Herramientas de Gestión basada en Web

Daniel Arias Figueroa

Trabajo de Tesis

Magister en Informática
Redes de Comunicaciones de Datos

Director: Javier Díaz

Diciembre de 1.999

Universidad Nacional de La Plata
Facultad de Informática
Calle 50 y 115 – 1° Piso
(1900) La Plata - Argentina

A la memoria de mi hermano
Julio Cesar Arias Figueroa

Agradecimientos

Mucha gente ayudo para que este trabajo fuera posible. Quisiera agradecer especialmente a mi director Javier Díaz, Mariela Finetti, Gustavo Gil, Miguel Luengo, Emilce Ottavianelli, Jorge Ramírez, Daniel Morales y Patricia Mac Gaul.

Agradezco también al Laboratorio de Investigación y Desarrollo en Tecnologías Informáticas – L.I.D.T.I. dependiente de la Facultad de Ciencias Exactas de la Universidad Nacional de Salta por la infraestructura aportada.

Por último y más importante quiero agradecer a mi esposa por su infinita paciencia y por soportar tantos viajes.

CONTENIDO

1 INTRODUCCIÓN

1.1 DESCRIPCIÓN GENERAL

El presente trabajo surge como una necesidad en el Campus de la Universidad Nacional de Salta, de contar con herramientas que faciliten al Ingeniero de redes realizar tareas de administración desde cualquier punto de la red, independizándolo de esta manera de la plataforma necesaria para ejecutar aplicaciones de gestión.

La Administración o Gestión basada en Web es la aplicación de la tecnología World Wide Web a redes y administración de dispositivos. Pretende aprovechar la amplia difusión de los navegadores como interfaz de usuario universal, para utilizarlos como interfaz para las aplicaciones de gestión. Debido a que esta tecnología es relativamente nueva, pocos investigadores se adentraron en el tema hasta ahora.

El principal beneficio de los mecanismos de Gestión basados en Web es que los desarrolladores de aplicaciones no tienen por qué conocer los detalles de los protocolos de gestión para manejar dispositivos remotos. Adicionalmente esto permite abstraer los diferentes protocolos y unificarlos con una única visión.

Este trabajo plantea por una parte inspeccionar el área de la Gestión basada en Web, y compararla con las Herramientas de Gestión tradicionales basadas en SNMP. Se hace énfasis en muchas características como seguridad, eficiencia, costo, interfaz amigable, etc. Por otro lado intenta desarrollar un conjunto de herramientas que sean rápidamente implementables y permitan al Ingeniero de red realizar algunas operaciones de administración en agentes del tipo pc/routers, ver estadísticas, estado y evolución de estos dispositivos.

1.2 ESTRUCTURA DE ESTE INFORME

Este informe está dividido en cinco partes. La primera es una recopilación de los dos protocolos de gestión mas difundidos, describiendo su arquitectura, especificaciones, ventajas y desventajas de su implementación.

Una segunda parte describe los elementos principales de la World Wide Web y como éstos pueden ser utilizados para desarrollar aplicaciones de gestión, define qué es Gestión basada en Web, tipos de arquitectura y presenta una comparación con las herramientas tradicionales de administración.

Una tercera parte describe la tecnología utilizada para llevar a cabo el diseño e implementación de las Herramientas de Administración a través del Web (SNMP Web Tools).

La cuarta parte describe la arquitectura de la aplicación desarrollada, su funcionalidad, especifica detalles de su construcción y se presenta un capitulo con los casos habituales de uso.

Por último, se anexa la mayor parte del código fuente de las herramientas desarrolladas, evitando presentar aquellos módulos que sean repetitivos.

En un CDROM se adjuntan todos los aplicativos y la documentación necesaria para instalar y utilizar las Herramientas de Gestión basada en Web.

Al finalizar este informe se encuentran las conclusiones y recomendaciones para el trabajo futuro.

Nota sobre el vocabulario utilizado

Es importante hacer notar que muchos términos técnicos utilizados en este informe no siempre tienen una traducción literal en español, por lo que se incluyen entre paréntesis las palabras o frases en inglés, para facilitar la comprensión del texto.

1.3 FINES Y OBJETIVOS

1.3.1 Objetivos Generales

- Inspeccionar el área de la Gestión basada en Web, y su comparación con las Herramientas de Gestión tradicionales basadas en SNMP desde el punto de vista de la seguridad, eficiencia, costo, interfaz amigable, escalabilidad, expansión y niveles de abstracción.

- Contar con herramientas que faciliten las tareas de administración desde cualquier punto de la red, independizándolo de la plataforma necesaria para ejecutar aplicaciones de gestión.

- Construir una infraestructura de gestión basada en el Web, que inicialmente soporte el protocolo de administración de redes SNMP, y que haga posible en trabajos posteriores la migración a otros protocolos de gestión.

1.3.2 Objetivos Específicos

- Adquirir los conocimientos necesarios sobre protocolos de gestión (SNMP y CMIP).

- Explorar los mecanismos de Gestión de red basados en Web.

- Permitir a los ingenieros de redes realizar las tareas de administración de sus dispositivos directamente desde el Web.

- Introducir un soporte de alto nivel que permita la utilización del Lenguaje Perl en el desarrollo de aplicaciones de gestión de redes.

- Implementar una Base de Datos que permita “Mantener Estado de Objetos Mib”, con el fin de poder realizar estadísticas y gráficos del estado y evolución de los dispositivos de red.

- Implementar la captura de SNMP Traps a través de la base de datos y generar e-mails automáticos a los administradores registrados.

- Soportar la administración de la base de datos a través de Web con lenguaje SQL.

- Construir una aplicación de gestión elemental que utilice los principios enunciados en los puntos anteriores.

1.3.3 Justificación Tecnológica

Actualmente las actividades de gestión usan técnicas y herramientas muchas veces aisladas e incompatibles entre sí. Se requiere por tanto de una pronta solución que además de simple, se convierta en una plataforma unificada para gestionar no solo redes, sino también sistemas y aplicaciones.

La Gestión basada en Web (Web based Management) es un acercamiento promisorio que puede proveer una solución de gestión realmente integrada.

Este trabajo plantea inspeccionar el área de la Gestión basada en Web, y desarrollar una primera fase que sea rápidamente implementable en el entorno para la cual fue desarrollada.

2 PROTOCOLO SNMP

En esta primera parte se trata la situación actual de los dos protocolos de gestión de red más importantes: el SNMP (Simple Network Management Protocol, protocolo simple de gestión de red) y el CMIP (Common Management Information Protocol, protocolo común de gestión de información). Se presentan las ventajas y desventajas de cada uno.

A finales de los años 70 las redes de computadoras experimentaron un espectacular crecimiento y empezaron a conectarse entre sí. A estas nuevas redes se les llamó inter-redes o internets. Pronto se hicieron muy difíciles de gestionar, y se hizo necesario el desarrollo de un protocolo de gestión.

El primer protocolo que se usó fue el SNMP [RFC 1157]. Se diseñó como un recurso provisional para "ir tirando" hasta que se desarrollara otro protocolo más elaborado.

En los años 80 aparecieron dos nuevos protocolos: por un lado, la segunda versión del SNMP, que incorporaba muchas de las funciones del original (que sigue en uso) e incluía nuevas características que mejoraban sus deficiencias. Por otro, el CMIP, que estaba mejor organizado y contenía muchas más funciones que las dos versiones del SNMP.

Pronto el público general tendrá que elegir entre CMIP [RFC 1189] y SNMPv2 [RFC 1441], y esta decisión será de gran importancia: no en vano una empresa gasta alrededor del 15% del presupuesto asignado a sistemas de información en gestión de red.

Los criterios de elección deben basarse en las necesidades del usuario, esto es, un buen sistema de seguridad de red, una interfaz amigable, implementación relativamente barata y una reducción del tiempo empleado en gestión. Estos serán algunos de los criterios que se seguirán en la comparación de ambos protocolos.

2.1 INTRODUCCIÓN A SNMP

Para el desarrollo de la gestión de redes en inter-redes basadas en TCP/IP, el IAB (Internet Activities Board) decidió seguir la estrategia de usar a corto plazo el Simple Network Management Protocol (SNMP) para gestionar los nodos, proponiendo para largo plazo la estructura de gestión de redes OSI. Se escribieron entonces dos documentos para definir la gestión de la información: RFC 1065 que definía la Estructura de la Información de Gestión (Structure of Management Information, SMI), y RFC 1066, que definía la Base de Información de Gestión (Management Information Base, MIB). Ambos documentos fueron diseñados para ser compatibles con la estructura SNMP y la de gestión de redes OSI.

Posteriormente se observó que los requerimientos de SNMP y los de gestión de redes OSI diferían más de lo esperado en un principio, por lo que los requerimientos de compatibilidad entre el SMI y el MIB fueron suspendidos.

La IAB ha designado al SNMP, a la SMI, y a la Internet MIB inicial como "Protocolos Estándar", con status de "Recomendado". Por medio de esta acción, la IAB recomienda que todas las implementaciones de IP y TCP sean gestionables por red, y los adopten y apliquen.

Así pues, la actual estructura para gestión de redes basadas en TCP/IP consiste en:

- Estructura e Identificación de la Información de Gestión para redes basadas en TCP/IP, que describe cómo se definen los objetos gestionados contenidos en el MIB tal y como se especifica en la RFC 1155.
- Protocolo de Gestión de Redes Simples, que define el protocolo usado para gestionar estos objetos, según se expone en la RFC 1157.

2.2 LA ARQUITECTUTA DE SNMP

Implícita en el modelo de arquitectura del SNMP existe una colección de estaciones de gestión de red y de elementos de red. Las estaciones ejecutan aplicaciones de administración que monitorizan y controlan los elementos de red. Los elementos de red son dispositivos como hosts, gateways, servidores de terminal, y parecidos, que poseen agentes de gestión para realizar las funciones de administración de red solicitadas por las estaciones de gestión de red. El SNMP es usado para transmitir información de administración entre las estaciones de gestión y los agentes en los elementos de red.

2.2.1 Propósitos de la arquitectura

El SNMP explícitamente minimiza el número y complejidad de las funciones de gestión realizadas por el propio agente de gestión. Esta meta es atractiva al menos en cuatro aspectos:

- El coste de desarrollo del software del agente de gestión necesario para soportar el protocolo se reduce acordemente.
- El grado de complejidad de funciones de gestión soportado remotamente se incrementa, posibilitando un uso completo de los recursos de Internet en la tarea de gestión imponiendo las mínimas restricciones posibles en la forma y sofisticación de herramientas de gestión.
- Los conjuntos simplificados de funciones de gestión son fácilmente entendibles y usados por los creadores de herramientas de gestión de red.

Un segundo objetivo del protocolo es que el paradigma funcional para monitorizar y controlar sea lo suficientemente flexible como para posibilitar aspectos de gestión y operación de la red adicionales y posiblemente no anticipados.

Un tercer propósito es que la arquitectura sea en lo posible independiente de los mecanismos de hosts o gateways particulares.

2.2.2 Elementos de la arquitectura

La arquitectura SNMP formula una solución al problema de gestión de redes en términos de los siguientes puntos:

- Alcance de la información de gestión comunicada por el protocolo.
- Representación de la información de gestión comunicada por el protocolo.
- Operaciones soportadas por el protocolo en la información de gestión.
- Forma y significado de los intercambios entre entidades de gestión.
- Forma y significado de las referencias a la información de gestión.

Alcance de la información de gestión

El alcance de la información de gestión transmitida por operaciones del SNMP es exactamente el representado por casos de todos los tipos de objetos no agregados, definidos en el estándar MIB [RFC 1156] de Internet, o definidos en cualquier otro sitio de acuerdo a las convenciones expuestas en el estándar SMI [RFC 1155] de Internet.

Representación de la información de gestión

La información de gestión se representa según el lenguaje ASN.1[1], que es especificado para la definición de tipos no agregados en el SMI. El SNMP utiliza un subconjunto bien definido de dicho lenguaje, incluyendo un subconjunto más complejo para la descripción de objetos gestionados y para describir las unidades de datos de protocolo (PDU's) utilizadas para gestionar esos objetos. Así mismo solo se utiliza un subconjunto de las reglas básicas de codificación del ASN.1, esto es, todas las codificaciones utilizan la forma de longitud definida.

Con el deseo de facilitar una futura transición a protocolos de gestión de redes basados en OSI, se procedió a la definición en el lenguaje ASN.1 de un SMI standard de Internet y de un MIB.

Operaciones soportadas por la información de gestión

El SNMP modela las funciones del agente de gestión como lecturas (get) o escrituras (set) de variables. Esta estrategia posee al menos dos consecuencias positivas:

- Limita el número esencial de funciones de gestión realizadas por el agente de gestión a dos.
- Evita introducir el soporte de comandos de gestión imperativos en la definición del protocolo.

[1] ISO/IEC 8824:1990

La estrategia plantea que la monitorización del estado de la red se puede basar a cualquier nivel de detalle en el sondeo (poll) de la información apropiada en la parte de los centros de monitorización. Un número limitado de mensajes no solicitados (traps) guían el objetivo y la secuencia del sondeo.

Las funciones de los pocos comandos imperativos actualmente soportados pueden ser fácilmente implementadas en este modelo de modo asíncrono.

Forma y significado de los intercambios

La comunicación de la información de gestión entre entidades de gestión se realiza en el SNMP por medio del intercambio de mensajes de protocolo.

El intercambio de mensajes SNMP sólo requiere un servicio de datagramas poco fiable, y todo mensaje se representa por un único datagrama de transporte.

Forma y significado de las referencias a objetos gestionados

El SMI [RFC 1155] requiere que la definición de un protocolo de gestión contemple:

- Resolución de referencias MIB ambiguas.
- Resolución de referencias MIB en presencia de múltiples versiones MIB.
- Identificación de casos particulares de tipos de objetos definidos en el MIB.

Resolución de referencias MIB ambiguas: debido a que el alcance de cualquier operación SNMP está conceptualmente confinado a los objetos relevantes a un único elemento de red, y ya que todas las referencias SMI a objetos MIB son por medio de nombres de variables únicos, no hay posibilidad de que una referencia SNMP a cualquier tipo de objeto definido en el MIB se pueda resolver entre múltiples casos de ese tipo.

Resolución de referencias entre versiones MIB: el objeto referenciado por cualquier operación SNMP es exactamente el especificado como parte de la operación de petición, o en el caso de una operación get-next su sucesor en el conjunto de MIB. En particular, una referencia a un objeto como parte de una versión del MIB estándar de Internet, no se aplica a ningún objeto que no sea parte de dicha versión, excepto en el caso de que la operación sea get-next, y que el nombre del objeto especificado sea el último lexicográficamente entre los nombres de todos los objetos presentados como parte de dicha versión.

Identificación de los casos de objetos: cada caso de un tipo de objeto definido en el MIB se identifica en las operaciones SNMP por un nombre único llamado su "nombre de variable". En general, el nombre de una variable SNMP es un identificador de objeto de la forma x.y, donde x es el nombre del tipo de objeto no agregado definido en el MIB, e y es un fragmento de un identificador de objeto que de forma única para dicho tipo de objeto, identifica el caso deseado. Esta estrategia de denominación admite la completa explotación de la semántica de la PDU GetNextRequest, dado que asigna nombres para variables relacionadas de forma que sean contiguas en el orden lexicográfico de todas las variables conocidas en el MIB.

2.3 SNMP: ESPECIFICACIONES DEL PROTOCOLO

El protocolo de administración de red es un protocolo de aplicación por el que las variables del MIB de un agente pueden ser inspeccionadas o alteradas.

Las entidades de protocolo se comunican entre sí mediante mensajes, cada uno formado únicamente por un datagrama UDP. Cada mensaje está formado por un identificador de versión, un nombre de comunidad SNMP y una PDU (Protocol Data Unit - Unidad de datos de protocolo). Estos datagramas no necesitan ser mayores que 484 bytes, pero es recomendable que las implementaciones de este protocolo soporten longitudes mayores.

Todas las implementaciones del SNMP soportan 5 tipos de PDU:

- GetRequest-PDU
- GetNextRequest-PDU
- GetResponse-PDU
- SetRequest-PDU
- Trap-PDU

2.3.1 Elementos de procedimiento

Se describirán a continuación las acciones que realiza una entidad de protocolo en una implementación SNMP. Definiremos dirección de transporte como una dirección IP seguida de un número de puerto UDP (Si se está usando el servicio de transporte UDP).

Cuando una entidad de protocolo envía un mensaje, realiza las siguientes acciones:

1. Construye la PDU apropiada como un objeto definido con el lenguaje ASN.1.
2. Pasa esta PDU, junto con un nombre de comunidad y las direcciones de transporte de fuente y destino, a un servicio de autentificación. Este servicio generará en respuesta otro objeto en ASN.1.
3. La entidad construye ahora un mensaje en ASN.1 usando el objeto que le ha devuelto el servicio de autentificación y el nombre de comunidad.
4. Este nuevo objeto se envía a la entidad destino usando un servicio de transporte.

Cuando una entidad de protocolo recibe un mensaje, realiza las siguientes acciones:

1. Hace un pequeño análisis para ver si el datagrama recibido se corresponde con un mensaje en ASN.1. Si no lo reconoce, el datagrama es descartado y la entidad no realiza más acciones.
2. Observa el número de versión. Si no concuerda descarta el datagrama y no realiza más acciones.

3. Pasa los datos de usuario, el nombre de comunidad y las direcciones de transporte de fuente y destino al servicio de autentificación. Si es correcto, este devuelve un objeto ASN.1. Si no lo es, envía una indicación de fallo. Entonces la entidad de protocolo puede generar una trampa (trap), descarta el datagrama y no realiza más acciones.
4. La entidad intenta reconocer la PDU. Si no la reconoce, descarta el datagrama. Si la reconoce, según el nombre de comunidad adopta un perfil y procesa la PDU. Si la PDU exige respuesta, la entidad iniciará la respuesta ahora.

2.3.2 Estructura de una PDU

Los datos que incluye una PDU genérica son los siguientes:

- RequestID: Entero que indica el orden de emisión de los datagramas. Este parámetro sirve también para identificar datagramas duplicados en los servicios de datagramas poco fiables.
- ErrorStatus: Entero que indica si ha existido un error. Puede tomar los siguientes valores, que se explicarán posteriormente:

  - ✓ noError (0)
  - ✓ tooBig (1)
  - ✓ noSuchName (2)
  - ✓ badValue (3)
  - ✓ readOnly (4)
  - ✓ genErr (5)

- ErrorIndex: Entero que en caso de error indica qué variable de una lista ha generado ese error.
- VarBindList: Lista de nombres de variables con su valor asociado. Algunas PDU quedan definidas sólo con los nombres, pero aún así deben llevar valores asociados. Se recomienda para estos casos la definición de un valor NULL.

2.3.3 GetRequest-PDU y GetNextRequest-PDU

Son PDU's que solicitan a la entidad destino los valores de ciertas variables. En el caso de GetRequest-PDU estas variables son las que se encuentran en la lista VarBindList; en el de GetNextRequest-PDU son aquellas cuyos nombres son sucesores lexicográficos de los nombres de las variables de la lista. Como se puede observar, GetNextRequest-PDU es útil para confeccionar tablas de información sobre un MIB.

Siempre tienen cero los campos ErrorStatus y ErrorIndex. Son generadas por una entidad de protocolo sólo cuando lo requiere su entidad de aplicación SNMP.

Estas PDU's siempre esperan como respuesta una GetResponse-PDU.

2.3.4 SetRequest-PDU

Ordena a la entidad destino poner a cada objeto reflejado en la lista VarBindList el valor que tiene asignado en dicha lista. Es idéntica a GetRequest-PDU, salvo por el identificador de PDU. Es generada por una entidad de protocolo sólo cuando lo requiere su entidad de aplicación SNMP. Espera siempre como respuesta una GetResponse-PDU.

2.3.5 GetResponse-PDU

Es una PDU generada por la entidad de protocolo sólo como respuesta a GetRequest-PDU, GetNextRequest-PDU o SetRequest-PDU. Contiene o bien la información requerida por la entidad destino o bien una indicación de error.

Cuando una entidad de protocolo recibe una GetRequest-PDU, una SetRequest-PDU o una GetNextRequest-PDU, sigue las siguientes reglas:

1. Si algún nombre de la lista (o el sucesor lexicográfico de un nombre en el caso de GetNextRequest-PDU) no coincide con el nombre de algún objeto en la vista del MIB al que se pueda realizar el tipo de operación requerido ("set" o "get"), la entidad envía al remitente del mensaje una GetResponse-PDU idéntica a la recibida, pero con el campo ErrorStatus puesto a 2 (noSuchName), y con el campo ErrorIndex indicando el nombre de objeto en la lista recibida que ha originado el error.
2. De la misma manera actúa si algún objeto de la lista recibida es un tipo agregado (como se define en el SMI), si la PDU recibida era una GetRequest-PDU.
3. Si se ha recibido una SetRequest-PDU y el valor de alguna variable de la lista no es del tipo correcto o está fuera de rango, la entidad envía al remitente una GetResponse-PDU idéntica a la recibida, salvo en que el campo ErrorStatus tendrá el valor 3 (badValue) y el campo ErrorIndex señalará el objeto de la lista que ha generado el error.
4. Si el tamaño de la PDU recibida excede una determinada limitación, la entidad enviará al remitente una GetResponse-PDU idéntica a la recibida, pero con el campo ErrorStatus puesto a 1 (tooBig).
5. Si el valor de algún objeto de la lista no puede ser obtenido (o alterado, según sea el caso) por una razón no contemplada en las reglas anteriores, la entidad envía al remitente una GetResponse-PDU idéntica a la recibida, pero con el campo ErrorStatus puesto a 5 (genErr), y el campo ErrorIndex indicando el objeto de la lista que ha originado el error.

Si no se llega a aplicar alguna de estas reglas, la entidad enviará al remitente una GetResponse-PDU de las siguientes características:

- Si es una respuesta a una GetResponse-PDU, tendrá la lista varBindList recibida, pero asignando a cada nombre de objeto el valor correspondiente.
- Si es una respuesta a una GetNextResponse-PDU, tendrá una lista varBindList con todos los sucesores lexicográficos de los objetos de la lista recibida, que estén en la vista del MIB relevante y que sean susceptibles de

ser objeto de la operación "get". Junto con cada nombre, aparecerá su correspondiente valor.

- Si es una respuesta a una SetResponse-PDU, será idéntica a esta, pero antes la entidad asignará a cada variable mencionada en la lista varBindList su correspondiente valor. Esta asignación se considera simultánea para todas las variables de la lista.

En cualquiera de estos casos el valor del campo ErrorStatus es 0 (noError), igual que el de ErrorIndex. El valor del campo requestID es el mismo que el de la PDU recibida.

2.3.6 Trap-PDU

Es una PDU que indica una excepción o trampa. Es generada por una entidad de protocolo sólo a petición de una entidad de aplicación SNMP.

Cuando una entidad de protocolo recibe una Trap-PDU [RFC 1215], presenta sus contenidos a su entidad de aplicación SNMP.

Los datos que incluye una Trap-PDU son los siguientes:

- enterprise: tipo de objeto que ha generado la trampa.
- agent-addr: dirección del objeto que ha generado la trampa.
- generic-trap: entero que indica el tipo de trampa. Puede tomar los siguientes valores:
  - ✓ coldStart (0)
  - ✓ warmStart (1)
  - ✓ linkDown (2)
  - ✓ linkUp (3)
  - ✓ authenticationFailure (4)
  - ✓ egpNeighborLoss (5)
  - ✓ enterpriseSpecific (6)
- specific-trap: entero con un código específico.
- time-stamp: tiempo desde la última inicialización de la entidad de red y la generación de la trampa.
- variable-bindings: lista tipo varBindList con información de posible interés.

Dependiendo del valor que tenga el campo generic-trap, se iniciarán unas u otras acciones:

- Trampa de arranque frío (coldStart): La entidad de protocolo remitente se está reinicializando de forma que la configuración del agente o la implementación de la entidad de protocolo puede ser alterada.
- Trampa de arranque caliente (warmStart): La entidad de protocolo remitente se está reinicializando de forma que ni la configuración del agente ni la implementación de la entidad de protocolo se altera.
- Trampa de conexión perdida (linkDown): La entidad de protocolo remitente reconoce un fallo en uno de los enlaces de comunicación representados en

la configuración del agente. Esta Trap-PDU contiene como primer elemento de la lista variable-bindings el nombre y valor de la interfaz afectada.

- Trampa de conexión establecida (linkUp): La entidad de protocolo remitente reconoce que uno de los enlaces de comunicación de la configuración del agente se ha establecido. El primer elemento de la lista variable-bindings es el nombre y el valor de la interfaz afectada.
- Trampa de fallo de autentificación (authenticationFailure): La entidad de protocolo remitente es la destinataria de un mensaje de protocolo que no ha sido autentificado.
- Trampa de pérdida de vecino EGP (egpNeighborLoss): Un vecino EGP con el que la entidad de protocolo remitente estaba emparejado ha sido seleccionado y ya no tiene dicha relación. El primer elemento de la lista variable-bindings es el nombre y el valor de la dirección del vecino afectado.
- Trampa específica (enterpriseSpecific): La entidad remitente reconoce que ha ocurrido algún evento específico. El campo specific-trap identifica qué trampa en particular se ha generado.

2.4 DEFINICIÓN DE UN MENSAJE

A continuación se representa la definición de un mensaje SNMP en lenguaje ASN.1:

```
RFC1157-SNMP DEFINITIONS ::= BEGIN

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

     -- Mensaje de alto nivel

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

               community          --nombre de comunidad
                    OCTET STRING,

               data
                    ANY          --se usa autentificación
          }

     -- Unidades de datos de protocolo

     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 ::=
    [1]
        IMPLICIT PDU
GetResponse-PDU ::=
    [2]
        IMPLICIT PDU
SetRequest-PDU ::=
    [3]
        IMPLICIT PDU

PDU ::=
    SEQUENCE {
        request-id
            INTEGER,

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

        error-index       -- a veces ignorado
            INTEGER,

        variable-bindings  -- los valores a veces ignorados
            VarBindList
    }

Trap-PDU ::=
    [4]
        IMPLICIT SEQUENCE {
            enterprise -- tipo de objeto que genera la
                    -- trampa, ver sysObjectID en [5]

                OBJECT IDENTIFIER,
```

```
            agent-addr -- dirección del objeto que genera
                NetworkAddress  -- la trampa

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

            specific-trap    -- código específico, presente
                INTEGER, -- incluso cuando generic-trap
                     -- no es enterpriseSpecific

            time-stamp        -- tiempo desde la última
                TimeTicks, -- (re)inicialización de la
                      -- entidad de red y la generación
                      -- de la trampa

            variable-bindings  -- información de interés
                VarBindList
        }

    -- duplas variable-valor

    VarBind ::=
        SEQUENCE {
            name
                ObjectName,
            value
                ObjectSyntax
        }

    VarBindList ::=
        SEQUENCE OF
            VarBind

    END
```

2.5 COEXISTENCIA ENTRE SNMPv1 Y SNMPv2

Se comentará a continuación qué hay que realizar para garantizar la compatibilidad y coexistencia de las dos versiones del protocolo SNMP [RFC 1908].

2.5.1 Información de gestión

La forma que tiene SNMPv2 para manejar los objetos gestionados no es mas que una extensión de SNMPv1. Así, ambas versiones utilizan el lenguaje ASN.1 para la notación. De hecho, lo que hace principalmente la versión 2 es normalizar la forma de definir los módulos MIB tal y como han dictado los años de experiencia trabajando con la primera versión.

Para que un módulo MIB o una declaración en SNMP se haga compatible con SNMPv2 necesita una serie de cambios. Normalmente estos cambios no exigen la invalidación de los objetos que contiene, ya que no son cambios muy graves. Son cambios referentes al vocabulario o la sintaxis (por ejemplo el guión se convierte en carácter prohibido en los nombres de variables), la definición de nuevos tipos (como integer32), o a la conversión de ciertas partes del MIB de opcionales a obligatorias (por ejemplo, ahora todos los objetos deben tener una cláusula DESCRIPTION). Hay cambios que son obligatorios y hay cambios que son sólo recomendados.

2.5.2 Operaciones de protocolo

Se considerarán dos áreas: el comportamiento del intermediario entre una entidad SNMPv2 y una agente SNMPv1, y el comportamiento de entidades de protocolo bilingües actuando como administradoras.

Comportamiento de un agente intermediario

Para conseguir la coexistencia a nivel de protocolo, se puede utilizar un mecanismo intermediario. Una entidad SNMPv2 actuando como agente puede ser implementada y configurada para realizar esta labor.

Paso de SNMPv2 a SNMPv1

Para convertir peticiones de una entidad SNMPv2 administradora en peticiones a una entidad SNMPv1 agente:

1. Si es una GetRequest-PDU, una GetNextRequest-PDU o SetRequest-PDU, el agente intermediario la pasa sin alterar.
2. Si es una GetBulkRequest-PDU, el intermediario pone los campos non-repeaters y max-repetitions a cero, y la convierte en una GetNextRequest-PDU.

Paso de SNMPv1 a SNMPv2

Para convertir respuestas enviadas de una entidad SNMPv1 agente hacia una entidad SNMPv2 administradora:

1. Si es una GetResponse-PDU, pasa por el intermediario sin alteraciones. No obstante hay que observar que aunque una entidad SNMPv2 nunca generará una PDU de respuesta con un campo error-status con un valor de "noSuchName", "badValue" o "readOnly", el agente intermediario no debe cambiar este campo. Así la entidad administradora podrá interpretar la respuesta correctamente. Si se recibe una GetResponse-PDU con el campo error-status con el valor "tooBig", el intermediario eliminará los contenidos del campo variable-bindings antes de propagar la respuesta. También aquí hay que señalar que aunque una entidad SNMPv2 nunca enviará una PDU de respuesta con un "tooBig" ante una GetBulkRequest-PDU, el agente itermediario debe propagar dicha respuesta.
2. Si se recibe una Trap-PDU, se convertirá en una Trap-PDU de SNMPv2. Esto se consigue colocando en el campo variable-bindings dos nuevos elementos: sysUpTime.0, que toma el valor del campo timestamp de la Trap-PDU, y snmpTrapOID.0, que se calcula así: Si el valor del campo generic-trap es "enterpriseSpecific", entonces el valor usado es la concatenación del campo enterprise de la PDU con dos subidentificadores: '0', y el valor del campo specific-trap. Si no es así, se utiliza el valor definido para las Trap-PDU en la versión 2. En este caso se pone un elemento más en el campo variable-bindings: snmpTrapEnterprise.0, que toma el valor del campo enterprise de la PDU. Los destinos de esta Trap-PDU versión 2 se determinan según la implementación del agente intermediario.

Comportamiento de un administrador bilingüe

Para conseguir la coexistencia a nivel de protocolo, una entidad de protocolo actuando como administradora podría soportar las dos versiones de SNMP. Cuando una aplicación de administración necesita contactar con una entidad de protocolo agente, la entidad administradora consulta una base de datos local para seleccionar el protocolo de gestión adecuado. Para dar transparencia a las aplicaciones, la entidad administradora debe mapear las operaciones como si fuera un agente intermediario.

2.6 VENTAJAS Y DESVENTAJAS DE SNMP

2.6.1 Ventajas de SNMP

La ventaja fundamental de usar SNMP es que su diseño es simple por lo que su implementación es sencilla en grandes redes y la información de gestión que se necesita intercambiar ocupa pocos recursos de la red. Además, permite al usuario elegir las variables que desea monitorizar sin mas que definir:

- El título de la variable.
- El tipo de datos de la variable.
- Si la variable es de sólo lectura o también de escritura.
- El valor de la variable.

Otra ventaja de SNMP es que en la actualidad es el sistema más extendido. Ha conseguido su popularidad debido a que fue el único protocolo que existió en un principio y por ello casi todos los fabricantes de dispositivos como puentes y enrutadores diseñan sus productos para soportar SNMP.

La posibilidad de expansión es otra ventaja del protocolo SNMP: debido a su sencillez es fácil de actualizar.

2.6.2 Desventajas de SNMP

El protocolo SNMP no es ni mucho menos perfecto. Tiene fallos que se han ido corrigiendo.

La primera deficiencia de SNMP es que tiene grandes fallos de seguridad que pueden permitir a intrusos acceder a información que lleva la red. Todavía peor, estos intrusos pueden llegar a bloquear o deshabilitar terminales.

La solución a este problema es sencilla y se ha incorporado en la nueva versión SNMPv2. Básicamente se han añadido mecanismos para resolver:

- Privacidad de los datos, que los intrusos no puedan tomar información que va por la red.
- Autentificación, para prevenir que los intrusos manden información falsa por la red.
- Control de acceso, que restringe el acceso a ciertas variables a determinados usuarios que puedan hacer caer la red.

El mayor problema de SNMP es que se considera tan simple que la información está poco organizada, lo que no lo hace muy acertado para gestionar las grandes redes de la actualidad. Esto se debe en gran parte a que SNMP se creó como un protocolo provisional y no ha sido sustituido por otro de entidad.

De nuevo este problema se ha solucionado con la nueva versión SNMPv2 que permite una separación de variables con más detalle, incluyendo estructuras de datos para hacer más fácil su manejo. Además SNMPv2 incluye dos nuevas PDU's orientadas a la manipulación de objetos en tablas.

3 PROTOCOLO CMIP

3.1 INTRODUCCIÓN

Tras la aparición de SNMP como protocolo de gestión de red, a finales de los 80, gobiernos y grandes corporaciones plantearon el Protocolo Común de Gestión de Información CMIP (Common Management Information Protocol) que se pensó podría llegar a ser una realidad debido al alto presupuesto con que contaba. En cambio, problemas de implementación han retrasado su expansión de modo que solo está disponible actualmente de forma limitada y para desarrolladores.

CMIP [RFC 1189] fue diseñado teniendo en cuenta a SNMP, solucionando los errores y fallos que tenía SNMP y volviéndose un gestor de red mayor y más detallado. Su diseño es similar a SNMP por lo que se usan PDUs (Protocol Data Unit) como 'variables' para monitorizar la red.

En CMIP las variables son estructuras de datos complejas con muchos atributos, que incluyen:

- variables de atributos: representan las características de las variables.
- variables de comportamiento: qué acciones puede realizar.
- notificaciones: la variable genera una indicación de evento cuando ocurre un determinado hecho.

3.2 GESTIÓN EN OSI

Como CMIP es un protocolo de gestión de red implementado sobre OSI conviene introducir el marco de trabajo OSI en lo que respecta a gestión, ya que será la base para CMIP.

La gestión OSI posibilita monitorizar y controlar los recursos de la red que se conocen como "objetos gestionados". Para especificar la estandarización de la gestión de red se determina:

- Modelo o grupo de modelos de la inteligencia de gestión, hay 3 principales:
  - ✓ *Modelo de organización:* describe la forma en que las funciones de gestión se pueden distribuir administrativamente. Aparecen los dominios como particiones administrativas de la red.
  - ✓ *Modelo funcional:* describe las funciones de gestión (de fallos, de configuración, de contabilidad, de seguridad...) y sus relaciones.
  - ✓ *Modelo de información:* provee las líneas a seguir para describir los objetos gestionados y sus informaciones de gestión asociadas. Reside en el MIB (Management Information Base).
- Estructura para registrar, identificar y definir los objetos gestionados.
- Especificación detallada de los objetos gestionados.
- Serie de servicios y protocolos para operaciones de gestión remotas.

3.3 FUNDAMENTOS DE CMIP

CMIP es un protocolo de gestión de red que se implementa sobre el modelo de Interconexión de Redes Abiertas OSI (Open Systems Interconnection) que ha sido normalizado por la ISO (International Organization for Standardization's) en sus grupos de trabajo OIW (OSI Implementors Workshop) y ONMF (OSI Network Management Forum). Además existe una variante del mismo llamado CMOT [RFC 1095] que se implementa sobre un modelo de red TCP/IP.

En pocas palabras, CMIP es una arquitectura de gestión de red que provee un modo de que la información de control y de mantenimiento pueda ser intercambiada entre un gestor (manager) y un elemento remoto de red. En efecto, los procesos de aplicación llamados gestores (managers) residen en las estaciones de gestión, mientras que los procesos de aplicación llamados agentes (agents) residen en los elementos de red.

CMIP define una relación igual a igual entre el gestor y el agente incluyendo lo que se refiere al establecimiento y cierre de conexión, y a la dirección de la información de gestión. Las operaciones CMIS (Common Management Information Services) se pueden originar tanto en gestores como en agentes, permitiendo relaciones simétricas o asimétricas entre los procesos de gestión. Sin embargo, la mayor parte de los dispositivos contienen las aplicaciones que sólo le permiten hacer de agente.

En esta presentación nos vamos a centrar en CMIP ya que si éste apenas ha tenido éxito menos aún lo ha tenido el CMOT.

La serie CMIP está compuesta por 6 protocolos que se esquematizan en la siguiente tabla:

Procesos de Gestión de Aplicaciones	
CMISE ISO 9595/9596	
ACSE ISO 8649/8650	ROSE ISO DIS 9072-1/2
ISO Presentación ISO	
ISO Sesión ISO	
ISO Transporte ISO	

Tabla 3.1. El CMIP Protocol Suite

Como se puede ver, un sistema CMIP debe implementar una serie de protocolos de los cuales el CMISE (Common Management Information Service Element) es el que trabaja mano a mano con CMIP: todas las operaciones de gestión de red que crea CMISE, el CMIP las mapea en una operación en el CMIP remoto.

3.4 PROTOCOLOS DE APLICACIÓN EN CMIP

Para comunicarse entre sí dos entidades de aplicación pares del gestor y del agente se utilizan APDU's (Application Protocol Data Units). Como hemos visto, CMIP está compuesto de los protocolos OSI que siguen:

- **ACSE (Association Control Service Element):** se utiliza para establecer y liberar asociaciones entre entidades de aplicación. El establecimiento lo puede realizar el agente o el gestor; durante el proceso se intercambian los títulos de la entidad de aplicación para identificarse, y los nombres del contexto de aplicación para establecer un contexto de aplicación.
  Servicios que ACSE proporciona a CMISE:
  - ✓ A-ASSOCIATE, servicio confirmado utilizado para inicializar la asociación entre entidades de aplicación.
  - ✓ A-RELEASE, servicio confirmado usado para liberar una asociación entre entidades de aplicación sin pérdida de información.
  - ✓ A-ABORT, servicio no confirmado que causa la liberación anormal de una asociación con una posible pérdida de información.
  - ✓ A-P-ABORT, servicio iniciado por el proveedor que indica la liberación anormal de la asociación del servicio de presentación con posible pérdida de información.

- **ROSE (Remote Operation Service Element):** es el equivalente OSI a una llamada de un procedimiento remoto. ROSE permite la invocación de una operación en un sistema remoto. CMIP usa los servicios orientados a conexión proporcionados por ROSE para todas las peticiones, respuestas y respuestas de error.
  Servicios que ROSE proporciona a CMISE:
  - ✓ RO-INVOKE, servicio no confirmado que es usado por un usuario de ROSE para invocar que una operación sea realizada por un ROSE invocado remoto.
  - ✓ RO-RESULT, servicio no confirmado que un ROSE invocado usa para contestar a una previa indicación RO-INVOKE en el caso de que se haya realizado con éxito.
  - ✓ RO-ERROR, servicio no confirmado que es usado por un usuario de ROSE invocado para contestar a una previa indicación RO-INVOKE en el caso de que haya fracasado.
  - ✓ RO-REJECT, servicio no confirmado utilizado por un usuario de ROSE para rechazar una petición (indicación RO-INVOKE) del otro.

- **CMISE (Common Management Information Service Element):** Proporciona los servicios básicos de gestión confirmados y no confirmados para reportar eventos y manipular datos de gestión. CMISE hace uso de los servicios proporcionados por ROSE y ACSE. Servicios de CMISE: se denominan unidades funcionales y se resumen en la tabla siguiente las denominadas 'stand alone' (luego hay otras tres más). El número que sigue a cada unidad funcional está definido por el

CMIP. También se especifica en cada caso si se trata de servicio confirmado o no confirmado.

Unidad Funcional	Primitivas de Servicio	Modo
Conf. Event report invoker(0)	M-EVENT-REPORT Req/Conf	C
Conf. Event report performer(1)	M-EVENT-REPORT Ind/Rsp	C
event report invoker(2)	M-EVENT-REPORT Req	U
event report performer(3)	M-EVENT-REPORT Ind	U
Confirmed get invoker(4)	M-GET Req/Conf	N/A
Confirmed get performer(5)	M-GET Ind/Rsp	N/A
Confirmed set invoker(6)	M-SET Req/Conf	C
Confirmed set performer(7)	M-SET Ind/Rsp	C
Set invoker(8)	M-SET Req	U
set performer(9)	M-SET Ind	U
Confirmed action invoker(10)	M-ACTION Req/Conf	C
Confirmed action performer(11)	M-ACTION Ind/Rsp	C
Action invoker(12)	M-ACTION Req	U
Action performer(13)	M-ACTION Ind	U
Confirmed create invoker(14)	M-CREATE Req/Conf	N/A
Confirmed create performer(15)	M-CREATE Ind/Rsp	N/A
Confirmed delete invoker(16)	M-DELETE Req/Conf	N/A
Confirmed delete performer(17)	M-DELETE Ind/Rsp	N/A
Multiple reply(18)	Linked Identification	N/A
Multiple object selection(19)	Scope, Filter, Sync.	N/A
Extended service(20)	Extended Presentation	N/A

C = confirmado, U = no confirmado, N/A = no aplicable
Tabla 3.2. Unidades Funcionales

Se distinguen dos elementos en la comunicación:

- **"invoker"** es el invocador, el que llama a la ejecución de una operación remota
- **"performer"** es el que realiza la operación solicitada por un sistema remoto.

Existen aparte otras tres unidades funcionales que sólo son válidas si se seleccionan junto con alguna de las vistas del tipo 'stand alone'. Todas ellas son PDU's.

Aparecen dos elementos en la interacción de la gestión de red, el gestor y el agente (gestionado), que negocian cuatro tipos de asociación que se establecen entre dos entidades de aplicación:

Event: el gestionado puede enviar un M-EVENT-REPORTS.

Event/Monitor: como el Event y además el gestor puede usar M-EVENT-REPORTS, peticiones M-GET y recibir respuestas M-GET.

Monitor/Control: el gestor implementa M-GET, M-SET, M-CREATE, M-DELETE y M-ACTION, sin permitir el reporte de eventos.
Full Manager/Agent: soporta todas las funciones.

Un sistema debe soportar al menos una de estas asociaciones, y puede hacer de gestor o gestionado pero no dentro de la misma asociación. Esta es una manera de reducir el tamaño del código: para determinado grupo de unidades funcionales se le da una asociación. Del mismo modo en la negociación, no se negocia cada unidad funcional sino una asociación, lo cual es más eficiente.

El proceso de negociación usa los servicios A-ASSOCIATE y A-RELEASE para determinar en una asociación quién va a ser el gestor o el gestionado, y el tipo de asociación. Estas son las reglas de negociación:

- Un sistema agente (gestionado) solo puede requerir una asociación Event y puede crearla sólo si tiene un evento que reportar y no tiene un gestor asociado.
- Un sistema gestor puede requerir cualquier tipo de asociación.
- Una asociación se crea mediante un requerimiento A-ASSOCIATE con el AE-TITLE del solicitante y la aplicación en uso. Entonces el receptor puede devolver un A-ASSOCIATE con su AE-TITLE para aceptarlo o un A-ASSOCIATE para rechazarlo.
- Un sistema gestionado puede pedir dentro de una asociación bajar a otra asociación (de Full a Monitor/Control o Event/Monitor o Event). El gestor puede rechazar la petición. A continuación se presentan de manera más detallada las unidades funcionales que soporta cada asociación.

Grupos de Unidades Funcionales	Event Report	Get	Set	Create/ Delete	Action	Mult. Reply	Mult.Object Select
1. Event Monitor	U	no	no	no	no	no	No
2. Event Sender	U	no	no	no	no	no	No
3. Monitoring Mgr.	U	si	no	no	no	no	No
4. Monitored Agent	U	si	no	no	no	no	No
5. Simple Manager	U	si	C	no	no	si	No*
6. Simple Agent	U	si	C	no	no	si	No*
7. Controlling Mgr.	U	si	U/C	si	no	si	Si
8. Controlled Agent	U	si	U/C	si	no	si	Si
9. Full Manager	U/C	si	U/C	si	U/C	si	Si
10. Full Agent	U/C	si	U/C	si	U/C	si	Si

C = confirmado, U = no confirmado
* Simple Managers y Agents deben soportar un nivel de alcance.
Tabla 3.3.

3.5 GESTIÓN PROXY

En nuestro contexto, un proxy (intermediario, en inglés) es un gestor habilitado para realizar acciones en nombre de otro gestor. Si el dispositivo gestionado no soporta CMIP, todo lo relativo a gestión que vaya a él se redirige al proxy, el cual le hará llegar la información adaptada al protocolo que soporte, y viceversa.

3.6 INFORMACIÓN DE GESTIÓN

SMI (Structure of Management Information) define la estructura lógica de la información de gestión y cómo se identifica y describe. Este SMI está diseñado para usarse tanto en SNMP como en CMIP, pero cada uno lo implementa de manera específica. SMI define las siguientes funciones:

- **Alcance:** se utiliza para identificar los objetos gestionados que van a ser filtrados.
- **Filtrado:** se usa para seleccionar un subconjunto de objetos gestionados que satisfacen ciertas condiciones.
- **Sincronización:** una vez filtrados, se puede operar bajo el método 'best effort' por el cual, si falla una operación en un objeto sigue con el resto, y el método 'atomic' en el que la operación se realiza en todos o en ninguno.

El único requerido en CMIP es el de 'best effort' aunque el otro también puede ser soportado.

MIB (Management Information Base) viene especificado por SMI y define los objetos gestionados en la actualidad. Los objetos gestionados vienen definidos totalmente especificando los atributos o propiedades que tiene el objeto. Se entiende por atributos a los elementos de información que solo se pueden manipular como un todo y se les da un identificador.

Los objetos se jerarquizan según están contenidos unos dentro de otros (jerarquía de contención) y según sus propiedades (jerarquía de herencia).

3.7 VENTAJAS Y DESVENTAJAS DE CMIP

3.7.1 Ventajas de CMIP

El principal beneficio que aporta el protocolo CMIP es que no sólo se puede enviar información de gestión de o hacia un terminal, sino que es posible desarrollar tareas que serían imposibles bajo SNMP. Por ejemplo, si un terminal no puede encontrar un servidor de ficheros en un tiempo predeterminado, CMIP notifica el evento al personal adecuado. En SNMP el usuario tendría que guardar el número de intentos de acceso al servidor mientras que en CMIP de esto se encarga el propio protocolo.

CMIP soluciona varios de los fallos de SNMP. Por ejemplo, tiene incluidos dispositivos de gestión de la seguridad que soportan autorizaciones, control de

acceso, contraseñas. Como resultado de la seguridad que de por sí proporciona CMIP no necesita de posteriores actualizaciones.

Otra ventaja de CMIP es que haya sido creado no sólo por gobiernos sino también por grandes empresas, en los que puede tener en el futuro un mercado fiel.

3.7.2 Desventajas de CMIP

Si todo lo dicho hace a CMIP tan bueno, uno puede preguntarse: ¿por qué no se usa? La respuesta es que CMIP significa también desventajas: CMIP requiere 10 veces más recursos de red que SNMP. En otras palabras, muy pocas redes de la actualidad son capaces de soportar una implementación completa de CMIP sin grandes modificaciones en la red (muchísima más memoria y nuevos protocolos de agente). Por eso mucha gente piensa que CMIP está destinado al fracaso.

La única solución es disminuir el tamaño de CMIP cambiando sus especificaciones. Así han aparecido varios protocolos que funcionan con la base de CMIP con menos recursos, pero todavía no ha llegado el momento de prescindir del tan extendido SNMP.

Otro problema de CMIP es su dificultad de programación: existe tal cantidad de variables que sólo programadores muy habilidosos son capaces de aprovechar todo su potencial.

3.8 CONCLUSIONES DE LA IMPLEMENTACIÓN

SNMP es un conjunto de especificaciones de comunicación de red muy simple que cubre los mínimos necesarios de gestión de red exigiendo muy poco esfuerzo a la red sobre el que SNMP está implementado.

CMIP es un sistema de gestión de red muy bien diseñado que mejora muchas de las deficiencias del SNMP. El costo de estas mejoras es haberse convertido en un sistema tan grande y complejo que sólo las redes mejor equipadas pueden soportarlo.

Por tanto, es recomendable implementar SNMP antes que CMIP por los enormes recursos de sistema que CMIP requiere.

4 WORLD WIDE WEB

En esta sección se da una definición de lo que es World Wide Web, se describen sus elementos principales, y como éstos pueden ser utilizados para desarrollar aplicaciones de gestión.

Se podría definir al WWW (World Wide Web) como una colección de recursos unidos por medio de hiperenlaces (hyperlinks) que pueden ser accedidos, transferidos o ejecutados remotamente desde cualquier lugar de la inter-red, desde un servidor HTTP a un cliente HTTP, utilizando HTTP [RFC 1945, 2616] como protocolo primario de transferencia.

Este principio es descripto en la siguiente figura.

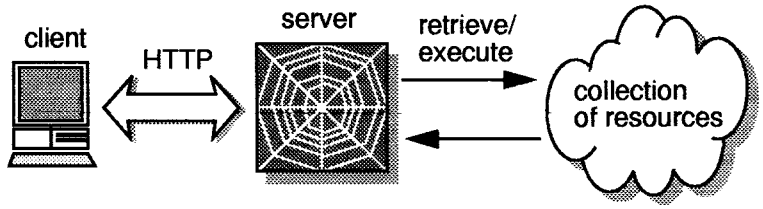

fig.4.1: Principios del World Wide Web

El cliente HTTP mencionado en la definición, puede ser cualquier visualizador (web browser), por ejemplo Netscape Communicator, o cualquier aplicación cliente que utilice HTTP como el protocolo primario de transferencia. Los recursos mencionados consisten principalmente de documentos HTML (Hypertext Markup Language) [RFC 1866]. Los documentos transferidos en la internet van codificados en un formato llamado MIME (Multipurpose Internet Mail Extensions) [RFC 1522].

4.1 HTML INTERACTIVO

Los documentos HTML pueden incluir tanto información estática como dinámica. Los documentos estáticos son almacenados en el servidor y generalmente no cambian, por el contrario, una página dinámica es creada instantáneamente como resultado de una consulta que el cliente envía al servidor.

HTML proporciona mucha flexibilidad a los clientes para interactuar con el servidor a través de los formularios. Esto se utiliza generalmente cuando un cliente necesita enviar una consulta de administración al Web (que puede ser una solicitud de un valor de un objeto gestionado o una operación para actualizar uno o más objetos).

4.2 HTML CLIENT-PULL

Muchos visualizadores proporcionan la capacidad para que los documentos HTML sean automáticamente refrescados o reemplazados por otros documentos después de transcurrir un cierto tiempo. Esta técnica es llamada “client-pull”, y es muy usada para construir aplicaciones de gestión que

necesitan monitorizar ciertos objetos. Para que un documento HTML se actualice automáticamente después de transcurrido un cierto intervalo, debe contener el siguiente encabezado:

```
<HTML><HEAD><META HTTP-EQUIV=”Refresh” CONTENT=1>
<TITLE>Documento Uno</TITLE></HEAD>
```

En este caso el documento es actualizado después de transcurrido un segundo. Si en su lugar se requiere cargar otro documento el encabezado se verá como:

```
<META HTTP-EQUIV=”Refresh” CONTENT=”12;
URL=http://server/doc.html>
```

Si este meta tag es utilizado, el documento será reemplazado por otro documento llamado *doc.html* después de 12 segundos.

En la figura 4.2 se describe el principio de un client-pull, en la cual un cliente realiza una solicitud al servidor a través de la conexión 1. El documento solicitado contiene un <META> tag [RFC 1866] que le dice al visualizador que lo reemplace por una nueva versión después de x segundos. Por lo tanto el navegador realiza la segunda conexión transcurridos los x segundos.

Un inconveniente con este tag es que no es un tag estándar de HTML, y por lo tanto no esta soportado por todos los visualizadores.

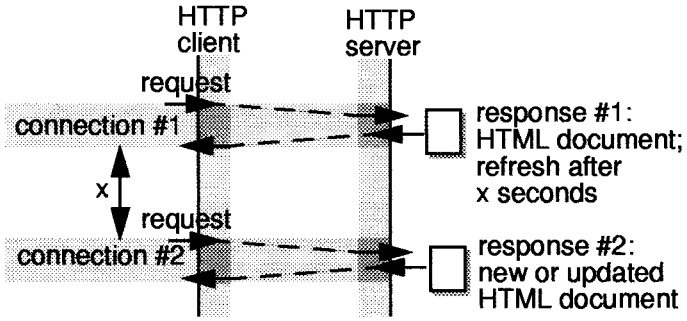

fig.4.2: Client Pull

4.3 VISUALIZADOR COMO INTERFAZ DE USUARIO DISTRIBUIDA

Los visualizadores presentan documentos HTML incluyendo varios formatos de texto y gráficos, esta característica puede ser considerada como una potente interfaz distribuida de usuario para aplicaciones de gestión que residan sobre el servidor. De este modo, HTML permite la visualización de información de gestión a los usuarios en forma de gráficos y estadísticas.

4.4 INDEPENDENCIA DE LA PLATAFORMA

La característica más importante es que HTML es un lenguaje independiente de la plataforma. Un documento puede ser presentado por

visualizadores, que están disponibles para muchas plataformas, virtualmente de la misma forma.

4.5 RECURSOS EJECUTABLES

El Web proporciona la capacidad de ampliar sus recursos a través de scripts CGI [RFC 1867, 2057] y también de bajar código Java y ejecutarlo localmente del lado del cliente. En la siguiente figura se describe el mecanismo de comunicación del servidor con el script CGI.

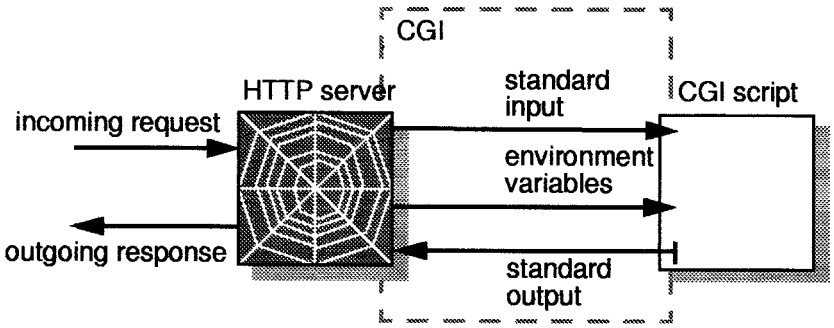

fig.4.3: Protocolo CGI

El servidor recibe una solicitud del cliente para ejecutar un recurso, esta solicitud es pasada al recurso ejecutable a través de la entrada estándar. Además, información adicional puede ser transferida a través de variables ambientales (tal como la dirección IP del cliente). El recurso pasa los resultados a través de la salida estándar y finalmente el servidor HTTP envía la respuesta al cliente.

La capacidad para crear documentos HTML dinámicamente es crucial si se utiliza HTML para la representación de información de administración por un visualizador, ya que esta información tiene una naturaleza dinámica.

La creación de documentos HTML dinámicos son descriptos en la siguiente figura:

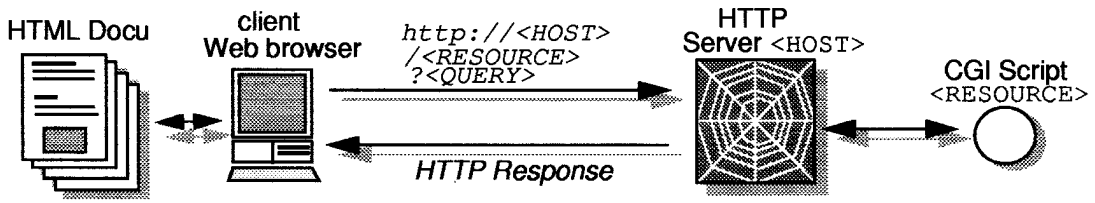

fig.4.4: Documentos dinámicos HTML

4.6 SERVER PUSH

En el punto 4.2, “client-pull[2]”, se vio como un script CGI puede retornar diferentes sub-documentos a intervalos regulares de tiempo. En este caso es necesaria una solicitud del cliente en cada tiempo. Si por ejemplo el servidor

[2] http://www1.netscape.com/assist/net_sites/pushpull.html

requiere enviar más información al cliente por un tiempo indefinido, se hace necesaria otra técnica.

Server-push es una técnica muy utilizada cuando se requiere notificar a un cliente de la ocurrencia de ciertos eventos. Permitiría por ejemplo capturar mensajes generados por un agente (SNMP traps).

Esta técnica es presentada en el siguiente gráfico de tiempo donde se puede apreciar claramente que la información de administración puede ser transferida automáticamente sin intervención del cliente, en oposición a “client-pull”.

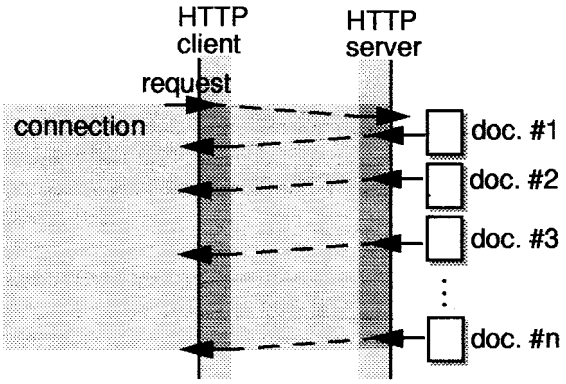

fig.4.5: Server push

4.7 JAVA

Java es un recurso ejecutable del lado del cliente soportado por la mayoría de los visualizadores. En la figura 4.6 se puede ver el principio de Java. Java es un lenguaje compilado generando un código objeto independiente de la plataforma llamado bytecode. Cuando un applet java[3] es solicitado desde un servidor, este es transferido a la máquina cliente, donde se realizan algunas verificaciones antes de ser ejecutado en el ambiente del visualizador. Este bytecode es ejecutado por un interprete de java.

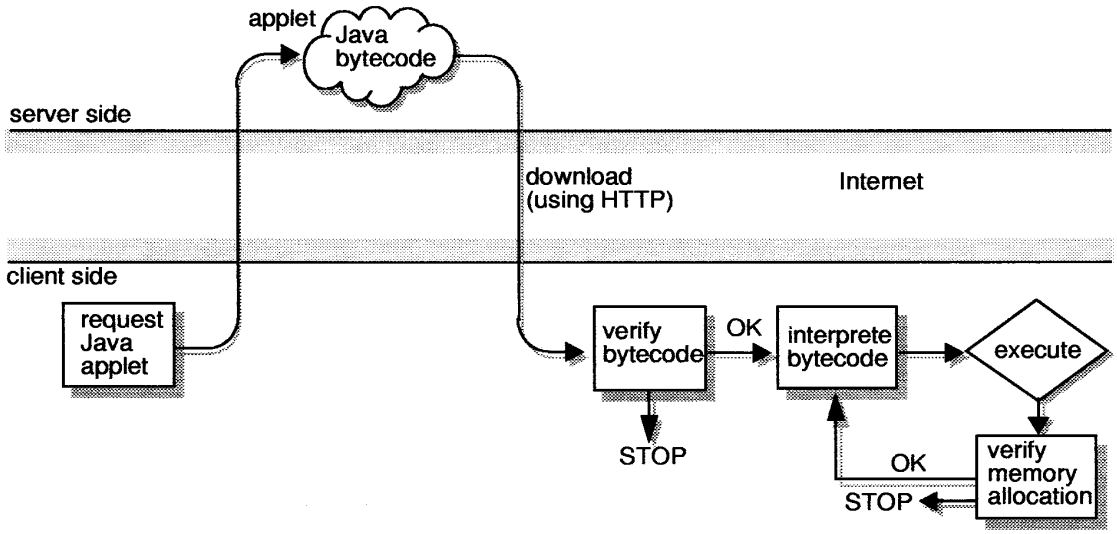

fig.4.6: Recuperación y ejecución de código Java

[3] http://www.javasoft.com/applets/index.html

Java proporciona muchas posibilidades para representaciones gráficas, lo que es vital para la creación de una interfaz amigable de usuario que dé soporte a aplicaciones sobre el Web.

También brinda la posibilidad de comunicación con el servidor a través del uso de sockets.

Java también proporciona una API que implementa un método para invocación remota de java llamado RMI[4] API. Esta API forma parte del Java Management API de Sun Microsystems. La idea detrás de RMI API es tener un mecanismo parecido a RPC (Remote Procedure Call) para java, que posibilite a los desarrolladores de aplicaciones distribuidas utilizar procedimientos que permitan el intercambio de parámetros entre entidades residentes en diferentes espacios de direcciones, todo esto soportado a bajo nivel por un mecanismo de sockets.

Este mecanismo es descripto en la figura 4.7.

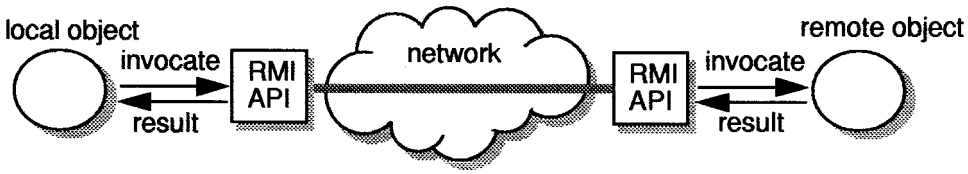

fig.4.7: Método de llamada remota

[4] Especificación del Protocolo: Java Remote Method Invocation Specification, http://www.javasoft.com/products/jdk/1.1/docs/guide/rmi/spec/rmiTOC.doc.html

5 GESTIÓN BASADA EN WEB

Existen diferentes versiones, que no son mutuamente excluyentes, acerca de lo que es Gestión Basada en Web (GbW). En esta sección se presenta una definición, a la que se hará referencia en todo el informe. Después de definir lo que es Gestión basada en Web, en la primera parte se explicará cómo la información de gestión puede ser transportada sobre el protocolo HTTP. En la segunda parte se presenta una clasificación de dos tipos diferentes de Gestión Basada en Web. En la tercera sección se realizará la comparación de la Gestión basada en Web con la Gestión tradicional de administración SNMP.

El término "Gestión Basada en Web" [Kasteleijn'97][5] puede definirse como:

Gestión Basada en Web es la aplicación de la tecnología WWW (World Wide Web) para propósitos de gestión. Esto significa utilizar HTTP como protocolo de transferencia, entre los servidores y clientes HTTP, para brindar información de gestión en formato HTML, texto plano, o alguna forma de codificación y realizar operaciones de administración sobre dispositivos y redes.

El prerequisito de utilizar HTTP como protocolo de transferencia, se fundamenta en que HTTP es el protocolo primario de transferencia en el Web.

Otro prerequisito es que la información de gestión sea transportada entre servidores HTTP y clientes HTTP.

Un cliente HTTP es cualquier aplicación que hace uso de los servicios provistos por un servidor HTTP. Esto puede ser un navegador o cualquier aplicación específica que utilice el protocolo HTTP.

Es importante enfatizar que las herramientas de gestión Java/RMI, no están dentro de la definición presentada aquí, ya que en éstas HTTP es únicamente utilizado para transferir código java desde el servidor al cliente y para el resto de la comunicación se utiliza RMI. Normalmente se hace referencia a esta arquitectura como "Java Based Management[6]".

En la figura 5.1 se presenta un esquema de los conceptos de Gestión basada en Web. En esta figura no se especifica cómo se realizan las operaciones de gestión ni tampoco cómo es intercambiada la información entre dispositivos de red y el servidor HTTP. En la figuras 5.2 y 5.3 se puede ver ampliada la región marcada con puntos.

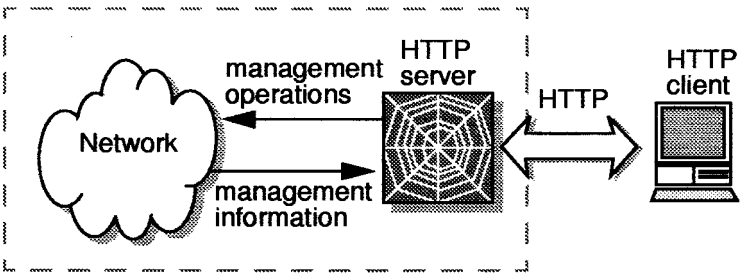

fig.5.1: Gestión Basada en Web: Servidor y Cliente HTTP intercambiando información de gestión

[5] Del trabajo de Wilco Kasteleijn "Web based Management"
[6] http://java.sun.com/JavaManagement/document/architecture/html/jmapi-arch.html

5.1 TRANSPORTANDO INFORMACIÓN DE GESTIÓN CON HTTP

La información transportada sobre HTTP desde el servidor al cliente debería realizarse en un formato legible por un usuario (como texto o HTML en el caso de que el cliente sea un navegador), o algún formato interpretado por una computadora. Este formato podrá ser cualquiera que sea entendido por alguna aplicación sin necesidad de código adicional.

No es conveniente transferir la PDU's completas del protocolo de gestión sobre HTTP, ya que esto implicaría una sobrecarga (overhead) innecesaria en el protocolo.

HTTP y los protocolos de gestión son implementados en la parte superior del modelo TCP/IP. Ambos requieren de la misma funcionalidad que hace posible este mapeo sobre el protocolo de transporte. Para mostrar esto, se presenta una tabla que resume las funciones básicas de un protocolo de gestión como SNMP, contrastado con las funciones ofrecidas por el protocolo HTTP:

Funciones básicas del protocolo SNMP	**Funciones básicas del protocolo HTTP**
Identificación de objetos e instancias administrables;	Identificación de recursos a través de URLs;
Realiza operaciones get, set y trap sobre los objetos administrables;	Realiza operaciones get, head y post sobre los recursos;
Autorización (community string)	Autorización (basic authorization)
Encoding (BER)	Encoding (MIME)

Es esta tabla se puede ver que ambos protocolos SNMP y HTTP proporcionan funcionalidad de autorización y codificación. Las community string usadas en SNMP se hacen innecesarias si las PDU's SNMP son transportadas sobre HTTP. Lo mismo para las Basic Ecoding Rules (BER)[7] usadas por SNMP.

En el caso de HTTP, se provee codificación a través del transporte de documentos MIME. El resto son funciones que identifican los objetos manejados y las instancias de esos objetos, además de realizar operaciones de gestión sobre ellos.

Los protocolos de gestión como SNMP o CMIP utilizan ASN.1 para describir los objetos e instancias manejados (texto ASCII), de la misma manera ASN.1 también podría ser utilizado para intercambiar información de gestión sobre HTTP (formato de texto plano, no HTML).

Finalmente tendrían que definirse las operaciones de gestión mediante strings, y proporcionar la forma para identificar objetos e instancias a través de esos strings.

[7] ISO/IEC 8825:1990

Esta representación basada en strings debería ser capaz de describir cualquier operación realizada por un protocolo de gestión, permitiendo construir clientes capaces de manejar agentes que utilicen diferentes protocolos.

5.2 SISTEMAS DE GESTIÓN BASADA EN WEB

Un Sistema de Gestión basada en Web[8], significa que un servidor HTTP se está ejecutando sobre un sistema o dispositivo administrable (elementos de red). Esto implica que los objetos gestionados por el sistema pueden ser administrados directamente a través de HTTP. El servidor HTTP es extendido con una funcionalidad extra capaz de permitir la comunicación con algún sistema o dispositivo específico (herramienta de administración) que controla los objetos gestionados. Esta idea es presentada en la figura 5.2. Allí se puede ver claramente que, en este caso, HTTP puede ser considerado como un protocolo de gestión.

Si la información exportada por el servidor está en un formato legible por el usuario (HTML), el conjunto de las herramientas de administración se ejecutan dentro de un mismo sistema (excepto por la interfaz de usuario que podría ser un navegador). Este esquema puede ser considerado como una interfaz de usuario distribuida de la herramienta de administración.

Si el servidor exporta información sobre HTTP en un formato que puede ser reconocido y entendido por una herramienta de administración del lado del cliente, la funcionalidad de administración podría residir parte en el sistema (del lado del servidor) y parte del lado del cliente, dependiendo del tipo de funciones de gestión a implementar.

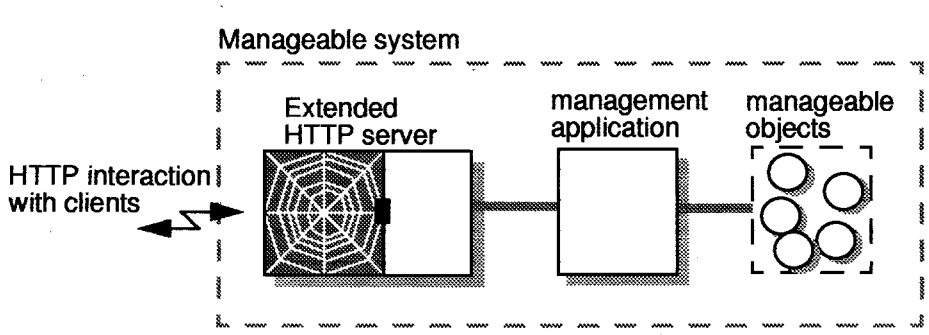

fig.5.2: Sistema de Gestión basada en Web

[8] Wilco Kasteleijn “Web based Management”

5.3 GESTIÓN BASADA EN WEB A TRAVÉS DE PROXY

Gestión basada en Web a través de Proxy[9], implica la existencia de un intermediario entre uno o más protocolos de gestión y HTTP. Este proxy es un servidor HTTP con funcionalidad extendida, capaz de convertir solicitudes HTTP en solicitudes del protocolo de gestión (request PDU's), y respuestas del protocolo de gestión (response PDU's) en respuestas HTTP. Con este esquema no hay necesidad de tener un servidor HTTP ejecutándose sobre los dispositivos o sistemas gestionados. En su lugar, la administración se realiza utilizando protocolos de gestión convencionales. Una aplicación agente (un agente SNMP) se ejecuta sobre esos dispositivos y sistemas gestionados. Este mecanismo es descripto en la figura 5.3.

Cuando la información exportada por el servidor se encuentra en un formato legible por el usuario (como HTML), todas las funciones de administración son realizadas por el proxy, el cual podrá ser considerado como una estación de gestión que administra elementos de red (agentes). La funcionalidad de este esquema implica simplemente una interfaz de usuario distribuida de la herramienta de administración.

Si el proxy exporta información de gestión sobre HTTP en un formato que puede ser reconocido y comprendido por alguna herramienta de administración del lado del cliente, las funciones de administración podrían residir una parte sobre el proxy del lado del servidor y otra parte del lado del cliente dependiendo del tipo de función a realizar.

Es importante destacar que cada uno de los mecanismos descriptos aquí no excluye al otro, ya que es posible una arquitectura híbrida entre Sistemas de Gestión basada en Web y Gestión basada en Web a través de Proxy.

Las características de este servidor proxy son las mismas que encontradas actualmente en cualquier web proxy server. Su implementación ofrece más posibilidades de seguridad, filtrado de información y control de acceso a los recursos.

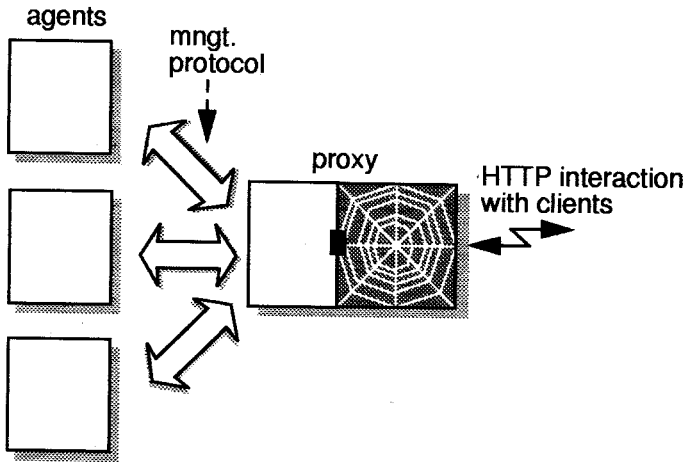

fig.5.3: Gestión basada en Web a través de Proxy

[9] Wilco Kasteleijn "Web based Management"

5.4 COMPARANDO SISTEMAS DE GESTIÓN CONVENCIONALES CON GbW

Para una mejor comprensión del valor de los sistemas de GbW, es necesario realizar la comparación entre la Gestión basada en Web y la forma convencional de administrar dispositivos y redes. Del análisis mencionado surgirán las ventajas y limitaciones de esta tecnología.

En la comparación se pone especial énfasis en las siguientes características:

- eficiencia
- escalabilidad
- niveles de abstracción de funcionalidad
- expansión
- costo
- interfaz amigable
- seguridad

5.4.1 Eficiencia

En esta sección se discuten los problemas de eficiencia de la versión 1 de SNMP, y se explica cómo las herramientas de GbW pueden mejorarla en ciertos casos. La idea no es reemplazar SNMPv1 por otro protocolo, sino combinarlo con la tecnología Web para hacer más eficaz el uso de agentes SNMPv1.

Limitación en el tiempo de respuesta de SNMPv1

SNMP fue desarrollado para proporcionar una implementación básica de un protocolo de administración en ambientes basados en TCP/IP. Muchas de las redes y dispositivos en Internet todavía son administrados con SNMPv1. Esta versión de SNMP tiene algunas limitaciones.

Como se describió en la primera parte de este informe, una de las mayores deficiencias de SNMPv1 es la recuperación de grandes volúmenes de datos, como lo es, por ejemplo, una tabla de rutas (routing table) completa. La razón es que SNMPv1 proporciona el getnext-request para recuperar el próximo elemento en orden lexicográfico. Esta es la única manera de recuperar una tabla MIB entera desde un agente, y consecuentemente se requiere una solicitud separada para cada entrada en la tabla.

Si bien SNMPv2 resuelve este problema con el uso de getbulk-request, existe el inconveniente de la escasa difusión en las redes de estos agentes.

El hecho de hacer muchas solicitudes SNMP, no es un problema en si el tiempo de respuesta es bajo (ejemplo 0.02 seg), pero si el tiempo de request-response es relativamente alto (por ejemplo 1 seg), es posible tener un retardo significativo para recuperar grandes tablas MIB. En la figura 5.4 se muestra un

diagrama de tiempo para representar el intervalo de intercambio de PDU's entre la estación de administración y el agente. En el ejemplo, los valores de 3 objetos que están en orden lexicográfico son requeridos desde un agente. En la figura 5.4a el tiempo de respuesta es bajo, mientras que en la figura 5.4b el retardo que tiene un paquete para viajar de un lado a otro es grande y por lo tanto puede pasar mucho tiempo antes de que la información solicitada pueda ser recuperada.

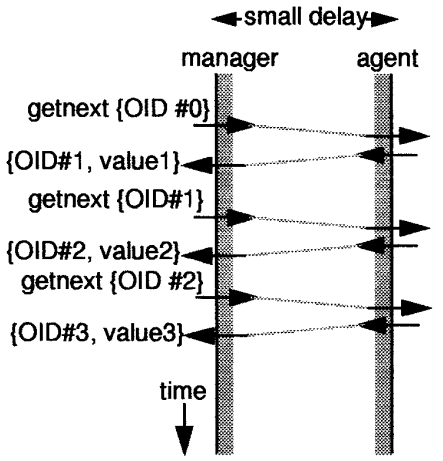

fig.5.4a: Tiempo corto entre el administrador y agente

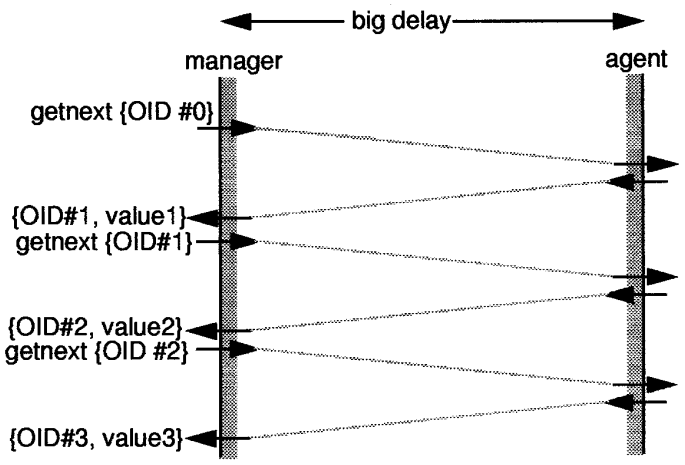

fig.5.4b: Tiempo largo entre el administrador y agente

Otro retardo significativo suele presentarse desde que el agente procesa la solicitud entrante hasta que retorna la respuesta. Considerando este retardo para una gran cantidad de solicitudes, también puede transcurrir un tiempo apreciable hasta recuperar una tabla MIB completa.

Resolviendo el tiempo de request-response de SNMPv1 con HTTP

Considerando el problema mencionado en la sección anterior, se podría concluir que el tiempo de request-response entre la estación de administración y el agente que gestiona debería ser bajo. Pero ¿qué ocurre si un usuario administra un dispositivo que se encuentra en una red remota y donde el tiempo de respuesta es alto? Esto puede ser un problema si no se desea esperar tanto tiempo para que la información de gestión esté disponible.

Una solución obvia a este dilema es utilizar alguna interfaz distribuida para la aplicación de administración que se encuentra ejecutándose en una estación sobre la misma red LAN que el agente. El tiempo de request-response en una LAN es generalmente bajo.

En particular una interfaz de usuario distribuida puede ser implementada con X11, ya que X11 está disponible para muchas plataformas y hace uso de TCP/IP. Pero X11 es una interfaz gráfica de usuario y lo único que se hace del lado del servidor X es representar gráficamente, lo que excluye cualquier posibilidad de usar la información exportada desde una estación de administración vía el cliente X, para cualquier software que se esté ejecutando

en la maquina servidor X. Otra desventaja de X Windows es que se requiere de un sistema rápido y potente para tener un rendimiento aceptable.

El principio de interfaz distribuida con X11 es presentado en la figura 5.5.

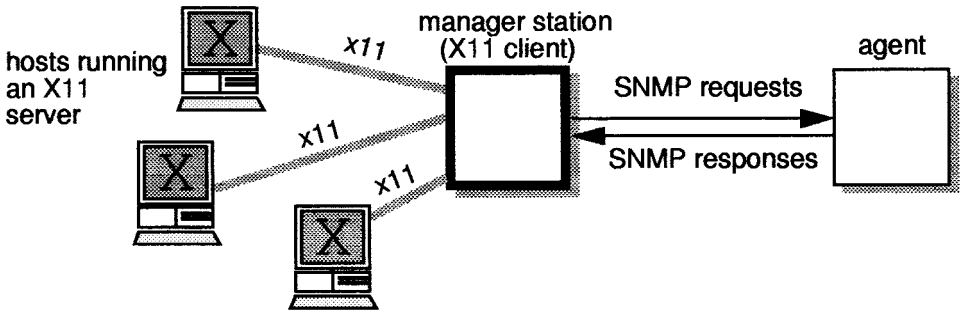

fig.5.5: Interfaz distribuida usando X11

Otro camino para implementar una interfaz distribuida de usuario es a través de Gestión basada en Web con Servidor Proxy, como se presentó anteriormente. Una aplicación de gestión se ejecuta sobre una estación ubicada en la misma red LAN que el agente que administra e intercambia la información de gestión con el cliente a través de un servidor HTTP.

Las ventajas de usar Gestión basada en Web en lugar de X11 son:

- Además de la capacidad como interfaz gráfica de usuario para los clientes, HTTP también puede transportar información que puede ser entendida por una aplicación de gestión corriendo en la máquina cliente.
- Como X11, las herramientas Web están disponibles hoy en día para muchas plataformas. Es mucho más común el uso de un navegador que el de interfaces X.
- La interfaz del navegador es una GUI[10] muy aceptada, ya que mucha gente conoce cómo trabaja.

La ineficiencia de HTTP

Un problema originado por el uso de HTTPv1.0 [RFC 1945] como protocolo de transferencia, es la ineficiencia que surge al tener conexiones TCP separadas para cada solicitud y respuesta entre el cliente y servidor. HTTPv1.1 [RFC 2616] resuelve este problema.

En caso de que HTTP sea utilizado para propósitos de administración con Gestión basada en Web, el problema es que típicamente podrían transcurrir entre 0.1 y 10 segundos antes de completarse la conexión TCP y el cliente esté en condiciones de poder enviar su solicitud al servidor.

[10] Articulo : The Difference Between Web Design and GUI Design, http://www.useit.com/alertbox/9705a.html

Por otro lado, el número de solicitudes HTTP requeridas por un cliente para recuperar la información de gestión deseada en general se mantiene razonablemente baja.

5.4.2 Escalabilidad

Esta sección discute los límites de escalabilidad que tiene la arquitectura de gestión SNMP. Primero se mencionan los problemas que pueden presentarse y posteriormente se explica cómo el uso herramientas de GbW puede evitar estos problemas de escalabilidad.

El problema de utilización de agentes SNMPv1

Los agentes SNMPv1 son generalmente incapaces de manejar muchas solicitudes simultáneamente, lo que puede ocurrir cuando el mismo agente es controlado por varias estaciones de administración. Si un agente es monitorizado con exceso, son necesarios recursos adicionales del sistema para poder manejar todas esas solicitudes simultáneas. Un agente SNMPv1 generalmente se ejecuta sobre un sistema con recursos limitados, ya que son muy costosos.

Afortunadamente esta situación no se presenta en forma frecuente.

Resolviendo el problema de utilización de agentes SNMPv1

De la sección anterior se podría concluir que los agentes SNMPv1 no deberían ser controlados por muchas estaciones de administración al mismo tiempo. Esto es un dilema si un grupo grande de gente, trabajando sobre distintos hosts, así lo requiere.

Una solución a este problema puede ser implementada con GbW a través de Proxy. El servidor HTTP está ejecutándose como un proxy entre el cliente que hace la solicitud y el agente que es monitorizado. El cliente solo puede hacer solicitudes al servidor HTTP, si hay muchos clientes éstos hacen una alta utilización del servidor y no de los agentes.

Esto significa que el servidor debe correr sobre una estación muy potente. El servidor es la única entidad que tiene control directo sobre los agentes y además puede regular el número de solicitudes enviadas a éstos.

Cuando el servidor recibe una solicitud, un script CGI se ejecuta y registra el tiempo en que ingresó el pedido. Este tiempo debe ser comparado con el de ingreso de la solicitud anterior. Luego ingresa una nueva solicitud, si el intervalo entre el nuevo pedido y el último es menor que un mínimo prefijado, el servidor envía una página HTML recuperada en la solicitud anterior (la cual se encuentra almacenada en el disco local).

Este intervalo debe ser elegido teniendo en cuenta que:

- Sea lo suficientemente pequeño para que el usuario tenga siempre información actualizada.
- Sea lo suficientemente grande para que los agentes SNMP no se sobrecarguen con solicitudes.

Un intervalo aceptable es de 1 minuto, pero debería ser configurable para cada implementación. Si este intervalo es elegido correctamente, los usuarios no lo percibirán y los agentes trabajarán sin sobrecargarse.

Este mecanismo de almacenamiento es descripto en la figura 5.6.

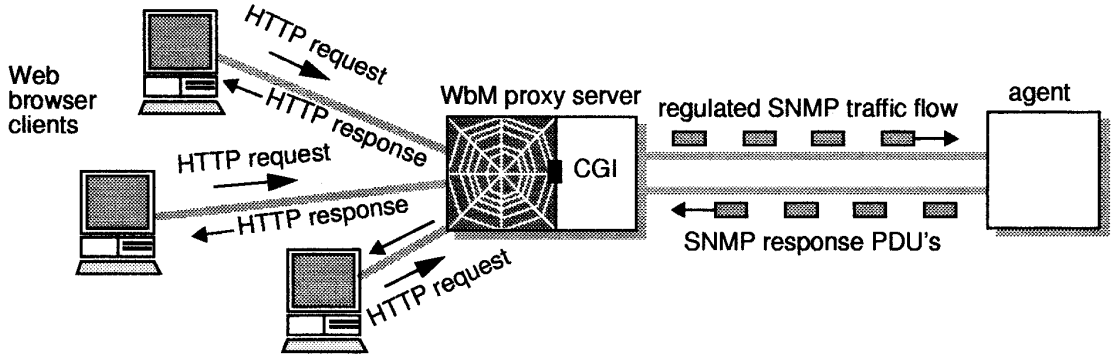

fig.5.6: Una Gestión basada en Web con Servidor Proxy permite controlar y regular el trafico hacia los agentes

El Servidor Proxy GbW puede regular el tráfico dirigido hacia los agentes, de este modo se asegura que éstos no se sobrecarguen con solicitudes.

Las herramientas de Gestión basada en Web pueden evitar o eludir de esta manera la limitación de que los agentes SNMP no puedan recibir muchas solicitudes simultáneamente. Para esto se requiere que las herramientas de GbW se ejecuten sobre estaciones potentes que sean capaces de manejar las solicitudes en exceso por parte de los clientes. Esto es mostrado en la figura 5.7. Esta solución es mejor que tener que implementar dispositivos de control en los agentes.

De este modo, en lugar de tener que reemplazar muchas partes en la red con hardware más potente, en caso de que los agentes SNMPv1 sean monitorizados en exceso, proxy GbW requiere reemplazar sólo un sistema (el servidor HTTP) para garantizar un tiempo aceptable de request-response.

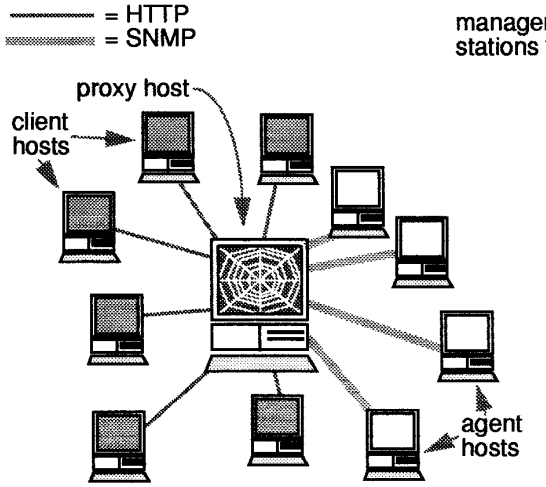

Fig.5.7a: Una solución basada en Proxy requiere sólo de una estación potente en caso de que la herramienta sea utilizada por muchos clientes.

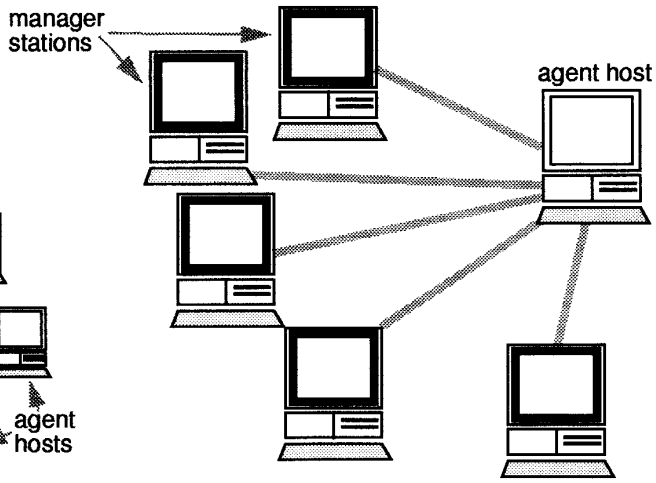

Fig.5.7b: Los hosts sobre los cuales corren agentes, deben ser potentes para que las estaciones de administración puedan hacer uso intensivo de los agentes.

5.4.3 Niveles de abstracción de funcionalidad

Es común en muchas redes LAN, tener una sola estación de administración controlando un gran número de agentes. Esta estación debe ser una máquina potente capaz de manejarlos a todos. Esto es presentado en la figura 5.8a.

Se podría pensar en un modelo gestor-agente capaz de distribuir las funciones de administración sobre un gran número de sitios, posibilitando múltiples herramientas de gestión sobre estaciones de mediana envergadura, ya que cada estación tendría menos agentes que atender. Ver figura 5.8b.

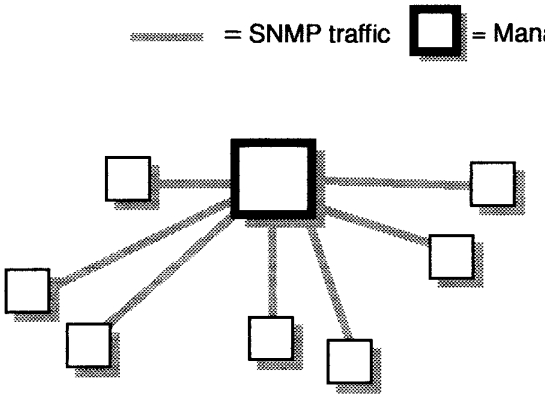

fig.5.8a: Estructura de administración SNMP: una estación central y muchos agentes.

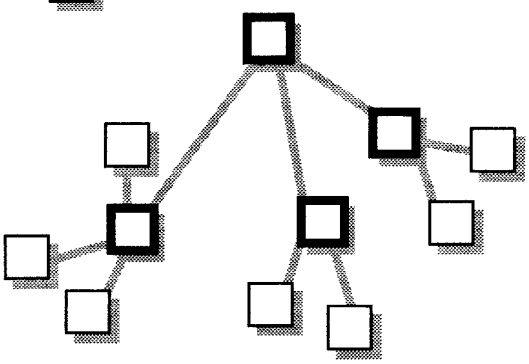

fig.5.8b: Funcionalidad de administración distribuida en varios sistemas.

Además esta solución permitiría tener múltiples capas de administración. Las herramientas de nivel superior podrían hacer uso de la información generada por la capa de nivel inferior.

La arquitectura convencional SNMP proporciona una forma distribuida de administración a través de agentes RMON [RFC 2613]. Un agente RMON permite monitorizar una sub-red entera de una manera mucho más flexible, sin tener agentes ejecutándose sobre cada dispositivo en esa sub-red. Los agentes RMON proveen capacidad para:

- Ver estadísticas (bajo nivel de utilización y estadísticas de error)
- Almacenar y ver información estadística todo el tiempo.
- Muestreo de alarmas.
- Ver el tráfico hacia y desde un host (ej. paquetes entrantes)
- Filtrar información estadística

Otro camino para implementar funcionalidad distribuida es mediante la GbW a través de servidor proxy. Las funciones de administración son distribuidas sobre los proxies y los clientes. Los proxies pueden realizar operaciones básicas SNMP y los clientes realizan las solicitudes HTTP.

El proxy puede proveer a los clientes información de gestión con un alto nivel de abstracción. Por ejemplo, puede ofrecer una tabla MIB completa haciendo una única solicitud request.

Además se puede proveer al cliente de todas las posibilidades de los mecanismos de filtrado y proporcionar información no sólo del estado de los agentes sino también estadísticas del comportamiento de los dispositivos en la red.

En la aplicación desarrollada el cliente es un visualizador (web browser) que realiza funciones de interfaz de usuario, pero el cliente podría ser una herramienta construida con propósitos específicos de administración utilizando HTTP para comunicarse con los proxies.

La información intercambiada podría basarse en un formato de string como el descripto anteriormente. Desafortunadamente todavía no hay un estándar para transmitir información de gestión sobre HTTP.

Se podría concluir que la GbW contribuye muy poco a la administración distribuida, ya que muchas de las capacidades de la GbW a través de proxy son también realizadas por RMON MIB [RFC 1757].

Algunas diferencias a considerar son:

- Las capacidades proporcionadas por RMON MIB son algo limitadas.
- HTTP es un protocolo simple. Se necesita muy poco código para que un cliente implemente HTTP. Por el contrario, SNMP debe ser implementado para cada herramienta de administración que usa el servicio de agentes RMON. SNMP requiere mas código que HTTP.
- HTTP permite el transporte de información de gestión en un formato legible por software (como el formato basado en strings) o en formato legible por el usuario (como HTML). La información de los agentes RMON puede ser solicitada únicamente con SNMP.

5.4.4 Expansión

En la sección anterior se mencionó que las herramientas tradicionales de gestión consumen muchos más recursos. La adición de nuevas funciones a estas herramientas no es una tarea fácil (el código fuente debe ser modificado). Los programadores se encuentran con el inconveniente de tener que utilizar el lenguaje original con que fue desarrollado el software. Comprender el código escrito por otra persona puede llevar mucho tiempo. El software tiene que ser completamente o en parte recompilado, lo que implica detenerlo temporalmente hasta tener la nueva versión disponible.

El agregado de nuevas funciones al software existente, debería poder hacerse de una manera más simple y rápida. Una herramienta de administración como *Tkined*[11] lo hace posible brindando un marco para extender la plataforma de gestión. Las ampliaciones son escritas en un lenguaje de scripts llamado *Tcl*[12], el cual reduce enormemente el tiempo de desarrollo y no hay necesidad de compilar el código fuente.

En general una herramienta expansible de gestión debería posibilitar lo siguiente:

- Prototipación: deberían desarrollarse versiones de nuevas funciones utilizando herramientas que permitan una rápida implementación.
- Creación de nuevas funciones en tiempo de ejecución: las herramientas existentes deberían quedar disponibles durante la creación de nuevas funciones.
- Compilación sólo de nuevas funciones
- Distribución de la funcionalidad sobre múltiples plataformas.

Las ampliaciones de las herramientas de administración distribuida son descriptas en la figura 5.9a. También pueden implementarse nuevas funciones administración distribuyendo las herramientas sobre muchos sitios (ver figura 5.9b).

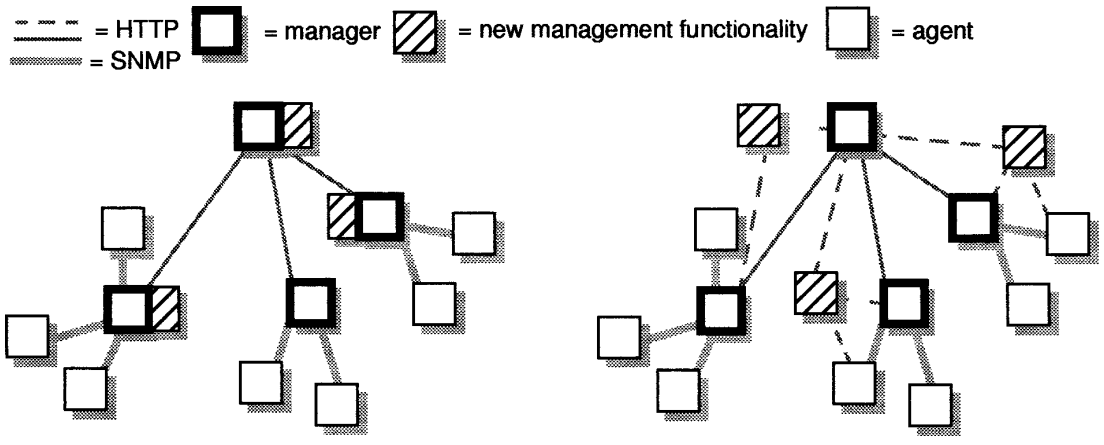

fig.5.9a: Nuevas funciones en herramientas distribuidas de gestión.

fig.5.9b: Agregar nuevas funciones, implementando herramientas de gestión en un ambiente distribuido.

[11] http://www.ibr.cs.tu-bs.de/projects/nm/scotty/tkined.html
[12] http://www.icemcfd.com/tcl/comparison.html

El primer punto habla de prototipos que sean rápidamente implementables. Ejemplo de herramientas que faciliten la prototipación son los scripts programming language[13] (que no requieren de compilación y permiten un tiempo de desarrollo reducido). Su empleo permitiría también elegir el método de programación preferido, y no depender del método o lenguaje usado en las funciones que ya existen en la herramienta. Una de las mayores ventajas de los scripts CGI es que pueden ser escritos prácticamente en cualquier lenguaje de programación, aunque para prototipar es preferible algún lenguaje como Perl o Tcl. Además los procesos que se comunican a través de CGI con el servidor también pueden ser implementados en cualquier lenguaje.

Ver figura 5.10.

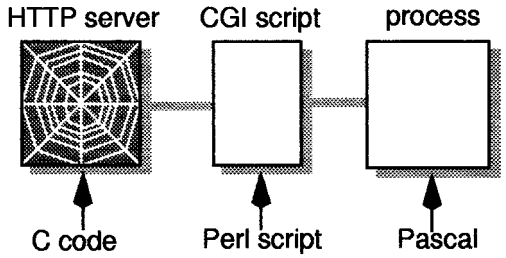

fig.5.10: Las aplicaciones que se comunican con el servidor Web pueden ser escritas en cualquier lenguaje

Tanto los procesos como los sripts CGI pueden ser creados mientras el servidor se encuentra operando. Todos los scripts CGI y aplicaciones quedan siempre disponibles mientras se agregan los nuevos.

Por último, los nuevos CGI y los nuevos procesos son compilados separadamente del sistema (esto es, servidor HTTP, scripts CGI e interfaz Web), así el tiempo de compilación y el tiempo de implementación de nuevas funciones es reducido en todo el sistema.

La desventaja de implementar Gestión basada en Web usando scripts CGI, es el costo de degradación de performance. El servidor HTTP ejecuta los scripts CGI como aplicaciones externas, lo cual es más lento que si se ejecutara como un procedimiento del mismo programa. Este es el caso de algunos scripts CGI escritos en lenguajes interpretados, donde el intérprete tiene que ser ejecutado antes de comenzar el script.

5.4.5 Costo

Ya se discutió anteriormente que muchas herramientas tradicionales de gestión requieren demasiado hardware para trabajar apropiadamente. Como el

[13] Comparación de Script Programming Languages
http://www.perl.com/pub/language/versus/index.html

hardware es costoso, en general se implementa una sola estación central para una red local entera. Los usuarios de esta estación pueden tener acceso desde la consola local a través de X11 como se explicó anteriormente.

También se discutió que Gestión basada en Web es una buena alternativa a X11 por las ventajas mencionadas.

Podemos decir que los requisitos necesarios son un servidor HTTP y algunos script CGI que deben ser escritos para servir de interfaz entre las herramientas de administración y el servidor. En lugar de una interfaz de usuario normal, las herramientas deben ser alteradas para interactuar con usuarios a través de HTML.

Finalmente, para poder acceder a estas herramientas de gestión es necesario un navegador, el cual cumple funciones de interfaz gráfica de usuario.

Una estación central con interfaz Web es descripta en la figura 5.11.

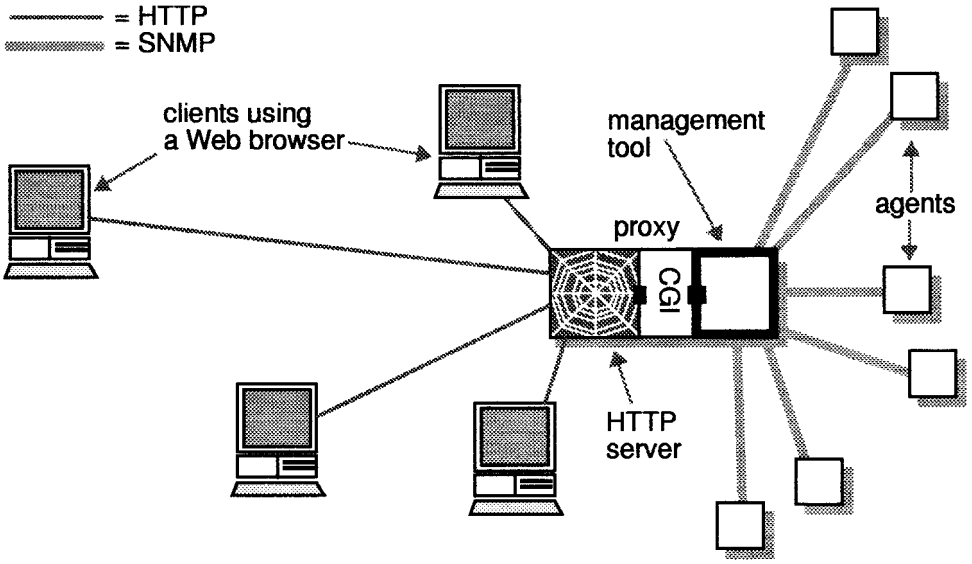

fig.5.11: Una estación central de administración que utiliza el web como interfaz de usuario

Una alternativa sería instalar servidores web en todo los dispositivos. Como se vio anteriormente, algunos vendedores equipan sus productos con servidores HTTP con funciones extendidas de gestión. Si todo los dispositivos pueden ser controlados vía HTTP, esto hace obviamente innecesario una estación central de administración sobre una máquina potente. Un administrador puede controlar los dispositivos de red a través de un navegador. Existen muchas posibilidades potenciales para proporcionar capacidades de administración a través del servidor web instalado en un dispositivo. Este tipo de solución generalmente tiene sus limitaciones. Las razones son las siguientes:

- Las herramientas de software son almacenadas en el dispositivo, y no todos los dispositivos tienen suficiente capacidad como una PC.
- Las herramientas son en general para un tipo de dispositivo de un vendedor. No todos los dispositivos pueden ser gestionados con las mismas herramientas.

- Las herramientas no pueden ser usadas para manejar varios dispositivos al mismo tiempo.

El segundo punto muestra la importancia de tener una interfaz de usuario estándar para el conjunto de herramientas de Gestión basada en Web. Si cada vendedor instala en sus dispositivos sus propias herramientas de administración, se plantea el peligro de que cada uno tenga su propia interfaz de usuario. Consecuentemente el administrador del sistema tendría que aprender a usar diferentes herramientas de diferentes proveedores. Esto es claramente una situación no deseable, y las implementaciones de Gestión basada en Web nunca serían un candidato serio para reemplazar las herramientas tradicionales.

La figura 5.12 muestra un ejemplo de Sistema de Gestión basada en Web donde un conjunto de agentes son gestionados a través de servidores HTTP y scripts CGI.

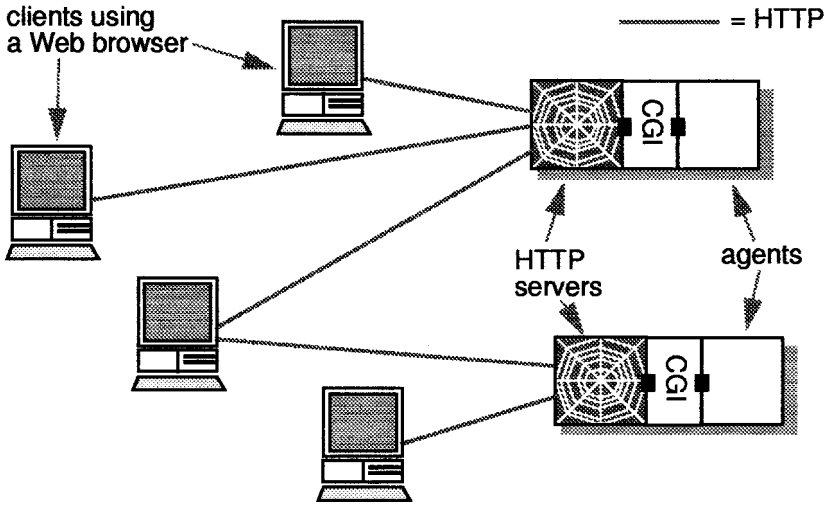

fig.5.12: Sistema de Gestión basada en Web

5.4.6 Interfaz amigable

Muchas herramientas convencionales de administración están equipadas con potentes interfaces gráficas de usuario (GUI). Un ejemplo de GUI es *Tk*. Tk[14] es una GUI para programar aplicaciones en lenguaje Tcl. Una herramienta que utiliza Tk es Tkined.

Las aplicaciones de Gestión basada en Web de ambas categorías descriptas en la sección anterior (*proxy y system*) pueden utilizar un navegador como interfaz de usuario. Las capacidades que brinda un navegador como GUI son simples pero efectivas, presentan páginas HTML, javascripts y java applets.

[14] http://dir.yahoo.com/Computers_and_Internet/Programming_Languages/Tcl_Tk/

Un navegador proporciona las siguientes características como GUI:

- hiperenlaces: una característica muy usada en interfaces gráficas como Windows para presentar páginas de ayuda.
- Botones clickeables.
- Formularios: permiten transferir una consulta simple y efectiva al servidor.
- Frames: permiten ver múltiples documentos en frames separados sobre una pantalla.
- Tablas.
- Bar charts.
- Javascripts tags: incrementa la interactividad de las páginas HTML.
- Java applets incrustados en páginas HTML.

Un navegador presenta páginas HTML, java script y java, ofreciendo así un rico conjunto de características GUI comparables con muchas GUI existentes para herramientas de administración. Pero éste no es el principal argumento para justificar el uso de esta tecnología, sino el hecho de que el mismo navegador que permite acceder a Internet permite controlar redes locales y remotas, y con las mismas características de GUI con las que uno está familiarizado.

5.4.7 Seguridad

Una de las mayores desventajas de la administración SNMPv1 convencional es que proporciona un mecanismo trivial de autenticación, por lo que SNMP es básicamente mejor para monitorizar que para controlar. SNMPv2 resuelve este problema proporcionando mayores facilidades de seguridad, pero no es muy común hoy en día en las redes. La pobre seguridad de SNMPv1 obliga a los administradores a bloquear todo el tráfico SNMP proveniente de la red externa.

De esta manera los usuarios externos no pueden tener acceso a ninguna información concerniente a la performance o disponibilidad de la red. Debería haber un camino para poder exportar información segura sin requerir el intercambio de PDU SNMP fuera del sistema.

Una solución simple a este problema es exportar información a través de HTTP. El único camino para los usuarios externos es ver cualquier información a través de un servidor HTTP extendido. El servidor recibe solicitudes y retorna los datos que obtiene indirectamente de la red. Esto implica que algunos procesos regulares de monitorización de dispositivos de red deben almacenar la información en alguna base de datos. El servidor HTTP puede ver sólo la información de estado desde esa base de datos, haciendo imposible a un cliente acceder directamente a un agente vía el servidor HTTP. Esta idea se muestra en la figura 5.13.

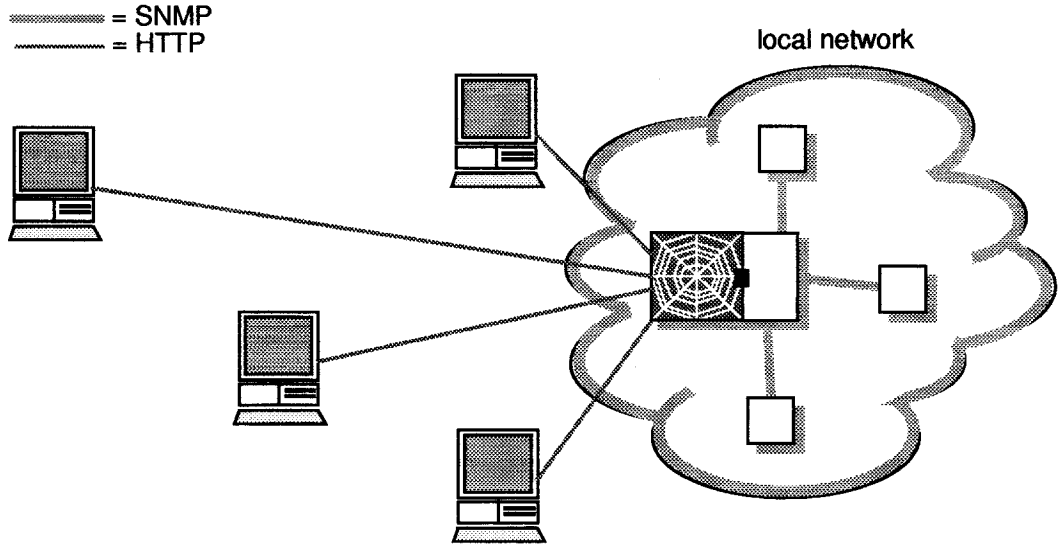

fig.5.13: Exportando información de Gestión a través de HTTP

El servidor HTTP puede ser ampliado con SSL[15] para proporcionar privacidad, integridad y autenticación. Juntos permiten ofrecer un servidor HTTP rico en mecanismos de seguridad.

Desafortunadamente, el uso de claves públicas para encriptación de mensajes HTTP desde el cliente al servidor y viceversa están limitados a 40 bits fuera de los EE.UU (estándar RSA utiliza 128 bits). Implementaciones con claves pequeñas implican menor tiempo para encontrarlas, y el uso de SSL depende de ello.

La figura 5.14 muestra estas dos situaciones: la utilización de un protocolo de administración seguro y el acceso de los clientes a la información sólo a través de un Proxy seguro.

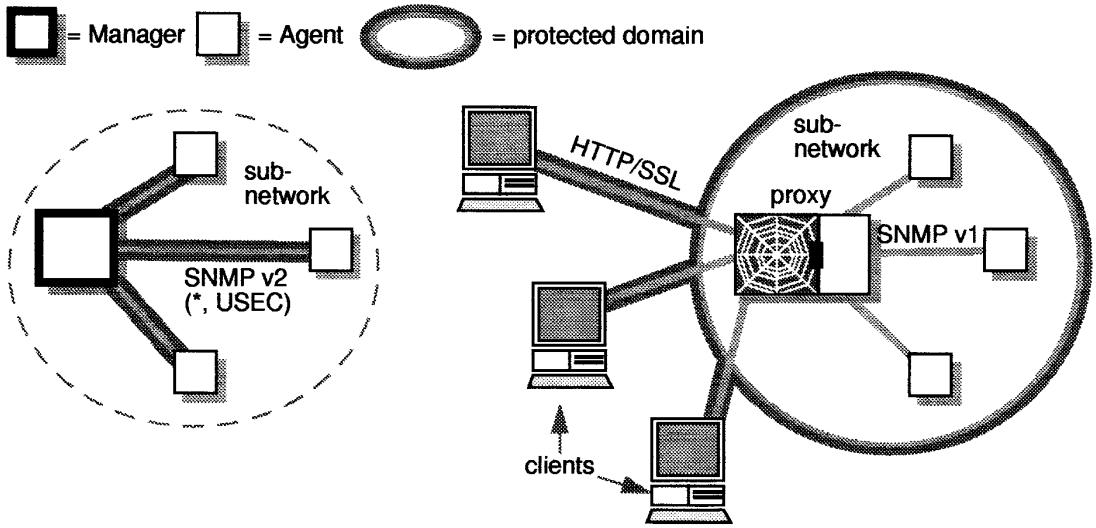

fig.5.14a: La estación de administración y los agentes se comunican a través de un protocolo seguro de gestión.

fig.5.14b: La seguridad es implementada a través de Gestión basada en Web con servidor Proxy. No es necesario un protocolo de gestión seguro.

[15] Especificación del Protocolo SSL: http://home.netscape.com/eng/ssl3/index.html

6 TECNOLOGÍA UTILIZADA

Si bien la mayoría de las herramientas y elementos principales de Gestión basada en Web ya fueron introducidos o descriptos en los capítulos anteriores, en esta sección se pretende mencionar aquellas herramientas que se utilizaron para la construcción de la aplicación:

- Protocolo de Administración SNMP
- Interface CGI - HTML
- Lenguaje Perl
- JavaScripts
- Socket
- Librerías SNMP para Perl
- PHP 3.0
- DBMS PostgreSQL
- Librerías GD 1.3
- Linux

6.1 SNMP

Los periféricos que tienen integradas las capacidades para SNMP ejecutan un paquete de software agente para administración, cargado como parte de un ciclo de arranque o incrustado en la memoria fija (firmware) del dispositivo. Estos dispositivos que tienen agentes SNMP se denominan “dispositivos administrados”.

Los dispositivos administrados por SNMP se comunican con el software servidor SNMP que está localizado en cualquier parte de la red. El dispositivo se comunica con el servidor de dos formas: por sondeo o por interrupción.

El administrador SNMP maneja el software general y las comunicaciones entre los dispositivos que utilizan el protocolo de comunicación SNMP.

Todos los dispositivos administrados por SNMP contienen el software agente SNMP y una base de datos llamada Base de Información sobre la Administración (Management Information Base, MIB).

La MIB tiene 126 áreas de información sobre el estado del dispositivo, el desempeño del dispositivo, sus conexiones hacia los diferentes dispositivos y su configuración. El administrador SNMP consulta al MIB a través del software agente y puede especificar los cambios hechos a la configuración. La mayor parte de los administradores SNMP consultan a los agentes en un intervalo regular, 15 minutos por ejemplo, a menos que el usuario indique otra cosa.

El software agente SNMP por lo general es bastante pequeño (comúnmente de 64KB) dado que el protocolo SNMP es sencillo. SNMP está diseñado para ser un protocolo de sondeo (polling), lo que quiere decir que el administrador produce mensajes para el agente. Los mensajes SNMP se colocan dentro de un datagrama UDP y se enrutan vía IP (aunque podrían utilizarse otros protocolos).

El puerto UDP 161 se utiliza para todos los mensajes, excepto para los traps, que llegan al puerto UDP 162. Los agentes reciben sus mensajes del administrador a través del puerto UDP 161 del agente.

Dado que UDP no tiene conexiones, no existe confiabilidad inherente en el envío de los mensajes. SNMP utiliza el sondeo, lo que ocupa una considerable cantidad de ancho de banda. Los intercambios entre SNMP y su sucesor, CMIP, en el futuro tomarán decisiones más difíciles concernientes al protocolo de administración.

6.2 CGI

CGI (Common Gateway Interface) es la interfaz entre un servidor con Protocolo de Transferencia de Hipertexto (HTTP) y los demás recursos del host servidor.

CGI proporciona el medio de comunicación que emplea un servidor Web para enviar información entre el visualizador (browser) y su propio programa. Es decir que a través de un CGI, el usuario puede interactuar con el servidor Web transfiriendo información entre el cliente Web y la aplicación CGI.

Los programas CGI pueden ser implementados en cualquier lenguaje. En particular para el desarrollo del presente trabajo, se utilizó PERL. La elección del lenguaje más conveniente a usar responde principalmente a dos criterios: tipo de programa a implementar y conocimiento del lenguaje a utilizar.

Para algunos tipos de programas, el uso de un lenguaje en particular puede facilitar las cosas, aún si no se conoce el mismo, en otros casos es posible que la programación en un lenguaje dado sea la más idónea pero es más fácil usar un lenguaje que se conozca.

Referencias interesantes:

http://developer.netscape.com:80/docs/manuals/enterprise/admnunix/programs.htm

http://developer.netscape.com:80/viewsource/lazar_cgi.html

http://developer.netscape.com:80/docs/manuals/enterprise/unix/contents.htm

6.3 PERL

Perl significa Practical Extraction and Report Language, algo así como lenguaje práctico de extracción y de informes. Es un lenguaje creado por Larry Wall con el objetivo principal de simplificar las tareas de administración de un sistema unix; actualmente (en su versión 5.004) se ha convertido en un lenguaje de propósito general, y una de las principales herramientas para desarrollar aplicaciones en Internet.

Es un lenguaje que hereda estructuras principalmente de los intérpretes de comandos de unix, especialmente el csh, y de otras utilidades estándar, como awk y sed. En realidad puede hacer todo lo que hace cualquiera de ellos y todos ellos juntos, la mayoría de las veces de forma más simple, comprensible y fácil de depurar.

Perl es un lenguaje interpretado, aunque internamente funciona como un compilador. Por eso se habla de scripts, y no de programas (concepto referido principalmente a programas compilados al lenguaje de máquina nativo del ordenador y sistema operativo en el que se ejecuta).

Aunque desarrollado originalmente en un entorno unix, actualmente hay versiones para casi todos los sistemas operativos: DOS, Windows 95, Windows NT, Amiga, MacOS.

Los scripts son compatibles entre las diversas plataformas, de forma que es un verdadero lenguaje multiplataforma. Muchos fabricantes lo incluyen en sus versiones de unix.

También es uno de los lenguajes más utilizados en la programación de CGI scripts, que son guiones o scripts que utilizan la interfaz CGI (Common Gateway Interface) para intercambio de información entre aplicaciones externas y servicios de información.

Finalmente se puede agregar que el mantenimiento y depuración de un programa en PERL es mucho más sencillo que la de cualquier programa en C.

Es posible encontrar mas información relacionada con Perl en las siguientes direcciones:
www.cs.calteach.edu/adam/Local/perl.html
www.spu.edu/teach/basic-perl/
www.yahoo.com/Computer/Languages/Perl

6.4 JAVASCRIPT

Javascript es una forma simplificada de Java que se utiliza para controlar un formulario o página Web. Se caracteriza por que su código se encuentra integrado dentro de la página Web. Cuando construimos una función en Javascript no es necesario compilarla y generar un fichero .class o applet como sucede con Java, sino que simplemente debemos incluir el código al principio de la página, siendo interpretado en la computadora cliente por el visualizador (browser).

Página de referencia:
http://developer.netscape.com/docs/manuals/javascript.html

6.5 SOCKETS

Los sockets no son más que puntos o mecanismos que permiten que un proceso se comunique (emita o reciba información) con otro proceso, incluso estando en distintas máquinas. Esta característica de interconectividad entre máquinas hace que el concepto de socket nos sea de gran utilidad.

La forma de hacer referencia a un socket por los procesos implicados es mediante un descriptor del mismo tipo que el utilizado para referenciar ficheros. Debido a esta característica, se podrán realizar redirecciones de los archivos de E/S estándar (descriptores 0, 1 y 2) a los sockets y así combinar entre ellos aplicaciones de la red.

La comunicación entre procesos a través de sockets se basa en la filosofía Cliente-Servidor: un proceso en esta comunicación actuará de proceso servidor creando un socket cuyo nombre conocerá el proceso cliente, el cual podrá "hablar" con el proceso servidor a través de la conexión con dicho socket.

Páginas de referencia:
http://www-bioeng.ucsd.edu/~fvetter/misc/socket_programming.html
http://www.cs.wvu.edu/~padabala/socket.html

6.6 PHP 3.0

Para la interfaz de consulta entre el servidor PostgreSQL y el Web, se utilizó PHP 3.0 (Hypertext Preprocessor). PHP es un lenguaje parecido a C que permite incrustar instrucciones SQL directamente en páginas HTML. Todo esto se realiza dentro de una página que contiene los *tags* necesarios en HTML y las consultas dentro de un tipo de *tag* especial, que es el que compila PHP. Estas páginas tienen terminación **.php3**.

PHP soporta el acceso a numerosos DBMS tales como mSQL, MySQL, PostgreSQL, Informix, Oracle, ODBC y otros.

Puede configurarse de dos formas, como una aplicación CGI o como un módulo del Servidor Web, siendo esta última mucho más eficiente y más segura.

La página principal de PHP es:
http://www.php.net.

6.7 POSTGRESQL

Para este trabajo se utilizó el motor de la base de datos PostgreSQL versión 6.2.1. PostgreSQL es un servidor de bases de datos relacionales orientado a objetos, multi-usuario y multitarea. PostgreSQL soporta un subconjunto del Estandar Ansi-SQL, específicamente no implementa sub-querys u outer-joins, pero por otro lado implementa características avanzadas como definición de clases, herencia, gestión de grandes objetos y arrays (desnormalización).

Actualmente está disponible para una gran variedad de plataformas unix y está en proyecto su desarrollo para plataformas Windows.

Para obtener más información dirigirse a la página:
http://www.postgreSQL.org.

6.8 LIBRERÍAS SNMP PARA EL LENGUAJE PERL

Esta librería está compuesta por dos módulos Perl 5, SNMP_Session.pm y Ber.pm, que proveen un acceso rudimentario a agentes remotos SNMP.

Esta librería difiere de otros paquetes SNMP en que es completamente stand-alone, y no necesita de ningún paquete adicional tal como CMU SNMP. Está escrita enteramente en Perl por Simon Leinen y no se necesita compilar ningún módulo. Utiliza el módulo Perl 5 Socket.pm, lo que lo hace altamente portable a otras plataformas que no son Unix.

Soporta las operaciones SNMP “get”, “get-next”, “set” y el envío y recepción de traps.

Para mas información sobre estas librerías dirigirse a:
www.switch.ch/misc/leinen/snmp/perl/index.html.

6.9 GD 1.3

Frecuentemente cuando se escriben aplicaciones de capturas de datos, se hace necesario mostrar resultados gráficamente. GD cuenta con una

biblioteca de funciones C llamada “libgd” que permite crear fácilmente imágenes GIF con líneas, arcos, texto y múltiples colores, directamente desde programas C, Perl, etc.

Estas librerías son necesarias para construir el preprocesador PHP3 y soportar así la posibilidad de crear imágenes dinámicas también desde páginas PHP3.

Páginas de referencia:
www.boutell.com/gd
www.genome.wi.mit.edu/pub/software/www/GD.html

6.10 LINUX

Si bien la plataforma de desarrollo e implementación es el sistema operativo Linux (Distribuciones Suse y RedHat), en todo momento se pensó que la aplicación fuera altamente portable. Se puede decir que aproximadamente el 80% del código es portable a plataformas Windows (utilizando WinPerl) sin cambios. El 20% restante que corresponde a la administración de la base de datos a través del Web, no es portable por la decisión de adoptar como motor de base de datos a PostgreSQL. Por esto al final del informe se propone como uno de los trabajos futuros, migrar el acceso de los datos a PHP sobre ODBC independizándonos de esta manera de la base de datos necesaria para ejecutar las herramientas.

Páginas principales de las distribuciones de Linux:
http://www.suse.com
http://www.redhat.com

7 DESCRIPCIÓN DE LA APLICACIÓN DESARROLLADA

Como se mencionó anteriormente, las Herramientas Web para la administración de redes son un conjunto de utilidades que permitirán al ingeniero de red realizar tareas de administración de agentes SNMP, utilizando un navegador como interfaz gráfica de usuario.

Esta información básicamente está compuesta por tablas de rutas, tablas de interfaces, tablas de direcciones y la posibilidad de un Mib Browser para monitorizar información mas específica.

Además las herramientas permiten algunas operaciones de escritura, siempre que el administrador esté autorizado para realizar un SNMP set-request.

En la figura 7.1 se puede ver la idea básica de las herramientas Web, presentando un modelo de alto nivel. Los clientes tienen acceso a las herramientas a través de solicitudes HTTP desde un navegador desde cualquier punto de la red.

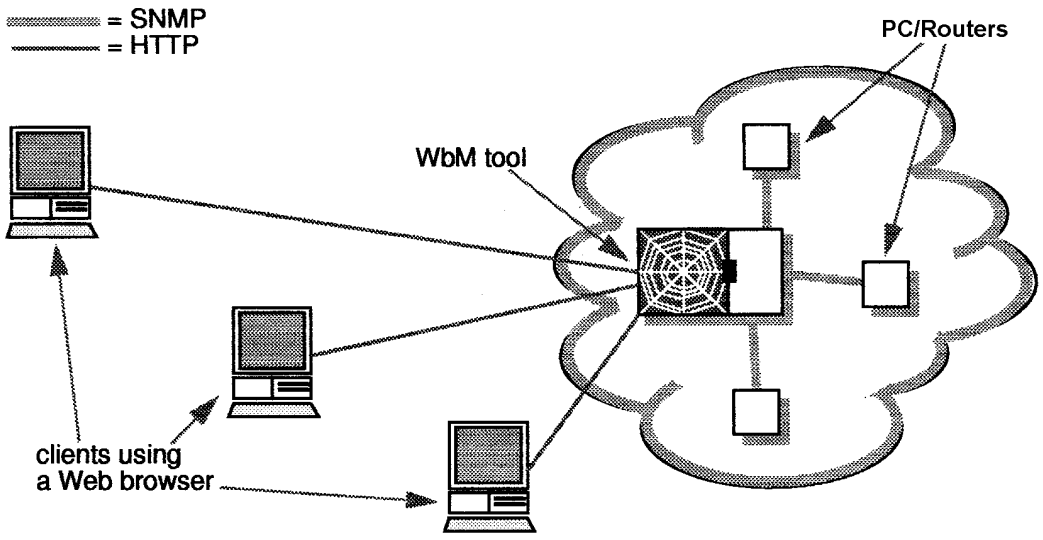

fig.7.1: Principio de las “Herramientas de Gestión basada en Web”

7.1 IMPLEMENTACIÓN

La aplicación desarrollada para la administración de redes consiste en varias herramientas que pueden ser utilizadas en conjunto o independientemente.

Para el diseño y estructura de las páginas HTML se utilizaron “páginas de estilo”, dándole una apariencia uniforme a toda la aplicación y permitiendo personalizarla especificando fondos, fuentes, colores, etc.

Las herramientas Web para la administración de redes constan de seis componentes principales:

- un servidor HTTP
- una colección de scripts CGI

- un proceso que recoge datos de los agentes registrados
- un proceso que captura los traps generados
- una base de datos
- módulo Snmp_tools.pm

La figura 7.2 describe la arquitectura de las “herramientas de gestión”.

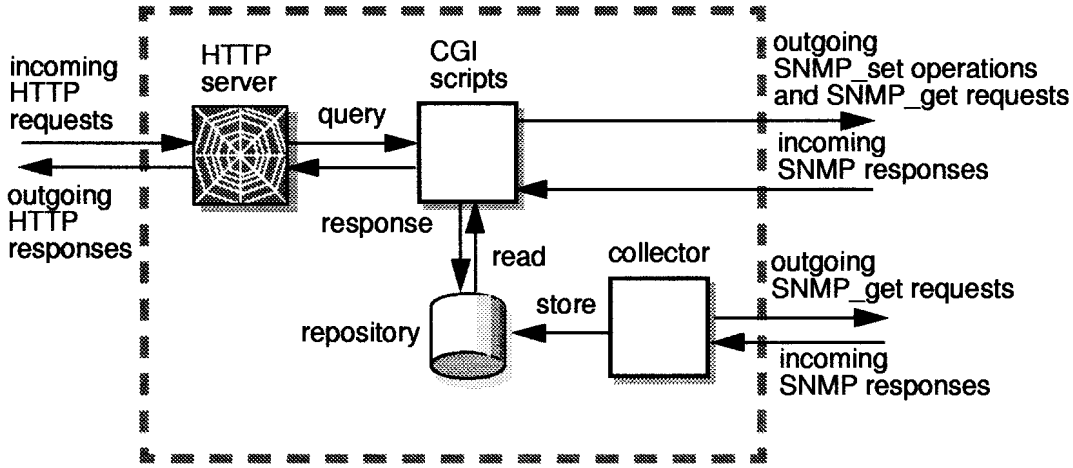

fig.7.2: Componentes de las Herramientas de Gestión basadas en Web

7.2 EL SERVIDOR HTTP

El servidor HTTP utilizado es un servidor estándar de distribución libre llamado apache v1.2.5. El servidor procesa las solicitudes de los clientes, reenvía la solicitud del cliente al script CGI y espera a que el script retorne los resultados. Luego estos resultados son enviados en una respuesta HTTP al cliente.

7.3 SCRIPTS CGI

Los scripts CGI y los procesos colectores fueron desarrollados en lenguaje Perl, utilizando una librería para la comunicación con los agentes SNMP. Esto permite una rápida implementación de la aplicación y un corto tiempo de desarrollo para adaptarla a las necesidades propias.

El empleo de Perl con las librerías SNMP resultó ser muy apropiado para implementar rápidos prototipos de gestión multiplataforma.

Las herramientas tienen una colección de varios scripts CGI. En general cada función se implementa mediante dos scripts. El primero genera el formulario HTML que permite al usuario ingresar algunos parámetros. Una vez que el formulario es llenado se envía al servidor, que a su vez reenvía la consulta al segundo script.

Este segundo script procesa la solicitud y retorna los resultados en una página HTML. Este método es descripto en la figura 7.3.

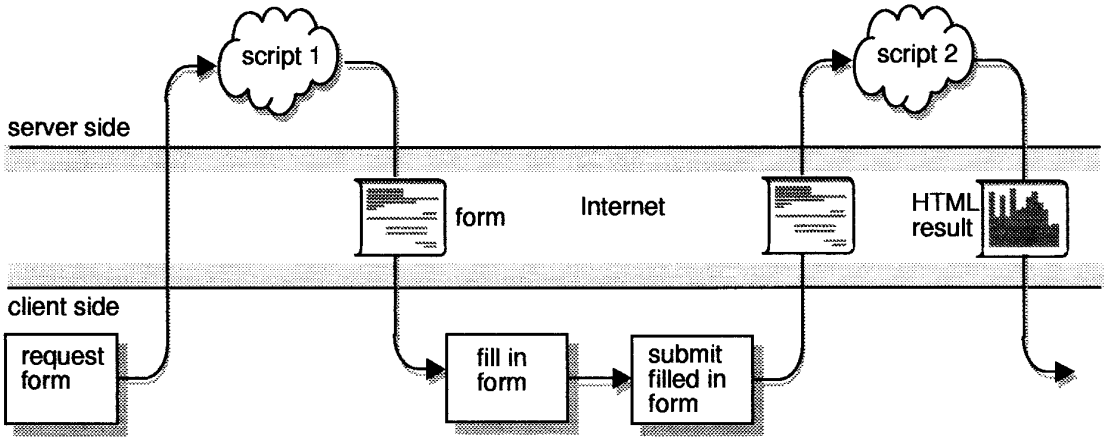

fig.7.3: Método de generación de resultados

Las utilidades están compuestas por:

- Network Monitor
- Discover Agents
- Set Request
- Send Trap
- Mib Web Browser

7.3.1 Network Monitor

El Network Monitor es una herramienta basada en una aplicación para Windows llamada “getif”. Tiene como objetivo permitir al administrador de redes monitorizar agentes SNMP desde una interfaz basada en Web. Esta herramienta posibilita la monitorización de los objetos Mib más interesantes de System (sysDescr, sysContact, sysName, sysLocalization, sysUpTime, SysObjectID, sysServices), Interfaces Table (ifIndex, ifDescr, ifType, ifMtu, ifSpeed, ifPhysAddress, ifAdminStatus, ifOperStatus), Adrresses (ipAdEntAddr, ipAdEntIndex, ipAdEntNetMask, ipAdEntBcastAddr), Routes Table (ipRouteDest, ipRouteIfIndex, ipRouteMetric1, ipRouteNextHop, ipRouteType, ipRouteProto, ipRouteAge, ipRouteMask, ipRouteInfo) y ARP Table (atIfIndex, atPhysAddress, atNetAddress), haciéndolos de esta manera accesibles a personas con poca o ninguna experiencia en la utilización del protocolo de administración SNMP.

Esta herramienta se desarrolló como un conjunto de módulos CGI independientes, implementados en lenguaje Perl con llamadas a Librería SNMP.

La interfaz de comunicación es estándar para todos los módulos, lo que permite la utilización de alguno de ellos en forma totalmente independiente invocándolo desde un navegador.

Los parámetros de invocación tienen el siguiente formato:

```
modulo.cgi?host=IP_host&community=Community&port=SNMP_Port
```

Para la funcionalidad de estos módulos se desarrollaron dos rutinas de nivel superior *snmpget* y *snmpgettable* que forman parte del módulo Perl SNMP_tools.pm. Estas rutinas utilizan las librerías SNMP, independizando así al desarrollador de detalles de implementación como manejo de buffers de entrada, salida, etc.

Descripción de Módulos:
- main : script cgi, página principal de la aplicación.
- form_mon : formulario que permite ingresar la dirección IP del agente a monitorizar, community string, SNMP port, etc.
- system.cgi : cgi perl que muestra las variables del objeto system.
- interfaces.cgi : cgi perl que muestra la tabla de interfaces del agente.
- addresses.cgi : cgi perl que muestra la tabla de direcciones (addressing table).
- routes.cgi : cgi perl que muestra la tabla de rutas del agente (routing table).
- arps.cgi : cgi que muestra la tabla de traducción de direcciones (address translation table).
- SNMP_tools.pm : módulo perl que contiene las rutinas genéricas para la aplicación (snmpget, snmpgettable, etc).
- BER.pm : módulo perl que forma parte de la librería SNMP.
- SNMP_Session.pm : módulo perl que forma parte de la librería SNMP.

7.3.2 Discover Agents

Esta herramienta permite descubrir nuevos agentes SNMP en la red, ingresando una dirección IP broadcast o multicast. Está escrita en Perl con llamadas a la librería SNMP y utilización de sockets para resolver los nombres DNS que los agentes encontrados pudieran tener.

Descripción de módulos:
- form_dis : formulario de ingreso de datos, dirección IP broadcast o multicast, community string, Port SNMP.
- discover.cgi : cgi perl que implementa la búsqueda de nuevos agentes en la red.
- SNMP_tools.pm : módulo perl que contiene las rutinas genéricas para la aplicación (snmpget, snmpgettable, etc).
- BER.pm : módulo perl que forma parte de la librería SNMP.
- SNMP_Session.pm : módulo perl que forma parte de la librería SNMP.

7.3.3 Set Request

Es una implementación de set-request basada en web. Permite alterar el contenido de determinadas variables.

Descripción de módulos:
- form_set : formulario de ingreso de datos. IP del agente, port, Object Id en formato ASN.1, tipo de dato y nuevo valor para el Objeto.
- setoid.cgi : cgi perl que implementa el set request

- SNMP_tools.pm : módulo perl que contiene las rutinas genéricas para la aplicación (snmpset, etc).
- BER.pm : módulo perl que forma parte de la librería SNMP.
- SNMP_Session.pm : módulo perl que forma parte de la librería SNMP.

7.3.4 Send Trap

Permite generar Traps de prueba desde una interfaz de usuario sencilla especificando destino, community string, Port SNMP Trap.

Descripción de módulos:
- form_trap : formulario de ingreso de datos. IP del agente, port, Object Id en formato ASN.1, tipo de dato y nuevo valor para el Objeto.
- Send_trap.cgi : cgi perl que implementa el send trap.
- SNMP_tools.pm : módulo perl que contiene las rutinas genéricas para la aplicación (snmpset, etc).
- BER.pm : módulo perl que forma parte de la librería SNMP.
- SNMP_Session.pm : módulo perl que forma parte de la librería SNMP.

7.3.5 Mib Web Browser

Esta herramienta permite recorrer la estructura del árbol MIB y monitorizar uno o varios agentes simultáneamente:

SNMP-MIB!system (1.3.6.1.2.1.1)
IF-MIB!interfaces (1.3.6.1.2.1.2)
RFC1213-MIB!at (1.3.6.1.2.1.3)
IP-MIB!ip (1.3.6.1.2.1.4)
IP-MIB!icmp (1.3.6.1.2.1.5)
TCP-MIB!tcp (1.3.6.1.2.1.6)
UDP-MIB!udp (1.3.6.1.2.1.7)
RFC1213-MIB!egp (1.3.6.1.2.1.8)
SNMP-SMI!transmission (1.3.6.1.2.1.10)
SNMP-MIB!snmp (1.3.6.1.2.1.11)

Enterprise MIBs:
CISCO-SMI!cisco (1.3.6.1.4.1.9)
TCPIPX-MIB!novell (1.3.6.1.4.1.23)
SUN-MIB!sun (1.3.6.1.4.1.42)

Esta herramienta se desarrolló a partir de un CGI, en lenguaje Perl con llamadas a Librería SNMP, que implementa el get-next-request.

La interfaz de comunicación es estándar, lo que permite su utilización en forma totalmente independiente invocándola desde un navegador.

Los parámetros de invocación tienen el siguiente formato:

```
mquery.cgi?hosts=IP_host&oids=Asn.1_object_id
```

Para la funcionalidad de esta herramienta se desarrolló una rutina de nivel superior snmpgettable2 que forma parte del módulo perl SNMP_tools.pm. Esta rutina utiliza la librería SNMP.

Descripción de módulos:
- mbrowser.cgi : cgi principal del Mib Web Browser.
- mquery.cgi : cgi perl que implementa la recuperación de los objetos (get-next-request) especificados en formato ASN.1
- setagent.cgi : cgi perl que permite especificar el o los agentes a monitorizar con el siguiente formato; <address>:<port>:<community>, si no se especifica, por defecto community es public y port es 161.
- SNMP_tools.pm : módulo perl que contiene las rutinas genéricas para la aplicación (snmpgettable2, etc).
- BER.pm : módulo perl que forma parte de la librería SNMP.
- SNMP_Session.pm : módulo perl que forma parte de la librería SNMP.

7.4 PROCESOS COLECTORES

Además del servidor HTTP, existen tres procesos corriendo continuamente. Uno de ellos monitoriza cada cierto intervalo, que puede ser configurado, los agentes que estén definidos en la base de datos. La información recuperada es también almacenada en la base de datos, permitiendo así “mantener estado de los agentes”, para posteriormente realizar gráficos y estadísticas.

El segundo proceso captura los traps producidos por los agentes, almacenando la información en la base de datos y generando un e-mail para el responsable del dispositivo que se encuentra definido en la base de datos.

Para los dispositivos incapaces de generar eventos, el proceso ping.pl permite verificar el estado a través de mensajes ICMP echo request-response, actualizando la base de datos y generando un e-mail cuando ocurre algún cambio de estado.

Componentes:
- State Monitor
- Traps Monitor
- Ping Monitor

7.4.1 State Monitor

Este módulo se ejecuta como un proceso crontab que recorre periódicamente la tabla Hosts generando un SNMP get-request por cada entrada que encuentra activa y almacenando los datos obtenidos en la tabla state. Debe ser instalado en unix como un proceso en la tabla crontab.

Está escrito en Perl con llamadas a la librería SNMP y acceso a la base de datos PostgreSQL a través de la librería estándar Pg.

7.4.2 Traps Monitor

Este módulo también se ejecuta como un proceso crontab que captura los traps generados, los almacena en la tabla traps y envía un e-mail con los detalles del suceso para los administradores registrados en la tabla users.

Está escrito en Perl con llamadas a la librería SNMP y acceso a la base de datos PostgreSQL a través de la librería estándar Pg.

7.4.3 Ping Monitor

Este módulo también se ejecuta como un proceso crontab que recorre periódicamente la tabla Ping generando un mensaje ICMP echo request por cada entrada que encuentra activa, actualizando el estado sobre la misma tabla y generando un e-mail a los administradores con detalles de la fecha y hora de producido el evento. Debe ser instalado en unix como un proceso en la tabla crontab.

Está escrito en Perl con acceso a la base de datos PostgreSQL a través de la librería estándar Pg.

7.5 LA BASE DE DATOS

La base de datos de SNMP es un conjunto de tablas que mantienen la información recogida por las Herramientas de Gestión basadas en Web.

Esta base de datos permite mantener estado de objetos Mib y está compuesta por siete tablas: Objects, Hosts, State, User, Traps, Ping y Devices.

La tabla Objects mantiene una lista de objetos mib interesantes para monitorizar que sirven de referencia para la tabla Hosts.

La tabla Hosts mantiene los agentes que serán monitorizados y sus objetos Mib de interés.

La tabla State permite mantener el estado y evolución de los objetos monitorizados almacenando los valores de las variables en cada instancia.

La tabla User almacena información de los Administradores y los agentes bajo su responsabilidad.

La tabla Traps mantiene información de los traps snmp que se generaron en los agentes administrados.

La tabla Ping permite verificar el estado (alcanzabilidad) de un host que carece de capacidad para generar eventos.

La tabla Devices permite un almacenamiento centralizado de las características principales de los dispositivos de red con propósitos de configuración.

La estructura de la base de datos es la siguiente:

Table: object

Field	Type	Extra
o_id	bpchar	Object Id
o_name	bpchar	Object Name

Table: host

Field	Type	Extra
h_id	int4	Host Id
ip	char16	Dir. IP
community	char16	Community
port	int4	SNMP Port
m_time	int4	Monitor Time
a_time	int4	Average Time
o_idi	bpchar	Object Id In
o_ido	bpchar	Object Id Out
monitor	bpchar	Monitor state

Table: state

Field	Type	Extra
h_id	int4	Host id
date	date	Date
time	time	Time
o_vali	int4	Object In Value
o_valo	int4	Object Out Value

Table: users

Field	Type	Extra
Agent_addr	char16	IP Agent
email	bpchar	E-mail del Administrador
nota	text	Algún detalle

Table: traps

Field	Type	Extra
date	date	Date
time	time	Time del trap
sender	char(16)	IP del agente originador
sender_port	int4	
community	char(16)	Community
enterprise	char(80)	
agent_addr	char16	
generic_id	int4	Generic Id
specific_id	int4	Specific Id
uptime	time	Uptime
bindings	text	Info. adicional

Table: ping

Field	Type	Extra
ip	char(15)	Dir. IP
pktcount	int	Cantidad de paquetes a enviar
pktlength	int	Longitud en bytes del paquete
monitor	char(7)	Monitorización enable/disable
data	char(50)	Info. adicional
state	char(5)	Estdo Up/Down del host

Table: devices

Field	Type	Extra

```
ip              char(15)  Dir. IP
sysname         char(20)  Nombre del sistema
sysdescr        char(200) Descripción
syslocation     char(50)  Ubicación física
syscontact      char(20)  Contacto
ifnumber        int       Cantidad de interfaces
serie           char(30)  Número de serie
provider        char(30)  Fabricante
data            char(50)  Info. adicional
```

Sobre las tablas Object, Hosts, User, Ping y Devices es posible realizar operaciones de selección, inserción, actualización y borrado.

En la tabla State se permite realizar consultas donde la cláusula where queda abierta totalmente al usuario, un tipo de consulta genérica que permite cualquier instrucción SQL sobre cualquier tabla, siempre que esté permitido.

La tabla Trap contiene la información de los traps snmp generados y solo pueden ser consultado y borrados.

Finalmente un tipo de gráfico genérico de performance y otro de utilización permite ver la evolución de los objetos monitorizados.

Toda esta interfaz de administración de la base de datos a través del Web fue desarrollada en PHP 3.0, HTML, Javascript con acceso al DBMS PostgreSQL.

7.6 MODULO SNMP_TOOLS.PM

Este módulo Perl, es utilizado por todas las herramientas web para administración de redes. Las funciones y procedimientos que contiene se describen a continuación:

- Snmpget : es un procedimiento de nivel superior que permite monitorizar varios objetos simples simultáneamente.
- Snmpgettable : es un procedimiento de nivel superior que permite recuperar tablas de objetos completas (Object Id y sus valores) especificando los objetos como definiciones perl.
- Snmpgettable2 : idem a la anterior pero especificando los objetos en formato ASN.1
- Snmpset : permite alterar el valor de un objeto especificado en formato ASN.1.
- fmi : representa velocidades en bytes/s, Kbytes/s, Mbutes/s, Gbytes/s
- bar1 : barra html genérica para la aplicación Network Monitor.
- bar2 : barra html genérica para la aplicación Mib Web Browser.
- Bar3 : barra html genérica para el resto de las utilidades.
- top : encabezado de todas la páginas html generadas por cgis.
- bottom : pie de todas las páginas html generadas por cgis.
- links : permite enlazar todos los cgis que forman parte del Network Monitor.
- readparse : procedimiento para leer los parámetros pasados a un cgi y transformarlos a un formato accesible desde el código.

- methget : complementa al procedimiento anterior, para verificar si el paso de parámetros es mediante GET o POST.
- ether_hex : función que transforma una dirección MAC a un formato estándar de presentación.
- toOID : dado un objeto en formato ASN.1 o mixto texto/ASN.1 retorna un objeto codificado.

8 CASOS DE USO

Como sabemos, la administración de redes es el proceso de controlar complejas redes de datos para maximizar su eficiencia y productividad. Para una mejor definición del ámbito de la Administración de Redes, la ISO[16] la divide en cinco áreas funcionales:

- Administración de Fallas (Fault Management): es el proceso de localizar problemas o fallas en la red.
- Administración de Configuración (Configuration Management): es el proceso de descubrir y configurar los dispositivos críticos en la red, es decir, los que controlan su comportamiento.
- Administración de Seguridad (Security Management): es el proceso de controlar el acceso a la información en la red.
- Administración de Rendimiento (Performance Management): involucra la medida del rendimiento de la red, hardware, software y medios de comunicación.
- Administración de Cuentas (Accounting Management): involucra el seguimiento de la utilización de la red por cada usuario o grupo de usuarios para asegurar los recursos suficientes.

Utilizando técnicas de Administración de fallas, el ingeniero puede localizar y resolver los problemas mucho más rápido que sin ellas.

Una herramienta de Administración de configuración puede ayudar al ingeniero a determinar, por ejemplo, que versión de software está instalada en cada dispositivo de la red.

La Administración de seguridad brinda una manera de monitorizar los puntos críticos de la red y proporcionar registros de auditoría para potenciar la seguridad.

Usando información del rendimiento de la red, el ingeniero puede asegurarse de que la red tiene la capacidad suficiente que los usuarios necesitan.

Con herramientas de administración de cuentas, el ingeniero puede conceder y remover permisos de acceso, o aprender sobre qué usuarios utilizan cuales recursos.

El propósito de este apartado es describir algunos casos de uso (desde la perspectiva de las áreas funcionales definidas por la ISO) que ayuden al ingeniero de red a interpretar la información proporcionada por las Herramientas de Gestión basada en Web.

La aplicación desarrollada se centró en brindar información correspondiente a las primeras tres áreas funcionales, Administración de Fallas, Administración de Configuración y Administración de Rendimiento, si bien es posible obtener información también aplicable a la Administración de Seguridad y Administración de Cuentas.

[16] ISO a través del Network Management Forum promueve la implementación de sistemas de administración.

8.1 ADMINISTRACIÓN DE FALLAS

Las Herramientas de gestión permiten testear la conectividad al nivel de la capa IP mediante mensajes ICMP (pings). Este tipo de herramienta es muy utilizada, particularmente si los hosts o dispositivos no disponen de capacidades sofisticadas para generar eventos.

Una vez que se produjo la falla (falta de conectividad) la herramienta alerta al administrador generando automáticamente un e-mail con la dirección IP y detalles de la fecha y hora.

Si los dispositivos en la red son lo suficientemente sofisticados como para reportar eventos, las herramientas disponen de un proceso que permite capturar traps e informar automáticamente al administrador definido en la base de datos mediante un e-mail.

8.1.1 Grupo System

Las Herramientas de Gestión basada en Web (específicamente Network Monitor) permiten monitorizar con cierto nivel de abstracción, los objetos más relevantes para la administración de fallas, como sysObjectID (que ayuda a clasificar las entidades por vendedor o a identificar al fabricante), sysServices (que nos dice a que nivel del modelo OSI opera el dispositivo) y sysUptime (que nos informa desde cuando el sistema se encuentra en funcionamiento).

Con la ayuda de la Base de Datos es posible definir una entrada en la tabla host para monitorizar el valor del sysUptime cada cierto intervalo, y determinar de esa manera en qué momento la entidad fue reinicializada.

Si al realizar una consulta sobre la tabla state se observa que el valor crece monótonamente, se entiende que el dispositivo está funcionando correctamente, si un valor es menor que el anterior, la entidad fue reinicializada desde el último muestreo. Si el valor disminuyó, se puede asumir que el agente se reinicializó o que el sysUptime alcanzó el valor máximo. Una forma de determinar cual de estos eventos ocurrió, es observar el tiempo transcurrido desde el último muestreo (por defecto 5 o 10 minutos) y el último valor conocido.

Todos estos objetos también pueden ser monitorizados desde el Mib Web Browser, permitiendo además el acceso a varias entidades simultáneamente.

8.1.2 Grupo Interfaces

A través de la aplicación Network Monitor, es posible ver el estado general de cada interfaz que resulta de los valores de los objetos ifAdminStatus, ifOperStatus como se puede observar en el siguiente cuadro:

IfAdminStatus	IfOperStatus	Estado General
1 (up)	1 (up)	Operacional
1 (up)	2 (down)	Falla

2 (down)	2 (down)	Administrativamente Off
3 (testing)	3 (testing)	Testing

Todas las otras combinaciones no son aplicables y si ocurrieran, la entidad podría estar trabajando inapropiadamente.

El objeto ifLastChange contiene el valor del sysUptime del momento en que la interfaz cambió a modo operacional.

8.1.3 Grupo IP

Si bien todos los objetos del grupo IP son útiles para la Administración de Fallas, la aplicación Network Monitor permite monitorizar los que considero más relevantes.

Es posible consultar la tabla de rutas y descubrir por ejemplo, cómo fue aprendida la información de ruteo (objetos ipRouteTable, ipRouteProto).

Otro grupo IP que puede ayudar también a resolver problemas es ipNetTo MediaTable. Estos objetos mapean direcciones de red IP a direcciones de otros protocolos. Un caso común es ARP (Address Resolution Protocol) tabla que mapea direcciones IP a direcciones MAC.

Además es posible acceder a todos los objetos de la tabla de rutas y tabla de traducción de direcciones desde el Mib Web Browser

8.1.4 Grupo SNMP

Si una entidad está recibiendo y enviando errores SNMP no necesariamente implica problemas en la red, esto podría significar que la entidad no está manejando apropiadamente los paquetes SNMP.

El número y tipo de error también podría indicar que la entidad está recibiendo paquetes SNMP con errores desde dispositivos en la red.

La solución de estos errores frecuentemente reside en la configuración de los agentes y estaciones administradoras.

Con el Mib Web Browser es posible monitorizar algunos de los siguientes objetos: snmpInASNParseErrs (Total de errores entrados ASN), snmpInNoSuchNames, snmpInBadValues, snmpOutTooBigs, snmpOutBad-Values, etc.

8.2 ADMINISTRACIÓN DE CONFIGURACIÓN

Las Herramientas de Gestión basada en Web permiten un almacenamiento centralizado de la información de los dispositivos de red, tal como nombre del dispositivo, descripción, número de serie, fabricante, ubicación física, contacto, cantidad de interfaces, etc.

A través de un mecanismo de auto descubrimiento (autodiscovery), la aplicación encuentra automáticamente las características básicas de un

dispositivo, las que posteriormente pueden ser editadas manualmente desde el Web actualizando los registros.

Las Herramientas de GbW tienen además características avanzadas, permitiendo una compleja manipulación de la información almacenada en la base de datos a través de SQL (Structured Query Language).

8.2.1 Grupo System

La aplicación Network Monitor permite observar los objetos sysDescr, sysLocation (ubicación física), sysContact (persona responsable) y sysName (nombre del sistema) que pueden ser usados para la configuración de un dispositivo o para resolver problemas. Para muchas entidades, la versión del software o sistema operativo está disponible a través del objeto sysDescr.

También es posible alterar el valor de algunos de estos objetos con la herramienta Web Set Request especificándolos en formato ASN.1.

8.2.2 Grupo Interfaces

Desde la aplicación Network Monitor es posible observar la configuración básica de las interfaces de una entidad. Por ejemplo, si el objeto ifDescr devuelve un valor “eth0”, el objeto ifType debe retornar el valor 6 especificando que se trata de una interfaz del tipo “Ethernet – CSMA/CD”. El objeto ifSpeed almacena la velocidad de la interfaz en bits por segundo, que en el caso de una ethernet retorna 10.000.000 – 10Mbps. El objeto ifMTU retorna el tamaño máximo del datagrama que puede manejar la interfaz.

Como vimos anteriormente el objeto ifAdminStatus nos dice si la interfaz se encuentra administrativamente activa. Con la herramienta Web Set Request es posible activar o desactivar remotamente una interfaz.

8.2.3 Grupo IP

Algunos dispositivos, como routers o switchs de nivel 3 están configurados para remitir datagramas IP. Con la aplicación Mib Web Browser es posible consultar el objeto ipForwarding para conocer la funcionalidad de la entidad. En algunos casos es posible que el objeto sysServices especifique funcionalidad de nivel-3 pero ipForwarding se encuentre deshabilitado por tratarse de un router Appletalk.

Otra información valiosa para propósitos de configuración es conocer a través de Network Monitor la dirección IP, máscara de sub-red y dirección de broadcast de una interfaz (objeto ipAddrTable).

El objeto ipRouteTable permite al ingeniero controlar el ruteo de paquetes IP. Si bien muchos de sus objetos están definidos como read-write permitiendo agregar nuevas rutas (ipRouteDest) o modificar las métricas (ipRouteMetric), todo depende de la implementación del agente (como ejemplo el agente de linux suse no permite alterar estos objetos).

Un administrador podría tener varias razones para controlar estos objetos, razones técnicas (rutear a través de caminos que de acuerdo a la experiencia generan menos errores) o razones políticas (rutear a través de caminos con menor costo).

8.2.4 Grupo TCP

La configuración del algoritmo de retransmisión puede afectar dramáticamente el rendimiento de las aplicaciones que utilizan el protocolo TCP. Observando con la aplicación Mib Web Browser los objetos tcpRtoAlgorithm, tcpRtoMin y tcpRtoMax se puede aprender acerca de la configuración actual del ambiente TCP. La modificación de estos objetos puede requerir reconfigurar parte del sistema operativo o en el peor de los casos cambiar la pila de protocolos TCP/IP.

El objeto tcpMaxConn (número de conexiones permitidas) puede ayudar a configurar la red para satisfacer las demandas de los usuarios junto al objeto tcpCurrEstab (número de conexiones actuales), ya que si un sistema puede manejar 10 conexiones y está sirviendo a 100 puede ser dañino para el rendimiento general.

8.3 ADMINISTRACIÓN DE RENDIMIENTO

Las Herramientas de Gestión basada en Web proporcionan una manera simple de conocer información básica de un dispositivo, sin tener que conocer detalles de la información de administración.

También es posible presentar gráficos históricos de utilización de las interfaces y porcentajes de error con el soporte de la base de datos, además de permitir consultas utilizando SQL (Structured Query Language).

8.3.1 Grupo Interfaces

La aplicación determina el número total de paquetes entrantes y salientes de una interfaz realizando la siguiente suma:

```
Total packets received = ifInUcastPkts + ifInNUcastPkts

Total packets send = ifOutUcastPkts + ifOutNUcastPkts
```

El porcentaje de errores de entrada y salida para una interfaz se puede obtener como:

```
Percent input errors = ifInErrors / total packets received

Percent output errors = ifOutErrors / total packets send
```

De la misma forma es posible calcular el porcentaje de paquetes descartados por una interfaz:

```
Percent in discards = ifInDiscards / total packets received

Percent out discards = ifOutDiscards / total packets send
```

Es importante observar estos datos, ya que los descartes y errores generalmente se producen por un mal funcionamiento de la interfaz, problemas de medios de comunicación, problemas de buffers del dispositivo, etc.

Debido a que muchos descartes se pueden producir por protocolos desconocidos, es interesante contrastar esta información con el objeto ifInUnknownProtos.

Las Herramientas de Gestión basada en Web calculan todos estos valores.

Además es posible calcular el porcentaje de utilización de una interfaz realizando muestreos a diferentes tiempos:

$$\text{Total bytes input} = \text{ifInOctects}_{\text{time y}} - \text{ifInOctects}_{\text{time x}}$$

$$\text{Total bytes output} = \text{ifOutOctects}_{\text{time y}} - \text{ifOutOctects}_{\text{time x}}$$

Luego se calcula el total de bytes por segundo:

```
Total bytes input per sec = Total bytes input / (y – x)

Total bytes output per sec = Total bytes output / (y – x)
```

Finalmente la utilización:

```
Utilization in = (total bytes input per sec * 8) / ifSpeed

Utilization out = (total bytes output per sec * 8) / ifSpeed
```

Las Herramientas de Gestión basada en Web permiten definir en la tabla host los agentes y sus objetos de interés a monitorizar a intervalos constantes. De esta manera es posible calcular el porcentaje de utilización de una interfaz.

Monitorizar el objeto ifOutQLen con el Mib Web Browser permitiría averiguar si existen paquetes en la cola de espera, y por lo tanto si un dispositivo tiene problemas para enviarlos. Una gran cantidad de paquetes esperando no implica un problema inmediatamente, pero si esto persiste por más tiempo puede indicar una congestión en la interfaz.

Otra manera de observar si existe congestión en la red es a través de los objetos ifOutDiscards y ifOutOctects. Si un dispositivo está descartando muchos paquetes salientes como indica ifOutDiscards y a su vez disminuye la cantidad de bytes salientes, como muestra ifOutOctects, la interfaz puede estar congestionada.

8.3.2 Grupo IP

Con este grupo la aplicación mide el porcentaje de tráfico IP entrante y saliente de una entidad con los siguientes cálculos:

$$\%\ \text{IP input} = \text{ipInReceives} \Big/ \sum_{\text{Cant.interfaces}} (\text{ifInUcastPkts} + \text{ifInNUcastPkts})$$

$$\%\ \text{IP output} = \text{ipOutRequest} \Big/ \sum_{\text{Cant.interfaces}} (\text{ifOutUcastPkts} + \text{ifOutNUcastPkts})$$

El porcentaje de errores de una interfaz es calculado de la siguiente manera:

$$\%\ \text{IP in errors} = (\text{ipInDiscards} + \text{ipInHdrErrors} + \text{ipInAddrErrors}) \Big/ \text{ipInReceives}$$

$$\%\ \text{IP out errors} = (\text{ipOutDiscards}) \Big/ \text{ipOutRequest}$$

El descarte de datagramas puede ocurrir por falta de recursos del sistema o por otros motivos que no permiten su procesamiento. Un gran porcentaje de paquetes que resultan en error puede ser una pista para determinar problemas de bajo rendimiento.

Algunos objetos permiten calcular errores que resultan de la fragmentación IP. Con la herramienta Mib Web Browser se pueden observar los valores de los objetos ipFragOKs (cantidad de datagramas que necesitan ser fragmentados), ipFragFail (cantidad de datagramas descartados porque la entidad necesita fragmentarlos y no puede hacerlo) e ipFragCreates (número de datagramas generados como resultado de la fragmentación).

Conocer estos datos puede ser de mucha ayuda, ya que permitiría determinar, por ejemplo, qué dispositivos están enviando y recibiendo una gran cantidad de datagramas fragmentados, o también la incidencia de la fragmentación en el rendimiento del sistema.

El objeto ipRouteDiscards nos puede decir si la entidad está descartando rutas válidas por falta de recursos y por lo tanto afectando el rendimiento de la red.

El objeto ipOutNoRoutes cuenta el número de veces que la entidad no dispone de una ruta válida para un datagrama. Si esta cantidad se incrementa, la entidad podría no ser capaz de retransmitir un paquete.

El objeto ipInUnknownProtos permite conocer si existen en la red clientes demandando un servicio sobre un protocolo que no existe. Si esto persiste por mucho tiempo pueden aparecer problemas de rendimiento.

Con la ayuda de la base de datos es posible conocer qué tan rápido está remitiendo paquetes una entidad para satisfacer los requerimientos de la red, realizando los cálculos siguientes:

$$\text{IP forwarding rate} = (\text{ipForwDatagrams}_{y} - \text{ipForwDatagrams}_{x}) / y - x$$

$$\text{IP input rate} = (\text{ipInReceives}_{y} - \text{ipInReceives}_{x}) / y - x$$

Si la entidad remite todos los paquetes recibidos, el forwarding rate tiene que ser igual al input rate.

8.3.3 Grupo ICMP

Todos lo objetos del grupo ICMP son aplicables al área de administración del rendimiento.

Una entidad generalmente debe procesar muchos paquetes ICMP entrantes. El consumo de procesador puede ser mínimo durante un período normal de tráfico, pero durante períodos ocupados este proceso puede afectar el rendimiento del sistema. Además, algunos paquetes como un Echo, requieren que un paquete Echo-response sea construido, lo cual consume más procesador.

La aplicación calcula el porcentaje de paquetes ICMP recibidos y enviados de la siguiente manera:

```
% packets ICMP input = icmpInMsgs / total packets input

% packets ICMP output = icmpOutMsgs / total packtes output
```

Que una entidad reciba o envíe muchos paquetes ICMP no necesariamente significa que tiene problemas, pero estas estadísticas podrían ayudar en el futuro.

8.3.4 Grupo TCP

Una conexión TCP puede fallar por una variedad de razones, por ejemplo, el sistema destino no existe o la red puede tener una falla. Conocer el número de rechazos al intentar una conexión puede ayudar a cuantificar la fiabilidad de la red. Una situación similar ocurre cuando una entidad envía un reset sobre una sesión establecida. Con el Mib Web Browser es posible observar los objetos tcpAttemptFail y tcpEstabResets y medir esta cantidad de rechazos.

Si la entidad está recibiendo segmentos TCP con errores, el objeto tcpInErrs se incrementa. Esto podría ser causado porque el sistema de origen encapsula incorrectamente los segmentos, pero también podría ser el resultado de muchos otros errores del sistema.

Con la ayuda de la base de datos es posible monitorizar cada cierto intervalo los objetos tcpInSegs y tcpOutSegs para chequear la cantidad de segmentos que ingresan y dejan la entidad. Estos datos pueden ser de ayuda para determinar el rendimiento de una aplicación que confía en el protocolo de transporte.

8.3.5 Grupo UDP

Monitorizar cada cierto intervalo, con la ayuda de la base de datos, los objetos udpInDatagrams y udpOutDatagrams pueden ser de mucha ayuda ya que el procesamiento de datagramas UDP afecta el rendimiento de la entidad.

El objeto udpNoPorts dice cuando una entidad está recibiendo datagramas para un protocolo desconocido o una aplicación que no está ejecutándose. Si este valor es significativo, puede ocurrir un problema de rendimiento.

Como en IP y TCP, udpInErrors almacena el número de datagramas UDP recibidos con error.

8.3.6 Grupo SNMP

Al igual que cualquier otra actividad que realice una entidad, SNMP puede afectar el rendimiento del sistema. Si se desea conocer qué porcentaje de recursos utiliza una entidad para manejar SNMP, la aplicación permite encontrar la cantidad de paquetes recibidos y enviados por la entidad a través de los objetos snmpInPkts, snmpOutPkts y el porcentaje sobre el total de paquetes manejados por la entidad.

```
% snmp packet input = snmpInPkts / total packet input

% snmp packet output = snmpOutPkts / total packet output
```

9 SITIO DE IMPLEMENTACIÓN

Las Herramientas de Gestión basadas en Web se encuentran implementadas en un servidor del L.I.D.T.I. – Laboratorio de Informática – Facultad de Ciencias Exactas – Universidad Nacional de Salta.

Puede accederse a ellas desde la siguiente dirección:

http://lidti.unsa.edu.ar/cgi-bin/msnmp/main

Es posible bajar una actualización de la aplicación desde el servidor FTP anónimo del Laboratorio:

ftp://lidti.unsa.edu.ar/hgbw

10 TRABAJO FUTURO

Analizando los resultados obtenidos, podemos decir que se cumplieron la mayoría de los objetivos propuestos. Se adquirió experiencia en el desarrollo de aplicaciones de Gestión basada en Web y se cuenta con una plataforma base de administración que puede ser fácilmente adaptada a las necesidades propias.

Actualmente las Herramientas de Gestión basada en Web se utilizan en el ámbito del Campus de la Universidad Nacional de Salta con propósitos de Administración de Fallas, Administración de Configuración y Administración de Rendimiento, según los requerimientos funcionales de Network Management definidos por la ISO[17].

La utilización de las mismas está aportando gran cantidad de información para su perfeccionamiento y la continuación del presente trabajo.

Sin embargo, todavía quedan muchas mejoras y ampliaciones sobre las cuales se puede continuar.

El trabajo futuro se centrará en la mejora de la interfaz de usuario, en mejorar la eficiencia en la recuperación de objetos y la inclusión de nuevas funciones:

<u>Interfaz de Usuario</u>

- Integrar el Mib Web Browser a la selección de objetos a monitorizar en la tabla Hosts, evitando al usuario confusiones en el manejo de los formatos ASN.1.

<u>Mejora de la eficiencia e inclusión de nuevas funciones</u>

- Permitir al usuario elegir los objetos de interés en el Network Monitor.
- Mejorar la eficiencia en la recuperación de tablas.
- Monitorización a intervalos diferentes para cada entrada en la tabla Hosts.
- Incluir gráficos estándares que representen promedios de monitoreos.
- Migrar el acceso a la base de datos a PHP sobre ODBC.
- Implementar seguridad.

[17] Allan Leinwand, Karen Fang Conroy - Network Management – A Practical Perspective – 2nd Edition

A CÓDIGO DE LA APLICACIÓN

En este anexo se presenta el código fuente de los módulos más relevantes de la aplicación desarrollada, evitando todos aquellos que sean repetitivos. Para un mayor detalle diríjase al CDROM adjunto.

A.1 MAIN

```
#!/usr/bin/perl
print "Content-Type: text/html\n\n";
##################################################################
# SNMP Web Tools: main
##################################################################
use SNMP_tools;
&top("SNMP Web Tools");
print <<ECHO;
<P>
<A HREF="/cgi-bin/msnmp/form_mon">Network Monitor</A>
This is a small utility that give information on network devices that
support SNMP protocol:
        - system information (Syscontact, SysobjectID, ...)
        - routing table
        - ARP table
        - interface table
        - addresses table
        - DNS lookup information<P>
<A HREF="/cgi-bin/msnmp/form_dis">Discover Agents</A>
Send a specific SNMP request to a IP address and wait for replies.
Can be used with a broadcast or multicast address to discover SNMP
agents ("Command Responder Applications") responding to a given
community and port.
<P>
<A HREF="/cgi-bin/msnmp/form_set">Set Request</A>
Send a specific SNMP set request with object in ASN.1 format.
<P>
<A HREF="/cgi-bin/msnmp/form_trap">Send Trap</A>
Send a specific SNMP Trap.
<P>
<A HREF="/cgi-bin/mbrowser/mbrowser.cgi">Mib Web Browser</A>
This browser is a
CGI script written in Perl.
It uses a highly portable SNMP implementation written entirely in
Perl.
 There is no need to install any external
SNMP package.
<P>
<A HREF="/SWtools/snmpdb.html">SNMP Database</A>
Database of Mib Object, Hosts, State, User administration, Traps.
ECHO
&bottom;
```

A.2 NETWORK MONITOR

```
#!/usr/bin/perl
print "Content-Type: text/html\n\n";
##################################################################
# Network Monitor: form_mon
##################################################################
```

```
use SNMP_tools;
&top("Network Monitor - SNMP Web Tools");
&bar1;
    print <<ECHO;
<FORM METHOD="POST" ACTION="/cgi-bin/msnmp/system.cgi">
<BR>
<TABLE BORDER=0>
      <CAPTION><B>Device Monitor</B></CAPTION>
<TR><TH ALIGN=RIGHT>Host: <TH ALIGN=LEFT><INPUT SIZE=30 NAME="host"
TYPE="TEXT">
<TR><TH ALIGN=RIGHT>Community: <TH ALIGN=LEFT><INPUT SIZE=30
NAME="community" TYPE="TEXT" VALUE="public">
<TR><TH ALIGN=RIGHT>SNMP Port: <TH ALIGN=LEFT><INPUT SIZE=6
NAME="port" TYPE="TEXT" VALUE="161">
<TR><TH ALIGN=RIGHT>Timeout (ms): <TH ALIGN=LEFT><INPUT NAME="timeout"
TYPE="TEXT" VALUE="2000">
<TR><TH ALIGN=RIGHT>Retries: <TH ALIGN=LEFT><SELECT NAME="retries">
      <OPTION>3
      <OPTION>2
      <OPTION>1
</SELECT>
</TABLE>
<BR><BR>
<INPUT NAME="clear" TYPE="RESET" VALUE="Clear">
<INPUT NAME="submit" TYPE="SUBMIT" VALUE="Submit">
</FORM>
ECHO
&bottom;


#!/usr/bin/perl
print "Content-Type: text/html\n\n";
######################################################################
# Network Monitor: addresses.cgi
#
######################################################################
use Socket;
use vars '$DEBUG';
my $DEBUG = 1;
use SNMP_tools;
%snmp::OIDS = (  'sysDescr' => '1.3.6.1.2.1.1.1.0',
                # ipAddrTable
                'ipAdEntAddr' => '1.3.6.1.2.1.4.20.1.1',
                'ipAdEntIfIndex' => '1.3.6.1.2.1.4.20.1.2',
                'ipAdEntNetMask' => '1.3.6.1.2.1.4.20.1.3',
                'ipAdEntBcastAddr' => '1.3.6.1.2.1.4.20.1.4',
                'ipAdEntReasmMaxSize' => '1.3.6.1.2.1.4.20.1.5',
                'CiscolocIfDescr' => '1.3.6.1.4.1.9.2.2.1.1.28',
             );

sub addresses {
  &ReadParse(*input);
  $router = $input{'host'};
  $community = $input{'community'};
  $port = $input{'port'};
  $sysName = $input{'sysName'};
  my($sysDescr)=snmpget($router,$community,$port,'sysDescr');
  if ($sysDescr eq "-1") {
      &top("Network Monitor - SNMP Web Tools");
      &bar1;
      print "<H2>$router did not respond to SNMP query !! </H2>\n";
```

```
      &pie;
      exit(0);
  }
  &top("Network Monitor - SNMP Web Tools");
  &bar1;
  print <<ECHO;
<TABLE BORDER=0>
<TR><TD ALIGN=RIGHT><B>System:</B></TD>
<TD ALIGN=LEFT>$sysName</TD>
<TR><TD ALIGN=RIGHT><B>Community:</B></TD>
<TD ALIGN=LEFT>$community</TD>
<TR><TD ALIGN=RIGHT><B>SNMP Port:</B></TD>
<TD ALIGN=LEFT>$port</TD>
<TR><TD ALIGN=RIGHT><B>Device:</B></TD>
<TD ALIGN=LEFT>$router</TD>
</TABLE>
<BR>
ECHO
  my @ipadent = snmpgettable($router,$community,$port, 'ipAdEntAddr');
  print STDERR "Ok ipAdEntAddr\n" if $DEBUG;
  my @ipadentif = snmpgettable($router,$community,$port,
'ipAdEntIfIndex');
  print STDERR "Ok ipAdEntIfIndex\n" if $DEBUG;
  my @ipadentmask = snmpgettable($router,$community,$port,
'ipAdEntNetMask');
  print STDERR "Ok ipAdEntNetMask\n" if $DEBUG;
  my @ipadentbcast = snmpgettable($router,$community,$port,
'ipAdEntBcastAddr');
  print STDERR "Ok ipAdEntBcastAddr\n" if $DEBUG;
  my @ipadentmax = snmpgettable($router,$community,$port,
'ipAdEntReasmMaxSize');
  print STDERR "Ok ipAdEntReasmMaxSize\n" if $DEBUG;
  my(%ipaddr, %ipmask, %ipbcast, %ipsize, $index);
  while (scalar @ipadentif){
    $index = shift @ipadentif;
    $ipaddr{$index} = shift @ipadent;
    $ipmask{$index} = shift @ipadentmask;
    $ipbcast{$index} = shift @ipadentbcast;
    $ipsize{$index} = shift @ipadentmax;
  }
  print <<ECHO;
<BR><BR><CENTER>
<TABLE BORDER=1>
      <CAPTION><H3>Addresses - Oid: 1.3.6.1.2.1.4.20</H3></CAPTION>
        <TR>
      <TD><B>Index</B></TD>
      <TD><B>IP</B></TD>
      <TD><B>Mask</B></TD>
      <TD><B>Broadcast</B></TD>
      <TD><B>Max Size</B></TD>
ECHO
  foreach $index ( sort { $a <=> $b } keys %ipaddr) {
    my $name="$router.$index";
    print <<ECHO;
<TR VALIGN=TOP>
<TD>$index</TD>
<TD>$ipaddr{$index}</TD>
<TD>$ipmask{$index}</TD>
<TD>$ipbcast{$index}</TD>
<TD>$ipsize{$index}</TD>
ECHO
```

```
  }
  print <<ECHO;
</TABLE></CENTER>
ECHO
&links($router,$community,$port,$sysName);
&bottom;
}
addresses;
exit(0);


#!/usr/bin/perl
print "Content-Type: text/html\n\n";
######################################################################
# Network Monitor: arps.cgi
######################################################################
use Socket;
use vars '$DEBUG';
my $DEBUG = 1;
use SNMP_tools;
use BER;
%snmp::OIDS = (     'sysDescr' => '1.3.6.1.2.1.1.1.0',
                # atTable
                'atIfIndex' => '1.3.6.1.2.1.3.1.1.1',
                'atPhysAddress' => '1.3.6.1.2.1.3.1.1.2',
                'atNetAddress' => '1.3.6.1.2.1.3.1.1.3'
              );

sub arps {
  &ReadParse(*input);
  $router = $input{'host'};
  $community = $input{'community'};
  $port = $input{'port'};
  $sysName = $input{'sysName'};

  my($sysDescr)=snmpget($router,$community,$port,'sysDescr');
  if ($sysDescr eq "-1") {
      &top("Network Monitor - SNMP Web Tools");
      &bar1;
      print "<H2>$router did not respond to SNMP query !! </H2>\n";
      &pie;
      exit(0);
  }
  &top("Network Monitor - SNMP Web Tools");
  &bar1;
  print <<ECHO;

<TABLE BORDER=0>
<TR><TD ALIGN=RIGHT><B>System:</B></TD>
<TD ALIGN=LEFT>$sysName</TD>
<TR><TD ALIGN=RIGHT><B>Community:</B></TD>
<TD ALIGN=LEFT>$community</TD>
<TR><TD ALIGN=RIGHT><B>SNMP Port:</B></TD>
<TD ALIGN=LEFT>$port</TD>
<TR><TD ALIGN=RIGHT><B>Device:</B></TD>
<TD ALIGN=LEFT>$router</TD>
</TABLE>
<BR>
ECHO
  my @atifindex = snmpgettable($router,$community,$port, 'atIfIndex');
  print STDERR "Ok atIfIndex\n" if $DEBUG;
```

```
  my @atphysaddr = snmpgettable($router,$community,$port,
'atPhysAddress');
  print STDERR "Ok atPhysAddress\n" if $DEBUG;
  my @atnetaddr = snmpgettable($router,$community,$port,
'atNetAddress');
  print STDERR "Ok atNetAddress\n" if $DEBUG;
  my(%atindex, %atphys,%atnet,$index);
  $i = 1;
  while ( scalar @atifindex){
    $index = $i;
    $atindex{$index} = shift @atifindex;
    $atphys{$index} = shift @atphysaddr;
    $atnet{$index} = shift @atnetaddr;
    $i++;
  }
  print <<ECHO;
<BR><BR><CENTER>
<TABLE BORDER=1>
      <CAPTION><H3>ARP Table - Oid: 1.3.6.1.2.1.3.1</H3></CAPTION>
        <TR>
      <TD><B>Index</B></TD>
      <TD><B>Physical Address</B></TD>
      <TD><B>Network Address</B></TD>
ECHO
  foreach $index ( sort { $a <=> $b } keys %atindex) {
    my $name="$router.$index";
    $physaddress = ether_hex($atphys{$index});
    print <<ECHO;
<TR VALIGN=TOP>
<TD>$atindex{$index}</TD>
<TD>$physaddress</TD>
<TD>$atnet{$index}</TD>
ECHO
  }
  print <<ECHO;
</TABLE></CENTER>
ECHO
&links($router,$community,$port,$sysName);
&bottom;
}
arps;
exit(0);


#!/usr/bin/perl
print "Content-Type: text/html\n\n";
######################################################################
# Network Monitor: interfaces.cgi
######################################################################
use Socket;
use vars '$DEBUG';
my $DEBUG = 1;
use SNMP_tools;
use BER;
%snmp::OIDS = (  'sysDescr' => '1.3.6.1.2.1.1.1.0',
                 'sysObjectID' => '1.3.6.1.2.1.1.2.0',
                 'sysUptime' => '1.3.6.1.2.1.1.3.0',
                 'sysContact' => '1.3.6.1.2.1.1.4.0',
                 'sysName' => '1.3.6.1.2.1.1.5.0',
                 'sysLocation' => '1.3.6.1.2.1.1.6.0',
                 'sysServices' => '1.3.6.1.2.1.1.7.0',
```

```
                'ifNumber' =>  '1.3.6.1.2.1.2.1.0',
                # Interfaces ....
                'ifIndex' => '1.3.6.1.2.1.2.2.1.1',
                'ifDescr' => '1.3.6.1.2.1.2.2.1.2',
                'ifType' => '1.3.6.1.2.1.2.2.1.3',
                'ifMtu' => '1.3.6.1.2.1.2.2.1.4',
                'ifSpeed' => '1.3.6.1.2.1.2.2.1.5',
                'ifPhysAddress' => '1.3.6.1.2.1.2.2.1.6',
                'ifAdminStatus' => '1.3.6.1.2.1.2.2.1.7',
                'ifOperStatus' => '1.3.6.1.2.1.2.2.1.8',
                # up 1, down 2, testing 3
                'ifLastChange' => '1.3.6.1.2.1.2.2.1.9',
                'ifInOctets' => '1.3.6.1.2.1.2.2.1.10',
                'ipAdEntAddr' => '1.3.6.1.2.1.4.20.1.1',
                'ipAdEntIfIndex' => '1.3.6.1.2.1.4.20.1.2',
                'CiscolocIfDescr' => '1.3.6.1.4.1.9.2.2.1.1.28',
             );

sub interfaces {
  my(%ifType_d)=('1'=>'Other(1)',
             '2'=>'regular-1822(2)',
             '3'=>'hdh1822(3)',
             '4'=>'ddnX25(4)',
             '5'=>'rfc-877x25(5)',
             '6'=>'Ethernet-Csma/Cd(6)',
             '7'=>'iso-88023-Csma/Cd(7)',
             '8'=>'iso-88024-TokenBus(8)',
             '9'=>'iso-88025-TokenRing(9)',
             '10'=>'iso-88026-Man(10)',
             '11'=>'starLan(11)',
             '12'=>'proteon-10Mbit(12)',
             '13'=>'proteon-80Mbit(13)',
             '14'=>'hyperchannel(14)',
             '15'=>'fddi(15)',
             '16'=>'lapb(16)',
             '17'=>'sdlc(17)',
             '18'=>'ds1(18)',
             '19'=>'e1(19)',
             '20'=>'basic-ISDN(20)',
             '21'=>'primary-ISDN(21)',
             '22'=>'prop-PointToPoint-Serial(22)',
             '23'=>'ppp(23)',
             '24'=>'software-Loopback(24)',
             '25'=>'eon(25)',
             '26'=>'ethernet-3Mbit(26)',
             '27'=>'nsip(27)',
             '28'=>'slip(28)',
             '29'=>'ultra(29)',
             '30'=>'ds3(30)',
             '31'=>'sip(31)',
             '32'=>'frame-relay(32)',
             '33'=>'rs232(33)',
             '34'=>'para(34)',
             '35'=>'arcnet(35)',
             '36'=>'arcnet-Plus(36)',
             '37'=>'atm(37)',
             '38'=>'miox25(38)',
             '39'=>'sonet(39)',
             '40'=>'x25ple(40)',
             '41'=>'iso-88022-llc(41)',
             '42'=>'localTalk(42)',
```

```
            '43'=>'smdsDxi(43)',
            '44'=>'frame-Relay-Service(44)',
            '45'=>'v35(45)',
            '46'=>'hssi(46)',
            '47'=>'hippi(47)',
            '48'=>'modem(48)',
            '49'=>'aal5(49)',
            '50'=>'sonet-Path(50)',
            '51'=>'sonet-VT(51)',
            '52'=>'smds-Icip(52)',
            '53'=>'prop-Virtual(53)',
            '54'=>'prop-Multiplexor(54)',
            '55'=>'100BaseVG(55)'
            );

  my(%ifAdmin_d)=('1'=>'Up(1)',
             '2'=>'Down(2)',
             '3'=>'Testing(3)',
             );

  my(%ifOper_d)=('1'=>'Up(1)',
             '2'=>'Down(2)',
             '3'=>'Testing(3)',
             );

  &ReadParse(*input);
  $router = $input{'host'};
  $community = $input{'community'};
  $port = $input{'port'};
  $sysName = $input{'sysName'};
  my($sysDescr)=snmpget($router,$community,$port,'sysDescr');
  if ($sysDescr eq "-1") {
      &top("Network Monitor - SNMP Web Tools");
      &bar1;
      print "<H2>$router did not respond to SNMP query !! </H2>\n";
      &pie;
      exit(0);
  }
  &top("Network Monitor - SNMP Web Tools");
  &bar1;
  print <<ECHO;
<TABLE BORDER=0>
<TR><TD ALIGN=RIGHT><B>System:</B></TD>
<TD ALIGN=LEFT>$sysName</TD>
<TR><TD ALIGN=RIGHT><B>Community:</B></TD>
<TD ALIGN=LEFT>$community</TD>
<TR><TD ALIGN=RIGHT><B>SNMP Port:</B></TD>
<TD ALIGN=LEFT>$port</TD>
<TR><TD ALIGN=RIGHT><B>Device:</B></TD>
<TD ALIGN=LEFT>$router</TD>
</TABLE>
<BR>
ECHO
  my @ipadent = snmpgettable($router,$community,$port, 'ipAdEntAddr');
  print STDERR "Ok Addresses\n" if $DEBUG;
  my @ipadentif = snmpgettable($router,$community,$port,
'ipAdEntIfIndex');
  print STDERR "Ok IfTable\n" if $DEBUG;
  my(%ipaddr, %iphost,$index);
  while (scalar @ipadentif){
    $index = shift @ipadentif;
```

```
    $ipaddr{$index} = shift @ipadent;
    $iphost{$index} =
      gethostbyaddr(pack('C4',split(/\./,$ipaddr{$index})), AF_INET);
    if ($iphost{$index} eq ''){
       $iphost{$index} = 'Not defined';
    }
  }
  my(@ifdescr) = snmpgettable($router,$community,$port, 'ifDescr');
  print STDERR "Ok IfDescr\n" if $DEBUG;
  my(@iftype) = snmpgettable($router,$community,$port, 'ifType');
  print STDERR "Ok IfType\n" if $DEBUG;
  my(@ifmtu) = snmpgettable($router,$community,$port, 'ifMtu');
  print STDERR "Ok IfMtu\n" if $DEBUG;
  my(@ifspeed) = snmpgettable($router,$community,$port, 'ifSpeed');
  print STDERR "Ok IfSpeed\n" if $DEBUG;
  my(@ifadminstatus) = snmpgettable($router,$community,$port,
'ifAdminStatus');
  print STDERR "Ok IfStatus\n" if $DEBUG;
  my(@ifoperstatus) = snmpgettable($router,$community,$port,
'ifOperStatus');
  print STDERR "Ok IfOperStatus\n" if $DEBUG;
  my(@ifindex) = snmpgettable($router,$community,$port, 'ifIndex');
  print STDERR "Ok IfIndex\n" if $DEBUG;
  my(@ifphysaddress) = snmpgettable($router,$community,$port,
'ifPhysAddress');
  print STDERR "Ok IfPhysAddress\n" if $DEBUG;

my(%sifdesc,%siftype,%sifspeed,%sifadminstatus,%sifoperstatus,%sciscod
escr,
%sifmtu,%sifphysaddress);
  while (scalar @ifindex) {
    $index = shift @ifindex;
    $sifdesc{$index} = shift @ifdescr;
    $siftype{$index} = shift @iftype;
    $sifmtu{$index} = shift @ifmtu;
    $sifspeed{$index} = shift @ifspeed;
    $sifphysaddress{$index} = shift @ifphysaddress;
#    if ($portmaster && $vendor) {
#      if ($siftype{$index} eq '23') {
#              if ($sifdesc{$index} eq 'ptpW1') {
#                      $sifspeed{$index} = 128000;
#              } else {
#                      $sifspeed{$index} = 76800;
#              }
#      } elsif ($siftype{$index} eq '28') {
#              $sifspeed{$index} = 38400;
#      } elsif ($siftype{$index} eq '6') {
#              $sifspeed{$index} = 10000000;
#      }
#    }
    $sifadminstatus{$index} = shift @ifadminstatus;
    $sifoperstatus{$index} = shift @ifoperstatus;
#    if ($ciscobox && $siftype{$index} != 18) {
#          $sciscodescr{$index} = "<BR>" . (shift @ciscodescr) if
@ciscodescr;
#    }
}
  print <<ECHO;
<BR><BR><CENTER>
<TABLE BORDER=1 >
      <CAPTION><H3>Interface - Oid: 1.3.6.1.2.1.2.2</H3></CAPTION>
```

```
        <TR>
      <TD><B>Index</B></TD>
      <TD><B>Admin</B></TD>
      <TD><B>Oper</B></TD>
      <TD><B>Type</B></TD>
      <TD><B>MTU</B></TD>
      <TD><B>Interface</B></TD>
      <TD><B>Bps</B></TD>
      <TD><B>DNS</B></TD>
      <TD><B>IP</B></TD>
      <TD><B>MAC</B></TD>
ECHO
  foreach $index ( sort { $a <=> $b } keys %sifdesc) {
    my $name="$router.$index";
    $physaddress = ether_hex($sifphysaddress{$index});
    print <<ECHO;
<TR>
<TD>$index</TD>
<TD>$ifAdmin_d{$sifadminstatus{$index}}</TD>
<TD>$ifOper_d{$sifoperstatus{$index}}</TD>
<TD>$ifType_d{$siftype{$index}}</TD>
<TD>$sifmtu{$index}</TD>
<TD>$sifdesc{$index}</TD>
<TD>$sifspeed{$index}</TD>
<TD>$iphost{$index}</TD>
<TD>$ipaddr{$index}</TD>
<TD>$physaddress</TD>
ECHO
  }
  print <<ECHO;
</TABLE></CENTER>
ECHO
&links($router,$community,$port,$sysName);
&bottom;
}
interfaces;
exit(0);


#!/usr/bin/perl
print "Content-Type: text/html\n\n";
######################################################################
# Network Monitor: routes.cgi
######################################################################
use Socket;
use vars '$DEBUG';
my $DEBUG = 1;
use SNMP_tools;

%snmp::OIDS = (  'sysDescr' => '1.3.6.1.2.1.1.1.0',
                # ipRouteTable
                'ipRouteDest' => '1.3.6.1.2.1.4.21.1.1',
                'ipRouteIfIndex' => '1.3.6.1.2.1.4.21.1.2',
                'ipRouteMetric1' => '1.3.6.1.2.1.4.21.1.3',
                'ipRouteMetric2' => '1.3.6.1.2.1.4.21.1.4',
                'ipRouteMetric3' => '1.3.6.1.2.1.4.21.1.5',
                'ipRouteMetric4' => '1.3.6.1.2.1.4.21.1.6',
                'ipRouteNextHop' => '1.3.6.1.2.1.4.21.1.7',
                'ipRouteType' => '1.3.6.1.2.1.4.21.1.8',
                'ipRouteProto' => '1.3.6.1.2.1.4.21.1.9',
                'ipRouteAge' => '1.3.6.1.2.1.4.21.1.10',
```

```
                'ipRouteMask' => '1.3.6.1.2.1.4.21.1.11',
                'ipRouteMetric5' => '1.3.6.1.2.1.4.21.1.12',
                'ipRouteInfo' => '1.3.6.1.2.1.4.21.1.13',
                'CiscolocIfDescr' => '1.3.6.1.4.1.9.2.2.1.1.28',
             );

  my(%ifType_d)= ('2'=>'Invalid(2)',
                  '3'=>'Direct(3)',
                  '4'=>'Indirect(4)',
                  );

sub routes {
  &ReadParse(*input);
  $router = $input{'host'};
  $community = $input{'community'};
  $port = $input{'port'};
  $sysName = $input{'sysName'};
  my($sysDescr)=snmpget($router,$community,$port,'sysDescr');
  if ($sysDescr eq "-1") {
      &top("Network Monitor - SNMP Web Tools");
      &bar1;
      print "<H2>$router did not respond to SNMP query !! </H2>\n";
      &pie;
      exit(0);
  }
  &top("Network Monitor - SNMP Web Tools");
  &bar1;
  print <<ECHO;
<TABLE BORDER=0>
<TR><TD ALIGN=RIGHT><B>System:</B></TD>
<TD ALIGN=LEFT>$sysName</TD>
<TR><TD ALIGN=RIGHT><B>Community:</B></TD>
<TD ALIGN=LEFT>$community</TD>
<TR><TD ALIGN=RIGHT><B>SNMP Port:</B></TD>
<TD ALIGN=LEFT>$port</TD>
<TR><TD ALIGN=RIGHT><B>Device:</B></TD>
<TD ALIGN=LEFT>$router</TD>
</TABLE>
<BR>
ECHO
  my @iprdest = snmpgettable($router,$community,$port, 'ipRouteDest');
  print STDERR "Ok ipRouteDest\n" if $DEBUG;
  my @iprindex = snmpgettable($router,$community,$port,
'ipRouteIfIndex');
  print STDERR "Ok ipRouteIfIndex\n" if $DEBUG;
  my @iprmetric1 = snmpgettable($router,$community,$port,
'ipRouteMetric1');
  print STDERR "Ok ipRouteMetric\n" if $DEBUG;
  my @iprnexthop = snmpgettable($router,$community,$port,
'ipRouteNextHop');
  print STDERR "Ok ipRouteNextHop\n" if $DEBUG;
  my @iprtype = snmpgettable($router,$community,$port, 'ipRouteType');
  print STDERR "Ok ipRouteType\n" if $DEBUG;
  my @iprproto = snmpgettable($router,$community,$port,
'ipRouteProto');
  print STDERR "Ok ipRouteProto\n" if $DEBUG;
  my @iprage = snmpgettable($router,$community,$port, 'ipRouteAge');
  print STDERR "Ok ipRouteAge\n" if $DEBUG;
  my @iprmask = snmpgettable($router,$community,$port, 'ipRouteMask');
  print STDERR "Ok ipRouteMask\n" if $DEBUG;
  my @iprinfo = snmpgettable($router,$community,$port, 'ipRouteInfo');
```

```
  print STDERR "Ok ipRouteInfo\n" if $DEBUG;
  my(%ipdest,
%ipmetric1,%iphop,%iptype,%ipproto,%ipage,%ipmask,%ipinfo,%ipindex,$in
dex);
$i = 1;
  while ( scalar @iprindex){
    $index = $i;
    $ipindex{$index} = shift @iprindex;
    $ipdest{$index} = shift @iprdest;
    $ipmetric1{$index} = shift @iprmetric1;
    $iphop{$index} = shift @iprnexthop;
    $iptype{$index} = shift @iprtype;
    $ipproto{$index} = shift @iprproto;
    $ipage{$index} = shift @iprage;
    $ipmask{$index} = shift @iprmask;
    $ipinfo{$index} = shift @iprinfo;
    $i++;
  }
  print <<ECHO;
<BR><BR><CENTER>
<TABLE BORDER=1>
<CAPTION><H3>Routes Table - Oid: 1.3.6.1.2.1.4.21</H3></CAPTION><TR>
      <TD><B>Index</B></TD>
      <TD><B>Destination IP</B></TD>
      <TD><B>Metric</B></TD>
      <TD><B>Next Hop</B></TD>
      <TD><B>Type</B></TD>
      <TD><B>Protocolo</B></TD>
      <TD><B>Age</B></TD>
      <TD><B>Mask</B></TD>
      <TD><B>Info</B></TD>
ECHO
  foreach $index ( sort { $a <=> $b } keys %ipindex) {
    my $name="$router.$index";
    print <<ECHO;
<TR VALIGN=TOP>
<TD>$ipindex{$index}</TD>
<TD>$ipdest{$index}</TD>
<TD>$ipmetric1{$index}</TD>
<TD>$iphop{$index}</TD>
<TD>$ifType_d{$iptype{$index}}</TD>
<TD>$ipproto{$index}</TD>
<TD>$ipage{$index}</TD>
<TD>$ipmask{$index}</TD>
<TD>$ipinfo{$index}</TD>
ECHO
  }
  print <<ECHO;</TABLE></CENTER>
ECHO
&links($router,$community,$port,$sysName);
&bottom;
}
routes;
exit(0);


#!/usr/bin/perl
print "Content-Type: text/html\n\n";
######################################################################
# Network Monitor: system.cgi
######################################################################
```

```
use Socket;
use vars '$DEBUG';
my $DEBUG = 1;
use SNMP_tools;

%snmp::OIDS = (  'sysDescr' => '1.3.6.1.2.1.1.1.0',
                 'sysObjectID' => '1.3.6.1.2.1.1.2.0',
                 'sysUptime' => '1.3.6.1.2.1.1.3.0',
                 'sysContact' => '1.3.6.1.2.1.1.4.0',
                 'sysName' => '1.3.6.1.2.1.1.5.0',
                 'sysLocation' => '1.3.6.1.2.1.1.6.0',
                 'sysServices' => '1.3.6.1.2.1.1.7.0',
                 'ifNumber' =>  '1.3.6.1.2.1.2.1.0',
              );
$| = 1;              # salida sin buffer
$router = '';
&ReadParse(*input);
$router = $input{'host'};
$community = $input{'community'};
$port = $input{'port'};
my($sysDescr,$sysContact,$sysName,$sysLocation,$sysUptime,$sysServices
,$ifNumber,
$sysObjectID) = snmpget($router,$community,$port,
'sysDescr','sysContact','sysName','sysLocation','sysUptime','sysServic
es',
'ifNumber','sysObjectID');

   $sysDescr =~ s/\r/<BR>/g;  # cambio returns por <BR>

   if ($sysDescr eq "-1") {
      &top("Network Monitor - SNMP Web Tools");
      &bar1;
      print "<H2>$router did not respond to SNMP query !! </H2>\n";
      &pie;
      exit(0);
   }
   &top("Network Monitor - SNMP Web Tools");
   &bar1;
   print <<ECHO;
<CENTER>
<TABLE BORDER=0>
<TR><TD ALIGN=RIGHT><B>Description:</B></TD>
<TD ALIGN=LEFT>$sysDescr</TD>
<TR><TD ALIGN=RIGHT><B>Contact:</B></TD>
<TD ALIGN=LEFT>$sysContact</TD>
<TR><TD ALIGN=RIGHT><B>System:</B></TD>
<TD ALIGN=LEFT>$sysName</TD>
<TR><TD ALIGN=RIGHT><B>Localization:</B></TD>
<TD ALIGN=LEFT>$sysLocation</TD>
<TR><TD ALIGN=RIGHT><B>Community:</B></TD>
<TD ALIGN=LEFT>$community</TD>
<TR><TD ALIGN=RIGHT><B>SNMP Port:</B></TD>
<TD ALIGN=LEFT>$port</TD>
<TR><TD ALIGN=RIGHT><B>Device:</B></TD>
<TD ALIGN=LEFT>$router</TD>
<TR><TD ALIGN=RIGHT><B>Ejecution time:</B></TD>
<TD ALIGN=LEFT>$sysUptime</TD>
<TR><TD ALIGN=RIGHT><B>SysObjectID:</B></TD>
<TD ALIGN=LEFT>$sysObjectID</TD>
<TR><TD ALIGN=RIGHT><B>SysServices:</B></TD>
<TD ALIGN=LEFT>$sysServices</TD>
```

```
<TR><TD ALIGN=RIGHT><B>Interfaces:</B></TD>
<TD ALIGN=LEFT>$ifNumber</TD>
</TABLE>
</CENTER>
<BR>
ECHO
&links($router,$community,$port,$sysName);
&bottom;
```

A.3 DISCOVER AGENTS

```
#!/usr/bin/perl
print "Content-Type: text/html\n\n";
######################################################################
# Discover Agents: form_dis
######################################################################
use SNMP_tools;
&top("Discover Agents - SNMP Web Tools");
&bar1;
print <<ECHO;
Send a specific SNMP request to a IP address and wait for replies.
Can be used with a broadcast or multicast address to discover SNMP
agents ("Command Responder Applications") responding to a given
community and port.
<FORM METHOD="POST" ACTION="/cgi-bin/msnmp/discover.cgi">
<BR>
<TABLE BORDER=0>
      <CAPTION><B>Device Monitor</B></CAPTION>
<TR><TH ALIGN=RIGHT>Host: <TH ALIGN=LEFT><INPUT SIZE=30 NAME="host"
TYPE="TEXT">
<TR><TH ALIGN=RIGHT>Community: <TH ALIGN=LEFT><INPUT SIZE=30
NAME="community" TYPE="TEXT" VALUE="public">
<TR><TH ALIGN=RIGHT>SNMP Port: <TH ALIGN=LEFT><INPUT SIZE=6
NAME="port" TYPE="TEXT" VALUE="161">
<TR><TH ALIGN=RIGHT>Timeout (ms): <TH ALIGN=LEFT><INPUT NAME="timeout"
TYPE="TEXT" VALUE="2000">
<TR><TH ALIGN=RIGHT>Retries: <TH ALIGN=LEFT><SELECT NAME="retries">
      <OPTION>3
      <OPTION>2
      <OPTION>1
</SELECT>
</TABLE>
<BR><BR>
<INPUT NAME="clear" TYPE="RESET" VALUE="Clear">
<INPUT NAME="submit" TYPE="SUBMIT" VALUE="Submit">
</FORM>
ECHO
&bottom;


#!/usr/bin/perl
print "Content-Type: text/html\n\n";
######################################################################
# Discover Agents: discover.cgi
######################################################################
use Socket;
use vars '$DEBUG';
my $DEBUG = 1;
use SNMP_tools;
```

```
use vars qw($MULTILINE_MATCHING);
require 5;
use SNMP_Session;
use BER;
use Socket;
  &ReadParse(*input);
  $host = $input{'host'};
  $community = $input{'community'};
  $port = $input{'port'};
  $sysName = $input{'sysName'};
  &top("Discover Agents - SNMP Web Tools");
  &bar1;
  print <<ECHO;
<P>
<TABLE BORDER=1>
        <CAPTION><H3></H3></CAPTION>
        <TR>
        <TD><B>Host</B></TD>
        <TD><B>Description</B></TD>
ECHO
my $sysDescr = encode_oid (1,3,6,1,2,1,1,1,0);
my $session = SNMP_Session->open ($host, $community, $port)
    || die "Cannot open SNMP session";
my @oids = ($sysDescr);
$session->send_query ($session->encode_get_request (@oids));
while ($session->wait_for_response ($session->timeout)) {
    my($response_length);
    $response_length
      = $session->receive_response_3 (SNMP_Session::get_response,
\@oids, 0);
    if ($response_length) {
      my $response = $session->pdu_buffer;
      my ($binding, $bindings, $oid, $value);
      eval {
          ($bindings) = $session->decode_get_response ($response);
      };
      while ($bindings ne '') {
          ($binding,$bindings) = decode_sequence ($bindings);
          ($oid,$value) = decode_by_template ($binding, "%O%@");
          ##print $pretty_oids{$oid}," => ",
          {
            my $string = pretty_print ($value);
            my $sender = $session->{'last_sender_addr'};
            local $MULTILINE_MATCHING = 1;
            $string =~ s/\n/\\n/g;
            $string =~ s/\r/\\r/g;
       #        print STDOUT pretty_address ($sender), "\n  ";
       #        print STDOUT $string, "\n";
            $name = pretty_address ($sender);
            print "<TR><TD>$name</TD><TD>$string</TD>";
          }
      }
    }
}
print "</TABLE>";
$session->close ();
&bottom;
1;
sub pretty_address
{
    my($addr) = shift;
```

```
    my($port,$ipaddr) = unpack_sockaddr_in($addr);
    my $hostname = gethostbyaddr ($ipaddr, AF_INET);
    return $hostname ? $hostname : ('['.inet_ntoa($ipaddr).']');
}
```

A.4 SET REQUEST

```
#!/usr/bin/perl
print "Content-Type: text/html\n\n";
######################################################################
# Set Request: form_set
######################################################################
use SNMP_tools;
&top("Set Request - SNMP Web Tools");
&bar3;
print <<ECHO;
<FORM METHOD="POST" ACTION="/cgi-bin/msnmp/getoid.cgi">
<BR>
<TABLE BORDER=0>
      <CAPTION><B>Get Object Id</B></CAPTION>
<TR><TH ALIGN=RIGHT>Host: <TH ALIGN=LEFT><INPUT SIZE=30 NAME="host"
TYPE="TEXT">
<TR><TH ALIGN=RIGHT>Community: <TH ALIGN=LEFT><INPUT SIZE=30
NAME="community" TYPE="TEXT" VALUE="public">
<TR><TH ALIGN=RIGHT>SNMP Port: <TH ALIGN=LEFT><INPUT SIZE=6
NAME="port" TYPE="TEXT" VALUE="161">
<TR><TH ALIGN=RIGHT>Timeout (ms): <TH ALIGN=LEFT><INPUT NAME="timeout"
TYPE="TEXT" VALUE="2000">
<TR><TH ALIGN=RIGHT>Retries: <TH ALIGN=LEFT><SELECT NAME="retries">
      <OPTION>3
      <OPTION>2
      <OPTION>1
</SELECT>
<TR><TH ALIGN=RIGHT>ASN.1 Object Id: <TH ALIGN=LEFT><INPUT SIZE=30
NAME="oid" TYPE="TEXT">
</TABLE>
<BR><BR>
<INPUT NAME="clear" TYPE="RESET" VALUE="Clear">
<INPUT NAME="submit" TYPE="SUBMIT" VALUE="Submit">
</FORM>
ECHO
&bottom;


#!/usr/bin/perl
print "Content-Type: text/html\n\n";
######################################################################
# Set Request :  getoid.cgi
######################################################################
  use Socket;
  use vars '$DEBUG';
  my $DEBUG = 1;

  require 5.002;
  use SNMP_Session;
  use BER;
  use SNMP_tools;

#sub main {
```

```
  my ($host, $oid, $community,$port,$response, $sysName);
  &ReadParse(*input);
  $host = $input{'host'};
  $community = $input{'community'};
  $port = $input{'port'};
  $sysName = $input{'sysName'};
  $oid = $input{'oid'};
  &top("Set Request - SNMP Web Tools");
  &bar3;
  ($response) = &snmpget($host, $community, $port, $oid);
  if ($response eq "-1") {
        print "<H2>$host did not respond to SNMP query !!</H2>\n";
  } else {
#        print "$response\n";
        &form_set;
  }
  &bottom;
#}
sub form_set {
print <<ECHO;
<FORM METHOD="POST" ACTION="/cgi-bin/msnmp/setoid.cgi">
<BR>
<TABLE BORDER=0>
        <CAPTION><B>Set Object Mib</B></CAPTION>
<TR><TH ALIGN=RIGHT>Host: <TH ALIGN=LEFT><INPUT SIZE=30 NAME="host"
TYPE="TEXT"
VALUE= $host>
<TR><TH ALIGN=RIGHT>Community: <TH ALIGN=LEFT><INPUT SIZE=30
NAME="community" TYPE="TEXT" VALUE= $community>
<TR><TH ALIGN=RIGHT>SNMP Port: <TH ALIGN=LEFT><INPUT SIZE=6
NAME="port" TYPE="TEXT" VALUE= $port>
<TR><TH ALIGN=RIGHT>Retries: <TH ALIGN=LEFT><SELECT NAME="retries">
        <OPTION>3
        <OPTION>2
        <OPTION>1
</SELECT>
<TR><TH ALIGN=RIGHT>Object Id: <TH ALIGN=LEFT><INPUT SIZE=30
NAME="oid" TYPE="TEXT" VALUE= $oid>
<TR><TH ALIGN=RIGHT>New Value: <TH ALIGN=LEFT><INPUT SIZE=50
NAME="value" TYPE="TEXT" VALUE= $response>
<TR><TH ALIGN=RIGHT>Type: <TH ALIGN=LEFT><SELECT NAME="type">
        <OPTION>string
        <OPTION>integer
        <OPTION>hex string
        <OPTION>decimal string
        <OPTION>ipaddress
        <OPTION>objid
        <OPTION>timeticks
        <OPTION>nullobj
</SELECT>
</TABLE>
<BR><BR>
<INPUT NAME="clear" TYPE="RESET" VALUE="Clear">
<INPUT NAME="submit" TYPE="SUBMIT" VALUE="Submit">
</FORM>
ECHO
}
#&getoid;
exit(0);
```

```
#!/usr/bin/perl
print "Content-Type: text/html\n\n";
#####################################################################
# Set Request :  setoid.cgi
#####################################################################
  use Socket;
  use vars '$DEBUG';
  my $DEBUG = 1;

  require 5.002;
  use SNMP_Session;
  use BER;
  use SNMP_tools;
  my ($host, $oid, $community,$port,$response, $sysName);
  &ReadParse(*input);
  $host = $input{'host'};
  $community = $input{'community'};
  $port = $input{'port'};
  $sysName = $input{'sysName'};
  $oid = $input{'oid'};
  $type = $input{'type'};
  $value = $input{'value'};
  &top("Set Request - SNMP Web Tools");
  &bar3;
  ($response) = &snmpget($host, $community, $port, $oid);
  if ($response eq "-1") {
        print "<H2>$host did not respond to SNMP query !!!</H2>\n";
  } else {
#        print "$response\n";
  }
  ($response) = &snmpset($host, $community, $oid, $type, $value);
  if ($response eq "-1") {
        print "<H2>$host did not respond to SNMP set !!!</H2>\n";
  } else {
        print "<H3> Set Request Ok !</H3>\n";
  }
  &bottom;
```

A.5 SEND TRAP

```
#!/usr/bin/perl
print "Content-Type: text/html\n\n";
#####################################################################
# Network Utils: form_trap
#####################################################################
use SNMP_tools;
&top("Send Trap - SNMP Web Tools");
&bar1;
print <<ECHO;
<FORM METHOD="POST" ACTION="/cgi-bin/msnmp/send_trap.cgi">
<BR>
<TABLE BORDER=0>
      <CAPTION><B>Parameters Traps</B></CAPTION>
<TR><TH ALIGN=RIGHT>Host: <TH ALIGN=LEFT><INPUT SIZE=30 NAME="host"
TYPE="TEXT">
<TR><TH ALIGN=RIGHT>Community: <TH ALIGN=LEFT><INPUT SIZE=30
NAME="community" TYPE="TEXT" VALUE="public">
<TR><TH ALIGN=RIGHT>Port Trap: <TH ALIGN=LEFT><INPUT SIZE=6
NAME="trapport"
```

```
TYPE="TEXT" VALUE="162">
<TR><TH ALIGN=RIGHT>Generic Trap: <TH ALIGN=LEFT><SELECT
NAME="gentrap">
      <OPTION VALUE="0">coldStart</OPTION>
      <OPTION VALUE="1">warmStart</OPTION>
      <OPTION VALUE="2">linkDown</OPTION>
      <OPTION VALUE="3">linkUp</OPTION>
      <OPTION VALUE="4">authenticationFailure</OPTION>
      <OPTION VALUE="5">egpNeighborLoss</OPTION>
      <OPTION VALUE="6">enterpriseSpecific</OPTION>
</SELECT>
<TR><TH ALIGN=RIGHT>Specific Trap: <TH ALIGN=LEFT><INPUT SIZE=30
NAME="spectrap" TYPE="TEXT" VALUE="0">
<TR><TH ALIGN=RIGHT>Bindings: <TH ALIGN=LEFT><INPUT SIZE=50
NAME="bindoid"
TYPE="TEXT" VALUE=" ">
<TH ALIGN=RIGHT>Value: <TH ALIGN=LEFT><INPUT SIZE=10 NAME="bindstr"
TYPE="TEXT"
VALUE=" ">
</TABLE>
<BR><BR>
<INPUT NAME="clear" TYPE="RESET" VALUE="Clear">
<INPUT NAME="submit" TYPE="SUBMIT" VALUE="Submit">
</FORM>
ECHO
&bottom;


#!/usr/bin/perl
print "Content-Type: text/html\n\n";
######################################################################
# Network Utilities: send_trap.cgi
######################################################################
use Socket;
use vars '$DEBUG';
my $DEBUG = 1;
use SNMP_tools;
use BER;
sub trap {

  &top("Send Trap - SNMP Web Tools");
  &bar1;
  &ReadParse(*input);
  $trap_receiver = $input{'host'};
  $trap_community = $input{'community'};
  $trap_port = $input{'trapport'};
  $genericTrap = $input{'gentrap'};
  $specificTrap = $input{'spectrap'};
  $bindoid = $input{'bindoid'};
  $bindstr = $input{'bindstr'};
my $start_time = time;
my $trap_session = SNMP_Session->open ($trap_receiver,
$trap_community,$trap_port);
  my ($if_index) = 1;
#  my $genericTrap = 2;       # linkDown
#  my $specificTrap = 0;
#  my @ifIndexOID = ( 1,3,6,1,2,1,2,2,1,1 );
#  my @ifDescrOID = ( 1,3,6,1,2,1,2,2,1,2 );
  my $upTime = int ((time - $start_time) * 100);
  my $myIpAddress = pack "CCCC", 10, 0, 0, 1;
  my @myOID = ( 1,3,6,1,4,1,2946,0,8,15 );
```

```
  warn "Sending trap failed"
    unless $trap_session->trap_request_send (encode_oid (@myOID),
                                     encode_ip_address ($myIpAddress),
                                     encode_int ($genericTrap),
                                     encode_int ($specificTrap),
                                     encode_timeticks ($upTime),
                                     [toOID($bindoid),
                                      encode_string($bindstr)]);
&bottom;
}
trap;
exit(0);
```

A.6 MIB WEB BROWSER

```
#!/usr/bin/perl
print "Content-Type: text/html\n\n";
######################################################################
# MIB SNMP BROWSER - mbrowser.cgi
######################################################################
use Socket;
use vars '$DEBUG';
my $DEBUG = 1;
use SNMP_tools;

&ReadParse(*input);
$hosts = $input{'hosts'};
&top("Mib Browser - SNMP Web Tools");
&bar2;
print <<ECHO;
<P>
Welcome to the Mini Web Brower - SNMP. This browser is a
CGI script written in Perl.
It uses a highly portable SNMP implementation written entirely in
Perl.
 There is no need to install any external
SNMP package.
Report bugs or any other comments to
<TT>lidti\@unsa.edu.ar</TT>
<HR>
<P><A HREF="setagent.cgi?hosts=">
Set Agents:</A> $hosts
<P>
<HR>
<H3>Official Internet MIBs:</H3>
<DL>
<DT>
<IMG ALT="" SRC="/icons/dir.gif">
<A HREF="system.cgi?hosts=$hosts">SNMPv2-MIB!system</A>
(1.3.6.1.2.1.1)
<DD>
<P>
<DT>
<IMG ALT="" SRC="/icons/dir.gif">
<A HREF="interfaces.cgi?hosts=$hosts">IF-MIB!interfaces</A>
(1.3.6.1.2.1.2)
<DD>
<P>
<DT>
```

```
<IMG ALT="" SRC="/icons/dir.gif">
<A HREF="at.cgi?hosts=$hosts">RFC1213-MIB!at</A>
(1.3.6.1.2.1.3)
<DD>
<P>
<DT>
<IMG ALT="" SRC="/icons/dir.gif">
<A HREF="ip.cgi?hosts=$hosts">IP-MIB!ip</A>
(1.3.6.1.2.1.4)
<DD>
<P>
<DT>
<IMG ALT="" SRC="/icons/dir.gif">
<A HREF="icmp.cgi?hosts=$hosts">IP-MIB!icmp</A>
(1.3.6.1.2.1.5)
<DD>
<P>
<DT>
<IMG ALT="" SRC="/icons/dir.gif">
<A HREF="tcp.cgi?hosts=$hosts">TCP-MIB!tcp</A>
(1.3.6.1.2.1.6)
<DD>
<P>
<DT>
<IMG ALT="" SRC="/icons/dir.gif">
<A HREF="udp.cgi?hosts=$hosts">UDP-MIB!udp</A>
(1.3.6.1.2.1.7)
<DD>
<P>
<DT>
<IMG ALT="" SRC="/icons/dir.gif">
<A HREF="egp.cgi?hosts=$hosts">RFC1213-MIB!egp</A>
(1.3.6.1.2.1.8)
<DD>
<P>
<DT>
<IMG ALT="" SRC="/icons/dir.gif">
<A HREF="transmission.cgi?hosts=$hosts">SNMPv2-SMI!transmission</A>
(1.3.6.1.2.1.10)
<DD>
<P>
<DT>
<IMG ALT="" SRC="/icons/dir.gif">
<A HREF="snmp.cgi?hosts=$hosts">SNMPv2-MIB!snmp</A>
(1.3.6.1.2.1.11)
<DD><P><DT><P><DT><P>
</DL>
<P>
<H3>Enterprise MIBs:</H3>
<DL>
<DT>
<IMG ALT="" SRC="/icons/dir.gif">
<A HREF="cisco.cgi?hosts=$hosts">CISCO-SMI!cisco</A>
(1.3.6.1.4.1.9)
<DD>The Structure of Management Information for the
Cisco enterprise.
<P>
<DT>
<IMG ALT="" SRC="/icons/dir.gif">
<A HREF="novell.cgi?hosts=$hosts">TCPIPX-MIB!novell</A>
(1.3.6.1.4.1.23)
```

```
<DD>
<P>
<DT>
<IMG ALT="" SRC="/icons/dir.gif">
<A HREF="sun.cgi?hosts=$hosts">SUN-MIB!sun</A>
(1.3.6.1.4.1.42)
<DD>
<P>
<DT>
</BODY></HTML>
ECHO
&bar2;
&bottom;


#!/usr/bin/perl
print "Content-Type: text/html\n\n";
######################################################################
# Mib Web Browser - SNMP: mquery.cgi
######################################################################
use Socket;
use vars '$DEBUG';
my $DEBUG = 1;
use SNMP_tools;
sub mquery {
  &ReadParse(*input);
  $oids = $input{'oids'};
  $hosts = $input{'hosts'};

  &top("Mib Browser - SNMP Web Tools");
  &bar2;

  @in = split(/,/,$hosts);
  $i=0;
  foreach $i(0 .. $#in) {
     # print $in[$i];
     ($router,$port,$community) = split(/:/,$in[$i]);
     if ($port eq "") { $port=161; }
     if ($community eq "") { $community=public;
  }
  print <<ECHO;
<B>Host:</B> $router
<B>Port:</B> $port
<B>Community:</B> $community<P>
ECHO
  my (@ifoids) = snmpgettable2($router,$community,$port,$oids);
  if ($ifoids[1] eq "-1") {
        print "<H2>$router did not respond to SNMP query !!</H2>\n";
      &bottom;
      exit(1);
  } else {
        print STDERR "Ok Objeto $oids\n" if $DEBUG;
  }
  my(%oid,$index);
  print <<ECHO;
<P>
<TABLE BORDER=1>
      <CAPTION><H3></H3></CAPTION>
        <TR>
      <TD><B>Object Mib</B></TD>
      <TD><B>Value</B></TD>
```

```
ECHO
#  foreach $index (keys %oid) {
  $i = 0;
  $cant = @ifoids;
  while ($i < $cant) {
     my $name="$router.$index";
     ($id,$value) = split(':',$ifoids[$i],2);
     print "<TR><TD>$id</TD><TD>$value</TD>";
     $i++;
  }
  print "</TABLE><P>";
  }
  #&bar2;
  &bottom;
}
mquery;
exit(0);


#!/usr/bin/perl
print "Content-Type: text/html\n\n";
######################################################################
# MIB SNMP BROWSER - setagent.cgi
######################################################################
use SNMP_tools;
&ReadParse(*input);
$hosts = $input{'hosts'};
&top("Mib Browser - SNMP Web Tools");
&bar2;
print <<ECHO;
<HTML><HEAD><title>SNMP MIB Browser
</title></HEAD><BODY>
<FORM ACTION="/cgi-bin/mbrowser/mbrowser.cgi" METHOD=POST>
<H3>Set SNMP agent addresses:</H3>
<INPUT TYPE="text" NAME=hosts VALUE="" SIZE=60>
<P>Agent addresses are either host names or internet addresses in
decimal dot notation like 170.210.201.193 . Multiple addresses
should be separated using ",". SNMP queries are send
to the well known SNMP port 161 using community public. Every
request will use a timeout of 5 seconds and 3 retries.<P>
You can specify alternate port numbers or community strings other
than public by using the syntax
&lt;address&gt;:&lt;port&gt;:&lt;community&gt;<P>
<INPUT TYPE="submit" VALUE="set and return">
</FORM>
<TABLE BORDER>
<TR><TH>IP Address</TH><TH>Port</TH><TH>Community</TH></TR>
</TABLE>
</BODY></HTML>
ECHO
#&bar2;
&bottom;


#!/usr/bin/perl
print "Content-Type: text/html\n\n";
######################################################################
# MINI SNMP BROWSER - system.cgi
######################################################################
use SNMP_tools;
&ReadParse(*input);
```

```
$hosts = $input{'hosts'};
&top("Mib Browser - SNMP Web Tools");
&bar2;
print <<ECHO;
<B>Agents:</B> $hosts
<P>
<HR><DL><DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.1.1">SNMPv2-
MIB!sysDescr</A>
(1.3.6.1.2.1.1.1)
<DD>A textual description of the entity.  This value should
include the full name and version identification of the
system's hardware type, software operating-system, and
networking software.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.1.2">SNMPv2-
MIB!sysObjectID</A>
(1.3.6.1.2.1.1.2)
<DD>The vendor's authoritative identification of the network
management subsystem contained in the entity.  This value is
allocated within the SMI enterprises subtree (1.3.6.1.4.1)
and provides an easy and unambiguous means for determining
`what kind of box' is being managed.  For example, if vendor
`Flintstones, Inc.' was assigned the subtree
1.3.6.1.4.1.4242, it could assign the identifier
1.3.6.1.4.1.4242.1.1 to its `Fred Router'.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.1.3">SNMPv2-
MIB!sysUpTime</A>
(1.3.6.1.2.1.1.3)
<DD>The time (in hundredths of a second) since the network
management portion of the system was last re-initialized.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.1.4">SNMPv2-
MIB!sysContact</A>
(1.3.6.1.2.1.1.4)
<DD>The textual identification of the contact person for this
managed node, together with information on how to contact
this person.  If no contact information is known, the value
is the zero-length string.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.1.5">SNMPv2-
MIB!sysName</A>
(1.3.6.1.2.1.1.5)
<DD>An administratively-assigned name for this managed node.
By convention, this is the node's fully-qualified domain
name.  If the name is unknown, the value is the zero-length
string.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
```

```
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.1.6">SNMPv2-
MIB!sysLocation</A>
(1.3.6.1.2.1.1.6)
<DD>The physical location of this node (e.g., `telephone
closet, 3rd floor').  If the location is unknown, the value
is the zero-length string.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.1.7">SNMPv2-
MIB!sysServices</A>
(1.3.6.1.2.1.1.7)
<DD>A value which indicates the set of services that this
entity may potentially offers.  The value is a sum.  This
sum initially takes the value zero, Then, for each layer, L,
in the range 1 through 7, that this node performs
transactions for, 2 raised to (L - 1) is added to the sum.
For example, a node which performs only routing functions
would have a value of 4 (2^(3-1)).  In contrast, a node
which is a host offering application services would have a
value of 72 (2^(4-1) + 2^(7-1)).  Note that in the context
of the Internet suite of protocols, values should be
calculated accordingly:
     layer      functionality
       1        physical (e.g., repeaters)
       2        datalink/subnetwork (e.g., bridges)
       3        internet (e.g., supports the IP)
       4        end-to-end  (e.g., supports the TCP)
       7        applications (e.g., supports the SMTP)
For systems including OSI protocols, layers 5 and 6 may also
be counted.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.1.8">SNMPv2-
MIB!sysORLastChange</A>
(1.3.6.1.2.1.1.8)
<DD>The value of sysUpTime at the time of the most recent
change in state or value of any instance of sysORID.
<P>
<DT>
<IMG ALT="" SRC="/icons/dir.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.1.9">SNMPv2-
MIB!sysORTable</A>
(1.3.6.1.2.1.1.9)
<DD>The (conceptual) table listing the capabilities of the
local SNMPv2 entity acting in an agent role with respect to
various MIB modules.  SNMPv2 entities having dynamically-
configurable support of MIB modules will have a
dynamically-varying number of conceptual rows.
<DL>
<DT>
<IMG ALT="" SRC="/icons/dir.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1..3.6.1.2.1.1.9.1">SNMPv2-
MIB!sysOREntry</A>
(1.3.6.1.2.1.1.9.1)
<DD>An entry (conceptual row) in the sysORTable.
<DL>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
```

```
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.1.9.1.1">SNMPv2-
MIB!sysORIndex</A>
(1.3.6.1.2.1.1.9.1.1)
<DD>The auxiliary variable used for identifying instances of
the columnar objects in the sysORTable.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.1.9.1.2">SNMPv2-
MIB!sysORID</A>
(1.3.6.1.2.1.1.9.1.2)
<DD>An authoritative identification of a capabilities statement
with respect to various MIB modules supported by the local
SNMPv2 entity acting in an agent role.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.1.9.1.3">SNMPv2-
MIB!sysORDescr</A>
(1.3.6.1.2.1.1.9.1.3)
<DD>A textual description of the capabilities identified by the
corresponding instance of sysORID.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.1.9.1.4">SNMPv2-
MIB!sysORUpTime</A>
(1.3.6.1.2.1.1.9.1.4)
<DD>The value of sysUpTime at the time this conceptual row was
last instanciated.
<P>
</DL>
<P>
</DL>
<P>
</DL>
</BODY></HTML>
ECHO
&bar2;
&bottom;


#!/usr/bin/perl
print "Content-Type: text/html\n\n";
######################################################################
# MINI WEB BROWSER - SNMP : interfaces.cgi
######################################################################
#
#
#
use SNMP_tools;
&ReadParse(*input);
$hosts = $input{'hosts'};
&top("Mib Browser - SNMP Web Tools");
&bar2;
print <<ECHO;
<B>Agents:</B> $hosts
<P>
<HR>
<DL>
<DT>
```

```
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.1">IF-
MIB!ifNumber</A>
(1.3.6.1.2.1.2.1)
<DD>The number of network interfaces (regardless of their
current state) present on this system.
<P>
<DT>
<IMG ALT="" SRC="/icons/dir.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.2">IF-
MIB!ifTable</A>
(1.3.6.1.2.1.2.2)
<DD>A list of interface entries.  The number of entries
is given by the value of ifNumber.
<DL>
<DT>
<IMG ALT="" SRC="/icons/dir.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.2.1">IF-
MIB!ifEntry</A>
(1.3.6.1.2.1.2.2.1)
<DD>An entry containing management information applicable
to a particular interface.
<DL>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.2.1.1">IF-
MIB!ifIndex</A>
(1.3.6.1.2.1.2.2.1.1)
<DD>A unique value, greater than zero, for each
interface.  It is recommended that values are assigned
contiguously starting from 1.  The value for each
interface sub-layer must remain constant at least from
one re-initialization of the entity's network
management system to the next re-initialization.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.2.1.2">IF-
MIB!ifDescr</A>
(1.3.6.1.2.1.2.2.1.2)
<DD>A textual string containing information about the
interface.  This string should include the name of the
manufacturer, the product name and the version of the
interface hardware/software.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.2.1.3">IF-
MIB!ifType</A>
(1.3.6.1.2.1.2.2.1.3)
<DD>The type of interface.  Additional values for ifType
are assigned by the Internet Assigned Numbers
Authority (IANA), through updating the syntax of the
IANAifType textual convention.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.2.1.4">IF-
MIB!ifMtu</A>
(1.3.6.1.2.1.2.2.1.4)
<DD>The size of the largest packet which can be
```

```
sent/received on the interface, specified in octets.
For interfaces that are used for transmitting network
datagrams, this is the size of the largest network
datagram that can be sent on the interface.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.2.1.5">IF-
MIB!ifSpeed</A>
(1.3.6.1.2.1.2.2.1.5)
<DD>An estimate of the interface's current bandwidth in
bits per second.  For interfaces which do not vary in
bandwidth or for those where no accurate estimation
can be made, this object should contain the nominal
bandwidth.  If the bandwidth of the interface is
greater than the maximum value reportable by this
object then this object should report its maximum
value (4,294,967,295) and ifHighSpeed must be used to
report the interace's speed.  For a sub-layer which
has no concept of bandwidth, this object should be
zero.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.2.1.6">IF-
MIB!ifPhysAddress</A>
(1.3.6.1.2.1.2.2.1.6)
<DD>The interface's address at its protocol sub-layer.
For example, for an 802.x interface, this object
normally contains a MAC address.  The interface's
media-specific MIB must define the bit and byte
ordering and the format of the value of this object.
For interfaces which do not have such an address
(e.g., a serial line), this object should contain an
octet string of zero length.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.2.1.7">IF-
MIB!ifAdminStatus</A>
(1.3.6.1.2.1.2.2.1.7)
<DD>The desired state of the interface.  The testing(3)
state indicates that no operational packets can be
passed.  When a managed system initializes, all
interfaces start with ifAdminStatus in the down(2)
state.  As a result of either explicit management
action or per configuration information retained by
the managed system, ifAdminStatus is then changed to
either the up(1) or testing(3) states (or remains in
the down(2) state).
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.2.1.8">IF-
MIB!ifOperStatus</A>
(1.3.6.1.2.1.2.2.1.8)
<DD>The current operational state of the interface.  The
testing(3) state indicates that no operational packets
can be passed.  If ifAdminStatus is down(2) then
ifOperStatus should be down(2).  If ifAdminStatus is
changed to up(1) then ifOperStatus should change to
```

```
up(1) if the interface is ready to transmit and
receive network traffic; it should change to
dormant(5) if the interface is waiting for external
actions (such as a serial line waiting for an incoming
connection); it should remain in the down(2) state if
and only if there is a fault that prevents it from
going to the up(1) state; it should remain in the
notPresent(6) state if the interface has missing
(typically, hardware) components.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.2.1.9">IF-
MIB!ifLastChange</A>
(1.3.6.1.2.1.2.2.1.9)
<DD>The value of sysUpTime at the time the interface
entered its current operational state.  If the current
state was entered prior to the last re-initialization
of the local network management subsystem, then this
object contains a zero value.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.2.1.10">IF-
MIB!ifInOctets</A>
(1.3.6.1.2.1.2.2.1.10)
<DD>The total number of octets received on the interface,
including framing characters.

Discontinuities in the value of this counter can occur
at re-initialization of the management system, and at
other times as indicated by the value of
ifCounterDiscontinuityTime.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.2.1.11">IF-
MIB!ifInUcastPkts</A>
(1.3.6.1.2.1.2.2.1.11)
<DD>The number of packets, delivered by this sub-layer to
a higher (sub-)layer, which were not addressed to a
multicast or broadcast address at this sub-layer.

Discontinuities in the value of this counter can occur
at re-initialization of the management system, and at
other times as indicated by the value of
ifCounterDiscontinuityTime.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.2.1.12">IF-
MIB!ifInNUcastPkts</A>
(1.3.6.1.2.1.2.2.1.12)
<DD>The number of packets, delivered by this sub-layer to
a higher (sub-)layer, which were addressed to a
multicast or broadcast address at this sub-layer.

Discontinuities in the value of this counter can occur
at re-initialization of the management system, and at
other times as indicated by the value of
ifCounterDiscontinuityTime.
```

```
This object is deprecated in favour of
ifInMulticastPkts and ifInBroadcastPkts.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.2.1.13">IF-
MIB!ifInDiscards</A>
(1.3.6.1.2.1.2.2.1.13)
<DD>The number of inbound packets which were chosen to be
discarded even though no errors had been detected to
prevent their being deliverable to a higher-layer
protocol.  One possible reason for discarding such a
packet could be to free up buffer space.

Discontinuities in the value of this counter can occur
at re-initialization of the management system, and at
other times as indicated by the value of
ifCounterDiscontinuityTime.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.2.1.14">IF-
MIB!ifInErrors</A>
(1.3.6.1.2.1.2.2.1.14)
<DD>For packet-oriented interfaces, the number of inbound
packets that contained errors preventing them from
being deliverable to a higher-layer protocol.  For
character-oriented or fixed-length interfaces, the
number of inbound transmission units that contained
errors preventing them from being deliverable to a
higher-layer protocol.

Discontinuities in the value of this counter can occur
at re-initialization of the management system, and at
other times as indicated by the value of
ifCounterDiscontinuityTime.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.2.1.15">IF-
MIB!ifInUnknownProtos
</A>(1.3.6.1.2.1.2.2.1.15)
<DD>For packet-oriented interfaces, the number of packets
received via the interface which were discarded
because of an unknown or unsupported protocol.  For
character-oriented or fixed-length interfaces that
support protocol multiplexing the number of
transmission units received via the interface which
were discarded because of an unknown or unsupported
protocol.  For any interface that does not support
protocol multiplexing, this counter will always be 0.

Discontinuities in the value of this counter can occur
at re-initialization of the management system, and at
other times as indicated by the value of
ifCounterDiscontinuityTime.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
```

```
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.2.1.16">IF-
MIB!ifOutOctets</A>
(1.3.6.1.2.1.2.2.1.16)
<DD>The total number of octets transmitted out of the
interface, including framing characters.
Discontinuities in the value of this counter can occur
at re-initialization of the management system, and at
other times as indicated by the value of
ifCounterDiscontinuityTime.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.2.1.17">IF-
MIB!ifOutUcastPkts</A>
(1.3.6.1.2.1.2.2.1.17)
<DD>The total number of packets that higher-level
protocols requested be transmitted, and which were not
addressed to a multicast or broadcast address at this
sub-layer, including those that were discarded or not
sent.

Discontinuities in the value of this counter can occur
at re-initialization of the management system, and at
other times as indicated by the value of
ifCounterDiscontinuityTime.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.2.1.18">IF-
MIB!ifOutNUcastPkts</A>
(1.3.6.1.2.1.2.2.1.18)
<DD>The total number of packets that higher-level
protocols requested be transmitted, and which were
addressed to a multicast or broadcast address at this
sub-layer, including those that were discarded or not
sent.
Discontinuities in the value of this counter can occur
at re-initialization of the management system, and at
other times as indicated by the value of
ifCounterDiscontinuityTime.
This object is deprecated in favour of
ifOutMulticastPkts and ifOutBroadcastPkts.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.2.1.19">IF-
MIB!ifOutDiscards</A>
(1.3.6.1.2.1.2.2.1.19)
<DD>The number of outbound packets which were chosen to
be discarded even though no errors had been detected
to prevent their being transmitted.  One possible
reason for discarding such a packet could be to free
up buffer space.
Discontinuities in the value of this counter can occur
at re-initialization of the management system, and at
other times as indicated by the value of
ifCounterDiscontinuityTime.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
```

```
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.2.1.20">IF-
MIB!ifOutErrors</A>
(1.3.6.1.2.1.2.2.1.20)
<DD>For packet-oriented interfaces, the number of
outbound packets that could not be transmitted because
of errors.  For character-oriented or fixed-length
interfaces, the number of outbound transmission units
that could not be transmitted because of errors.

Discontinuities in the value of this counter can occur
at re-initialization of the management system, and at
other times as indicated by the value of
ifCounterDiscontinuityTime.
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.2.1.21">IF-
MIB!ifOutQLen</A>
(1.3.6.1.2.1.2.2.1.21)
<DD>The length of the output packet queue (in packets).
<P>
<DT>
<IMG ALT="" SRC="/icons/doc.gif">
<A HREF="mquery.cgi?hosts=$hosts&oids=1.3.6.1.2.1.2.2.1.22">IF-
MIB!ifSpecific</A>
(1.3.6.1.2.1.2.2.1.22)
<DD>A reference to MIB definitions specific to the
particular media being used to realize the interface.
It is recommended that this value point to an instance
of a MIB object in the media-specific MIB, i.e., that
this object have the semantics associated with the
InstancePointer textual convention defined in RFC
1903.  In fact, it is recommended that the media-
specific MIB specify what value ifSpecific should/can
take for values of ifType.  If no MIB definitions
specific to the particular media are available, the
value should be set to the OBJECT IDENTIFIER { 0 0 }.
<P>
</DL>
<P>
</DL>
<P>
</DL>
</BODY></HTML>
ECHO
&bar2;
&bottom;
```

A.7 PROCESOS COLECTORES

```
#####################################################################
#   Colector: state.pl
#####################################################################
#!/usr/bin/perl
#  Get State of Oject Mib from Table Host Management
# Run crontab
# 0,5,10,15,20,25,30,35,40,45,50,55 * * * * /home/danny/state.pl
use SNMP_tools;
use Pg;
```

```
$conn = Pg::connectdb("dbname = snmpdb");
Pg::doQuery($conn,"select * from host", \@ary);
for $i (0 .. $#ary) {
#  for $j (0 .. $#{$ary[$i] } ) {
#  ($h_id,$ip,$community,$port,$o_idi,$o_ido) = @ary[$i];
   $h_id = $ary[$i][0];
   $ip   = $ary[$i][1];
   $community = $ary[$i][2];
   $port   = $ary[$i][3];
   $m_time = $ary[$i][4];
   $a_time = $ary[$i][5];
   $o_idi  = $ary[$i][6];
   $o_ido  = $ary[$i][7];
   $monitor= $ary[$i][8];
#  }
   if ($monitor ne "disable") {
#   print "$h_id $ip $port *$monitor*";
($ip,$na) = split(/ /,$ip);
($community,$na) = split(/ /,$community);
($port,$na) = split(/ /,$port);
($o_idi,$na) = split(/ /,$o_idi);
($o_ido,$na) = split(/ /,$o_ido);
#print $ip,$community,$port,$o_idi,$o_ido;
      ($o_vali,$o_valo) = &snmpget($ip, $community, $port, $o_idi,
$o_ido);
      if ($o_vali ne "-1")
      {
         chop ($date = `/bin/date '+%d-%m-%y'`);
         chop ($time = `/bin/date '+%H:%M'`);
         $command = "insert into state (h_id,date,time,o_vali,o_valo)
values ($h_id,'$date','$time',$o_vali,$o_valo)";
         $result = $conn->exec($command);
         if ($result eq '') {
             print STDERR "Insert Error into State...\n";
         }
         else {
             print STDERR "Insert Ok!\n";
         }
      }
      else
      {
      print STDERR "$host did not respond to SNMP query\n";
      }
   }
}


#####################################################################
#   Colector: trap.pl
#####################################################################
#!/usr/bin/perl -w
#
# Listen for SNMP traps, decode and print them
#
package main;
use Pg;
use SNMP_Session "0.60";
use BER;
use Socket;
$conn = Pg::connectdb("dbname = snmpdb");
my $session = SNMPv1_Session->open_trap_session ();
```

```
my ($trap, $sender, $sender_port);
while (($trap, $sender, $sender_port) = $session->receive_trap ()) {
    print_trap ($session,$trap,$sender,$sender_port);
}
1;
sub print_trap ($$) {
    my ($this,$trap,$sender,$sender_port) = @_;
    my ($encoded_pair, $oid, $value);
    my ($community, $ent, $agent, $gen, $spec, $dt, $bindings)
      = $this->decode_trap_request ($trap);
    my ($binding, $prefix);
    $enterprise = BER::pretty_oid ($ent);
    $agentaddr = inet_ntoa ($agent);
    $sender = inet_ntoa ($sender);
    $uptime = BER::pretty_uptime_value ($dt);
    $prefix = "     bindings: ";
    $bind = '';
    while ($bindings) {
      ($binding,$bindings) = decode_sequence ($bindings);
      ($oid,$value) = decode_by_template ($binding, "%O%@");
      $bind = $bind.BER::pretty_oid ($oid)." => ".pretty_print
($value)."\n";
      $prefix = "  ";
    }
    chop ($date = `/bin/date '+%d-%m-%y'`);
    chop ($time = `/bin/date '+%H:%M'`);
    print STDERR "Received trap from ($sender).$sender_port\n";
    print STDERR "\n";
    print STDERR "    Community: ".$community."\n";
    print STDERR "   Enterprise: ".$enterprise."\n";
    print STDERR "Agent Address: ".$agentaddr."\n";
    print STDERR "   Generic Id: ".$gen."\n";
    print STDERR "  Specific Id: ".$spec."\n";
    print STDERR "       Uptime: ".$uptime."\n";
    print STDERR "     Bindings: \n";
    print STDERR "$bind\n";
    $command = "insert into traps
(date,time,sender,sender_port,community,enterprise,agent_addr,generic_
id,
specific_id,uptime,bindings) values
('$date','$time','$sender',$sender_port,'$community','$enterprise','$a
gentaddr',
$gen,$spec,'$uptime','$bind')";
    $result = $conn->exec($command);
    if ($result eq '') {
       print STDERR "Insert Database Error into Traps...\n";
    }
    else {
       print STDERR "Insert Database Traps Ok!\n";
    }
    # Mail delivery
    $mailprog = '/usr/lib/sendmail';
    Pg::doQuery($conn,"select email from admin where agent_addr =
'$sender'", \@ary);
    for $i (0 .. $#ary) {
        $mailto = $ary[$i][0];
        $username = "ada\@lidti.unsa.edu.ar";
        $realname = "Administrator Traps";
        $comments = " ";
        open (MAIL, "|$mailprog $mailto") || die "Can't open
$mailprog!\n";
```

```
        print MAIL "From: $username\n";
        print MAIL "Reply-to: $username\n";
        print MAIL "Subject: Mail from daemon Trap \n\n";
        print MAIL "$username ( $realname ) sent the following\n";
        print MAIL "Mail from the daemon Trap:\n\n";
        print MAIL "------------------------------------------------
------\n";
        print MAIL "Received trap from ($sender).$sender_port\n";
        print MAIL "\n";
        print MAIL "    Community: ".$community."\n";
        print MAIL "   Enterprise: ".$enterprise."\n";
        print MAIL "Agent Address: ".$agentaddr."\n";
        print MAIL "   Generic Id: ".$gen."\n";
        print MAIL "  Specific Id: ".$spec."\n";
        print MAIL "       Uptime: ".$uptime."\n";
        print MAIL "     Bindings: \n";
        print MAIL "$bind\n";
        print MAIL "------------------------------------------------
------\n";
        print MAIL "$comments";
        close (MAIL);
        print STDERR "Send mail for $mailto...\n";
    }
}


######################################################################
# Colector: Ping.pl
######################################################################
#!/usr/bin/perl
require 5.003;
use Getopt::Std;
use File::Basename;
use Config;
$Prog_name = basename($0);
use Pg;
$conn = Pg::connectdb("dbname = snmpdb");
Pg::doQuery($conn,"select * from ping", \@ary);
for $i (0 .. $#ary) {
   $ip = $ary[$i][0];
   $pktc = $ary[$i][1];
   $pktl = $ary[$i][2];
   $monitor = $ary[$i][3];
   $data   = $ary[$i][4];
   $state  = $ary[$i][5];
   if ($monitor ne "disable") {
      print "$ip $pktc $pktl $monitor $data $state\n";
      ($pt_min, $pt_avg, $pt_max) = ping($ip, $pktl, $pktc);
      if ($pt_min ne "-1")
      {
       $stateup = "up";
      }
      else
      {
       $stateup = "down????";
       email($ip);
         print STDERR "Host $ip unreachable\n";
      }
      $command = "update ping set state = '$stateup' where ip =
'$ip'";
      $result = $conn->exec($command);
```

```
      if ($result eq '') {
          print STDERR "Update Error into Ping...\n";
      }
      else {
          print STDERR "Update Ok!\n";
      }
   }
}
exit(0);
#####################################################################
# ping host retrieve and return min, avg, max round trip time
#
sub ping {
      my($host, $length, $count) = @_;
      my(%ping, $ping_output);

      # List of known ping programs
      %ping = (
            'linux'     =>    "/bin/ping -c $count -s $length $host",
            'solaris' =>      "/usr/sbin/ping -s $host $length $count",
            );
      unless (defined($ping{'linux'})) {
            print STDERR "${Prog_name}: FATAL: Not yet configured for
$Config{'osname'}\n";
            exit(1);
            }

      unless (open(PING, "$ping{'linux'} 2>&1 |")) {
            print STDERR "${Prog_name}: FATAL: Can't open
$ping{$Config{'osname'}}: $!";
            exit(1);
            }

      while (<PING>) {
            $ping_output .= $_;
            close(PING), return($1,$2,$3)
                  if m|^round-trip(?: \(ms\) )? min/avg/max =
(\d+)(?:\.\d+)?/(\d+)(?:\.\d+)?/(\d+)(?:\.\d+)?|;
            }
#     print "${Prog_name}: ERROR: Could not find ping summary for
$host\n";
#     print "${Prog_name}: INFO: The output of the ping command
$ping{$Config{'osname'}} was:\n";

      close(PING), return(-1,-1,-1);
}
#############################################################
# Mail delivery
#
sub email {
    my($p1) = @_;
    $mailprog = '/usr/lib/sendmail';
    ($hostip,$tash) = split(' ',$p1);
    chop ($date = `/bin/date '+%d-%m-%y'`);
    chop ($time = `/bin/date '+%H:%M'`);
    Pg::doQuery($conn,"select email from admin where agent_addr =
'$hostip'", \@ary2);
    for $i (0 .. $#ary2) {
        $mailto = $ary2[$i][0];
        $username = "ada\@lidti.unsa.edu.ar";
        $realname = "Ping Reports";
```

```
        $comments = " ";
        open (MAIL, "|$mailprog $mailto") || die "Can't open
$mailprog!\n";
        print MAIL "From: $username\n";
        print MAIL "Reply-to: $username\n";
        print MAIL "Subject: Mail from daemon Ping\n\n";
        print MAIL "$username ( $realname ) sent the following\n";
        print MAIL "Mail from daemon Ping:\n\n";
        print MAIL "-------------------------------------------------
------\n";        print MAIL "Host Unreacheble ($hostip)\n";
        print MAIL "\n";
        print MAIL "Date: $date  Time: $time\n";
        print MAIL "\n";
        print MAIL "Please state verify.\n";
        print MAIL "-------------------------------------------------
------\n";        print MAIL "$comments";
        close (MAIL);
        print STDERR "Send mail for $mailto...\n";
    }
}
```

A.8 MODULO SNMP_TOOLS.PM

```
#####################################################################
#   Modulo Perl: SNMP_tools.pm
#####################################################################
package SNMP_tools;
require 5.002;
#use strict;
use vars qw(@ISA @EXPORT $VERSION);
use Exporter;
use BER "0.58";
use SNMP_Session "0.59";
use Socket;
$VERSION = '0.57';
@ISA = qw(Exporter);
@EXPORT = qw(snmpget snmpgettable snmpgettable2 snmpgetnext snmpwalk
snmpset snmptrap snmpmapOID top bottom bar1 bar2 bar3 links ReadParse
ether_hex toOID);
sub snmpget {
        my($host,$community,$port,@varList) = @_;
        my(@enoid, $var,$response, $bindings, $binding, $value,
$inoid,
                $outoid, $upoid,$oid,@retvals);
        grep ($_=toOID($_), @varList);
        srand();
        my $session = SNMP_Session->open ($host , $community, $port);
        if ($session->get_request_response(@varList)) {
                $response = $session->pdu_buffer;
                ($bindings) = $session->decode_get_response
($response);
                $session->close ();
                while ($bindings) {
                        ($binding,$bindings) = decode_sequence
($bindings);
                        ($oid,$value) = decode_by_template ($binding,
"%O%@");
                        my $tempo = pretty_print($value);
                        $tempo=~s/\t/ /g;
                        $tempo=~s/\n/ /g;
```

```
                        $tempo=~s/^\s+//;
                        $tempo=~s/\s+$//;
                        push @retvals,  $tempo;
                }
                return (@retvals);
        } else {
                return (-1,-1);
        }
}
sub snmpgettable{
  my($host,$community,$port,$var) = @_;
  my($next_oid,$enoid,$orig_oid,
     $response, $bindings, $binding, $value, $inoid,$outoid,
     $upoid,$oid,@table,$tempo,$doid);
#    unless $snmpget::OIDS{$var};
# $orig_oid = encode_oid(split /\./, $snmpget::OIDS{$var});
  $orig_oid = toOID($var);
  $enoid=$orig_oid;
  srand();
  my $session = SNMP_Session->open ($host , $community, $port);
  for(;;)  {
    if ($session->getnext_request_response(($enoid))) {
         $response = $session->pdu_buffer;
         ($bindings) = $session->decode_get_response ($response);
         ($binding,$bindings) = decode_sequence ($bindings);
         ($next_oid,$value) = decode_by_template ($binding, "%O%@");
         # quit once we are outside the table
         last unless BER::encoded_oid_prefix_p($orig_oid,$next_oid);
         $doid = BER::pretty_oid($next_oid);
         $tempo = pretty_print($value);
         #print "$var: '$tempo'\n";
         $tempo=~s/\t/ /g;
         $tempo=~s/\n/ /g;
         $tempo=~s/^\s+//;
         $tempo=~s/\s+$//;
         push @table, $tempo;
    } else {
         return (-1,-1);
    }
    $enoid=$next_oid;
  }
  $session->close ();
  return (@table);
}

sub snmpgettable2{
  my($host,$community,$port,$var) = @_;
  my($next_oid,$enoid,$orig_oid,
     $response, $bindings, $binding, $value, $inoid,$outoid,
     $upoid,$oid,@table,$tempo,@toids,$doid);
#    unless $snmpget::OIDS{$var};
$orig_oid = encode_oid(split('\.',$var));
#  $orig_oid = toOID($var);
#  $orig_oid = encode_oid(split /\./, $snmpget::OIDS{$var});
  $enoid=$orig_oid;
  srand();
  my $session = SNMP_Session->open ($host , $community, $port);
  for(;;)  {
    if ($session->getnext_request_response(($enoid))) {
         $response = $session->pdu_buffer;
         ($bindings) = $session->decode_get_response ($response);
```

```
        ($binding,$bindings) = decode_sequence ($bindings);
        ($next_oid,$value) = decode_by_template ($binding, "%O%@");
        # quit once we are outside the table
        last unless BER::encoded_oid_prefix_p($orig_oid,$next_oid);
        $doid = BER::pretty_oid($next_oid);
        $tempo = pretty_print($value);
        #print "$var: '$tempo'\n";
        $tempo=~s/\t/ /g;
        $tempo=~s/\n/ /g;
        $tempo=~s/^\s+//;
        $tempo=~s/\s+$//;
        push @table, "$doid:$tempo";
    } else {
        return (-1,-1);
    }
    $enoid=$next_oid;
  }
  $session->close ();
  return (@table);
}

sub fmi {
  my($number) = $_[0];
  my(@short);
  @short = ("Bytes/s","kBytes/s","MBytes/s","GBytes/s");
  my $digits=length(""."$number);
  my $divm=0;
  while ($digits-$divm*3 > 4) { $divm++; }
  my $divnum = $number/10**($divm*3);
  return sprintf("%1.1f %s",$divnum,$short[$divm]);
}

sub bar1 {
print <<ECHO;
<BR>
<CENTER><H5>
<BR>
<B><A HREF="form_mon">home</A> &#32;&#166;
<A HREF="back.html">back</A> &#32;&#166;
<A HREF="next.html">next</A> &#32;&#166;
<A HREF="main">menu</A> &#32;&#166;
<A HREF="search.html#search4d">search</A> &#32;&#166;
<A HREF="contact.html">contact</A> &#32;&#166;
<A HREF="comment.html">comment</A> &#32;&#166;
<A HREF="helpinfo.html">helpinfo</A></B><BR>
<BR>
</H5></CENTER>
ECHO
}

sub bar2 {
print <<ECHO;
<BR>
<CENTER><H5>
<BR>
<B><A HREF="/cgi-bin/mbrowser/mbrowser.cgi">home</A> &#32;&#166;
<A HREF="back.html">back</A> &#32;&#166;
<A HREF="next.html">next</A> &#32;&#166;
<A HREF="/cgi-bin/msnmp/main">menu</A> &#32;&#166;
<A HREF="search.html#search4d">search</A> &#32;&#166;
<A HREF="contact.html">contact</A> &#32;&#166;
```

```
<A HREF="comment.html">comment</A> &#32;&#166;
<A HREF="helpinfo.html">helpinfo</A></B><BR>
<BR>
</H5></CENTER>
ECHO
}

sub bar3 {
print <<ECHO;
<BR>
<CENTER><H5>
<BR>
<B><A HREF="/cgi-bin/msnmp/form_set">home</A> &#32;&#166;
<A HREF="back.html">back</A> &#32;&#166;
<A HREF="next.html">next</A> &#32;&#166;
<A HREF="/cgi-bin/msnmp/main">menu</A> &#32;&#166;
<A HREF="search.html#search4d">search</A> &#32;&#166;
<A HREF="contact.html">contact</A> &#32;&#166;
<A HREF="comment.html">comment</A> &#32;&#166;
<A HREF="helpinfo.html">helpinfo</A></B><BR>
<BR>
</H5></CENTER>
ECHO
}

sub top {
my($title) = @_;
chop ($date = `/bin/date '+%d-%m-%y'`);
chop ($time = `/bin/date '+%H:%M:%S'`);
print <<ECHO;
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML>
<HEAD>
<TITLE>$title</TITLE>
<LINK REL=STYLESHEET TYPE="text/css" HREF="http://lidti/snmp.styl">
</HEAD>
<BODY>
<H3>$title</H3>
Date:$date Time:$time
<HR>
ECHO
}

sub bottom {
my($text) = @_;
print <<ECHO;
<BR><BR>
<HR>
<H6><ADDRESS>
L.I.D.T.I. -
(Laboratorio de Investigacion y Desarrollo en Tecnologias
Informaticas)<BR>
Facultad de Ciencias Exactas - Universidad Nacional de Salta<BR>
<A NAME= "bottom"></A>
E-mail: <A HREF="mailto:lidti\@unsa.edu.ar">
lidti\@unsa.edu.ar</A><BR>
</ADDRESS></H6>
<BODY>
<HTML>
ECHO
}
```

```
sub links {
my($router,$community,$port,$sysName) = @_;
print <<ECHO;
<CENTER>
<BR><BR>
<IMG ALT="" SRC="/icons/mon.gif">
<A HREF="/cgi-
bin/msnmp/system.cgi?host=$router&community=$community&port=$port&sysN
ame=$sysName">
System</A>
<IMG ALT="" SRC="/icons/mon.gif">
<A HREF="/cgi-
bin/msnmp/interfaces.cgi?host=$router&community=$community&port=$port&
sysName=
$sysName">Interfaces</A>
<IMG ALT="" SRC="/icons/mon.gif">
<A HREF="/cgi-
bin/msnmp/addresses.cgi?host=$router&community=$community&port=$port&s
ysName=
$sysName">Addresses</A>
<IMG ALT="" SRC="/icons/mon.gif">
<A HREF="/cgi-
bin/msnmp/routes.cgi?host=$router&community=$community&port=
$port&sysName=$sysName">Routes Table</A>
<IMG ALT="" SRC="/icons/mon.gif">
<A HREF="/cgi-
bin/msnmp/arps.cgi?host=$router&community=$community&port=
$port&sysName=$sysName">ARP Table</A>
<BR><BR>
</CENTER>
ECHO
}

sub ReadParse {
  local (*in) = @_ if @_;
  local ($i, $key, $val);

  # Read in text
  if (&MethGet) {
    $in = $ENV{'QUERY_STRING'};
  } elsif ($ENV{'REQUEST_METHOD'} eq "POST") {
    read(STDIN,$in,$ENV{'CONTENT_LENGTH'});
  }

  @in = split(/&/,$in);

  foreach $i (0 .. $#in) {
    # Convert plus's to spaces
    $in[$i] =~ s/\+/ /g;

    # Split into key and value.
    ($key, $val) = split(/=/,$in[$i],2); # splits on the first =.

    # Convert %XX from hex numbers to alphanumeric
    $key =~ s/%(..)/pack("c",hex($1))/ge;
    $val =~ s/%(..)/pack("c",hex($1))/ge;

    # Associate key and value
    $in{$key} .= "\0" if (defined($in{$key})); # \0 is the multiple
separator
```

```
    $in{$key} .= $val;

  }

  return length($in);
}

# MethGet
# Return true if this cgi call was using the GET request, false
otherwise

sub MethGet {
  return ($ENV{'REQUEST_METHOD'} eq "GET");
}

## ether_hex (HEX_STRING)
##
## Converts a raw hex representation into the common form used in
## Ethernet addresses, e.g. "080020830069" becomes
## "08:00:20:83:00:69".
##
sub ether_hex ($) {
  my ($string) = @_;
  $string =~ s/([0-9a-f][0-9a-f])/$1:/g;
  $string =~ s/:$//;
  $string;
}

sub snmpset {
        my($host,$community,@varList) = @_;
        my(@enoid, $response, $bindings, $binding, $inoid,$outoid,
                $upoid,$oid,@retvals);
        my ($type,$value);
        while (@varList) {
                $oid   = toOID(shift @varList);
                $type  = shift @varList;
                $value = shift @varList;
                ($type eq 'string') && do {
                        $value = encode_string($value);
                        push @enoid, [$oid,$value];
                        next;
                };
                ($type eq 'int') && do {
                        $value = encode_int($value);
                        push @enoid, [$oid,$value];
                        next;
                };
                die "Unknown SNMP type: $type";
        }
        srand();
        my $session = SNMP_Session->open ($host , $community, 161);
        if ($session->set_request_response(@enoid)) {
                $response = $session->pdu_buffer;
                ($bindings) = $session->decode_get_response
($response);
                $session->close ();
                while ($bindings) {
                        ($binding,$bindings) = decode_sequence
($bindings);
                        ($oid,$value) = decode_by_template ($binding,
"%O%@");
```

```
                        my $tempo = pretty_print($value);
                        $tempo=~s/\t/ /g;
                        $tempo=~s/\n/ /g;
                        $tempo=~s/^\s+//;
                        $tempo=~s/\s+$//;
                        push @retvals,  $tempo;
                }
                return (@retvals);
        } else {
                return (-1,-1);
        }
}

#
#  Given an OID in either ASN.1 or mixed text/ASN.1 notation, return
an
#  encoded OID.
#
sub toOID {
        my $var = shift;
        if ($var =~ /^([a-z]+[^\.]*)/i) {
#     push @enoid,  encode_oid((split /\./, $snmpget::OIDS{$var}));
                my $oid = $snmp::OIDS{$1};
                if ($oid) {
                        $var =~ s/$1/$oid/;
                } else {
                        die "Unknown SNMP var $var\n"
                }
        }
        encode_oid((split /\./, $var));
}
1;
```

A.9 HOJA DE ESTILO

```
#####################################################################
#   Hoja de Etilo: snmp.styl
#####################################################################
BODY {background: url(fondo.gif)}
   {font-family:arial;font-style:normal;color:black;}
H1 {font-family:arial;font-style:normal;color:black;}
H2 {font-family:arial;font-style:normal;color:red;}
H3 {font-family:arial;font-style:normal;color:black;}
H4 {font-family:arial;font-style:normal;color:black;}
H5 {font-family:arial;font-style:normal;color:black;}
H6 {font-family:arial;font-style:normal;color:black;}
P  {font-family:arial;font-style:normal;color:black;}
A  {font-family:arial;font-style:normal;color:blue;}
B  {font-family:arial;font-style:normal;color:black;}
DT {font-family:arial;font-style:normal;color:black;}
DL {font-family:arial;font-style:normal;color:black;}
DD {font-family:arial;font-style:normal;color:black;}
TR {font-family:arial;font-style:normal;color:black;}
TD {font-family:arial;font-style:normal;color:black;}
TH {font-family:arial;font-style:normal;color:black;}
TABLE {font-family:arial;font-style:normal;color:black;}
TABLE {border: thin red}
```

A.10 CÓDIGO PHP

main.html
****

```
<html>
<div align=center>
<P><P>
<H1>SNMP Web Tools</H1>
<P>
<H2>Database Management</H2>
<H4>DBMS: PostgreSQL</H4>
</html>
```

prev1.html
****

```
<html>
<head>
<title>Presentacion1</title>
</head>
<body>
<li><a href="/cgi-bin/msnmp/main" target="_top">Main</a>
</body>
</html>
```

prev2.html
****

```
<html>
<head>
<title>Presentacion2</title>
</head>
<body>
<h3>Database Scheme</h3>
<li><a href="/cgi-bin/php/SWtools/tbl_scheme.php3?table=object"
target="pag3">Table Object</a>
<li><a href="/cgi-bin/php/SWtools/tbl_scheme.php3?table=host"
target="pag3">Table Host</a>
<li><a href="/cgi-bin/php/SWtools/tbl_scheme.php3?table=state"
target="pag3">Table State</a>
<li><a href="/cgi-bin/php/SWtools/tbl_scheme.php3?table=traps"
target="pag3">Table Traps</a>
<li><a href="/cgi-bin/php/SWtools/tbl_scheme.php3?table=admin"
target="pag3">Table Users Administration</a>
<h3>Mib Object</h3>
<li><a href="/cgi-bin/php/SWtools/sel_object.php3"
target="pag3">Select</a>
<li><a href="/cgi-bin/php/SWtools/for_object.php3"
target="pag3">Insert</a>
<li><a href="/cgi-bin/php/SWtools/up1_object.php3"
target="pag3">Update</a>
<li><a href="/cgi-bin/php/SWtools/dl1_object.php3"
target="pag3">Delete</a>
<h3>Users Administration</h3>
<li><a href="/cgi-bin/php/SWtools/sel_admin.php3"
target="pag3">Select</a>
<li><a href="/cgi-bin/php/SWtools/for_admin.php3"
target="pag3">Insert</a>
<li><a href="/cgi-bin/php/SWtools/up1_admin.php3"
target="pag3">Update</a>
```

```
<li><a href="/cgi-bin/php/SWtools/dl1_admin.php3"
target="pag3">Delete</a>
<h3>Hosts Management</h3>
<li><a href="/cgi-bin/php/SWtools/sel_host.php3"
target="pag3">Select</a>
<li><a href="/cgi-bin/php/SWtools/for_host.php3"
target="pag3">Insert</a>
<li><a href="/cgi-bin/php/SWtools/up1_host.php3"
target="pag3">Update</a>
<li><a href="/cgi-bin/php/SWtools/dl1_host.php3"
target="pag3">Delete</a>
<h3>Table State</h3>
<li><a href="/cgi-bin/php/SWtools/sl1_state.php3"
target="pag3">Select</a>
<li><a href="/cgi-bin/php/SWtools/sl1_gen.php3" target="pag3">Generic
Select</a>
<h3>Graphics</h3>
<li><a href="/cgi-bin/php/SWtools/graph_perf.php3"
target="pag3">Performance</a>
<li><a href="/cgi-bin/php/SWtools/graph_comp.php3"
target="pag3">Comparative In-Out</a>
<h3>Table Traps</h3>
<li><a href="/cgi-bin/php/SWtools/sel_traps.php3"
target="pag3">Generic Select</a>
<li><a href="/cgi-bin/php/SWtools/sl1_traps.php3"
target="pag3">Select</a>
<li><a href="/cgi-bin/php/SWtools/dl1_trap.php3"
target="pag3">Delete</a>
</body></html>
```

snmpdb.html

```
**************************************************************
<html><head>
      <meta name ="keywords"
            contens ="snmp, tools, management">
      <meta name = "description"
            contens="Snmp Web Tools">
      <meta name ="author"
            contens ="Daniel Arias Figueroa">
<title> Union </title>
<head><frameset cols ="20%,*">
      <frameset rows ="10%,*">
            <frame name ="pag1" src="prev1.html">
            <frame name ="pag2" src="prev2.html">
      </frameset>
      <frame name ="pag3" src="main.html">
</frameset>
</html>
```

header.inc

```
**************************************************************
<html><head>
<title>SNMP Web Tools - DBMS</title>
<LINK REL=STYLESHEET TYPE="text/css" HREF="http://lidti/snmp.styl">
```

footer.inc

```
**************************************************************
<DIV ALIGN=left>
```

```
<P><P>
<HR>
<H6><ADDRESS>
L.I.D.T.I. -
(Laboratorio de Investigacion y Desarrollo en Tecnologias
Informaticas)<BR>
Facultad de Ciencias Exactas - Universidad Nacional de Salta<BR>
<A NAME= "bottom"></A>
E-mail: <A HREF="mailto:lidti\@unsa.edu.ar"> lidti@unsa.edu.ar</A><BR>
</ADDRESS></H6>
<BODY>
<HTML>
```

sel_host.php3

```
**************************************************************
<?PHP
require("header.inc");
?>
<DIV ALIGN="center">
<H3>View Host Records</H3>
<TABLE BORDER="1">
<TR>
<TH>Host Id</TH>
<TH>IP</TH>
<TH>Community</TH>
<TH>Port</TH>
<TH>Monitor Time</TH>
<TH>Average Time</TH>
<TH>Object Id In</TH>
<TH>Object Id Out</TH>
<TH>Monitor</TH>
</TR>
<?PHP
        $conn = pg_Connect("localhost", "5432", "", "", "snmpdb");
        if (!$conn) {
                echo "Connect - An error occurred.\n";
                exit;
        }
        $result = pg_Exec($conn,
                "SELECT *
                        FROM host
                        ORDER BY h_id;");
        if (!$result) {
            echo "</TABLE>";
                echo "Select - An error occurred.\n";
                exit;
        }
        $num = pg_NumRows($result);
        $i = 0;
        while ($i < $num) {
                echo "<TR><TD>";
                echo pg_Result($result, $i, "h_id");
                echo "</TD><TD>";
                echo pg_Result($result, $i, "ip");
                echo "</TD><TD>";
                echo pg_Result($result, $i, "community");
                echo "</TD><TD>";
                echo pg_Result($result, $i, "port");
                echo "</TD><TD>";
                echo pg_Result($result, $i, "m_time");
```

```
                echo "</TD><TD>";
                echo pg_Result($result, $i, "a_time");
                echo "</TD><TD>";
                echo pg_Result($result, $i, "o_idi");
                echo "</TD><TD>";
                echo pg_Result($result, $i, "o_ido");
                echo "</TD><TD>";
                echo pg_Result($result, $i, "monitor");
                echo "</TD><TR>";
                $i++;
        }
        pg_FreeResult($result);
        pg_Close($conn);
?>
</TABLE>
<?PHP
require("footer.inc");
?>
```

for_host.php3

```
**************************************************************
<?PHP
require("header.inc");
?>
<SCRIPT LANGUAGE="JavaScript">
var v=0;

function reference()
{

document.formu.o_idi.value=document.formu.o_ref.options[document.formu
.o_ref.selectedIndex].value
}
function reference2()
{

document.formu.o_ido.value=document.formu.o_ref2.options[document.form
u.o_ref2.selectedIndex].value
}
</SCRIPT>
</HEAD>
<BODY>
<DIV ALIGN="center">
<H3>Insert Host Record</H3>
<FORM NAME=formu METHOD="post" ACTION="/cgi-
bin/php/SWtools/ins_host.php3">
<TABLE BORDER="1">
<TR><TD>New Host Id:</TD>
<TD><INPUT NAME="h_id" SIZE=5></TR>
<TR><TD>IP:</TD>
<TD><INPUT NAME="ip" SIZE=15></TR>
<TR><TD>Community:</TD>
<TD><INPUT NAME="community" SIZE=15 VALUE="public"></TR>
<TR><TD>Port:</TD>
<TD><INPUT NAME="port" SIZE=7 VALUE="161"></TR>
<TR><TD>Monitor Time:</TD>
<TD><INPUT NAME="m_time" SIZE=5 VALUE=5></TR>
<TR><TD>Average Time:</TD>
<TD><INPUT NAME="a_time" SIZE=5 VALUE=30></TR>
<TR><TD>Object Reference:</TD>
```

```
<TD><SELECT NAME="o_ref" SIZE="1" onChange="reference()">
<?PHP
        $conn = pg_Connect("localhost", "5432", "", "", "snmpdb");
        if (!$conn) {
            echo "</TABLE>";
                echo "Connect - An error occurred.\n";
                exit;
        }
        $result = pg_Exec($conn,
                "SELECT * FROM object ORDER BY o_id;");
        if (!$result) {
            echo "</TABLE>";
                echo "Select - An error occurred.\n";
                exit;
        }
        $num = pg_NumRows($result);
        $i = 0;
        while ($i < $num) {
                printf("<OPTION VALUE=
%s>",pg_Result($result,$i,"o_id"));
                printf(" %s",pg_Result($result, $i, "o_id"));
                printf(" %s </OPTION>",pg_Result($result, $i,
"o_name"));
                $i++;
        }
?>
</SELECT>
<TR><TD>Object Id In:</TD>
<TD><INPUT NAME="o_idi" SIZE=40></TR>
<TR><TD>Object Reference:</TD>
<TD><SELECT NAME="o_ref2" SIZE="1" onChange="reference2()">
<?PHP
        $num = pg_NumRows($result);
        $i = 0;
        while ($i < $num) {
                printf("<OPTION VALUE=
%s>",pg_Result($result,$i,"o_id"));
                printf(" %s",pg_Result($result, $i, "o_id"));
                printf(" %s </OPTION>",pg_Result($result, $i,
"o_name"));
                $i++;
        }
        pg_FreeResult($result);
        pg_Close($conn);
?>
</SELECT>
<TR><TD>Object Id Out:</TD>
<TD><INPUT NAME="o_ido" SIZE=40></TR>
<TR><TD>Monitor:</TD>
<TD><SELECT NAME="monitor" SIZE="1">
<OPTION>enable
<OPTION>disable
</SELECT>
</TABLE><P>
<INPUT TYPE="submit">
<INPUT TYPE="reset">
</FORM>
<?PHP
require("footer.inc");
?>
```

ins_host.php3

```
*************************************************************
<?PHP
require("header.inc");
?>
<DIV ALIGN="center">
<H3>Insert Host Record</H3>
<?PHP
        $conn = pg_Connect("localhost", "5432", "", "", "snmpdb");
        if (!$conn) {
                echo "Connect - An error occurred.\n";
                exit;
        }
        pg_Exec($conn,
             "INSERT INTO host
                VALUES ($h_id,'$ip','$community',

$port,$m_time,$a_time,'$o_idi','$o_ido','$monitor');");
        if (!$conn) {
                echo "Insert - An error occurred.\n";
                exit;
        }
      else {
                echo "Database Updated !\n";
        }
        pg_Close($conn);
?>
<?PHP
require("footer.inc");
?>
```

up1_host.php3

```
*************************************************************
<?PHP
require("header.inc");
?>
<DIV ALIGN="center">
<H3>Update Host Record</H3>
<FORM METHOD="post" ACTION="/cgi-bin/php/SWtools/up2_host.php3">
Host-Id  IP  Community      Object-Id-In      Object-Id-Out
<PRE>
<SELECT NAME="h_id" SIZE=5>
<?PHP
        $conn = pg_Connect("localhost", "5432", "", "", "snmpdb");
        if (!$conn) {
                echo "Connect - An error occurred.\n";
                exit;
        }
        $result = pg_Exec($conn,
                "SELECT *
                        FROM host
                        ORDER BY ip;");
        if (!$result) {
                echo "Select - An error occurred.\n";
                exit;
        }
        $num = pg_NumRows($result);
        $i = 0;
        while ($i < $num) {
```

```
                printf("<OPTION VALUE=
%s>",pg_Result($result,$i,"h_id"));
                printf("%s  ",pg_Result($result, $i, "h_id"));
                printf("%s  ",pg_Result($result, $i, "ip"));
                printf("%s  ",pg_Result($result, $i, "community"));
                printf("%s  ",trim(pg_Result($result, $i, "o_idi")));
                printf("%s  </OPTION>\n",trim(pg_Result($result, $i,
"o_ido")));
                $i++;
        }
        pg_FreeResult($result);
        pg_Close($conn);
?>
</PRE>
</SELECT>
<P>
<INPUT TYPE="submit">
<INPUT TYPE="reset">
</FORM>
<?PHP
require("footer.inc");
?>
```

up2_host.php3

```
**************************************************************
<?PHP
require("header.inc");
?>
<DIV ALIGN="center">
<h3>Update Host Record</h3>
<FORM METHOD="post" ACTION="/cgi-bin/php/SWtools/up3_host.php3">
<TABLE BORDER="1">
<TR>
<TD>Host Id:</TD>
<?PHP
        if (!$h_id) {
                echo "No selected record.\n";
                exit;
        }
        $conn = pg_Connect("localhost", "5432", "", "", "snmpdb");
        if (!$conn) {
                echo "Connect - An error occurred.\n";
                exit;
        }
        $result = pg_Exec($conn,
                "SELECT *
                        FROM host WHERE h_id = $h_id;");
        if (!$result) {
            echo "</TABLE>";
                echo "Select - An error occurred.\n";
                exit;
        }
        $num = pg_NumRows($result);
        echo "<TD><INPUT NAME=h_id SIZE=5 VALUE =";
        echo pg_Result($result,0,"h_id"),"></TR>";
        echo "<TR><TD>IP:</TD>";
        echo "<TD><INPUT NAME=ip SIZE=15 VALUE=";
        echo pg_Result($result,0,"ip"),"></TR>";
        echo "<TR><TD>Community:</TD>";
        echo "<TD><INPUT NAME=community SIZE=15 VALUE=";
```

```
        echo pg_Result($result,0,"community"),"></TR>";
        echo "<TR><TD>Port:</TD>";
        echo "<TD><INPUT NAME=port SIZE=7 VALUE=";
        echo pg_Result($result,0,"port"),"></TR>";
        echo "<TR><TD>Monitor Time:</TD>";
        echo "<TD><INPUT NAME=m_time SIZE=5 VALUE=";
        echo pg_Result($result,0,"m_time"),"></TR>";
        echo "<TR><TD>Average Time:</TD>";
        echo "<TD><INPUT NAME=a_time SIZE=5 VALUE=";
        echo pg_Result($result,0,"a_time"),"></TR>";
        echo "<TR><TD>Object Id In:</TD>";
        echo "<TD><INPUT NAME=o_idi SIZE=40 VALUE=";
        echo pg_Result($result,0,"o_idi"),"></TR>";
        echo "<TR><TD>Object Id Out:</TD>";
        echo "<TD><INPUT NAME=o_ido SIZE=40 VALUE=";
        echo pg_Result($result,0,"o_idi"),"></TR>";
        echo "<TR><TD>Monitor:</TD>";
        echo "<TD><SELECT NAME=monitor SIZE=1>";
        echo "<OPTION>enable";
        echo "<OPTION>disable";
        echo "</SELECT></TR>";
        pg_FreeResult($result);
        pg_Close($conn);
?>
</TABLE>
<P>
<INPUT TYPE="submit">
<INPUT TYPE="reset">
</FORM>
<?PHP
require("footer.inc");
?>
```

up3_host.php3

```
****************************************************************
<?PHP
require("header.inc");
?>
<DIV ALIGN="center">
<h3>Update Host Record</h3>
<?PHP
        $conn = pg_Connect("localhost", "5432", "", "", "snmpdb");
        if (!$conn) {
                echo "Connect - An error occurred.\n";
                exit;
        }
        pg_Exec($conn,
                "UPDATE host SET ip = '$ip', community = '$community',
            port = $port, m_time = $m_time, a_time = $a_time,
            o_idi = '$o_idi', o_ido = '$o_ido', monitor = '$monitor'
                        WHERE h_id = $h_id");
        if (!$conn) {
                echo "Update - An error occurred.\n";
                exit; }
      else {
                echo "Database Updated !\n";
        }
        pg_Close($conn);
?>
<?PHP
```

```
require("footer.inc");
?>
```

dl1_host.php3

```
**************************************************************
<?PHP
require("header.inc");
?>
<DIV ALIGN="center">
<h3>Delete Host Record</h3>
<FORM METHOD="post" ACTION="/cgi-bin/php/SWtools/dl2_host.php3">
Host-Id  IP  Community      Object-Id-In      Object-Id-Out
<PRE>
<SELECT NAME="h_id" SIZE=5>
<?PHP
        $conn = pg_Connect("localhost", "5432", "", "", "snmpdb");
        if (!$conn) {
                echo "Connect - An error occurred.\n";
                exit;
        }
        $result = pg_Exec($conn,
                "SELECT *
                        FROM host
                        ORDER BY ip;");
        if (!$result) {
                echo "Select - An error occurred.\n";
                exit;
        }
        $num = pg_NumRows($result);
        $i = 0;
        while ($i < $num) {
                printf("<OPTION VALUE=
%s>",pg_Result($result,$i,"h_id"));
                printf("%s  ",pg_Result($result, $i, "h_id"));
                printf("%s  ",pg_Result($result, $i, "ip"));
                printf("%s  ",pg_Result($result, $i, "community"));
                printf("%s  ",trim(pg_Result($result, $i, "o_idi")));
                printf("%s  </OPTION>\n",trim(pg_Result($result, $i,
"o_ido")));
                $i++;
        }
        pg_FreeResult($result);
        pg_Close($conn);
?>
</PRE>
</SELECT>
<P>
<INPUT TYPE="submit">
<INPUT TYPE="reset">
</FORM>
<?PHP
require("footer.inc");
?>
```

dl2_host.php3

```
**************************************************************
<?PHP
require("header.inc");
?>
```

```
<DIV ALIGN="center">
<H3>Delete Host Records</H3>
<?PHP
        if (!$h_id) {
                echo "No selected record.\n";
                exit;
        }
        $conn = pg_Connect("localhost", "5432", "", "", "snmpdb");
        if (!$conn) {
                echo "Connect - An error occurred.\n";
                exit;
        }
        pg_Exec($conn,
                "DELETE FROM host
                        WHERE h_id = $h_id");
        if (!$conn) {
                echo "Delete - An error occurred.\n";
                exit;
        }
      else {
                echo "Database Updated !\n";
      }
        pg_Close($conn);
?>
<?PHP
require("footer.inc");
?>
```

sl1_state.php3

```
**************************************************************
<?PHP
require("header.inc");
?>
<DIV ALIGN="center">
<H3>Query State Table</H3>
<FORM METHOD="post" ACTION="/cgi-bin/php/SWtools/sl2_state.php3">
<TABLE BORDER="0"><TR>
<TD>Input body of the "where" Clause:</TD>
<TD><INPUT NAME="where" SIZE="50"></TR>
</TABLE>
<?PHP
?><P>
<INPUT TYPE="submit">
<INPUT TYPE="reset">
</FORM>
<DIV ALIGN="left">
Examples:
<li>state.h_id = xx</li>
<li>date > 'dd-mm-yy'</li>
<li>time = 'hh:mm'</li>
<li>state.h_id = xx and date > 'dd-mm-yy'</li>
<li>state.h_id = xx and date <= 'dd-mm-yy' and time = 'hh:mm'</li>
<?PHP
require("footer.inc");
?>
```

sl2_state.php3

```
**************************************************************
<?PHP
```

```
require("header.inc");
?>
<DIV ALIGN="center">
<h3>View State Records</h3>
<TABLE BORDER="1"><TR>
<TH>Host Id</TH>
<TH>IP</TH>
<TH>Date</TH>
<TH>Time</TH>
<TH>Object Value In</TH>
<TH>Object Value Out</TH></TR>
<?PHP
        $conn = pg_Connect("localhost", "5432", "", "", "snmpdb");
        if (!$conn) {
            echo "</TABLE>";
                echo "Connect - An error occurred.\n";
                exit;
        }
      if ($where) {
            $where = sprintf("and %s",$where);
      }
        $result = pg_Exec($conn,
                "SELECT host.h_id,ip,date,time,o_vali,o_valo
                        FROM host,state WHERE host.h_id = state.h_id
$where
                        ORDER BY h_id,date,time;");
        if (!$result) {
            echo "</TABLE>";
                echo "Select - An error occurred.\n";
                exit;
        }
        $num = pg_NumRows($result);
        $i = 0;
        while ($i < $num) {
                echo "<TR><TD>";
                echo pg_Result($result, $i, 0);
                echo "</TD><TD>";
                echo pg_Result($result, $i, "ip");
                echo "</TD><TD>";
                echo pg_Result($result, $i, "date");
                echo "</TD><TD>";
                echo pg_Result($result, $i, "time");
                echo "</TD><TD>";
                echo pg_Result($result, $i, "o_vali");
                echo "</TD><TD>";
                echo pg_Result($result, $i, "o_valo");
                echo "</TD><TR>";
                $i++;
        }
        pg_FreeResult($result);
        pg_Close($conn);
?>
</TABLE>
<?PHP
require("footer.inc");
?>
```

graph_perf.php3

```
************************************************************
<?PHP
```

```
require("header.inc");
?>
<DIV ALIGN="center">
<h3>Graphic Date Records</h3>
<FORM METHOD="post" ACTION="/cgi-bin/php/SWtools/graph_perf2.php3">
Host-Id  IP  Community      Object-Id-In      Object-Id-Out
<PRE>
<SELECT NAME="h_id" SIZE=5>
<?PHP
        $conn = pg_Connect("localhost", "5432", "", "", "snmpdb");
        if (!$conn) {
                echo "Connect - An error occurred.\n";
                exit;
        }
        $result = pg_Exec($conn,
                "SELECT *
                        FROM host
                        ORDER BY ip;");
        if (!$result) {
                echo "Select - An error occurred.\n";
                exit;
        }
        $num = pg_NumRows($result);
        $i = 0;
        while ($i < $num) {
                printf("<OPTION VALUE=
%s>",pg_Result($result,$i,"h_id"));
                printf("%s  ",pg_Result($result, $i, "h_id"));
                printf("%s  ",pg_Result($result, $i, "ip"));
                printf("%s  ",pg_Result($result, $i, "community"));
                printf("%s  ",trim(pg_Result($result, $i, "o_idi")));
                printf("%s  </OPTION>\n",trim(pg_Result($result, $i,
"o_ido")));
                $i++;
        }
        pg_FreeResult($result);
        pg_Close($conn);
?>
</SELECT></PRE><P>
Input Date:<INPUT NAME="date" SIZE="15"><P>
Object In <INPUT TYPE="radio" NAME="oval" VALUE="o_vali">
Object Out<INPUT TYPE="radio" NAME="oval" VALUE="o_valo"><P>
<INPUT TYPE="submit">
<INPUT TYPE="reset">
</FORM>
<?PHP
require("footer.inc");
?>
```

graph_perf2.php3

```
*****************************************************************
<?PHP
        $out= "<DIV ALIGN=center><H3>Graphic Performance</H3>";

        if (!$date or !$h_id) {
                echo $out;
                echo "No selected records !";
                exit;
        }
        $conn = pg_Connect("localhost", "5432", "", "", "snmpdb");
```

```
  if (!$conn) {
          echo $out;
          echo "Connect - An error occurred.\n";
          exit;
  }
  $result = pg_Exec($conn,
        "SELECT * FROM state
          WHERE h_id = $h_id and date = '$date'
      ORDER BY time ;");
  if (!$result) {
          echo $out;
          echo "Select - An error occurred.\n";
          exit;
  }
  $num = pg_NumRows($result);
  if (!$num) {
          echo "<DIV ALIGN=center>";
          echo "<H3>Graphic Performance</H3>";
          echo "No records !";
          exit;
  }
  if ($oval == "o_vali") {
     $result2 = pg_Exec($conn,
        "SELECT min(o_vali), max(o_vali) FROM state
          WHERE h_id = $h_id and date = '$date' ;");}
else {
     $result2 = pg_Exec($conn,
        "SELECT min(o_valo), max(o_valo) FROM state
          WHERE h_id = $h_id and date = '$date' ;");}
  $min=pg_Result($result2, 0,0);
  $max=pg_Result($result2, 0,1);
$datestr= sprintf('Date:%s',$date);
$x=700; $y=300;
$x1=70; $y1=50;
$x2=$x-$x1; $y2=$y-$y1;
$k = ($y2-$y1)/($max-$min);
$z = ($x2-$x1)/$num;
  Header("Content-type: image/gif");
  header("Content-type: image/gif");
  $im = imagecreate($x,$y);
$white = ImageColorAllocate($im,255,255,255);
$green = ImageColorAllocate($im,0,255,0);
$blue  = ImageColorAllocate($im,0,0,255);
$black = ImageColorAllocate($im,0,0,0);
$gray  = ImageColorAllocate($im,192,192,192);
ImageFilledRectangle($im,$x1,$y1,$x2,$y2,$gray);
ImageString($im,4,$x1,$y2+20,$value,$black);
  if ($oval == "o_vali") {
  ImageString($im,2,$x1,$y2+35,'Object Value In:',$black);
  ImageString($im,2,$x1+100,$y2+35,'Green',$green); }
else {
  ImageString($im,2,$x1,$y2+35,'Object Value Out:',$black);
  ImageString($im,2,$x1+105,$y2+35,'Blue',$blue); }
ImageString($im,5,250,10,'Graphic Performance',$black);
ImageString($im,4,0,$y1-25,$datestr,$black);
ImageString($im,5,$x2,$y2,'Sample',$black);
ImageString($im,4,0,$y1,$max,$black);
ImageString($im,4,0,$y2-15,$min,$black);
  $i = 0; $inter=$z;$xx=0;
  if ($oval == "o_vali") {
    while ($i < $num) {
```

```
            $val = pg_Result($result, $i, "o_vali");
          ImageLine($im,$x1+$xx,$y2,$x1+$xx,$y2-($val-
$min)*$k,$green);
          $xx=$xx+$inter;
          $i++;
        }}
      else {
          while ($i < $num) {
            $val = pg_Result($result, $i, "o_valo");
          ImageLine($im,$x1+$xx,$y2,$x1+$xx,$y2-($val-$min)*$k,$blue);
          $xx=$xx+$inter;
          $i++;
        }
      }
        ImageGif($im);
        pg_FreeResult($result);
        pg_FreeResult($result2);
        pg_Close($conn);
        ImageDestroy($im);
?>
```

graph_util2.php3

```
**************************************************************
<?PHP
        $out= "<DIV ALIGN=center><H3>Graphic Utilization</H3>";

        if (!$date or !$h_id) {
                echo $out;
                echo "No selected records !";
                exit;
        }
        $conn = pg_Connect("localhost", "5432", "", "", "snmpdb");
        if (!$conn) {
                echo $out;
                echo "Connect - An error occurred.\n";
                exit;
        }
        $result1 = pg_Exec($conn,
              "SELECT * FROM host
                WHERE h_id = $h_id;");
        if (!$result1) {
                echo $out;
                echo "Select - An error occurred.\n";
                exit;
        }
        $sec = pg_Result($result1, $i, "m_time");
        $result = pg_Exec($conn,
              "SELECT * FROM state
                WHERE h_id = $h_id and date = '$date'
            ORDER BY time;");
        if (!$result) {
                echo $out;
                echo "Select - An error occurred.\n";
                exit;
        }
        $num = pg_NumRows($result);
        if (!$num) {
                echo "<DIV ALIGN=center>";
                echo "<H3>Graphic Utilization</H3>";
                echo "No records !";
```

```
                exit;
          }
        $datestr= sprintf('Date:%s  Rows:%s',$date,$num);
        $x=700; $y=300;
        $x1=63; $y1=50;
        $x2=$x-$x1; $y2=$y-$y1;
        $z = ($x2 - $x1)/($num-1);
          Header("Content-type: image/gif");
          $im = imagecreate($x,$y);
        $white = ImageColorAllocate($im,255,255,255);
        $green = ImageColorAllocate($im,0,255,0);
        $blue  = ImageColorAllocate($im,0,0,255);
        $black = ImageColorAllocate($im,0,0,0);
        $gray  = ImageColorAllocate($im,192,192,192);
        ImageFilledRectangle($im,$x1,$y1,$x2,$y2,$gray);
        ImageString($im,4,$x1,$y2+20,$value,$black);
        ImageString($im,2,$x1,$y2+25,'Utilization In:',$black);
        ImageString($im,2,$x1+100,$y2+25,'Green',$green);
        ImageString($im,2,$x1,$y2+35,'Utilization Out:',$black);
        ImageString($im,2,$x1+105,$y2+35,'Blue',$blue);
        ImageString($im,5,250,10,'Graphic Utilization',$black);
        ImageString($im,4,0,$y1-25,$datestr,$black);
        ImageString($im,5,$x2,$y2,'Time',$black);
        ImageString($im,4,0,$y1,'% 100',$black);
        ImageString($im,4,0,$y2-15,'% 0',$black);
          $i = 0; $inter=$z;$xx=$z;
          $valix = pg_Result($result, $i, "o_vali");
          $valox = pg_Result($result, $i, "o_valo");
        $i++;$xia=$x1;$yia=$y2;
        $xoa=$x1;$yoa=$y2;
        $upixel = ($y2-$y1)/100;
        ImageLine($im,$x1,$y2,$x1,$y1,$black);
          while ($i < $num) {
              $valiy = pg_Result($result, $i, "o_vali");
            $dif1 = $valiy - $valix;
            if($dif1 < 0) { $dif1=0; }
            $utili = ((($dif1)*8 / 300) / $ifspeed)*100;
              $valix = $valiy;
            $y3 = $y2 - ($utili * $upixel);
            ImageLine($im,$xia,$yia,$x1+$xx,$y3,$green);
            ImageLine($im,$x1+$xx,$y2,$x1+$xx,$y1,$black);
              $xia = $x1+$xx; $yia = $y3;
              $valoy = pg_Result($result, $i, "o_valo");
            $dif2 = $valoy - $valox;
            if($dif2 < 0) { $dif2=0; }
            $utilo = ((($dif2)*8 / 300) / $ifspeed)*100;
              $valox = $valoy;
            $y3 = $y2 - ($utilo * $upixel);
            ImageLine($im,$xoa,$yoa,$x1+$xx,$y3,$blue);
              $xoa = $x1+$xx; $yoa = $y3;
            $xx=$xx+$inter;
            $i++;
        }
          ImageGif($im);
          pg_FreeResult($result);
          pg_FreeResult($result1);
          pg_Close($conn);
          ImageDestroy($im);
?>
```

tbl_scheme.php3

****

```
<?PHP
require("header.inc");
?>
<div align="center">
<h3>Database Schema</h3>
Table:
<?
echo "<b>$table</b><p>";
$link = pg_Connect("localhost", "5432", "", "", "snmpdb");
if (!$link) {
   echo "Connect - An error occurred.\n";
   exit;
}
$result = pg_exec($link,"select * from $table");
$num_of_fields = pg_numfields($result);
if (!$result)
   {
   pg_die();
   }
else
   {
   ?>
   <table border=1>
   <tr>
   <th>Field</th>
   <th>Type</th>
<!--
   <th>Null</th>
   <th>Default</th>
-->

   <th>Extra</th>
   </tr>
   <?
   $i = 0;
   while ($i < $num_of_fields)
         {
       $query = "db=$db&table=$table";

      $extra = pg_exec($link, "SELECT ic.relname
             FROM pg_class bc,
                  pg_class ic,
                  pg_index i,
                  pg_attribute a
             WHERE
      not bc.relname~'pg_.*'
      and i.indrelid = bc.oid
                and i.indexrelid = ic.oid
                and i.indkey[0] = a.attnum
                and a.attrelid = bc.oid
                and i.indproc = '0'::oid");
       if (pg_numrows($extra) < 1) {
            $Extra ="";
       } else {
            $extra = pg_fetch_row($extra, 0);
            $Extra = $extra[0];
       }
       $indexed = 0;
       $field = pg_fieldname($result, $i);
```

```
         $type = pg_fieldtype($result, $i);
        if (substr($type, 0, 1) == "_") {
                $type = substr($type, 1, strlen($type)). "[]";
        }
       $i++;
       ?>
         <tr>
         <td><? echo $field;?> </td>
         <td><? echo $type;?> </td>

      <!--
      <td><? if ($row[Null] == "") { echo "NO";} else {echo
$row[Null];}?> </td>
         <td><? echo $row["Default"];?> </td>
      -->

         <td><? if ($Extra == $field) {
                  echo "indexed";
                  $indexed = 1;
            } ?> </td>

         </tr>
         <?
         }
   ?>
   </table>
   <?
   }
?>
<table border=1>
<?
/*
$result = mysql_db_query($db, "SHOW KEYS FROM $table");
if (!$result)
   {
   pg_die();
   }
else
   {
   for ($i=0 ; $i<pg_numrows($result); $i++)
       {
       $row = mysql_fetch_array($result);
       echo "<tr>";
       if ($row[Key_name] == "PRIMARY")
          {
          $sql_query = urlencode("ALTER TABLE $table DROP PRIMARY
KEY");
          $zero_rows = urlencode("Primary key has been dropped.");
          }
       else
          {
          $sql_query = urlencode("ALTER TABLE $table DROP INDEX
$row[Key_name]");
          $zero_rows = urlencode("Index $row[Key_name] has been
dropped.");
          }
       echo "<td>$row[Key_name]</td><td>$row[Column_name]</td><td><a
href=sql.php3?$query&sql_query=$sql_query&zero_rows=$zero_rows>Drop</a
></td>";
       echo "</tr>";
       }
```

```
   }
*/
?>

</table>
<?PHP
require("footer.inc");
?>
```

REFERENCIAS

[Leinwand] Allan Leinwand, Karen Fang Conroy - Network Management – A Practical Perspective – 2nd Edition - Addison Wesley

[Black] Uyless Black - Network Management Standards – Second Edition – McGraw-Hill

[Butzen] Fred Butzen, Dorothy Forbes - The Linux Database – MIS:Press

[Husain] Kamram Husain – Perl 5 al Descubierto – Editorial Prentice Hall

[Herrmann] Eric Herrmann – CGI Programming with Perl 5 – Sams Net

[Bernaus] Albert Bernaus, Jaime Blanco – Diseño y Programación para Internet – Infor Book Ediciones.

[Kasteleijn] Wilco Kasteleijn – Web based Management – University of Twente – Departament of Computer Science. 1997;

[Tanenbaum] Andrew S. Tanenbaum -Redes de Computadoras – Tercera Edición.

[Linux Suse] Howtos y Documentación del Sistema Operativo Linux Suse 5.3, 6.0 y 6.1.

[Comer] Stevens Comer – Internetworking with TCP/IP Volume I,II,II. - Prentice Hall

[Luotonen] Ari Luotonen – Web Proxy Servers – Prentice Hall -1998

[Arias-Díaz] Daniel Arias Figueroa – Javier Díaz – Herramientas Web de administración a través del Web – Paper presentado en el Congreso ICIE'99 – UBA – Julio de 1999. L.I.D.T.I. – L.I.N.T.I.

[Berners-Lee] T. Berners-Lee, R. Fielding, H. Frystyk, RFC 1945: Hypertext Transfer Protocol – HTTP/1.0, MIT/LCS, May 1996;

[Netscape] Netscape Comunications, The Javascript Authoring guide, http:/home.netscape.com/eng/mozilla/2.02/hadbook/javascript, 1995-1996;

[Berners-Lee] T. Berners-Lee, D. Connolly, RFC 1866: Hypertext Markup Language – 2.0, MIT/W3C, 1995

[Freier,Karlton] A.O. Freier, P. Karlton, P.C. Kocher, Internet Draft: He SSL Protocol Versión 3.0, Netscape Communications, 1996.

[Netscape] Netscape Comunications, An exploration of dynamic documents, Netscape online Documentation, http://www1.netscape.com/assist/net_sites/pushpull.html, 1997;

[Sun] Sun Microsistems, Java Remote Method Invocation Specification,http://www.javasoft.com:80/products/jdk/1.1/docs/guide/rmi/spec/rmiTOC.doc.html, 1996-1997;

[Newman] M. Newman, getting started with Tkined, http://wwwsnmp.cs.utwente.nl/~schoenw/scotty/docs/getstart.html, University of Twente, Enschede – NL, Departament of Computer Science, 1997.

[Apache Group] He Apache Group, Information on the Apache HTTP Server Project, http://www.apache.org/info.html;

[Fielding...] R.Fielding, J. Gettys,J. Mogul, H. Frystyk, T. Berners-Lee, RFC 2068: Hypertext Trasfer Protocol – HTTP/1.1, UC Irvine, DEC, MIT/LCS, 1997;