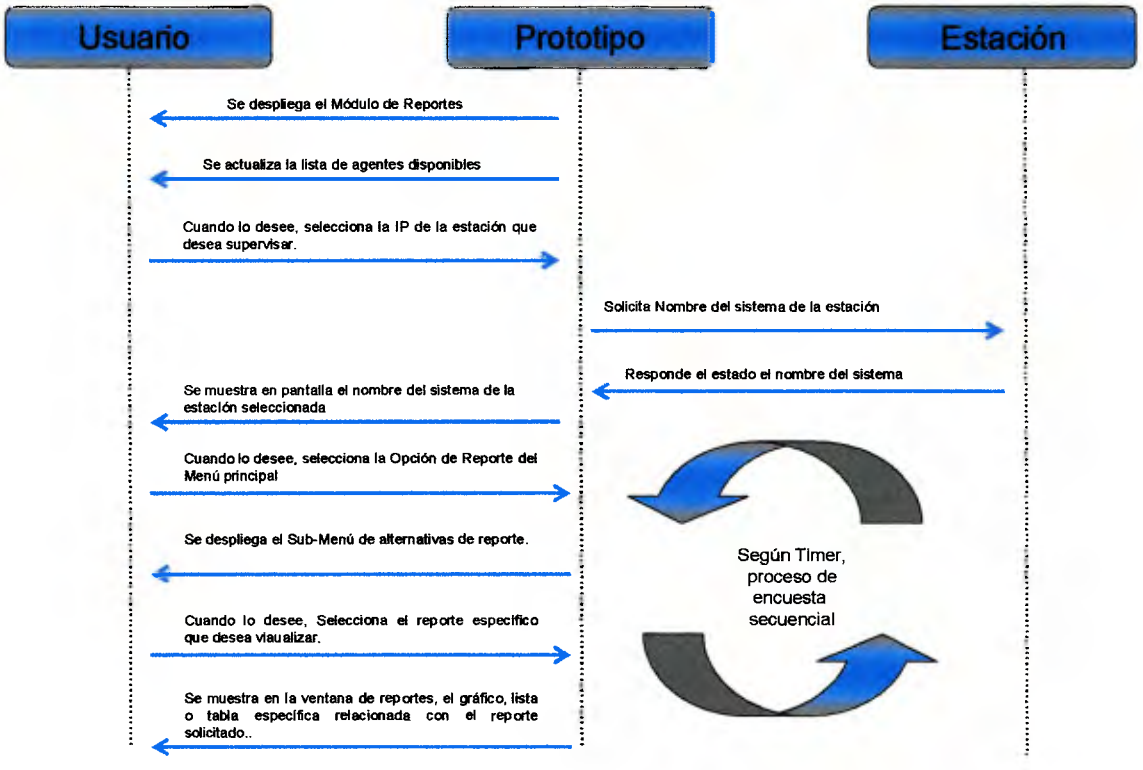



*Figura 10.. Diagrama de Secuencia Módulo Reporte*
*Fuente: Elaboración Propia*

# ANEXO I

## REQUEST FOR COMMENTS 1157

*SISTEMA DE CONTROL, SUPERVISIÓN Y GESTIÓN DE REDES BASADAS EN IPV4.*

# RFC 1157 (RFC1157)



## RFC 1157 - Simple Network Management Protocol (SNMP)

---

Network Working Group
Request for Comments: 1157
Obsoletes: RFC 1098

J. Case
SNMP Research
M. Fedor
Performance Systems International
M. Schoffstall
Performance Systems International
J. Davin
MIT Laboratory for Computer Science

May 1990

A Simple Network Management Protocol (SNMP)

Table of Contents

## 1. Status of this Memo

This RFC is a re-release of RFC 1098, with a changed "Status of this Memo" section plus a few minor typographical corrections. This memo defines a simple protocol by which management information for a network element may be inspected or altered by logically remote users. In particular, together with its companion memos which describe the structure of management information along with the management information base, these documents provide a simple, workable architecture and system for managing TCP/IP-based internets and in particular the Internet.

The Internet Activities Board recommends that all IP and TCP implementations be network manageable. This implies implementation of the Internet MIB (RFC-1156) and at least one of the two recommended management protocols SNMP (RFC-1157) or CMOT (RFC-1095). It should be noted that, at this time, SNMP is a full Internet standard and CMOT is a draft standard. See also the Host and Gateway Requirements RFCs for more specific information on the applicability of this standard.

Please refer to the latest edition of the "IAB Official Protocol Standards" RFC for current information on the state and status of standard Internet protocols.

Distribution of this memo is unlimited.

## 2. Introduction

As reported in RFC 1052, IAB Recommendations for the Development of Internet Network Management Standards [1], a two-prong strategy for network management of TCP/IP-based internets was undertaken. In the short-term, the Simple Network Management Protocol (SNMP) was to be used to manage nodes in the Internet community. In the long-term, the use of the OSI network management framework was to be examined. Two documents were produced to define the management information: RFC 1065, which defined the Structure of Management Information (SMI) [2], and RFC 1066, which defined the Management Information Base (MIB) [3]. Both of these documents were designed so as to be

compatible with both the SNMP and the OSI network management framework.

This strategy was quite successful in the short-term: Internet-based network management technology was fielded, by both the research and commercial communities, within a few months. As a result of this,

As reported in RFC 1109, Report of the Second Ad Hoc Network Management Review Group [4], the requirements of the SNMP and the OSI network management frameworks were more different than anticipated. As such, the requirement for compatibility between the SMI/MIB and both frameworks was suspended. This action permitted the operational network management framework, the SNMP, to respond to new operational needs in the Internet community by producing documents defining new MIB items.

The IAB has designated the SNMP, SMI, and the initial Internet MIB to be full "Standard Protocols" with "Recommended" status. By this action, the IAB recommends that all IP and TCP implementations be network manageable and that the implementations that are network manageable are expected to adopt and implement the SMI, MIB, and SNMP.

As such, the current network management framework for TCP/IP- based internets consists of: Structure and Identification of Management Information for TCP/IP-based Internets, which describes how managed objects contained in the MIB are defined as set forth in RFC 1155 [5]; Management Information Base for Network Management of TCP/IP-based Internets, which describes the managed objects contained in the MIB as set forth in RFC 1156 [6]; and, the Simple Network Management Protocol, which defines the protocol used to manage these objects, as set forth in this memo.

As reported in RFC 1052, IAB Recommendations for the Development of Internet Network Management Standards [1], the Internet Activities Board has directed the Internet Engineering Task Force (IETF) to create two new working groups in the area of network management. One group was charged with the further specification and definition of elements to be included in the Management Information Base (MIB). The other was charged with defining the modifications to the Simple Network Management Protocol (SNMP) to accommodate the short-term needs of the network vendor and operations communities, and to align with the output of the MIB working group.

The MIB working group produced two memos, one which defines a Structure for Management Information (SMI) [2] for use by the managed

objects contained in the MIB. A second memo [3] defines the list of managed objects.

The output of the SNMP Extensions working group is this memo, which incorporates changes to the initial SNMP definition [7] required to attain alignment with the output of the MIB working group. The changes should be minimal in order to be consistent with the IAB's directive that the working groups be "extremely sensitive to the need to keep the SNMP simple." Although considerable care and debate has gone into the changes to the SNMP which are reflected in this memo, the resulting protocol is not backwardly-compatible with its predecessor, the Simple Gateway Monitoring Protocol (SGMP) [8]. Although the syntax of the protocol has been altered, the original philosophy, design decisions, and architecture remain intact. In order to avoid confusion, new UDP ports have been allocated for use by the protocol described in this memo.

## 3. The SNMP Architecture

Implicit in the SNMP architectural model is a collection of network management stations and network elements. Network management

gateways, terminal servers, and the like, which have management agents responsible for performing the network management functions requested by the network management stations. The Simple Network Management Protocol (SNMP) is used to communicate management information between the network management stations and the agents in the network elements.

## 3.1. Goals of the Architecture

The SNMP explicitly minimizes the number and complexity of management functions realized by the management agent itself. This goal is attractive in at least four respects:

(1) The development cost for management agent software necessary to support the protocol is accordingly reduced.

(2) The degree of management function that is remotely supported is accordingly increased, thereby admitting fullest use of internet resources in the management task.

(3) The degree of management function that is remotely supported is accordingly increased, thereby imposing the fewest possible restrictions on the form and sophistication of management tools.

(4) Simplified sets of management functions are easily understood and used by developers of network management tools.

A second goal of the protocol is that the functional paradigm for monitoring and control be sufficiently extensible to accommodate additional, possibly unanticipated aspects of network operation and management.

A third goal is that the architecture be, as much as possible, independent of the architecture and mechanisms of particular hosts or particular gateways.

## 3.2. Elements of the Architecture

The SNMP architecture articulates a solution to the network management problem in terms of:

(1) the scope of the management information communicated by the protocol,

(2) the representation of the management information communicated by the protocol,

(3) operations on management information supported by the protocol,

(4) the form and meaning of exchanges among management entities,

(5) the definition of administrative relationships among management entities, and

(6) the form and meaning of references to management information.

### 3.2.1. Scope of Management Information

the SNMP is exactly that represented by instances of all non-aggregate object types either defined in Internet-standard MIB or defined elsewhere according to the conventions set forth in Internet-standard SMI [5].

Support for aggregate object types in the MIB is neither required for conformance with the SMI nor realized by the SNMP.

### 3.2.2. Representation of Management Information

Management information communicated by operation of the SNMP is represented according to the subset of the ASN.1 language [9] that is specified for the definition of non-aggregate types in the SMI.

The SGMP adopted the convention of using a well-defined subset of the ASN.1 language [9]. The SNMP continues and extends this tradition by utilizing a moderately more complex subset of ASN.1 for describing managed objects and for describing the protocol data units used for managing those objects. In addition, the desire to ease eventual transition to OSI-based network management protocols led to the definition in the ASN.1 language of an Internet-standard Structure of Management Information (SMI) [5] and Management Information Base (MIB) [6]. The use of the ASN.1 language, was, in part, encouraged by the successful use of ASN.1 in earlier efforts, in particular, the SGMP. The restrictions on the use of ASN.1 that are part of the SMI contribute to the simplicity espoused and validated by experience with the SGMP.

Also for the sake of simplicity, the SNMP uses only a subset of the basic encoding rules of ASN.1 [10]. Namely, all encodings use the definite-length form. Further, whenever permissible, non-constructor encodings are used rather than constructor encodings. This restriction applies to all aspects of ASN.1 encoding, both for the top-level protocol data units and the data objects they contain.

### 3.2.3. Operations Supported on Management Information

The SNMP models all management agent functions as alterations or inspections of variables. Thus, a protocol entity on a logically remote host (possibly the network element itself) interacts with the management agent resident on the network element in order to retrieve (get) or alter (set) variables. This strategy has at least two positive consequences:

(1) It has the effect of limiting the number of essential management functions realized by the management agent to two: one operation to assign a value to a specified configuration or other parameter and another to retrieve such a value.

(2) A second effect of this decision is to avoid introducing into the protocol definition support for imperative management commands: the number of such commands is in practice ever-increasing, and the semantics of such commands are in general arbitrarily complex.

The strategy implicit in the SNMP is that the monitoring of network state at any significant level of detail is accomplished primarily by polling for appropriate information on the part of the monitoring center(s). A limited number of unsolicited messages (traps) guide the timing and focus of the polling. Limiting the number of unsolicited messages is consistent with the goal of simplicity and minimizing the amount of traffic generated by the network management

The exclusion of imperative commands from the set of explicitly supported management functions is unlikely to preclude any desirable management agent operation. Currently, most commands are requests either to set the value of some parameter or to retrieve such a value, and the function of the few imperative commands currently supported is easily accommodated in an asynchronous mode by this management model. In this scheme, an imperative command might be realized as the setting of a parameter value that subsequently triggers the desired action. For example, rather than implementing a "reboot command," this action might be invoked by simply setting a parameter indicating the number of seconds until system reboot.

### 3.2.4. Form and Meaning of Protocol Exchanges

The communication of management information among management entities is realized in the SNMP through the exchange of protocol messages. The form and meaning of those messages is defined below in Section 4.

Consistent with the goal of minimizing complexity of the management agent, the exchange of SNMP messages requires only an unreliable datagram service, and every message is entirely and independently represented by a single transport datagram. While this document specifies the exchange of messages via the UDP protocol [11], the mechanisms of the SNMP are generally suitable for use with a wide variety of transport services.

### 3.2.5. Definition of Administrative Relationships

The SNMP architecture admits a variety of administrative relationships among entities that participate in the protocol. The entities residing at management stations and network elements which communicate with one another using the SNMP are termed SNMP application entities. The peer processes which implement the SNMP, and thus support the SNMP application entities, are termed protocol entities.

A pairing of an SNMP agent with some arbitrary set of SNMP application entities is called an SNMP community. Each SNMP community is named by a string of octets, that is called the community name for said community.

An SNMP message originated by an SNMP application entity that in fact belongs to the SNMP community named by the community component of said message is called an authentic SNMP message. The set of rules by which an SNMP message is identified as an authentic SNMP message for a particular SNMP community is called an authentication scheme. An implementation of a function that identifies authentic SNMP messages according to one or more authentication schemes is called an authentication service.

Clearly, effective management of administrative relationships among SNMP application entities requires authentication services that (by the use of encryption or other techniques) are able to identify authentic SNMP messages with a high degree of certainty. Some SNMP implementations may wish to support only a trivial authentication service that identifies all SNMP messages as authentic SNMP messages.

For any network element, a subset of objects in the MIB that pertain to that element is called a SNMP MIB view. Note that the names of the object types represented in a SNMP MIB view need not belong to a

single sub-tree of the object type name space.

An element of the set { READ-ONLY, READ-WRITE } is called an SNMP access mode.

A pairing of a SNMP access mode with a SNMP MIB view is called an SNMP community profile. A SNMP community profile represents specified access privileges to variables in a specified MIB view. For every variable in the MIB view in a given SNMP community profile, access to that variable is represented by the profile according to the following conventions:

(1) if said variable is defined in the MIB with "Access:" of "none," it is unavailable as an operand for any operator;

(2) if said variable is defined in the MIB with "Access:" of "read-write" or "write-only" and the access mode of the given profile is READ-WRITE, that variable is available as an operand for the get, set, and trap operations;

(3) otherwise, the variable is available as an operand for the get and trap operations.

(4) In those cases where a "write-only" variable is an operand used for the get or trap operations, the value given for the variable is implementation-specific.

A pairing of a SNMP community with a SNMP community profile is called a SNMP access policy. An access policy represents a specified community profile afforded by the SNMP agent of a specified SNMP community to other members of that community. All administrative relationships among SNMP application entities are architecturally defined in terms of SNMP access policies.

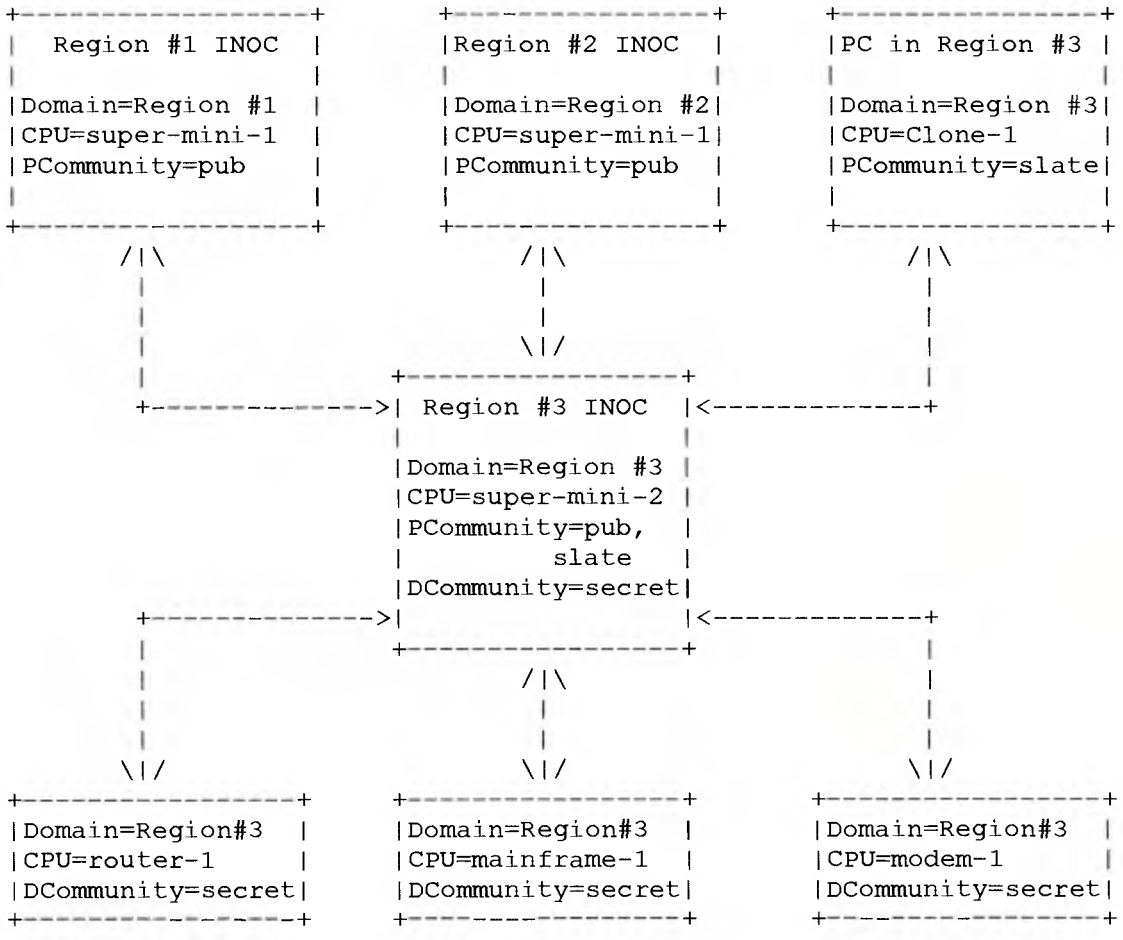
For every SNMP access policy, if the network element on which the SNMP agent for the specified SNMP community resides is not that to which the MIB view for the specified profile pertains, then that policy is called a SNMP proxy access policy. The SNMP agent associated with a proxy access policy is called a SNMP proxy agent. While careless definition of proxy access policies can result in management loops, prudent definition of proxy policies is useful in at least two ways:

(1) It permits the monitoring and control of network elements which are otherwise not addressable using the management protocol and the transport protocol. That is, a proxy agent may provide a protocol conversion function allowing a management station to apply a consistent management

framework to all network elements, including devices such as modems, multiplexors, and other devices which support different management frameworks.

(2) It potentially shields network elements from elaborate access control policies. For example, a proxy agent may implement sophisticated access control whereby diverse subsets of variables within the MIB are made accessible to different management stations without increasing the complexity of the network element.

By way of example, Figure 1 illustrates the relationship between management stations, proxy agents, and management agents. In this example, the proxy agent is envisioned to be a normal Internet Network Operations Center (INOC) of some administrative domain which has a standard managerial relationship with a set of management




Domain: the administrative domain of the element
PCommunity: the name of a community utilizing a proxy agent
DCommunity: the name of a direct community

Figure 1
Example Network Management Configuration

3.2.6. Form and Meaning of References to Managed Objects

The SMI requires that the definition of a conformant management protocol address:

(1) the resolution of ambiguous MIB references,

(2) the resolution of MIB references in the presence multiple MIB versions, and

(3) the identification of particular instances of object types defined in the MIB.

3.2.6.1. Resolution of Ambiguous MIB References

Because the scope of any SNMP operation is conceptually confined to objects relevant to a single network element, and because all SNMP references to MIB objects are (implicitly or explicitly) by unique variable names, there is no possibility that any SNMP reference to any object type defined in the MIB could resolve to multiple instances of that type.

3.2.6.2. Resolution of References across MIB Versions

The object instance referred to by any SNMP operation is exactly that specified as part of the operation request or (in the case of a get-next operation) its immediate successor in the MIB as a whole. In particular, a reference to an object as part of some version of the Internet-standard MIB does not resolve to any object that is not part of said version of the Internet-standard MIB, except in the case that the requested operation is get-next and the specified object name is lexicographically last among the names of all objects presented as part of said version of the Internet-Standard MIB.

#### 3.2.6.3. Identification of Object Instances

The names for all object types in the MIB are defined explicitly either in the Internet-standard MIB or in other documents which conform to the naming conventions of the SMI. The SMI requires that conformant management protocols define mechanisms for identifying individual instances of those object types for a particular network element.

Each instance of any object type defined in the MIB is identified in SNMP operations by a unique name called its "variable name." In general, the name of an SNMP variable is an OBJECT IDENTIFIER of the form x.y, where x is the name of a non-aggregate object type defined in the MIB and y is an OBJECT IDENTIFIER fragment that, in a way

specific to the named object type, identifies the desired instance.

This naming strategy admits the fullest exploitation of the semantics of the GetNextRequest-PDU (see Section 4), because it assigns names for related variables so as to be contiguous in the lexicographical ordering of all variable names known in the MIB.

The type-specific naming of object instances is defined below for a number of classes of object types. Instances of an object type to which none of the following naming conventions are applicable are named by OBJECT IDENTIFIERs of the form x.0, where x is the name of said object type in the MIB definition.

For example, suppose one wanted to identify an instance of the variable sysDescr The object class for sysDescr is:

```
iso org dod internet mgmt mib system sysDescr
 1   3   6     1      2    1    1       1
```

Hence, the object type, x, would be 1.3.6.1.2.1.1.1 to which is appended an instance sub-identifier of 0. That is, 1.3.6.1.2.1.1.1.0 identifies the one and only instance of sysDescr.

#### 3.2.6.3.1. ifTable Object Type Names

The name of a subnet interface, s, is the OBJECT IDENTIFIER value of the form i, where i has the value of that instance of the ifIndex object type associated with s.

For each object type, t, for which the defined name, n, has a prefix of ifEntry, an instance, i, of t is named by an OBJECT IDENTIFIER of the form n.s, where s is the name of the subnet interface about which i represents information.

For example, suppose one wanted to identify the instance of the variable ifType associated with interface 2. Accordingly, ifType.2 would identify the desired instance.

The name of an AT-cached network address, x, is an OBJECT IDENTIFIER of the form 1.a.b.c.d, where a.b.c.d is the value (in the familiar "dot" notation) of the atNetAddress object type associated with x.

The name of an address translation equivalence e is an OBJECT IDENTIFIER value of the form s.w, such that s is the value of that instance of the atIndex object type associated with e and such that w is the name of the AT-cached network address associated with e.

For each object type, t, for which the defined name, n, has a prefix of atEntry, an instance, i, of t is named by an OBJECT IDENTIFIER of the form n.y, where y is the name of the address translation equivalence about which i represents information.

For example, suppose one wanted to find the physical address of an entry in the address translation table (ARP cache) associated with an IP address of 89.1.1.42 and interface 3. Accordingly, atPhysAddress.3.1.89.1.1.42 would identify the desired instance.

3.2.6.3.3. ipAddrTable Object Type Names

The name of an IP-addressable network element, x, is the OBJECT IDENTIFIER of the form a.b.c.d such that a.b.c.d is the value (in the familiar "dot" notation) of that instance of the ipAdEntAddr object type associated with x.

For each object type, t, for which the defined name, n, has a prefix of ipAddrEntry, an instance, i, of t is named by an OBJECT IDENTIFIER of the form n.y, where y is the name of the IP-addressable network element about which i represents information.

For example, suppose one wanted to find the network mask of an entry in the IP interface table associated with an IP address of 89.1.1.42. Accordingly, ipAdEntNetMask.89.1.1.42 would identify the desired instance.

3.2.6.3.4. ipRoutingTable Object Type Names

The name of an IP route, x, is the OBJECT IDENTIFIER of the form a.b.c.d such that a.b.c.d is the value (in the familiar "dot" notation) of that instance of the ipRouteDest object type associated with x.

For each object type, t, for which the defined name, n, has a prefix of ipRoutingEntry, an instance, i, of t is named by an OBJECT IDENTIFIER of the form n.y, where y is the name of the IP route about which i represents information.

For example, suppose one wanted to find the next hop of an entry in the IP routing table associated with the destination of 89.1.1.42. Accordingly, ipRouteNextHop.89.1.1.42 would identify the desired instance.

3.2.6.3.5. tcpConnTable Object Type Names

The name of a TCP connection, x, is the OBJECT IDENTIFIER of the form a.b.c.d.e.f.g.h.i.j such that a.b.c.d is the value (in the familiar

"dot" notation) of that instance of the tcpConnLocalAddress object type associated with x and such that f.g.h.i is the value (in the familiar "dot" notation) of that instance of the tcpConnRemoteAddress object type associated with x and such that e is the value of that

such that j is the value of that instance of the tcpConnRemotePort object type associated with x.

For each object type, t, for which the defined name, n, has a prefix of tcpConnEntry, an instance, i, of t is named by an OBJECT IDENTIFIER of the form n.y, where y is the name of the TCP connection about which i represents information.

For example, suppose one wanted to find the state of a TCP connection between the local address of 89.1.1.42 on TCP port 21 and the remote address of 10.0.0.51 on TCP port 2059. Accordingly, tcpConnState.89.1.1.42.21.10.0.0.51.2059 would identify the desired instance.

3.2.6.3.6. egpNeighTable Object Type Names

The name of an EGP neighbor, x, is the OBJECT IDENTIFIER of the form a.b.c.d such that a.b.c.d is the value (in the familiar "dot" notation) of that instance of the egpNeighAddr object type associated with x.

For each object type, t, for which the defined name, n, has a prefix of egpNeighEntry, an instance, i, of t is named by an OBJECT IDENTIFIER of the form n.y, where y is the name of the EGP neighbor about which i represents information.

For example, suppose one wanted to find the neighbor state for the IP address of 89.1.1.42. Accordingly, egpNeighState.89.1.1.42 would identify the desired instance.

## 4. Protocol Specification

The network management protocol is an application protocol by which the variables of an agent's MIB may be inspected or altered.

Communication among protocol entities is accomplished by the exchange of messages, each of which is entirely and independently represented within a single UDP datagram using the basic encoding rules of ASN.1 (as discussed in Section 3.2.2). A message consists of a version identifier, an SNMP community name, and a protocol data unit (PDU). A protocol entity receives messages at UDP port 161 on the host with which it is associated for all messages except for those which report traps (i.e., all messages except those which contain the Trap-PDU). Messages which report traps should be received on UDP port 162 for further processing. An implementation of this protocol need not accept messages whose length exceeds 484 octets. However, it is recommended that implementations support larger datagrams whenever feasible.

It is mandatory that all implementations of the SNMP support the five PDUs: GetRequest-PDU, GetNextRequest-PDU, GetResponse-PDU, SetRequest-PDU, and Trap-PDU.

```
RFC1157-SNMP DEFINITIONS ::= BEGIN

 IMPORTS
      ObjectName, ObjectSyntax, NetworkAddress, IpAddress, TimeTicks
              FROM RFC1155-SMI;

 -- top-level message

         Message ::=
                 SEQUENCE {
```

```
                        INTEGER {
                            version-1(0)
                        },

                    community       -- community name
                        OCTET STRING,

                    data            -- e.g., PDUs if trivial
                        ANY         -- authentication is being used
                }

-- protocol data units

        PDUs ::=
                CHOICE {
                    get-request
                        GetRequest-PDU,

                    get-next-request
                        GetNextRequest-PDU,

                    get-response
                        GetResponse-PDU,

                    set-request
                        SetRequest-PDU,

                    trap
                        Trap-PDU
                 }

-- the individual PDUs and commonly used
-- data types will be defined later

END
```

## 4.1. Elements of Procedure

This section describes the actions of a protocol entity implementing the SNMP. Note, however, that it is not intended to constrain the internal architecture of any conformant implementation.

In the text that follows, the term transport address is used. In the case of the UDP, a transport address consists of an IP address along with a UDP port. Other transport services may be used to support the SNMP. In these cases, the definition of a transport address should be made accordingly.

The top-level actions of a protocol entity which generates a message are as follows:

(1) It first constructs the appropriate PDU, e.g., the GetRequest-PDU, as an ASN.1 object.

(2) It then passes this ASN.1 object along with a community name its source transport address and the destination transport address, to the service which implements the desired authentication scheme. This authentication

service returns another ASN.1 object.

(3) The protocol entity then constructs an ASN.1 Message object, using the community name and the resulting ASN.1 object.

(4) This new ASN.1 object is then serialized, using the basic encoding rules of ASN.1, and then sent using a transport service to the peer protocol entity.

Similarly, the top-level actions of a protocol entity which receives a message are as follows:

(1) It performs a rudimentary parse of the incoming datagram to build an ASN.1 object corresponding to an ASN.1 Message object. If the parse fails, it discards the datagram and performs no further actions.

(2) It then verifies the version number of the SNMP message. If there is a mismatch, it discards the datagram and performs no further actions.

(3) The protocol entity then passes the community name and user data found in the ASN.1 Message object, along with the datagram's source and destination transport addresses to the service which implements the desired authentication scheme. This entity returns another ASN.1 object, or signals an authentication failure. In the latter case, the protocol entity notes this failure, (possibly) generates a trap, and discards the datagram and performs no further actions.

(4) The protocol entity then performs a rudimentary parse on the ASN.1 object returned from the authentication service to build an ASN.1 object corresponding to an ASN.1 PDUs object. If the parse fails, it discards the datagram and performs no further actions. Otherwise, using the named SNMP community, the appropriate profile is selected, and the PDU is processed accordingly. If, as a result of this processing, a message is returned then the source transport address that the response message is sent from shall be identical to the destination transport address that the original request message was sent to.

### 4.1.1. Common Constructs

Before introducing the six PDU types of the protocol, it is appropriate to consider some of the ASN.1 constructs used frequently:

```
-- request/response information

RequestID ::=
        INTEGER

ErrorStatus ::=
        INTEGER {
            noError(0),
            tooBig(1),
            noSuchName(2),
            badValue(3),
            readOnly(4)
            genErr(5)
        }

ErrorIndex ::=
        INTEGER

-- variable bindings
```

```
VarBind ::=
        SEQUENCE {
            name
                ObjectName,

            value
                ObjectSyntax
        }

VarBindList ::=
        SEQUENCE OF
            VarBind
```

RequestIDs are used to distinguish among outstanding requests. By use of the RequestID, an SNMP application entity can correlate incoming responses with outstanding requests. In cases where an unreliable datagram service is being used, the RequestID also provides a simple means of identifying messages duplicated by the network.

A non-zero instance of ErrorStatus is used to indicate that an

exception occurred while processing a request. In these cases, ErrorIndex may provide additional information by indicating which variable in a list caused the exception.

The term variable refers to an instance of a managed object. A variable binding, or VarBind, refers to the pairing of the name of a variable to the variable's value. A VarBindList is a simple list of variable names and corresponding values. Some PDUs are concerned only with the name of a variable and not its value (e.g., the GetRequest-PDU). In this case, the value portion of the binding is ignored by the protocol entity. However, the value portion must still have valid ASN.1 syntax and encoding. It is recommended that the ASN.1 value NULL be used for the value portion of such bindings.

### 4.1.2. The GetRequest-PDU

The form of the GetRequest-PDU is:

```
GetRequest-PDU ::=
    [0]
        IMPLICIT SEQUENCE {
            request-id
                RequestID,

            error-status            -- always 0
                ErrorStatus,

            error-index             -- always 0
                ErrorIndex,

            variable-bindings
                VarBindList
        }
```

The GetRequest-PDU is generated by a protocol entity only at the request of its SNMP application entity.

Upon receipt of the GetRequest-PDU, the receiving protocol entity responds according to any applicable rule in the list below:

(1) If, for any object named in the variable-bindings field, the object's name does not exactly match the name of some

view, then the receiving entity sends to the originator of the received message the GetResponse-PDU of identical form, except that the value of the error-status field is noSuchName, and the value of the error-index field is the index of said object name component in the received

message.

(2) If, for any object named in the variable-bindings field, the object is an aggregate type (as defined in the SMI), then the receiving entity sends to the originator of the received message the GetResponse-PDU of identical form, except that the value of the error-status field is noSuchName, and the value of the error-index field is the index of said object name component in the received message.

(3) If the size of the GetResponse-PDU generated as described below would exceed a local limitation, then the receiving entity sends to the originator of the received message the GetResponse-PDU of identical form, except that the value of the error-status field is tooBig, and the value of the error-index field is zero.

(4) If, for any object named in the variable-bindings field, the value of the object cannot be retrieved for reasons not covered by any of the foregoing rules, then the receiving entity sends to the originator of the received message the GetResponse-PDU of identical form, except that the value of the error-status field is genErr and the value of the error-index field is the index of said object name component in the received message.

If none of the foregoing rules apply, then the receiving protocol entity sends to the originator of the received message the GetResponse-PDU such that, for each object named in the variable-bindings field of the received message, the corresponding component of the GetResponse-PDU represents the name and value of that variable. The value of the error- status field of the GetResponse-PDU is noError and the value of the error-index field is zero. The value of the request-id field of the GetResponse-PDU is that of the received message.

### 4.1.3. The GetNextRequest-PDU

The form of the GetNextRequest-PDU is identical to that of the GetRequest-PDU except for the indication of the PDU type. In the ASN.1 language:

```
GetNextRequest-PDU ::=
    [1]
        IMPLICIT SEQUENCE {
            request-id
                RequestID,

            error-status        -- always 0
                ErrorStatus,

            error-index         -- always 0
                ErrorIndex,

            variable-bindings
                VarBindList
```

The GetNextRequest-PDU is generated by a protocol entity only at the request of its SNMP application entity.

Upon receipt of the GetNextRequest-PDU, the receiving protocol entity responds according to any applicable rule in the list below:

(1) If, for any object name in the variable-bindings field, that name does not lexicographically precede the name of some object available for get operations in the relevant MIB view, then the receiving entity sends to the originator of the received message the GetResponse-PDU of identical form, except that the value of the error-status field is noSuchName, and the value of the error-index field is the index of said object name component in the received message.

(2) If the size of the GetResponse-PDU generated as described below would exceed a local limitation, then the receiving entity sends to the originator of the received message the GetResponse-PDU of identical form, except that the value of the error-status field is tooBig, and the value of the error-index field is zero.

(3) If, for any object named in the variable-bindings field, the value of the lexicographical successor to the named object cannot be retrieved for reasons not covered by any of the foregoing rules, then the receiving entity sends to the originator of the received message the GetResponse-PDU of identical form, except that the value of the error-status field is genErr and the value of the error-index field is the index of said object name component in the received message.

If none of the foregoing rules apply, then the receiving protocol entity sends to the originator of the received message the GetResponse-PDU such that, for each name in the variable-bindings field of the received message, the corresponding component of the

GetResponse-PDU represents the name and value of that object whose name is, in the lexicographical ordering of the names of all objects available for get operations in the relevant MIB view, together with the value of the name field of the given component, the immediate successor to that value. The value of the error-status field of the GetResponse-PDU is noError and the value of the errorindex field is zero. The value of the request-id field of the GetResponse-PDU is that of the received message.

4.1.3.1. Example of Table Traversal

One important use of the GetNextRequest-PDU is the traversal of conceptual tables of information within the MIB. The semantics of this type of SNMP message, together with the protocol-specific mechanisms for identifying individual instances of object types in the MIB, affords access to related objects in the MIB as if they enjoyed a tabular organization.

By the SNMP exchange sketched below, an SNMP application entity might extract the destination address and next hop gateway for each entry in the routing table of a particular network element. Suppose that this routing table has three entries:

| Destination | NextHop | Metric |
|---|---|---|

```
10.0.0.99                       89.1.1.42       5
9.1.2.3                         99.0.0.3        3
10.0.0.51                       89.1.1.42       5
```

The management station sends to the SNMP agent a GetNextRequest-PDU containing the indicated OBJECT IDENTIFIER values as the requested variable names:

```
GetNextRequest ( ipRouteDest, ipRouteNextHop, ipRouteMetric1 )
```

The SNMP agent responds with a GetResponse-PDU:

```
GetResponse (( ipRouteDest.9.1.2.3 =  "9.1.2.3" ),
        ( ipRouteNextHop.9.1.2.3 = "99.0.0.3" ),
        ( ipRouteMetric1.9.1.2.3 = 3 ))
```

The management station continues with:

```
GetNextRequest ( ipRouteDest.9.1.2.3,
        ipRouteNextHop.9.1.2.3,

        ipRouteMetric1.9.1.2.3 )
```

The SNMP agent responds:

```
GetResponse (( ipRouteDest.10.0.0.51 = "10.0.0.51" ),
        ( ipRouteNextHop.10.0.0.51 = "89.1.1.42" ),
        ( ipRouteMetric1.10.0.0.51 = 5 ))
```

The management station continues with:

```
GetNextRequest ( ipRouteDest.10.0.0.51,
        ipRouteNextHop.10.0.0.51,
        ipRouteMetric1.10.0.0.51 )
```

The SNMP agent responds:

```
GetResponse (( ipRouteDest.10.0.0.99 = "10.0.0.99" ),
        ( ipRouteNextHop.10.0.0.99 = "89.1.1.42" ),
        ( ipRouteMetric1.10.0.0.99 = 5 ))
```

The management station continues with:

```
GetNextRequest ( ipRouteDest.10.0.0.99,
        ipRouteNextHop.10.0.0.99,
        ipRouteMetric1.10.0.0.99 )
```

As there are no further entries in the table, the SNMP agent returns those objects that are next in the lexicographical ordering of the known object names. This response signals the end of the routing table to the management station.

### 4.1.4. The GetResponse-PDU

The form of the GetResponse-PDU is identical to that of the GetRequest-PDU except for the indication of the PDU type. In the ASN.1 language:

```
GetResponse-PDU ::=
    [2]
        IMPLICIT SEQUENCE {
            request-id
                RequestID
```

```
                    error-status
                        ErrorStatus,

                    error-index
                        ErrorIndex,

                    variable-bindings
                        VarBindList
                }
```

The GetResponse-PDU is generated by a protocol entity only upon receipt of the GetRequest-PDU, GetNextRequest-PDU, or SetRequest-PDU, as described elsewhere in this document.

Upon receipt of the GetResponse-PDU, the receiving protocol entity presents its contents to its SNMP application entity.

### 4.1.5. The SetRequest-PDU

The form of the SetRequest-PDU is identical to that of the GetRequest-PDU except for the indication of the PDU type. In the ASN.1 language:

```
        SetRequest-PDU ::=
            [3]
                IMPLICIT SEQUENCE {
                    request-id
                        RequestID,

                    error-status        -- always 0
                        ErrorStatus,

                    error-index         -- always 0
                        ErrorIndex,

                    variable-bindings
                        VarBindList
                }
```

The SetRequest-PDU is generated by a protocol entity only at the request of its SNMP application entity.

Upon receipt of the SetRequest-PDU, the receiving entity responds according to any applicable rule in the list below:

(1) If, for any object named in the variable-bindings field,

the object is not available for set operations in the relevant MIB view, then the receiving entity sends to the originator of the received message the GetResponse-PDU of identical form, except that the value of the error-status field is noSuchName, and the value of the error-index field is the index of said object name component in the received message.

(2) If, for any object named in the variable-bindings field, the contents of the value field does not, according to the ASN.1 language, manifest a type, length, and value that is consistent with that required for the variable, then the receiving entity sends to the originator of the received message the GetResponse-PDU of identical form, except that the value of the error-status field is badValue, and the value of the error-index field is the

(3) If the size of the Get Response type message generated as described below would exceed a local limitation, then the receiving entity sends to the originator of the received message the GetResponse-PDU of identical form, except that the value of the error-status field is tooBig, and the value of the error-index field is zero.

(4) If, for any object named in the variable-bindings field, the value of the named object cannot be altered for reasons not covered by any of the foregoing rules, then the receiving entity sends to the originator of the received message the GetResponse-PDU of identical form, except that the value of the error-status field is genErr and the value of the error-index field is the index of said object name component in the received message.

If none of the foregoing rules apply, then for each object named in the variable-bindings field of the received message, the corresponding value is assigned to the variable. Each variable assignment specified by the SetRequest-PDU should be effected as if simultaneously set with respect to all other assignments specified in the same message.

The receiving entity then sends to the originator of the received message the GetResponse-PDU of identical form except that the value of the error-status field of the generated message is noError and the value of the error-index field is zero.

### 4.1.6. The Trap-PDU

The form of the Trap-PDU is:

```
Trap-PDU ::=
    [4]

        IMPLICIT SEQUENCE {
            enterprise          -- type of object generating
                                -- trap, see sysObjectID in [5]
                OBJECT IDENTIFIER,

            agent-addr          -- address of object generating
                NetworkAddress, -- trap

            generic-trap        -- generic trap type
                INTEGER {
                    coldStart(0),
                    warmStart(1),
                    linkDown(2),
                    linkUp(3),
                    authenticationFailure(4),
                    egpNeighborLoss(5),
                    enterpriseSpecific(6)
                },

            specific-trap       -- specific code, present even
                INTEGER,        -- if generic-trap is not
                                -- enterpriseSpecific

            time-stamp          -- time elapsed between the last
              TimeTicks,        -- (re)initialization of the network
                                -- entity and the generation of the
                                   trap
```

```
        variable-bindings   -- "interesting" information
            VarBindList
    }
```

The Trap-PDU is generated by a protocol entity only at the request of the SNMP application entity. The means by which an SNMP application entity selects the destination addresses of the SNMP application entities is implementation-specific.

Upon receipt of the Trap-PDU, the receiving protocol entity presents its contents to its SNMP application entity.

The significance of the variable-bindings component of the Trap-PDU is implementation-specific.

Interpretations of the value of the generic-trap field are:

#### 4.1.6.1. The coldStart Trap

A coldStart(0) trap signifies that the sending protocol entity is reinitializing itself such that the agent's configuration or the protocol entity implementation may be altered.

#### 4.1.6.2. The warmStart Trap

A warmStart(1) trap signifies that the sending protocol entity is reinitializing itself such that neither the agent configuration nor the protocol entity implementation is altered.

#### 4.1.6.3. The linkDown Trap

A linkDown(2) trap signifies that the sending protocol entity recognizes a failure in one of the communication links represented in the agent's configuration.

The Trap-PDU of type linkDown contains as the first element of its variable-bindings, the name and value of the ifIndex instance for the affected interface.

#### 4.1.6.4. The linkUp Trap

A linkUp(3) trap signifies that the sending protocol entity recognizes that one of the communication links represented in the agent's configuration has come up.

The Trap-PDU of type linkUp contains as the first element of its variable-bindings, the name and value of the ifIndex instance for the affected interface.

#### 4.1.6.5. The authenticationFailure Trap

An authenticationFailure(4) trap signifies that the sending protocol entity is the addressee of a protocol message that is not properly authenticated. While implementations of the SNMP must be capable of generating this trap, they must also be capable of suppressing the emission of such traps via an implementation-specific mechanism.

#### 4.1.6.6. The egpNeighborLoss Trap

An egpNeighborLoss(5) trap signifies that an EGP neighbor for whom

the sending protocol entity was an EGP peer has been marked down and the peer relationship no longer obtains.

The Trap-PDU of type egpNeighborLoss contains as the first element of its variable-bindings, the name and value of the egpNeighAddr instance for the affected neighbor.

4.1.6.7. The enterpriseSpecific Trap

A enterpriseSpecific(6) trap signifies that the sending protocol entity recognizes that some enterprise-specific event has occurred. The specific-trap field identifies the particular trap which occurred.

5. Definitions

```
RFC1157-SNMP DEFINITIONS ::= BEGIN

 IMPORTS
     ObjectName, ObjectSyntax, NetworkAddress, IpAddress, TimeTicks
         FROM RFC1155-SMI;

     -- top-level message

     Message ::=
             SEQUENCE {
                 version          -- version-1 for this RFC
                     INTEGER {
                         version-1(0)
                     },

                 community        -- community name
                     OCTET STRING,

                 data             -- e.g., PDUs if trivial
                     ANY          -- authentication is being used
             }

     -- protocol data units

     PDUs ::=
             CHOICE {
                         get-request
                             GetRequest-PDU,

                         get-next-request
                             GetNextRequest-PDU,

                         get-response
                             GetResponse-PDU,

                         set-request
                             SetRequest-PDU,

                         trap
                             Trap-PDU
                     }

     -- PDUs

     GetRequest-PDU ::=
         [0]
             IMPLICIT PDU

     GetNextRequest-PDU ::=
```

```
GetResponse-PDU ::=
    [2]
        IMPLICIT PDU

SetRequest-PDU ::=
    [3]
        IMPLICIT PDU

PDU ::=
        SEQUENCE {
           request-id
                INTEGER,

            error-status      -- sometimes ignored
                INTEGER {
                    noError(0),
                    tooBig(1),
                    noSuchName(2),
                    badValue(3),
                    readOnly(4),
                    genErr(5)
                },

            error-index       -- sometimes ignored
               INTEGER,

            variable-bindings -- values are sometimes ignored
                VarBindList
        }

Trap-PDU ::=
    [4]
       IMPLICIT SEQUENCE {
            enterprise        -- type of object generating
                              -- trap, see sysObjectID in [5]

                OBJECT IDENTIFIER,

            agent-addr        -- address of object generating
                NetworkAddress, -- trap

            generic-trap      -- generic trap type
                INTEGER {
                    coldStart(0),
                    warmStart(1),
                    linkDown(2),
                    linkUp(3),
                    authenticationFailure(4),
                    egpNeighborLoss(5),
                    enterpriseSpecific(6)
                },

            specific-trap  -- specific code, present even
                INTEGER,   -- if generic-trap is not
                           -- enterpriseSpecific

            time-stamp     -- time elapsed between the last
                TimeTicks, -- (re)initialization of the
                           network
                           -- entity and the generation of the
                           trap
```

```
                    VarBindList
            }

    -- variable bindings

    VarBind ::=
            SEQUENCE {
                name
                    ObjectName,

                value
                    ObjectSyntax
            }

    VarBindList ::=
            SEQUENCE OF
               VarBind

    END
```

## 6. Acknowledgements

This memo was influenced by the IETF SNMP Extensions working group:

Karl Auerbach, Epilogue Technology
K. Ramesh Babu, Excelan
Amatzia Ben-Artzi, 3Com/Bridge
Lawrence Besaw, Hewlett-Packard
Jeffrey D. Case, University of Tennessee at Knoxville
Anthony Chung, Sytek
James Davidson, The Wollongong Group
James R. Davin, MIT Laboratory for Computer Science
Mark S. Fedor, NYSERNet
Phill Gross, The MITRE Corporation
Satish Joshi, ACC
Dan Lynch, Advanced Computing Environments
Keith McCloghrie, The Wollongong Group
Marshall T. Rose, The Wollongong Group (chair)
Greg Satz, cisco
Martin Lee Schoffstall, Rensselaer Polytechnic Institute
Wengyik Yeong, NYSERNet

## 7. References

[1] Cerf, V., "IAB Recommendations for the Development of Internet Network Management Standards", RFC 1052, IAB, April 1988.

[2] Rose, M., and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based internets", RFC 1065, TWG, August 1988.

[3] McCloghrie, K., and M. Rose, "Management Information Base for Network Management of TCP/IP-based internets", RFC 1066, TWG, August 1988.

[4] Cerf, V., "Report of the Second Ad Hoc Network Management Review Group", RFC 1109, IAB, August 1989.

[5] Rose, M., and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based Internets",

[6] McCloghrie, K., and M. Rose, "Management Information Base for Network Management of TCP/IP-based Internets", RFC 1156, Hughes LAN Systems and Performance Systems International, May 1990.

[7] Case, J., M. Fedor, M. Schoffstall, and J. Davin, "A Simple Network Management Protocol", Internet Engineering Task Force working note, Network Information Center, SRI International, Menlo Park, California, March 1988.

[8] Davin, J., J. Case, M. Fedor, and M. Schoffstall, "A Simple Gateway Monitoring Protocol", RFC 1028, Proteon, University of Tennessee at Knoxville, Cornell University, and Rensselaer Polytechnic Institute, November 1987.

[9] Information processing systems - Open Systems Interconnection, "Specification of Abstract Syntax Notation One (ASN.1)", International Organization for Standardization, International Standard 8824, December 1987.

[10] Information processing systems - Open Systems Interconnection, "Specification of Basic Encoding Rules for Abstract Notation One (ASN.1)", International

Organization for Standardization, International Standard 8825, December 1987.

[11] Postel, J., "User Datagram Protocol", RFC 768, USC/Information Sciences Institute, November 1980.

## Security Considerations

Security issues are not discussed in this memo.

## Authors' Addresses

Jeffrey D. Case
SNMP Research
P.O. Box 8593
Knoxville, TN 37996-4800

Phone: (615) 573-1434

Email: case@CS.UTK.EDU

Mark Fedor
Performance Systems International
Rensselaer Technology Park
125 Jordan Road
Troy, NY 12180

Phone: (518) 283-8860

Email: fedor@patton.NYSER.NET

Martin Lee Schoffstall
Performance Systems International
Rensselaer Technology Park

Phone: (518) 283-8860

Email: schoff@NISC.NYSER.NET

James R. Davin
MIT Laboratory for Computer Science, NE43-507
545 Technology Square
Cambridge, MA 02139

Phone: (617) 253-6020

EMail: jrd@ptt.lcs.mit.edu

Comment on RFC 1157

Comments about this RFC:

- RFC 1157: por favor profesionalicen los RFC, dan pena. Es demasiado costoso trabajar a... by intereseado (5/13/2004)
- RFC 1157: Hello, An excellent site. I am able to learn many stotras. Very good work. ... by <Default> (12/9/2006)
- RFC 1157: Hi, great work, It's a wonderful site. http://cheap-levitra.drzeo.org ... by <Default> (11/7/2006)
- RFC 1157: Is there is standard for MOSY format. If yes, form where I could get it. by raheel (3/3/2005)
- RFC 1157: Hello, This site is simply very good... http://uturare.thehostcity.com/seven... by None (12/24/2006)
- RFC 1157: Hello, Very good site with nice info's http://baby-gift-new-sibling.nice-sea... by TaN (10/22/2006)
- RFC 1157: Very nice site. It is useful for everybody. http://ionamin-on.tripod.com/ion... by Merlin (10/16/2006)
- RFC 1157: Hello, This site is cool! http://vocational-schools.orhat.info by <Default> (11/24/2006)
- RFC 1157: Hi, It's a great site http://insurother.tripod.com/carisoprodol-online-tc/ ... by LoWyeR (10/24/2006)
- RFC 1157: Hi! You have a cool homepage! http://valtrex-over-the-counter.amrud.org ... by <Default> (11/12/2006)
- RFC 1157: Hello, This site is simply very good... http://otoquqisoweol.thehostcity.com... by None (12/18/2006)
- RFC 1157: open my comment by octieu (12/16/2003)
- RFC 1157: Hello, Nice site. I am able to learn many stotras. Very good work. http://te... by None (12/10/2006)
- RFC 1157: Hello, Nice site. I am able to learn many stotras. Very good work. http://pi... by None (12/16/2006)
- RFC 1157: click this blue link to see me naked by click this blue link to see me naked (10/3/2005)
- RFC 1157: Your comment will not be displayed immediately by Andy (10/6/2004)
- RFC 1157: You are commenting on RFC 1157. Once accepted, your comment will be displayed... by Andy (10/6/2004)
- RFC 1157: Wow, design is much better from my last visit ,my respect! http://voltc.ifra... by Shanchik (10/19/2006)

(11/17/2006)
- RFC 1157: Hello, Great site! http://opewepo.275mb.com/order-levitra-online.html by <Default> (11/29/2006)

Previous: RFC 1156 - Management Information Base for network management of TCP/IP-based internets

Next: RFC 1158 - Management Information Base for network management of TCP/IP-based internets: MIB-II

[ RFC Index | RFC Search | Usenet FAQs | Web FAQs | Documents | Cities ]