ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"Implementación del Monitoreo Web de una red WAN, integrando un sistema de alarmas SMTP, basado en SNMPv2"

TESINA DE SEMINARIO

Previa a la obtención del título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:

Evelyn Emperatriz Carrasco Echeverría

José Andrés Castillo Macías

GUAYAQUIL – ECUADOR

AÑO 2014

AGRADECIMIENTO

A Dios, por su compañía y guía a lo largo de la carrera, siendo quien nos ha brindado una vida llena de oportunidades, crecimiento y felicidad.

A nuestro director, Ing. Washington Medina por su valiosa aportación en el desarrollo de nuestro proyecto de graduación.

A nuestro delegado, Ph. D. Freddy Villao por sus enseñanzas y consejos motivándonos a alcanzar nuevas metas.

DEDICATORIA

Este trabajo está dedicado a Dios, por su amor, bendiciones y fuerzas en cada momento de mi vida. A mis padres por ser mis pilares fundamentales, por su incondicional apoyo y motivación a ser cada día mejor. A mis hermanas, por estar presente y apoyarme en mi crecimiento y a mis amigos por su ayuda incondicional y compañía en todo momento.

Evelyn Carrasco E.

Este logro tiene una dedicación especial a Dios por haberme dado las fuerzas necesarias para seguir adelante y culminar cada etapa de mi vida, a mis padres por brindarme su apoyo en todo momento en especial en los momentos más difíciles, a mis hermanos y amigos que intervinieron para que este día pueda celebrar este acontecimiento tan importante.

José Castillo M.

TRIBUNAL DE SUSTENTACIÓN

_____

M.Sc. Washington Medina M.

PROFESOR DEL SEMINARIO DE GRADUACIÓN

_____

PH. D. Freddy Villao Quezada

PROFESOR DELEGADO POR LA UNIDAD ACADÉMICA

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de esta Tesina, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL".

(Reglamento de exámenes y títulos profesionales de la ESPOL)

_____

Evelyn Emperatriz Carrasco Echeverría

_____

José Andrés Castillo Macías

RESUMEN

El presente proyecto consiste en el monitoreo Web de una red WAN, integrando un sistema de alarmas SMTP, basado en SNMPv2; es decir, mediante la herramienta de monitoreo en la web observaremos remotamente el comportamiento de nuestro sistema en cualquier momento, al igual que garantizar el óptimo funcionamiento de los equipos de red y del sistema en general.

En el primer capítulo se especifica de manera clara y detallada el proyecto a realizar, enfocando los puntos de investigación que nos condujo al desarrollo del proyecto, los objetivos y las limitaciones con las que nos encontraremos.

En el segundo capítulo se tiene el fundamento teórico detallando las generalidades de SNMPv2, así como el lenguaje PERL para el monitoreo web, servidores web y aplicaciones del lado del servidor.

En el capítulo tres se desarrolla la aplicación para la generación de las alarmas SMTP, se explica detalladamente la configuración del código fuente y los scripts que se ejecutarán para el funcionamiento de la aplicación web.

En el capítulo cuatro se crea la base de datos con la que se va a trabajar en el proyecto, se levanta y configura el servidor web y se detalla de manera minuciosa el diseño de la aplicación.

En el capítulo cinco se observarán los resultados del monitoreo, ejecutando los scripts creados, se detallan todos los escenarios posibles del comportamiento de la red y se analizará las alarmas enviadas al administrador de red.

ÍNDICE GENERAL

ABREVIATURAS

SNMPv1	Simple Network Management Protocol version 1
SNMPv2	Simple Network Management Protocol version 2
SNMPv3	Simple Network Management Protocol version 3
PDU	Protocol Data Unit
NMS	Network Management Systems
MIB	Base de información de administración
MD	Managed Device
CRLF	Retorno de carro – Salto de línea
SMTP	Simple Mail Transfer Protocol
WAN	Wide Area Network
LAN	Local Area Network
DBI	Database Independent Interface
SQL	Structured Query Language
TCP	Transmission Control Protocol
OID	Object Indentifier
SMI	Structure of Managed Information
SW	Software

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Tiempo atrás, los sistemas de red eran esquematizados de manera sencilla y con pequeña organización, el cual si ocurría un daño no era necesario que los administradores realicen cambios en ese momento, debido a que su corrección no era crítica. El monitoreo web era una idea lejana de aplicación en las redes y lo tenían únicamente las grandes empresas de Telecomunicaciones.

Los encargados de gestionar los sistemas de redes en las organizaciones, pierden monitoreo de los equipos al retirarse de las oficinas. Es en esta ausencia que se pueden presentar inconvenientes en los elementos de red.

La poca información del procesamiento a tiempo real de los equipos disminuye el conocimiento de su estado, aumenta el tiempo de respuesta de solución y se pierde continuidad del monitoreo.

Debido al crecimiento y desarrollo de las redes en la actualidad es necesario conocer el estado de nuestro sistema en todo momento. El monitoreo proactivo permite tener control a tiempo real de detalles de la red y sus equipos, ya que detecta los errores que se pueden presentar y ayuda a que se puedan corregir evitando futuros conflictos y daños costosos.

Este monitoreo se basa en inteligencia de equipos de red, que permite anticiparse a los problemas, resolverlos y de esta manera mejorar notablemente la imagen y los servicios.

Para el presente proyecto se implementará un sistema de monitoreo en la que tendremos acceso vía web y visualizaremos el estado de los componentes de nuestra red de tres estaciones que simularán tres ciudades, Guayaquil como punto matriz, Portoviejo y Machala como sucursales.

CAPÍTULO 1

1. DESCRIPCION GENERAL DEL PROYECTO

1.1 INTRODUCCIÓN

Para una empresa, mantener sus equipos de red en correcto funcionamiento día a día se ha vuelto una necesidad. La detección a tiempo de las incidencias que se pueden presentar se ha vuelto prioridad para poder brindar excelente servicio a sus clientes.

Con el avance de la tecnología se vuelve importante la administración de la red mediante monitoreo proactivo que notifique falencias que se presenten y permita la gestión de los equipos mediante herramientas que ayuden a mantener la información organizada, optimizando el uso y

desempeño de los equipos de red utilizados en el sistema, de manera sencilla y eficaz

Existe el protocolo SNMPv2 que facilita la realización de un monitoreo mediante comunidades y permite gestión de la red, así como de la constante observación de los equipos mediante aplicación Web a tiempo real realizada en el Lenguaje de programación Perl, previniendo daños en los elementos de red y reduciendo el tiempo de solución de las incidencias presentadas.

1.2 OBJETIVOS

El diseño del proyecto pretende alcanzar los siguientes objetivos:

1.2.1 Objetivo general

Implementar monitoreo Web de una red WAN, considerando un sistema de alarmas SMTP, basado en SNMPv2.

1.2.2 Objetivos específicos

- Analizar el principio de funcionamiento del protocolo SNMPv2 aplicado en el monitoreo de una red WAN.
- Estudiar y comprender el lenguaje de programación PERL utilizado en la elaboración de la aplicación web.

- Crear una aplicación web que realice consultas SNMP por medio de scripts con lenguaje de programación PERL.
- Verificar resultados de la aplicación mediante el monitoreo implementado, analizando alarmas recibidas vía e-mail.

1.3 JUSTIFICACIÓN

En Ecuador, las grandes empresas de Telecomunicaciones son las que poseen un sistema de monitoreo completo que les permite conocer eventos en la red y el estado de los equipos de sus clientes. La problemática que presentan las diversas instituciones en sus redes en horarios fuera de oficina y contando con la tecnología actual, se puede desarrollar programas y aplicaciones que faciliten un monitoreo constante del estado de la red y funcionamiento de los equipos, y notifique a los administradores de algún fallo mediante alarmas.

Actualmente, estas organizaciones invierten grandes sumas de dinero en aplicaciones que los mantengan informados de su sistema de red, quedando inconformes por el escaso detalle técnico que se tiene de sus equipos. La implementación que se quiere realizar ayudará de gran manera a solucionar este problema; además de su bajo costo de aplicación, su acceso es fácil, mejorará la visualización del analista de red y garantizará la disponibilidad y funcionalidad de los equipos de red a tiempo real.

1.4 METODOLOGÍA

Para la elaboración de nuestro monitoreo web, se ha divido el trabajo en partes seccionadas que facilitarán el análisis y desarrollo del mismo.

Iniciaremos realizando la implementación de la red WAN configurando los equipos de red como base de nuestro proyecto. Crearemos una base de datos MYSQL donde se registrarán todos los elementos a utilizar, y ejecutaremos el script, que actualizará la información de la base de datos. Desarrollaremos la página en HTML que muestre la información que nos dejarán los scripts y se levantará un servidor web APACHE donde se muestra una interfaz con la información de los equipos.
Se generará un sistema de alarmas que finalmente mostrará vía e-mail al administrador el momento en el que detecte fallas en el sistema de red.

1.5 LIMITACIONES

Entre las limitaciones que tenemos para la realización del proyecto, tenemos:

- Se realizará la implementación de la aplicación web únicamente con programación PERL lo que nos restringe utilizar otras librerías que no son propias ni compatibles con el lenguaje.
- La optimización de la aplicación fuera del lugar de trabajo depende del acceso al correo electrónico que tenga el administrador de red

CAPÍTULO 2

2. FUNDAMENTO TEÓRICO

2.1 Protocolo SNMP versión 2 (SNMPv2).

Para la descripción del protocolo SNMP versión 2, nos basaremos en los aspectos más relevantes de SNMP versión 1.

SNMP o SNMPv1 (Simple Network Managment Protocol) es un protocolo de la capa de aplicación para gestión de equipos de red. Plantea una arquitectura basada en agente - servidor y permite a los administradores supervisar el funcionamiento de su red, resolver problemas y planear su crecimiento. Su principal tarea es la supervisión y la administración de equipos de una red informática. El protocolo también permite realizar tareas de gestión, tales como la modificación y la aplicación de una nueva configuración a través de la modificación remota de ciertas variables. Las variables que son accesibles a través de SNMP están organizadas en

jerarquías. Estas jerarquías, y otro tipo de datos (tales como el tipo y la descripción de la variable), se describen por Bases de Información de Gestión (MIB).

Los principales problemas de la versión 1 son la autenticación del origen del mensaje (PDU), la protección de los mensajes de divulgación y la colocación de controles de acceso en la base de la información gestionada (MIB). [1]

SNMPv2 nace de las mejoras debido a las falencias vistas en SNMPv1. Fue diseñado en 1993 y debía ser una evolución de su predecesor. Las operaciones Get, GetNext y Set utilizados en SNMPv1 son exactamente los mismos como los utilizados en SNMPv2. Sin embargo, SNMPv2 añade y mejora el protocolo de operaciones. En SNMPv2, si un valor es solicitado múltiples ocasiones en una petición GET, habrá respuestas múltiples, mientras que para la versión 1, no se da ninguna respuesta, sólo un mensaje de error. En la versión 1, los Traps tenían un formato diferente de todas las PDU. La versión 2 simplifica los Traps dando ellos el mismo formato que las PDU get y set. [1]

Éste protocolo define dos nuevas operaciones: GetBulk e Inform. La Operación GetBulk es utilizada por el NMS para recuperar de manera eficiente grandes bloques de datos. GetBulk llena un mensaje de respuesta con la mayor cantidad de datos que caben. La operación Inform permite enviar información de captura a otro NMS y luego recibe una respuesta. La última área que SNMPv2 debe mejorar era la seguridad, y esto llevó al desarrollo de SNMPv2 versión-variantes". [1]

Las operaciones de protocolo en SNMPv2 son diferentes a las de SNMPv1, puesto que requirió algunas modificaciones en el formato de las PDU. Sin embargo, las operaciones de protocolo son las mismas para todas las variaciones de SNMPv2. Las diferencias entre las versiones de SNMPv2 es el área de implementación de seguridad. [1]

Características generales de SNMPv2: [2]

- Amplia el modelo de comunicaciones haciendo posible tener una gestión centralizada y distribuida.
- El sistema posee la capacidad de funcionar tanto en agentes como en gestores.
- Se tiene la capacidad de comunicación entre gestores con la posibilidad de mantener la jerarquía en la gestión.
- Mayor eficiencia en la transferencia de la información.
- Soporta una señalización extendida de errores.
- Permite el uso de varios servicios de transporte.

2.1.1 Componentes básicos

Una red administrada con SNMP consiste de tres componentes claves:

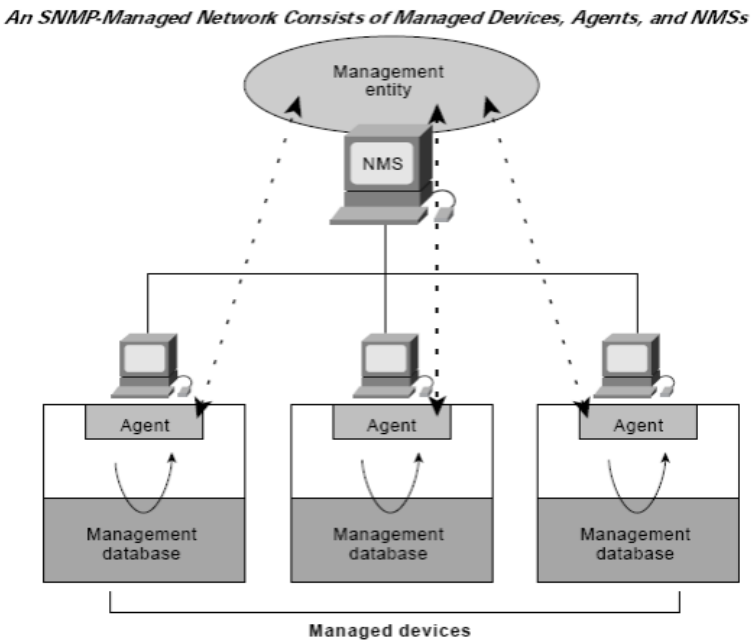
- Dispositivos administrados (*MD*)
- Agentes (MIB)
- NMS (*Network Management System)*

Un Managed Deviced (MD) reside en una red administrada conteniendo un agente SNMP. Recopila información permitiendo que sea disponible al NMS mediante SNMP. Los MD son equipos terminales que pueden ser tales como routers y servidores de acceso, switches, hubs, computadoras anfitrionas o impresoras. [3]

Los NMS´s facilitan gran parte de recursos de procesamiento y memoria utilizados para la administración de la red. Ejecuta aplicaciones que controlan los MD. [3]

Un agente es un módulo de software (SW) de gestión de red que reside en un MD. Un Agente tiene conocimiento local de información (sobre su memoria, número de paquetes recibidos - enviados, direcciones IP, rutas, etc.) y traduce esa información en un formato compatible con SNMP. [4]

En la Figura 2.1 se observan los componentes básicos de SNMP, donde se destaca la jerarquía con la que funciona el protocolo.



Fuente: Castro, C. Gestión de Redes de Computadores (2006). [4]

Figura 2.1 Componentes básicos SNMPv2.

2.1.2 Comandos Básicos

En SNMPv2 se tienen los comandos básicos utilizados en SNMPv1 los cuales son:

Los MD (Managed Devices) son controlados utilizando 4 comandos SNMP básicos: [3]

a) *READ* controla los Dispositivos Administrados siendo utilizado por un NMS que examina diferentes variables pertenecientes a los dispositivos.

b) *WRITE* controla los MD utilizados por un NMS que cambia los valores de las variables almacenadas.
c) *TRAP* es utilizado por los Dispositivos Administrados para reportar eventos de forma asíncrona a los NMS. Cuando cierto tipo de eventos ocurren, un MD envía un TRAP hacia el NMS.
d) Las operaciones de recorrido o *Traversal Operations* verifican qué variables son soportadas por los MD y obtienen valores de una tabla similar a una routing-table.

2.1.3 Base de información de administración SNMP (MIB)

2.1.3.1 Aspecto General

Un MIB (Management Information Base – Base de información de gestión), formada por diferentes variables SNMP, es una base de datos, el cual se caracteriza por tener un idioma independiente del sistema destino. [5]

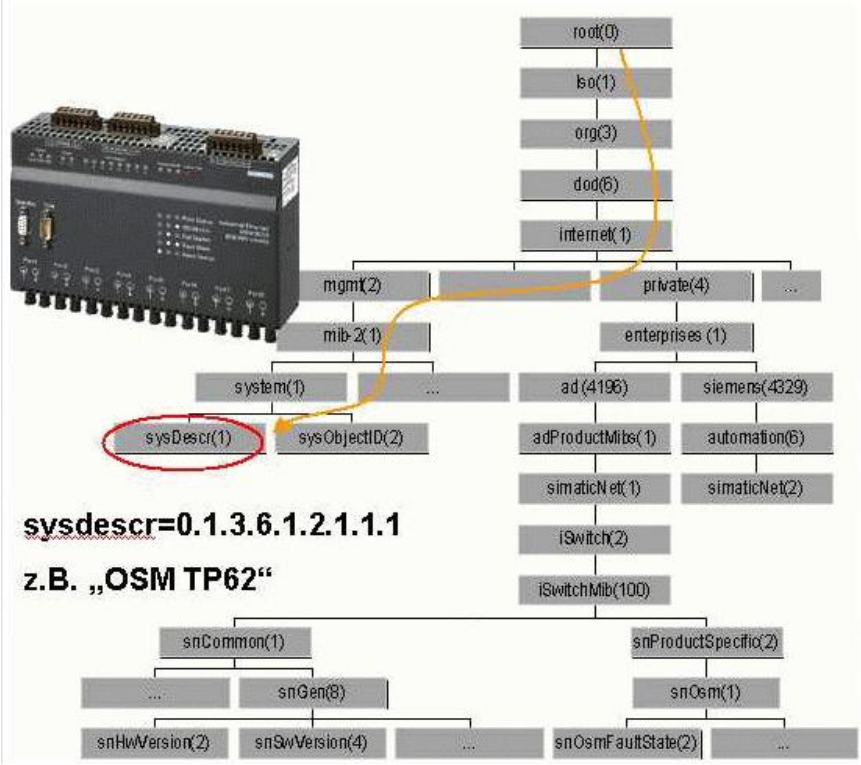
Gracias a esta estandarización independiente, se puede controlar elementos de diferentes fabricantes en una red diversa.

Se puede utilizar un MIB privado descrito por un fabricante en la supervisión de una red de componentes no estándar. De esta forma, también se pueden observar los valores que no estén contemplados con el estándar. La información a la que se accede a través del MIB privado lo describe el fabricante. [5]

2.1.3.2 Base de Información Administrativa MIB de SNMP.

Un MIB describe todos los elementos SNMP (variables SNMP) que se encuentran en la red. El OID (Object Indentifier – Identificador de objetos) indica donde está ubicado el objeto MIB. Se asigna la dirección de forma fija en los objetos MIB estándar. Los objetos MIB privados siempre se almacenan en el directorio "Enterprise". [5]

Tal como se observa en la Figura 2.2, el objeto estándar "sysdescr" (dirección del objeto 0.1.3.6.1.2.1.1.1) contiene una descripción en forma de texto de los componentes SNMP.



Fuente: Soporte de Automatización Siemens [5]

Figura 2.2 Vista general de la estructura

Los MIB privados deben ser conocidos en una estación de administración para poder ser leídos. En caso contrario los objetos no se pueden direccionar. Para poder escribir y leer inmediatamente objetos MIB, se utilizan los llamados buscadores MIB, disponibles en las aplicaciones de administración de red. [5]

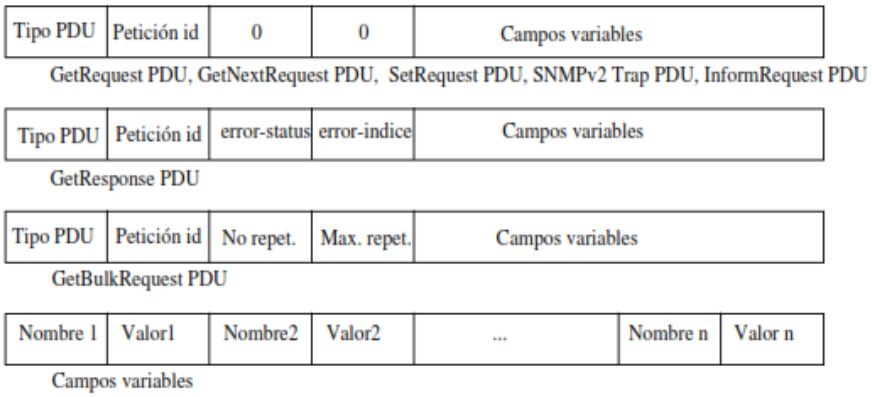
Se utiliza el compilador MIB para que éstos puedan integrarse en el sistema al que corresponde.

Mientras que el MIB analiza aspectos del sistema (por ejemplo información estática del flujo, nodos de gestión, aviso de errores), el SMI crea el entorno que necesita el MIB para su ejecución, de modo que describe el modo de intercambio de datos con el protocolo y el tipo de representación de objeto. [5]

2.1.4 Mensajes SNMP

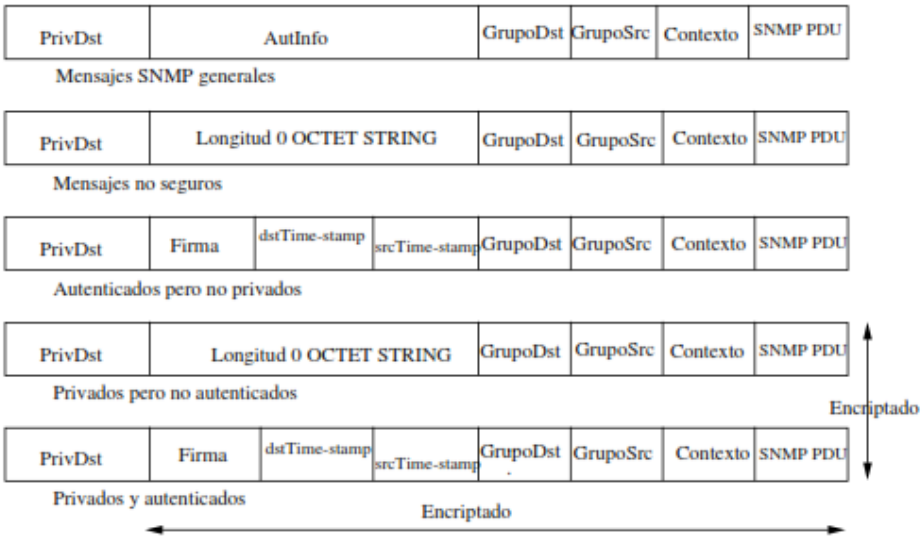
Cuando los programas de administración SNMP envían solicitudes a un elemento de red, el software del agente de ese elemento recibe las solicitudes y recupera la información de las MIB. Luego, el agente vuelve a enviar la información solicitada al programa de administración SNMP que lo inició. Para realizar estas tareas, el agente utiliza los tipos de mensaje siguientes: GetBulkRequest, GetNextRequest, GetRequest, Inform, Report, Response, SNMP v2-Trap y SetRequest. [6] Véase la Figura 2.3

Cuatro de estos tipos de mensajes son protocolos de solicitud y respuesta simples en los que SNMP utiliza el Protocolo de datagramas de usuario (UDP, User Datagram Protocol). Esto significa que existe la posibilidad de que una solicitud del sistema de administración no llegue al agente y de que la respuesta del agente no llegue al sistema de administración. SNMP es un protocolo de red sin conexión, por lo que no existen garantías de que los mensajes SNMP lleguen a su destino. [6]. Véase la Figura 2.4



Fuente: Barba Marti, A. Gestión de red. [6]

Figura 2.3 Formatos de PDU para SNMPv2



Fuente: Barba Marti, A. Gestión de red. [6]

Figura 2.4. Formatos de mensajes de SNMPv2

Sin embargo, el hecho de su incompatibilidad con la versión SNMPv1 y la mayor complejidad añadida a las plataformas están desaprobando sus futuras implementación.

El desacuerdo en el consorcio sobre las recomendaciones acerca de seguridad propuestas en SNMPv2 ha propiciado finalmente su incorporación en una nueva versión SNMPv3. [6]

2.1.4.1 PDU de SNMP v2

De acuerdo con la solicitud RFC 3416, cada PDU de SNMPv2 especifica una operación en particular tales como: GetRequest, GetNextRequest, GetBulkRequest, SetRequest, SNMP v2-Trap, InformRequest y Report; los cuales se describen en lo siguiente: [7]

2.1.4.1.1 GetRequest

A través de este mensaje el NMS solicita al agente retornar el valor de una variable mediante su nombre. En respuesta el agente envía una respuesta indicando el éxito o fracaso de la petición. Si alguna variable provoca un error, se usan los campos noSuchObject o noSuchInstance, que se comunica junto a su identificador de objeto para enviar el mensaje dentro de los campos variables. También se pueden utilizar los campos error-status y error-index. Si la respuesta es de gran tamaño se genera una nueva PDU con el error tooBig y con el campo variable vacío. [2]

2.1.4.1.2 GetNextRequest

Igual en sintaxis y significado que en SNMP v1, salvo que la respuesta no es elemental, ya que GetNextRequest en SNMP v2 procesa tantas variables como sea posible. [2]

2.1.4.1.3 GetBulkRequest

Esta PDU es una de las grandes mejoras de SNMPv2, ya que minimiza el número de peticiones permitiendo el intercambio de grandes cantidades de información. GetBulkRequest tiene una primera parte de peticiones de valores de objetos, con el número de objetos como el valor del campo non-repeaters, La segunda parte comienza a partir del objeto non-repeaters+1. Se devolverá un valor de la misma terminología igual al campo max-repetitions. [2]

2.1.4.1.4 SetRequest

Se diferencia de SNMPv1 en la respuesta por dos motivos: Primero se comprueban posibles condiciones de error validando todos los pares de variables con su valor. Segundo se modifican los valores por los recibidos en la PDU SetRequest. [2]

2.1.4.1.5 SNMPv2 – Trap

Una entidad SNMPv2 la crea y envía y actúa como agente en algún evento inusual. Usa el mismo formato que las demás PDUs, salvo GetBulkRequest. [2]

2.1.4.1.6 InformRequest

Esta PDU es enviada por una entidad SNMPv2 actuando como gestor, para completar la información de gestión. Cuando la entidad receptora lee una InformRequest elabora un envío con valores iguales a los campos que en la PDU entrante, a menos que sea de gran tamaño. [2]

2.2 LENGUAJE DE PROGRAMACION PERL

2.2.1 Descripción

Perl es un lenguaje contemplado para trabajar en cadenas de caracteres, archivos y procesos. Este trabajo se ve reducido por el gran número de operadores a disposición del usuario. [8]

Es un lenguaje libre de uso, eso quiere decir que es gratuito. Antes estaba muy asociado a la plataforma Uníx, pero en la actualidad está disponible en otros sistemas operativos como Windows. Es interpretado, al igual que muchos otros lenguajes de Internet. Esto quiere decir que el código en Perl no se compila sino que cada vez que

se ejecuta, lee el código y se pone en marcha interpretando lo que está escrito. Además es mucho más amplio a partir de otros lenguajes, ya que se puede hacer llamar a subprogramas escritos en otros lenguajes y viceversa. [9]

Se tienen dos requerimientos básicos para trabajar con Perl:

- Un editor de texto.
- El intérprete de Perl, el cual va a ejecutar el programa.

La característica principal de este lenguaje es que posee una combinación de los lenguajes más usados por los programadores, especialmente con el lenguaje estructurado C.

El lenguaje Perl se percibe habitualmente como un lenguaje intermedio, debido a su sintaxis, entre los shell scripts y la programación en C. Esto porque los programas en Perl son una sucesión de líneas de instrucciones similares a los shell scripts, la cual, no tiene un procedimiento principal como MAIN en lenguaje C, lenguaje con el que presenta gran similitud en sintaxis y funciones lo que facilita el trabajo de este lenguaje. [10]

Debido a esto, Perl es un lenguaje utilizado en los siguientes campos:

1. La administración de sistemas operativos.
2. La creación de formularios en la Web.

La administración de sistemas operativos, debido a sus características, es muy fuerte en la creación de pequeños programas que pueden ser utilizados como filtros. Además, hay plataformas bases para casi todas las plataformas que existen.

La creación de formularios en la Web, se ha usado desde sus inicios para escribir scripts CGI, los cuales realizan el intercambio de información, es decir, se encargan de analizar y responder la información que el cliente www envía al servidor a través de un formulario.

Una de las características más importantes es que el lenguaje Perl es más rápido que la mayoría de lenguajes, pero no es pre-compilado. Esto se debe a que los programas son analizados, interpretados y compilados en su totalidad por el intérprete Perl antes de su ejecución. [10]

2.2.2 Características

El lenguaje de Programación PERL posee varias características importantes, las cuales se mencionan a continuación:

1. Es un lenguaje práctico (fácil de usar, eficiente, y completo), además de que está enfocado hacia programadores con conocimientos del lenguaje.

2. Se puede programar cualquier necesidad presentada, ya que existen una serie de librerías y módulos para cualquier requerimiento.

3. Es rápido de crear, ya que no posee funciones que pueden ser importantes, pero retrasan el proceso de desarrollo de la aplicación del lenguaje.

4. Éste lenguaje es de difícil aprendizaje, aunque su poder de alcance justifica su estudio.

5. Se puede utilizar en varios entornos, tales como, Windows, OS/2, Linux, sin realizar cambios de código, únicamente introduciendo, de acuerdo al sistema operativo, el intérprete de Perl correspondiente.

6. Soportan una variedad de prototipos de programación, como la orientada a objetos, estructural y funcional. Así mismo no es necesario seguir ningún prototipo en particular.

7. Tiene incorporado un gran número de módulos disponibles y un sistema de procesamiento de texto.

8. Desde la consola de comandos ofrece ayuda en línea. Por ejemplo, para obtener ayuda sobre la función print, se debe de escribir en una ventana MSDOS: perldoc –f Print.

9. Se ejecuta desde la línea de comandos de una ventana del sistema operativo.

10. Es un lenguaje case-sensitive, lo cual un texto puede presentarse en mayúsculas o minúsculas [10].

2.2.3 Estructura del Lenguaje

2.2.3.1 Tipos de Datos

Una característica de mayor importancia en Perl es que no es necesario declarar una variable para comenzar a utilizarla, se la puede utilizar directamente, lo cual facilita el desarrollo del programa [8].

Existen tres tipos básicos de variables:

1. Escalar: Un escalar puede almacenar números, strings, referenciar a otras variables. Empiezan por el caracter $.

2. Arreglos: Sirven para agrupar un conjunto de variables de tipo escalar. Las variables array empiezan por el carácter @.

3. Hash: Consiste básicamente en un arreglo en el cual se accede a sus distintos elementos a través de una clave en lugar de por un índice. Empiezan por el carácter %.

2.2.3.2 Estructuras de control

El lenguaje Perl controla un conjunto de instrucciones que permiten elaborar de manera óptima el desarrollo de un programa. Se denominan estructuras de control porque ejecutan un conjunto de instrucciones cuando se verifica una condición o ejecuta iterativamente un bloque de instrucciones mientras una expresión sea válida. [11]

1. La instrucción if

   Es muy parecida a la utilizada en C. Cuando esta expresión resulta verdadera, se ejecuta la instrucción o el bloque de instrucciones descritos por llaves. Caso contrario, cuando es falsa ejecuta la instrucción o bloque de instrucciones descritas fuera del bloque.

2. La instrucción while.

   Ejecuta secuencialmente un bloque de instrucciones mientras una expresión sea verdadera, evaluando la comprobación en cada iteración delimitado por las llaves.

3. La instrucción for.

   La instrucción for permite ejecutar iterativamente un conjunto de instrucciones asignando un valor a una variable que permite controlar el número de repeticiones.

4. La instrucción foreach.
   Asocia reiteradamente un elemento de la lista a una variable. Esta sucesión de valores sirve para parametrizar la ejecución del bloque de instrucción. [12]

Las instrucciones for y foreach son equivalentes. Sin embargo, la utilización de una de estas instrucciones se justifica generalmente por el contexto.

5. La instrucción last.
   La instrucción last detiene la ejecución del lazo actual y se ejecuta la instrucción que sigue al bloque.

6. La instrucción next.
   Es idéntica a la instrucción continue en C. Interrumpe la ejecución del bloque actual y prosigue la ejecución en la iteración siguiente. Esta instrucción no interrumpe completamente la ejecución del lazo; la expresión que controla el bucle se evalúa. Si el resultado de la expresión es válido, el bucle se ejecuta de nuevo.

7. La instrucción until.
   Ejecuta un bloque mientras no se verifique si es verdadero un número determinado de veces.

8. La instrucción unless.

   Esta instrucción es análoga al if, salvo que permite considerar la no verificación de la prueba.

2.2.3.3 Operadores

2.2.3.3.1 Operadores Aritméticos básicos

Como se observa en la Tabla 2.1, entre los operadores aritméticos más utilizados se tiene:

Tabla 2.1 Operadores Aritméticos

Ejemplo	Tipo	Resultado
$a+$b	Suma	Suma de $a más $b
$a-$b	Resta	Resultado de la resta $a menos $b
$a*$b	Multiplicación	Producto de $a por $b
$a/$b	División	Resultado $a entre $b
$a%$b	Residuo (Modulus)	Es el residuo de la divisón de $a entre $b
$a**$b	Potenciar	$a a la protencia de $b

Fuente: Lizama, U. Operadores de Perl [13]

2.2.3.3.2 Operadores de cadena (String)

Para evitar confusiones, las variables de cadena (string) en Perl tienen sus propios operadores. Estos operadores son y x.

2.2.3.3.3 Operadores de Asignación

Los operadores de asignación como su nombre dicen asignan el valor del lado izquierdo de un = a la variable que se encuentre del lado derecho.

2.2.3.3.4 Operadores de Comparación

Estos operadores sirven para comparar dos variables, el resultado de la comparación puede regresar 1 si es verdadero o 0 si es falso. [13] Véase la Tabla 2.2

Tabla 2.2 Operadores de Comparación

Tipo	Númerico	Cadena
Mayor A	>	gt
Menor A	<	lt
Igual A	==	eq
No Igual A	!=	ne
Menor o Igual A	<=	le
Mayor o Igual A	>=	ge

Fuente: Lizama, U. Operadores de Perl [13]

2.2.3.3.5 Operadores de auto-incrementación y auto-decrementación.

Estos operadores sencillamente suman o restan 1 a la variable deseada. Únicamente depende del orden en que se coloca el operador en que hace las cosas. Si se coloca el operador antes que la variable, esta primero se incrementa y luego regresa su valor, en cambio si el operador viene después la variable primero regresa su valor y después lo cambia.

2.2.3.3.6 Operadores Lógicos

Estos operadores son muy usados al momento de evaluar acciones para controlar el flujo de nuestro programa y para controlar variables.

Observamos la Tabla 2.3 los Operadores Lógicos que se utilizan. [13]

Tabla 2.3 Operadores Lógicos

Ejemplo	Versión Corta	Versión Textual	Significado
$a and $b; $a && b	&&	and	regresa verdadero si las variables $a y $b estan definidas y no son 0
$a or $b; $a\|\|$b	||	or	regresa verdadero si alguna de las variables $a ó $b esta definida y no es 0
!$a; not $a	!	not	Regresá el contrario de lo que la expresión regresaría

Fuente: Lizama, U. Operadores de Perl [13]

2.2.3.4 Subrutinas

La subrutina puede estar en cualquier parte del fichero. Su definición es global: una vez definidas se pueden usar desde cualquier parte o fichero durante la ejecución del script. Si dos subrutinas tienen el mismo nombre, se aplica la última que se lee (en este caso lanza un warning). En las subrutinas, las variables definidas son globales. [14]

2.3 SERVIDOR WEB

Un servidor web es un programa que sirve para atender y responder a las diferentes peticiones de los navegadores, procesando una aplicación del

lado del servidor, proporciona los recursos que soliciten usando el protocolo HTTP o el protocolo HTTPS (la versión cifrada y autenticada).

Un servidor web básico cuenta con un esquema de funcionamiento muy simple, basado en ejecutar infinitamente el siguiente bucle:

- Espera peticiones en el puerto TCP indicado (el estándar por defecto para HTTP es el 80).
- Recibe una petición de un navegador.
- Busca el recurso solicitado.
- Responde utilizando la misma conexión por la que recibió petición.
- Regresa al segundo punto.

Un servidor web que cumpla con lo detallado en el esquema anterior atenderá de manera eficiente todos los requisitos básicos de los servidores HTTP, con la limitación que sólo podrá servir ficheros estáticos. [15]

2.3.1 ARQUITECTURA

2.3.1.1 PETICIÓN GET

Un servidor web realiza operaciones mediante el protocolo HTTP, de la capa de aplicación del Modelo OSI. Al protocolo HTTP habitualmente se le asigna el puerto TCP 80. Las peticiones al servidor se realizan mediante HTTP utilizando el método de petición GET en el que se solicita el recurso a través de la URL.

2.3.1.2 PETICIÓN POST

Este tipo de petición HTTP es el segundo más utilizado. Para enviar los datos al servidor se incluyen en el cuerpo de la petición con las cabeceras HTTP. Por lo general se asocia con los formularios web en el que los datos pueden ser cifrados para enviarlos de manera segura al servidor. [15]

En la tabla 2.4 observamos la estructura de una petición POST.

Tabla 2.4 Estructura de una petición POST

Estructura típica de una petición POST		Muestra
Petition type	POST url HTTP/1.1	POST comment.php HTTP/1.1
Referer	http-url-referer	index.php
Content-Length	contentlenght-int	63
Origin	http-url-origin	http://es.wikipedia.org
User-Agent	useragent-string	Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US) ...
Content-Type	content-type-string	application/x-www-form-urlencoded
Accept	mimetypes-accepted-string	application/xml,application/xhtml+xml ...
Accept-Language	language-accepted-string	es-ES,es;q=0.8
Accept-Charset	charset-accepted-string	ISO-8859-1,utf-8;q=0.7,*;q=0.3
Cookie	phpsessid-string	PHPSESSID=gm0ugf96iojuldio8i51u92716
Accept-Encoding	accept-encoding-string	gzip,deflate,sdch
Content	Content-string	&data=4&lang=es+es

Fuente: Stallings, W. "SNMP, SNMPv2, SNMPv3, and R MON 1 and 2" [15]

2.3.2 FUNCIONAMIENTO

El Servidor web responde adecuadamente a las peticiones realizadas por un cliente (navegador web), mediante una página web que se mostrará en el navegador o exhibiendo el respectivo mensaje si se produjo algún tipo de error. Por ejemplo al escribir www.google.com en nuestro navegador realiza una petición HTTP al servidor de esta dirección. El servidor responde al navegador enviando el código HTML de la página, al recibir el código este es interpretado por el cliente y lo exhibe en pantalla. Como podemos notar el cliente es el encargado de interpretar el código HTML, el servidor tan sólo se limita a transferir el código de la página sin llevar a cabo ninguna interpretación de la misma.

Adicionalmente los Servidores web pueden entregar aplicaciones web, que son códigos que se ejecutan cuando se realizan ciertas peticiones o respuestas HTTP.

Aplicaciones en el lado del cliente: Este tipo de aplicaciones son totalmente manejadas por el cliente, este se encarga de ejecutarlas en la máquina del usuario. El servidor web proporciona el código de la aplicación al cliente y mediante el navegador las ejecuta.

Aplicaciones en el lado del servidor: el servidor web ejecuta la aplicación generando un código HTML, el servidor toma este código y lo envía al navegador mediante el protocolo HTTP.

Las aplicaciones de servidor son la mejor opción para realizar aplicaciones web. Ya que al ejecutarse ésta en el servidor y no en la máquina del cliente. [4]

2.3.3 SERVIDOR WEB LOCAL

Un Servidor Web Local como su nombre lo indica es el que se encuentra en una red local al equipo de referencia. El Servidor web Local puede instalarse en cualquiera de los equipos que conforman una red local.

Por ejemplo si deseamos acceder al servidor Web y este se encuentra instalado en el mismo equipo podemos utilizar la dirección 127.0.0.1 de Loopback. Por lo tanto el puerto TCP 80 se obvia y los archivos se guardan en un directorio que es determinado por medio de la configuración.

2.3.4 APLICACIÓN DEL LADO DEL SERVIDOR

Una aplicación del lado del servidor debe estar escrita mediante un lenguaje de programación, tal como se muestra en la Tabla 2.5. El programa es diseñado con el fin de que el servidor web lo procese y realice una determinada acción.

Tabla 2.5 Lenguajes de Programación

Lenguaje	Fecha de primera versión estable	Sistema operativo	Ultima versión estable
PHP	1995	Multiplataforma	5.3.5
ASP.Net	1998	Windows (Algunas versiones)	4.0
Perl	1987	Multiplataforma	5.12.3
Python	1991	Multiplataforma	3.2.0
Ruby	1995	Multiplataforma	1.9.3-p125

Fuente: Stallings, W. "SNMP, SNMPv2, SNMPv3, and R MON 1 and 2" [15]

2.4 SERVIDOR APACHE

El Servidor Apache HTTP es un servidor Web de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otra, su fácil configuración, robustez y estabilidad hacen que cada vez millones de servidores utilicen este programa.

Su función es la de aceptar las peticiones de páginas que provienen de los visitantes que acceden a nuestro sitio web y gestionar o denegar su

entrega, según las políticas de seguridad establecidas. Esto, implica muchas funcionalidades que debe cubrir, como responder de manera eficiente, ya que puede recibir un sinnúmero de peticiones HTTP.

- Puede restringir el acceso a los ficheros que no se quieran 'exponer', gestionar autentificaciones de usuarios o filtrado de peticiones.
- Gestión de la información a transmitir en función de su formato e informar adecuadamente al navegador que está solicitando dicho recurso.
- Almacenar las peticiones recibidas, errores que se han producido y en general toda información que puede ser registrada y analizada para obtener las estadísticas de acceso al sitio web. [1]

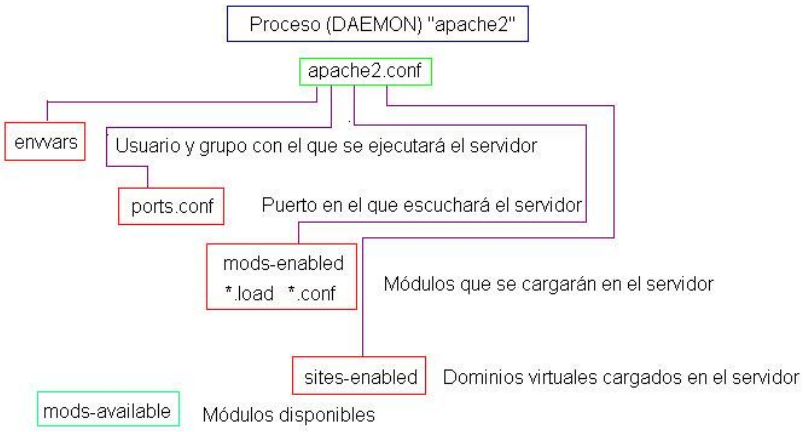
2.4.1 CONFIGURACIÓN

La configuración del servidor Apache se guarda en un archivo de texto nombrado: httpd.conf que se encuentra en la ruta C:\Apache\conf, lo podemos editar en cualquier editor de texto como el Bloc de notas se recomienda Notepad++, abre el archivo httpd.conf y edita manualmente las líneas que se indican:

- Todas las líneas que comienzan con el símbolo # son comentarios, explican en cada sección las distintas opciones.
- En la configuración debe encontrarse la palabra Listen, que indica el puerto y dirección IP por el que el servidor va a recibir las peticiones; se pueda usar de dos maneras, el servidor va recibir peticiones solo de la misma PC: Listen localhost:80 o recibirá peticiones de otras máquinas en una red local: Listen 80.

- Se configura también DocumentRoot, aquí se especifica la ruta de la carpeta local que contendrá las páginas y archivos a servir.
- En la desripción de <Directory> establece los permisos necesarios al directorio anterior.

Esta es la configuración con los parámetros esenciales para comenzar a utilizar Apache. [1]



Fuente: Digital Learning [16].

Figura 2.5 Estructura del servidor Apache2.

CAPÍTULO 3

3. APLICACIÓN PARA GENERACIÓN DE ALARMAS VIA SMTP.

Parte de nuestro proyecto es la generación de alarmas vía SMTP donde no será necesario estar en el lugar donde se encuentran nuestros equipos para darnos cuenta de algún error en la red. Podemos conocer el funcionamiento de los mismos, a tiempo real, de forma remota, accediendo vía web para conocer el status de cada uno de los equipos de red.

Con el objetivo de proteger nuestros elementos de red, evitar que un router se apague, o simplemente controlar que un equipo tenga su procesamiento en porcentajes estables, se ha desarrollado en lenguaje PERL un código que nos facilite el envío y recepción de e-mails automáticos notificando errores, realizando un recorrido de la base de datos, comparando los diferentes parámetros de funcionamiento de los equipos con sus niveles óptimos y enviar alarmas vía e-mail en caso de algún daño.

Es posible recibir alarmas, para estar alertados de posibles fallos en el procesamiento, saturación en los enlaces que ayudarán a tomar medidas preventivas y evitar daños en los elementos de red.

Se eligió el programa Perl debido a que es un programa completo, eficiente y fácil de usar, muy parecido al lenguaje C aprendido en materias correspondientes al flujo de la carrera de Ingeniería en Electrónica y Telecomunicaciones, además de las aplicaciones que se le pueden dar a los campos para desarrollar aplicaciones web que es la que implementaremos en este capítulo.

3.1 CREACIÓN Y ENVÍO DE PAQUETES SMTP CON PERL

Utilizando lenguaje de programación Perl, se desarrolló una función que nos ayudará a la elaboración de los mensajes que serán emitidos vía SMTP al administrador de la red.

Para la elaboración del código se tendrá presente los parámetros establecidos para aplicaciones web. Como base, utilizamos una función que crea un **Email :: Simple**, a partir de un conjunto de parámetros nombrados. El valor del parámetro header, es una lista de referencias que contiene un conjunto de cabeceras que se crearán. El valor del parámetro body, es un valor escalar que espera el contenido del cuerpo del mensaje. Los finales de línea en el cuerpo serán normalizados con CRLF.

A continuación, se detalla el código realizado por los autores para la emisión automática de e-mails en caso de presentarse fallos en los equipos pertenecientes a nuestra red.

```
# librerias
use Email::Simple;
use Email::Simple::Creator;
use Email::Sender::Simple 'sendmail';
use Email::Sender::Transport::SMTP;

# Envío del correo
my $email = Email::Simple->create(
header => [
From    => '"Evelyn" <ecarrasc@espol.edu.ec>',
                        To      =>
'emperatrice_87@hotmail.com',
                        Subject => 'prueba',
                    ],
                    body => "prueba",);

              my $transport =
Email::Sender::Transport::SMTP->new({
    host => '200.105.239.3', ##Servidor SMTP
    port => 25,
  });

  sendmail($email, { transport => $transport });
```

El diseño de la función utilizado para la emisión de e-mails consta de:
Librerías de e-mails utilizadas para crear, enviar y transporta el e-mail vía SMTP.

Se llama a la función Email :: Simple que es la que con parámetros establecidos, nos ayudará con le emisión del e-mail.

Se inicializa con el header de la función, llenando los campos FROM que es desde donde se enviará el e-mail, TO que es la dirección electrónica que recibirá el e-mail, SUBJECT que es el asunto del mensaje y BODY que es el cuerpo del mensaje que se enviará.

Se culmina la función enviando como parámetro la función TRANSPORT, que es la que enviará el mensaje mediante protocolo SMTP, especificando la IP del servidor y el puerto de envió de correos que es el puerto 25.

Para el desarrollo del código detallado en PERL de la función que se utilizará para el envío de e-mails se inicia con UPDATE, comando que actualiza columnas en registros de tabla existentes con nuevos valores. La cláusula SET indica qué columna modificar y los valores que puede recibir. La cláusula WHERE, si se da, especifica qué registros deben actualizarse.

Se continúa con un recorrido de la base de datos mediante un lazo for, que realiza pruebas de ping a cada una de las IPs registradas. Si ésta responde a esta prueba, su estado se registra como UP, de lo contrario se registra como DOWN.

Mientras realiza el recorrido, compara el estado anterior de las IPs con el estado actual, el cual si es diferente, llama a la función que envía e-mail al administrador para que tenga conocimiento del cambio de estado del equipo.

En el e-mail que se enviará, además de los datos de header específicos, se adjuntará la IP que está cambiando de estado y el estado actual en el que se encuentra.

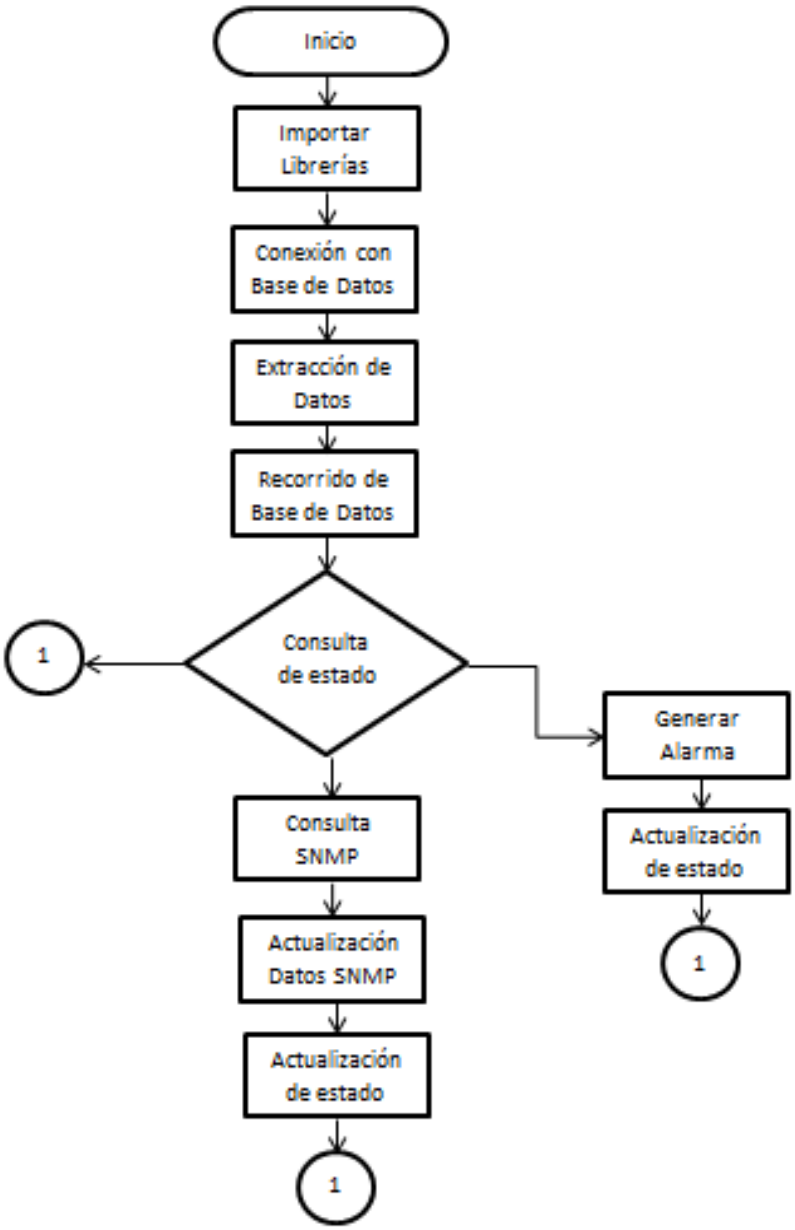
A continuación se detalla el desarrollo del código para el envío de alarmas SMTP.

```
     UPDATE enlace SET uestado = '$estado'  WHERE ip=
'ip_actual'
     for(;;) {
           $sth->execute();
           while ( @registro=$sth->fetchrow_array()) {
           if ($p->ping($ip_actual, 3)) {
$dbh->do("UPDATE enlace SET estado = 'UP' WHERE
ip='$ip_actual'");
      }else {
$dbh->do("UPDATE enlace SET estado = 'DOWN' WHERE ip=
'$ip_actual'"); }
     if(!( uestado eq estado))#uestado es diferente a
estado{
       my $email = Email::Simple->create(
           header => [From    => '"Monitoreo"
<jcastillom@gye.puntonet.ec>',
                        To   => 'jcastillo@telconet.ec',
                    Subject => 'MONITOREO',],
           body => "El enlace $NOMBRE con ip $ip_actual
está ahora $estado ",);
                my $transport =
Email::Sender::Transport::SMTP->new({
    host => 'mail.telconet.ec', #Servidor SMTP
    port => 25,
  });
  sendmail($email, { transport => $transport });
      }
     $dbh->do("UPDATE enlace SET uestado = '$estado'  WHERE
ip= '$ip_actual'");
     }
}
```

3.2 Diseño de emisión de alarmas SMTP.

Parte importante del proyecto es la generación de alarmas SMTP al momento que ocurra algún evento en la red. La emisión de alarmas SMTP notifica a los administradores del sistema de posibles eventos presentados en la red, por lo que se podrá tomar decisiones oportunas y evitar mayores incidencias.

Para el aviso de alarma se ha diseñado un diagrama de flujo que indicará paso a paso cómo funcionará el proyecto realizado. Véase Figura 3.1



Fuente: Diagrama de Flujo, autoría de los integrantes del proyecto

Figura 3.1 Diagrama de Flujo de elaboración del proyecto.

3.3 IMPLEMENTACIÓN DE LAS ALARMAS SMTP.

Como diseño de éste proyecto se realizará la implementación de una red WAN que consta de tres routers, los mismos que representan tres ciudades diferentes donde se encuentra ubicada la Matriz y sucursales de la empresa: Guayaquil, Machala y Portoviejo. Se configura en cada uno de los equipos la red en la que van a iniciar su funcionamiento:

- Guayaquil con la IP 192.168.2.3 (Matriz)
- Portoviejo con la IP 192.168.2.4 (Sucursal 1)
- Machala con la IP 192.168.2.5 (Sucursal 2)

Se mantendrá el monitoreo de los equipos mediante la aplicación Web en donde se observará el estado y funcionamiento de los equipos a tiempo real. Se realizará la implementación de acuerdo al diseño realizado de la siguiente manera:

Importar Librerías

Este proceso es esencial en el desarrollo de la aplicación, se aprovechará las pocas limitaciones que tiene Perl con respecto a los otros lenguajes de script, además de la amplia gama de librerías que nos ofrece este lenguaje para poder utilizarlas y así permitirnos abarcar la mayor cantidad de procesos utilizados en los diferente lenguajes de programación.

El poder interactuar con diferentes leguajes en uno solo nos permite encontrar diferentes soluciones para un mismo problema, esta característica ayuda a la fortaleza del programa y al desarrollo del código fuente de la aplicación.

Para la implementación de la aplicación utilizaremos las siguientes librerías en el código de programación:

```
use DBI;
use Email::Simple;
use Email::Simple::Creator;
use Email::Sender::Simple 'sendmail';
use Email::Sender::Transport::SMTP;
```

Conexión con la base de datos

En este bloque lograremos conectarnos con la base de datos mediante una serie de parámetros como son la identificación y seguridad de la misma, luego de esto se mostrará un mensaje de ERROR si no se pudo conectar con la base de datos. Asimismo se mostrará un aviso si la conexión tuvo éxito.

A continuación se observa el código desarrollado para la conexión con la base de datos y las ejecuciones que se realizarán en caso de respuesta positiva o negativa.

```
#Declaraciones
my $host="servidor"; #Servidor donde se aloja la base de
datos
my $base_datos="basedatos";  #Nombre de las base de datos
my $usuario="basedatos";  #Usuario de la BD
my $clave="password";  #Password de la BD
my $driver="mysql";  #Utilizamos el driver de mysql

#Conectamos con la BD. Si no podemos, mostramos un mensaje
de error
my $dbh = DBI-> connect ("dbi:$driver:database=$base_datos;
host=$host", $usuario, $clave)
|| die "\nError al abrir la base datos: $DBI::errstr\n";

#Mostramos aviso en caso de éxito
print "\nSe ha conectado con la BD $base_datos del driver
$driver\n";
```

```
#Realizamos la etapa de preparación de la sentencia
my $sth = $dbh->prepare("SELECT id,nombre FROM
articulos;");

#Realizamos la etapa de ejecución de la sentencia
$sth->execute();
```

Extracción de Datos

Se extraerá la información de cada uno de los registros de la base de datos para poder realizar las diferentes consultas y dependiendo del caso poder modificar algún estado de la misma.

En la generación de alarmas solo se preguntará y se modificará el campo estado que es el que determinará si un enlace esta UP o DOWN.

Recorrido de la base de datos

En este proceso se realiza el recorrido registro a registro de la base de datos para realizar los diferentes procesos sobre la misma, al mismo tiempo se presentaran por pantalla el resultado de cada uno de los valores en los registros.

A continuación se detalla el código utilizado para la extracción de datos y el recorrido de la base de datos.

```
#Realizamos la etapa de extracción de datos.
while ( @registro=$sth->fetchrow_array()) {
}
#Realizamos el recorrido de la base de datos. Mostramos los
registros.
while ( @registro=$sth->fetchrow_array()) {
print "Id:$registro[0]  Nombre: $registro[1]\n";
}
```

Generar alarma

Es uno de los procesos más importantes en nuestro diagrama ya que en este se va a generar la alarma cuando un enlace modifique su estado, se creará un paquete SMTP, el cual va a ser enviado mediante el protocolo, especificando la IP del servidor y el puerto de envió de correos que es el puerto 25.

El paquete SMTP está compuesto por dos partes importantes: el primero de ellos el header de la función que contiene los campos FROM que es desde donde se enviará el e-mail, TO que es la dirección electrónica que recibirá el e-mail, SUBJECT que es el asunto del mensaje y el segundo es el BODY que es el cuerpo del mensaje que se enviará, este deberá contener toda la información extraída de la base de datos.

Por último se utiliza la función TRANSPORT, que es la que enviará el mensaje mediante protocolo SMTP especificando la IP del servidor, como se muestra en el siguiente código:

```
my $email = Email::Simple->create(
                    header => [
                        From    => '"Monitoreo"
<jocastil@espol.edu.ec>',
                        To      => 'ecarrasc@espol.edu.ec',
                        Subject => 'MONITOREO',
                    ],
                    body => "El enlace $NOMBRE con ip
$ip_actual está ahora $estado ",
                );
               my $transport =
Email::Sender::Transport::SMTP->new({
    host => 'mail.espol.edu.ec', #Servidor SMTP
    port => 25,
  });
  sendmail($email, { transport => $transport });
```

Actualización de estados

Se modificará el estado de cada uno de los enlaces, luego de realizar la respectiva consulta a cada uno de los registros de la base de datos.

El comando UPDATE actualiza columnas en registros de tabla existentes con nuevos valores. La cláusula SET indica qué columna modificar y los valores que puede recibir. La cláusula WHERE, si se da, especifica qué registros deben actualizarse.

A continuación se muestra el código realizado para la actualización de datos:

```
$dbh->do("UPDATE enlace SET estado = 'UP' WHERE ip=
'$ip_actual'");
     }
     else {
$dbh->do("UPDATE enlace SET estado = 'DOWN' WHERE ip=
'$ip_actual'");
```

Consultas SNMP

Este proceso se ejecutará siempre y cuando el enlace se encuentre UP, aquí se detallará toda la información correspondiente a los equipos que conforman la red, la cual estará guardada en cada uno de los registros de la base de datos.

Los campos de los registros se detallan a continuación:

- Nombre
- IP
- Estado actual
- Ultimo estado
- Hostname
- Update
- OS
- Marca

3.4 Interfaz con base de datos MYSQL

Una de las tareas más comunes en un script en Perl con acceso a base de datos, es obtener datos de la misma. Para las consultas sobre bases de datos en Perl, se utiliza el lenguaje estándar SQL.

El proceso de obtener datos usando DBI se divide en cuatro etapas:

- *Preparación*. En esta etapa se analiza y valida la sentencia SQL y se devuelve un manejador de sentencia representando la consulta en la base de datos. Se utiliza el método prepare (con la consulta SQL como parámetro) del manejador de la base de datos, devolviendo el manejador de la sentencia.

- *Ejecución*. En esta etapa se ejecuta el manejador de sentencia obtenido en la etapa anterior. Se hace la consulta y se almacenan los datos en las estructuras de datos correspondientes de la base de datos, aunque en esta etapa, el script en Perl no puede acceder a los datos obtenidos. Se utiliza el método execute del manejador de sentencia.

- *Extracción*. En esta etapa se extraen los datos de la base de datos usando el manejador de sentencia. Se almacenan los datos consultados, en las estructuras de datos de Perl, para que sean posteriormente manipulados por el script.

- *Liberación*. En esta fase se liberan los recursos ocupados por el manejador de sentencia y por la base de datos. Se utiliza el método finish del manejador de sentencia.

Los registros de la base de datos contienen la información de los equipos que conforman nuestra red, estos deberán tener los siguientes campos:

Nombre, IP, Estado actual, Ultimo estado, Hostname, Update, OS, Marca

Estos campos se van a actualizar cada vez que la respuesta de ping sea exitosa, caso contrario estos campos tomarán el valor de NULL, y se presentará el estado del enlace hacia el equipo como DOWN. Además del estado se verificará porcentaje de procesamiento, uptime del equipo y estado de la red.

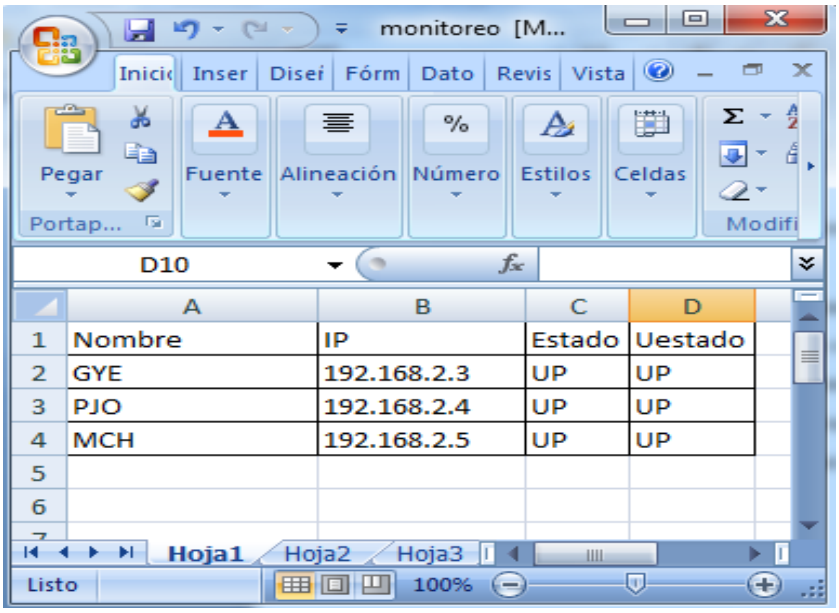
CAPÍTULO 4

4. DISEÑO DE LA APLICACIÓN WEB

Para un sistema de monitoreo es muy importante tener a disposición una aplicación web de la cual se pueda acceder a toda hora y desde cualquier parte del mundo; en este capítulo se desarrollará un programa el cual nos permita visualizar el nombre, ip, estado y el nivel de procesamiento de cada uno de los equipos que conforman la red.

4.1 CREACION DE LA BASE DE DATOS

Para este proyecto se trabajará con tres routers marca CISCO, modelo 881; la información de éstos se ingresará en la base de datos mediante la ejecución de un script llamado ScriptMonitoreo,pl. Éste pequeño código abre y lee el archivo monitoreo.xls, obtiene la información de cada una de sus celdas los asigna a los campos: nombre, ip, estado y uestado, los cuales llenan los registros de la base de datos. Véase Figura 4.1



Fuente: Código fuente, autoría de los integrantes del proyecto

Figura 4.1 Monitoreo.xls

A continuación, se detalla el código fuente del archivo ScriptMonitoreo.pl, se puede observar que el nombre de nuestra base de datos será ENLACE.

```
use strict;
use Spreadsheet::Read;
use Data::Dumper;
my $xls = ReadData ("monitoreo.xls");
use DBI;
my $dbh = DBI->connect('DBI:mysql:monitoreo', 'root',
'jacm') || die "Could not connect to database:
$DBI::errstr";
$dbh->do('delete from enlace');
for my $cell (map { "A$_" }2 .. 10) {
$_ = $xls->[1]{$cell};
      my $nombre = $xls->[1]{$cell};
      $cell =~ s/A/B/;
      my $ip  = $xls->[1]{$cell};
```

```
        $cell =~ s/B/C/;
        my $estado  = $xls->[1]{$cell};
        $cell =~ s/C/D/;
        my $uestado  = $xls->[1]{$cell};
        if($ip != ''){
         $dbh->do('INSERT INTO
enlace(nombre,ip,estado,uestado) VALUES(?,?,?,?)',
undef,$nombre, $ip,$estado,$uestado);
        }
}
$dbh->disconnect();
exit;
```

4.2 LEVANTAMIENTO Y CONFIGURACION DEL SERVIDOR WEB

Para el levantamiento del servidor web se ejecutó el demonio APACHE httpd.exe previamente instalado, el puerto 80 queda como puerto escucha para receptar las peticiones http desde los hosts remotos.

En la Figura 4.2 se encuentra la pantalla donde se observa el levantamiento del servidor web.

Fuente: Dominio Apache, autoría de los integrantes del proyecto.

Figura 4.2 Levantamiento del Servidor Web

Se puede observar que el estado del servidor es “LISTENING” lo que significa que está listo para establecer sesiones TCP en el puerto 80.

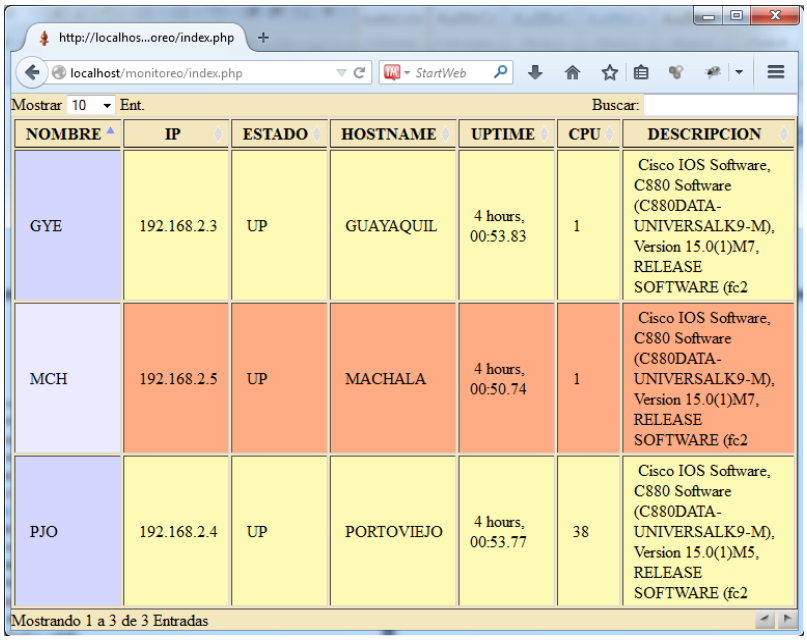
A continuación se observa el archivo de configuración del servidor apache llamado httpd.conf que se encuentra dentro del directorio APACHE en el cual se define ciertas políticas como el número de puerto escucha y políticas de acceso al servidor.

```
1      ServerName localhost:80
2      <Directory "c:/wamp/www/">
3           Order Deny,Allow
4           Allow from 192.168.2.0/24
5           Deny from all
6      </Directory>
```

En el código se establece el puerto 80 como puerto escucha y se establece políticas de seguridad permitiendo el acceso solo a la red de gestión y negando redes ajenas a esta.

4.3 DISEÑO E IMPLEMENTACION DE LA PÁGINA WEB

El diseño de la página web se implementará mediante el generador de código html PHP, para la presentación de los datos se utilizará un código en javascript que le dará mayor dinamismo en procesos de búsqueda y ordenamiento de los registros que conforman la base de datos, La visualización de los datos se refrescará automáticamente cada seis segundos. Véase la Figura 4.3



Fuente: Diseño de la página Web, autoría de los integrantes del proyecto.

Figura 4.3 Diseño de la página web.

Para la actualización de los datos se creará una clase llamada “conexión” que interactuará directamente con la base de datos; todo este proceso lo realiza el archivo index.php que se detallará en el siguiente punto del capítulo.

A continuación se detalla el código de fuente de la clase Conexión:

```
<?php
 class conection{

    function conectarMySql(){
        $host="localhost";
        $user="root";
        $clave="jacm";
        $name_db="monitoreo";
        //Conectar a mysql con el manejo de errores
```

```
        $conex=@mysql_connect($host,$user,$clave) or die("No se
puede conectar a la base de datos");
        //Seleccionar la base de datos
        mysql_select_db($name_db,$conex);
        return $conex;
    }

    function ejecutarQuery($sql){
        $conn=$this->conectarMySql();
        if($conn!=null){
            return mysql_query($sql,$conn);
        }else{
            return null;
        }
    }
 }
?>
```

4.4 DESARROLLO DEL CÓDIGO FUENTE DEL SCRIPT

El código fuente que se detalla a continuación es el que se encarga del diseño de la página web mediante el generador de código html PHP, adicionalmente realiza la interacción con la base de datos mediante la clase "conexión" y presenta la información actualizada que obtiene de la base de datos mediante consultas SQL, este proceso se repite y se muestra en la página web cada seis segundos.

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"
/>
<META HTTP-EQUIV="REFRESH"
CONTENT="6;URL=http://localhost/monitoreo/index.php">
<link href="style.css" rel="stylesheet" type="text/css" />
<link rel="icon" href="images/icon.ico" />

       <style type="text/css" title="currentStyle">
```

```
                    @import "media/css/demo_page.css";
                    @import "media/css/demo_table.css";
             </style>
             <script type="text/javascript" language="javascript"
src="media/js/jquery.js"></script>
             <script type="text/javascript" language="javascript"
src="media/js/jquery.dataTables.js"></script>
             <script type="text/javascript" charset="utf-8">
                    $(document).ready(function() {
                           $('#listado_auto').dataTable();
            });
             </script>
</head>
<body>
    <?php
    require_once ("clases/conexion.php");
    $obj = new conection;
    $sql = "SELECT  * FROM monitoreo.enlace ";

    $res = $obj->ejecutarQuery($sql);

    echo "<table border=1 id='listado_auto' class='display'>
            <thead>
                <tr>
                <th>NOMBRE</th>
<th>IP</th>
<th>ESTADO</th>
<th>HOSTNAME</th>
<th>UPTIME</th>
<th>CPU</th>
  <th>DESCRIPCION</th>
                </tr>
            </thead>
            <tbody>";
    while ($data = mysql_fetch_array($res)) {
          echo "<td> " . $data['nombre'] . "</td>
       <td> " . $data['ip'] . "</td>
   <td> " . $data['estado'] . "</td>
   <td> " . $data['hostname'] . "</td>
  <td> " . $data['uptime'] . "</td>
    <td> " . $data['cpu'] . "</td>
  <td> " . $data['description'] . "</td>
        </tr>    ";}
    echo " </tbody>
    </table>";
    ?>
 </body>
</html>
```

CAPÍTULO 5

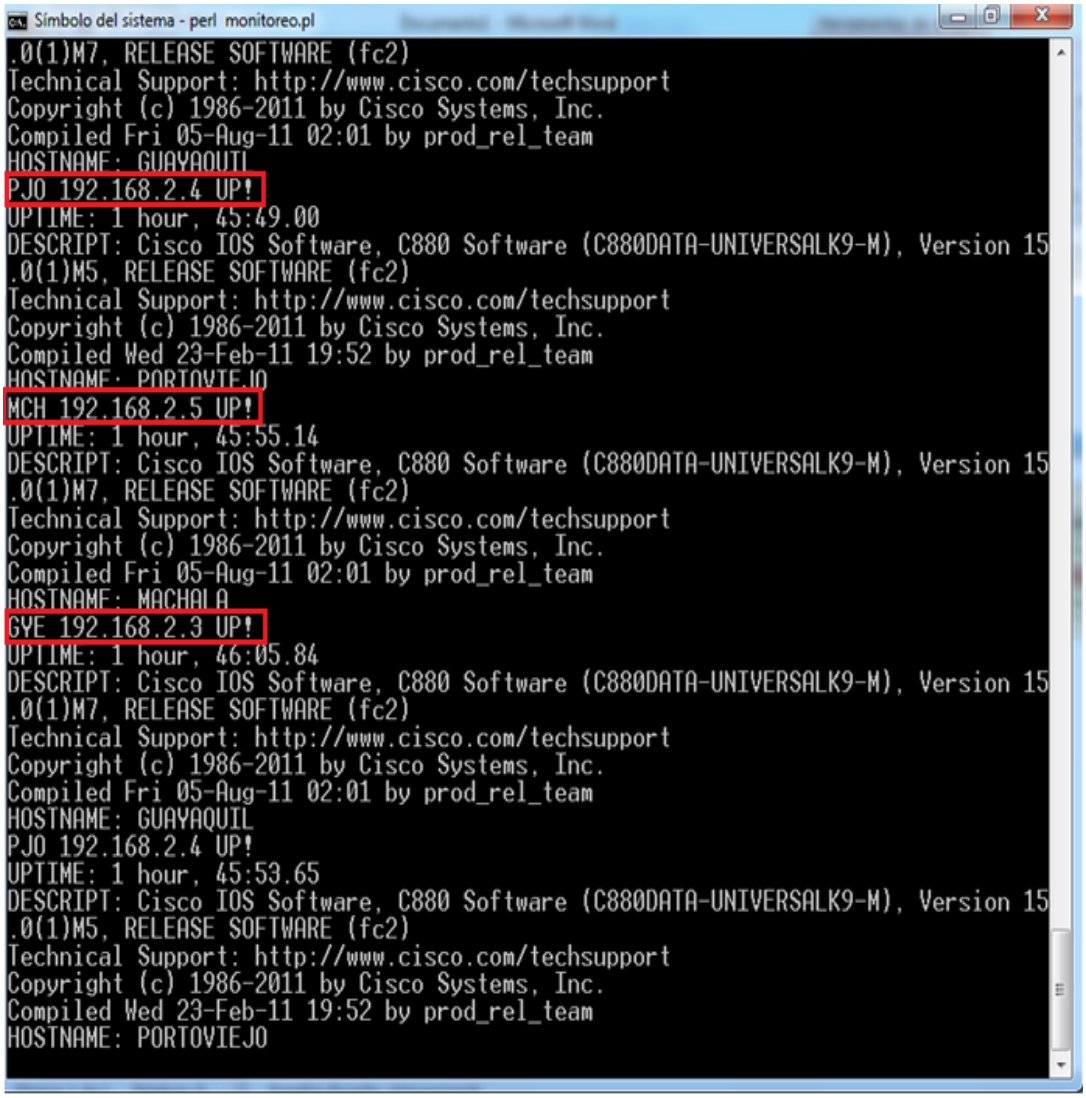
5.PRUEBAS DE MONITOREO Y GESTION DE RED

En este capítulo se presentan los resultados del trabajo en conjunto de la aplicación para la generación de alarmas vía SMTP, así como de la aplicación WEB donde observaremos el monitoreo. Se observa las diferentes situaciones que se pueden presentar en la red y se analiza las medidas preventivas para el buen funcionamiento de los equipos.

5.1 EJECUCION DEL SCRIPT

Se inician las pruebas de monitoreo poniendo en ejecución el script del programa fuente. Se realiza el barrido del código realizando paso a paso el monitoreo que se solicita, observándose los equipos operativos, su

UPTIME, su descripción como equipo, el HOSTNAME y el CPU PROCESS. Véase la Figura 5.1



Fuente: Código fuente, autoría de los integrantes del proyecto.

Figura 5.1 Script del programa fuente.

Debido a que es monitoreo a tiempo real, el script está en constante funcionamiento, donde se observan los diferentes valores de los equipos de red que integran el programa.

El Script está configurado en un servidor en oficina, por lo que sólo se visualiza fuera de ella con acceso remoto, que por motivos de seguridad, únicamente los administradores de red deben tener.

5.2 MONITOREO DE LA RED MEDIANTE APLICACIÓN WEB.

La visualización que tiene el administrador de red en su monitoreo es la aplicación web, que es en sus diferentes campos muestra el estado actual de los equipos.

Se tiene correcto funcionamiento de red cuando en el campo ESTADO, los elementos se observan UP y en el campo CPU, su funcionamiento es menor al 50%. Véase Figura 5.2.

Mostrar 10 Ent. Buscar:

NOMBRE	IP	ESTADO	HOSTNAME	UPTIME	CPU	DESCRIPCION
GYE	192.168.2.3	UP	GUAYAQUIL	7 hours, 32:55.14	1	Cisco IOS Software, C880 Software (C880DATA-UNIVERSALK9-M), Version 15.0(1)M7, RELEASE SOFTWARE (fc2
MCH	192.168.2.5	UP	MACHALA	7 hours, 32:56.73	2	Cisco IOS Software, C880 Software (C880DATA-UNIVERSALK9-M), Version 15.0(1)M7, RELEASE SOFTWARE (fc2
PJO	192.168.2.4	UP	PORTOVIEJO	7 hours, 32.54.91	1	Cisco IOS Software, C880 Software (C880DATA-UNIVERSALK9-M), Version 15.0(1)M5, RELEASE SOFTWARE (fc2

Mostrando 1 a 3 de 3 Entradas

Fuente: Aplicación Web, autoría de los integrantes del proyecto.

Figura 5.2 Aplicación WEB para el monitoreo de la red.

En caso de observar DOWN Esto no quiere decir que al aumentar el porcentaje de procesador de CPU el equipo estaría averiado. Un router tiene un buen procesamiento hasta un 70%. En el código de éste proyecto se configuró como límite de procesador de CPU 50% a manera de prevención, para revisión del motivo por el cual va en aumento, que pueden ser:

- Saturación en la red,
- Mal funcionamiento de la red interna,
- Equipos de red interna mal conectados,
- Router averiado.

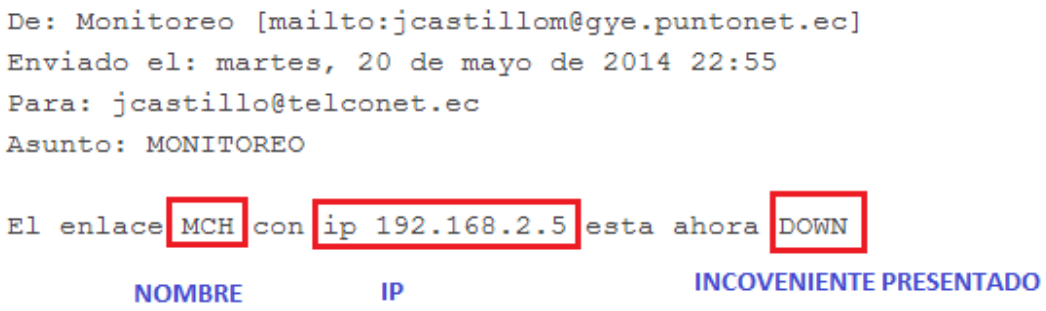
En el campo IP se observa la IP que se configuró en el equipo de las distintas ciudades en red, se observa también el HOSTNAME de los routers que se tienen en observación y la DESCRIPCION de los equipos.
En el campo DESCRIPCION se observa el Software del router, el modelo del equipo y la versión.

Debido a que es un constante monitoreo, el programa se va actualizando en la aplicación web, determinando a tiempo real los valores que presentan los equipos. Es el caso del UPTIME, que muestra el tiempo en que los equipos han sido encendidos. Se puede determinar en éste campo, si el equipo ha tenido un reinicio y se debe verificar el motivo del mismo, pues puede mostrar una falla en éste router así como fallas eléctricas en el punto donde se encuentra. Como medida de prevención se tiene como referencia el UPTIME de los demás equipos.

5.3 Gestión sobre la red utilizando SMTP

Cuando verificamos por monitoreo que un equipo no se encuentra en óptimas condiciones, llega un mail al correo electrónico del administrador de red indicando qué equipo está fallando.

De acuerdo al código ya realizado se notificará mediante correo electrónico la falla que presenta la red de la siguiente manera: NOMBRE del equipo, IP, e inconveniente presentado. Véase Figura 5.3



Fuente: E-mail de cambio de estado, autoría de los integrantes del proyecto.

Figura 5.3 Alarma via SMTP de fallo en la red

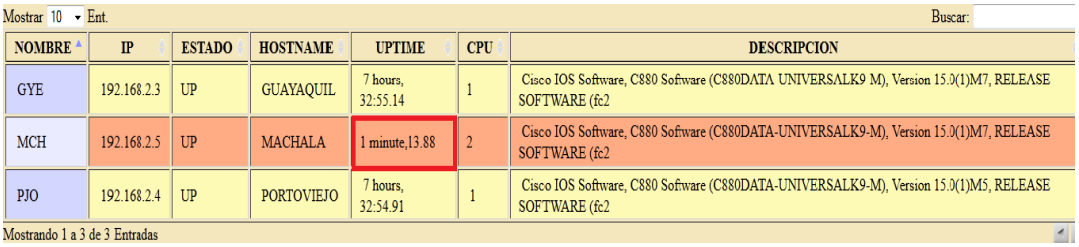
El administrador de red es notificado inmediatamente el estado del equipo cambia, por lo que puede tomar medidas inmediatas reduciendo el tiempo de solución del inconveniente.

Vía SMTP se da a conocer:

- El cambio de ESTADO de un router: Al instante que un equipo se desconecta de la red cambia su estado de UP a DOWN. Los motivos

que provocan este cambio de estado pueden ser varios, tales como, desconexión de la interfaz WAN de la red, problemas en el equipo o fallas eléctricas en el punto donde se encuentra instalado el router.

- En caso de problemas eléctricos, al retornar la energía en el punto, en la aplicación WEB se observará el UPTIME actual del equipo. Véase Figura 5.4



Mostrar 10 Ent. Buscar:

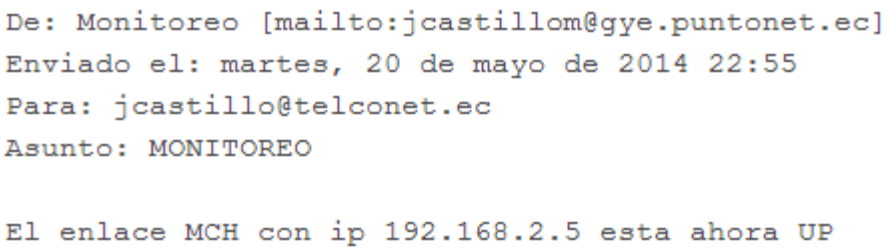
NOMBRE	IP	ESTADO	HOSTNAME	UPTIME	CPU	DESCRIPCION
GYE	192.168.2.3	UP	GUAYAQUIL	7 hours, 32:55.14	1	Cisco IOS Software, C880 Sofware (C880DATA UNIVERSALK9 M), Version 15.0(1)M7, RELEASE SOFTWARE (fc2
MCH	192.168.2.5	UP	MACHALA	1 minute,13.88	2	Cisco IOS Software, C880 Sofware (C880DATA-UNIVERSALK9-M), Version 15.0(1)M7, RELEASE SOFTWARE (fc2
PJO	192.168.2.4	UP	PORTOVIEJO	7 hours, 32:54.91	1	Cisco IOS Software, C880 Sofware (C880DATA-UNIVERSALK9-M), Version 15.0(1)M5, RELEASE SOFTWARE (fc2

Mostrando 1 a 3 de 3 Entradas

Fuente: Uptime del equipo, autoría de los integrantes del proyecto.

Figura 5.4 UPTIME del router MACHALA luego de un reinicio por falla eléctrica.

Para alertar al administrador de red sobre la caída del equipo, en este caso MACHALA, se notifica via SMTP a su correo tal como se muestra en la Figura 5.3. Luego de que se recupera gestión del router se notifica por el mismo medio al administrador de red. Véase Figura 5.5

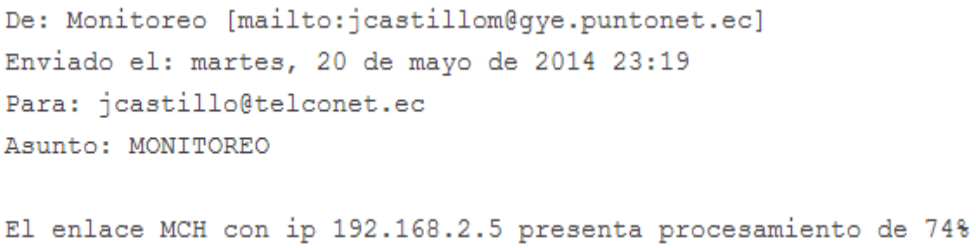


Fuente: E-mail de cambio de estado (UP) de equipo, autoría de los integrantes del proyecto.

Figura 5.5 Alarma via SMTP de cambio de estado de equipo.

- Alto porcentaje de procesamiento de CPU: Se configuró en el código fuente que al llegar el procesador del CPU al 50% envíe alarma SMTP como medida de prevención, por lo que revisar el motivo de éste aumento es necesario, así se evitan problemas mayores en la red.

  En la Figura 5.6 se muestra el e-mail de alerta que le llegará al administrador de red indicando que hay alto porcentaje de procesador de CPU.



Fuente: E-mail de cambio de alto porcentaje de procesamiento, autoría de los integrantes del proyecto.

Figura 5.6 Alarma via SMTP de alto porcentaje de procesamiento de CPU.

5.4 ANÁLISIS DE RESULTADOS

Luego de realizar las pruebas para monitoreo de la red se observa que previene grandes inconvenientes en el sistema; con el monitoreo a tiempo real se conoce el estado actual de los equipos con lo que se garantiza el buen funcionamiento de la red.

La creación de alarmas vía SMTP ayuda de gran manera a los administradores de red a que estén informados en el momento preciso que ocurre algún evento en el sistema, por lo que se pueden tomar decisiones preventivas y correctivas disminuyendo el tiempo de incidencias presentadas.

La aplicación WEB permite al usuario visualizar, mediante interfaz gráfica, los valores actuales de la red sin necesidad de estar presentes en la empresa. Se observa mediante browser el funcionamiento del sistema a tiempo real.

CONCLUSIONES

1. SNMPv2 es la evolución de SNMPv1, perteneciente a la capa de aplicación, nos ayuda a planificar, organizar y administrar el funcionamiento de la red, teniendo gran alcance de gestión vía remota, facilitando la solución de los problemas que se presentan. Tiene una implementación sencilla en grandes redes y utiliza pocos recursos por lo que es muy solicitada.

2. El Lenguaje Perl es una excelente alternativa para el diseño de la página Web, pues es de desarrollo práctico y portable de varias plataformas. Se estudió dicho lenguaje lo que facilitó la ejecución rápida de los scripts y cuenta con varias librerías gratuitas, a su vez nos permitió interactuar con diferentes programas y así llevar un mejor control y manejo de la aplicación.

3. Se demostró que con la utilización de un generador de código HTML PHP se realiza consultas SNMP y presenta en pantalla la información actualizada de la base de datos; en caso de ocurrir algún evento, los datos son nuevamente actualizados por código Perl, poniendo en aviso al cliente mediante un correo electrónico.

4. Al instante de tener fallas, tales como, caída de un router o elevado procesamiento de CPU se comprobó que el administrador de red fue notificado para su inmediata revisión, con la información necesaria del equipo que está presentado inconvenientes.

5. Se pudo comprobar que la alarma SMTP debido a procesamiento elevado del CPU se la tiene como medida de prevención, pues se configuró el aviso de Procesador Elevado cuando esté en el 50% de su funcionamiento, teniendo como límite real el 70%.

6. Se demostró que el tiempo de notificación al administrador de red (alarma SMTP), presenta un desfase de tiempo de 20 a 30 segundos. Éste se presenta por la ejecución del lazo descrito en nuestro programa y la generación de la alarma enviada. La afectación es mínima debido a que el administrador de red es notificado en corto tiempo para la toma de decisiones y resolución de fallas.

RECOMENDACIONES

1. El crecimiento de las Telecomunicaciones motiva a la actualización y desarrollo de nuestra red, por lo que es de vital importancia monitorear cada uno de los elementos que forman parte de la misma; de esta manera se garantizará un buen funcionamiento, nos ayudará a prevenir daños y nos informará sobre los cambios presentados en tiempo real para la toma de decisiones.

2. Para la implementación de alarmas SMTP se debe tomar en cuenta que el puerto de transmisión de correo electrónico que se debe configurar en el código en Perl es el puerto 25 y en el diseño de la aplicación Web se debe tener presente configurar el puerto 80, que es el puerto determinado para esta aplicación.

3. Sería conveniente configurar el correo electrónico en un celular inteligente que tenga el administrador de red para que el e-mail de cambio de estado de un equipo de red sea visto al instante.

4. Se recomienda tener contratado el servicio de backup en el punto Matriz. Así, si se presenta algún evento de última milla en este punto, tendremos contingencia para continuar el monitoreo del sistema.

BIBLIOGRAFÍA

[1] Bibbs, E., Brandon, M. (2006). Comparison of SNMP Version 1, 2 and 3. Extraído el 17 de Abril de 2013.

[2] Universidad de Jaén departamento de ingeniería electrónica de telecomunicación y automática (2007). SNMPv2. Recuperado el 18 de abril de 2014, de http://www4.ujaen.es/~mdmolina/grr/Tema%203.pdf

[3] Infotelecommil, 2009, Extraído el 20 de Junio del 2013, desde http://infotelecommil.webcindario.com/librostelecom/SNMP.pdf

[4] Castro, C. Gestión de Redes de Computadores (2006). Extraído el 17 de Abril de 2013, desde http://www.ebah.com.br/content/ABAAAfctoAG/snmp

[5] Siemens (n/d). Extraído el 25 de Mayo de 2014, desde http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&objId=32447945&load=treecontent&lang=es&siteid=cseus&aktprim=0&objaction=csview&extranet=standard&viewreg=WW

[6] Barba Marti, A. Gestión de red, Ediciones OPC, Universidad Politécnica de Catalunya, España 2001. Extraído el 6 de abril de 2013.

[7] Belloso Chacín, R. Planificación y Gestión de Red, Venezuela 2010. Extraído el 20 de Mayo del 2013, desde http://www.urbe.edu/info-consultas/web-profesor/12697883/archivos/planificacion-gestion-red/Unidad-II.pdf

[8] Barquero Chaves, I. Características del lenguaje Perl 5.0 y su aplicación como herramienta de desarrollo en la elaboración de un Servidor Web, Costa Rica 2007. Extraído el 18 de Junio del 2013, desde http://www.di-mare.com/adolfo/cursos/2007-2/pp-Perl.pdf

[9] Macario, P., Villafranca, D. Ingeniería del Software II, Julio 2008. Extraído el 10 de Junio del 2013, desde http://www.inf-cr.uclm.es/www/mpolo/asig/0708/tutorJavaWeb.pdf

[10] Salazar Ruiz, Iván. Proyecto Final PFC_ISR, Universidad del País Vasco, España 2011. Extraído el 25 de Mayo del 2014, desde http://adimen.si.ehu.es/~rigau/teaching/EHU/PFCs/IvanSalazar2010-2011/PFC_ISR_Memoria_08_07_11.odt

[11] Programación Perl, Marzo 2010. Extraído el 10 de Junio del 2013, desde http://trabajodeprogramacionperl.blogspot.com/2010_03_01_archive.html

[12] Ximena. Programación Perl, Colombia 2010. Extraído el 25 de Junio del 2013, desde http://trabajodeprogramacionperl.blogspot.com/2010/03/estructuras-de-control.html

[13] Lizama, U. Operadores de Perl, Extraído el 1 de Julio del 2013, desde http://perlenespanol.com/tutoriales/bases_de_perl/operadores_de_perl.html

[14] Alba, A. Madrid 2011. Extraído el 25 de Mayo del 2014, desde http://www.slideshare.net/aalbagarcia/perl3-subrutinas

[15] Stallings, W. “SNMP, SNMPv2, SNMPv3, and R MON 1 and 2” 3ª Edición, Addison Wesley, 1999. Extraído el 17 de Abril de 2013.

[16] Digital Learning. (n/d). Extraído el 22 de mayo de 2013, desde http://www.digitallearning.es/blog/apache-servidor-web-configuracion-apache2-conf/