INSTITUTO TECNOLOGICO Y DE ESTUDIOS
SUPERIORES DE MONTERREY
CAMPUS MONTERREY
PROGRAMA DE GRADUADOS EN ELECTRONICA,
COMPUTACION, INFORMACION Y COMUNICACIONES

DISEÑO E IMPLEMENTACION DE ALGORITMO PARA LA DETECCION PROACTIVA DE FALLAS EN REDES EMPRESARIALES DE DATOS

T E S I S
PRESENTADA COMO REQUISITO PARCIAL
PARA OBTENER EL GRADO ACADEMICO DE:
MAESTRO EN ADMINISTRACION
DE TELECOMUNICACIONES

POR
ARTURO LEV SERVIN NIEMBRO

MONTERREY, NUEVO LEON JULIO DE 2004

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS MONTERREY

PROGRAMA DE GRADUADOS EN ELECTRÓNICA, COMPUTACIÓN, INFORMACIÓNY COMUNICACIONES

DISEÑO E IMPLEMENTACION DE ALGORITMO PARA LA DETECCION PROACTIVA DE FALLAS EN REDES EMPRESARIALES DE DATOS

TESIS

PRESENTADA COMO REQUISITO PARCIAL PARA OBTENER EL GRADO ACADEMICO DE:

MAESTRO EN ADMINISTRACION DE TELECOMUNICACIONES

POR:

ARTURO LEV SERVIN NIEMBRO

MONTERREY , N.L. JULIO DE 2001

DISEÑO E IMPLEMENTACION DE ALGORITMO PARA LA DETECCION PROACTIVA DE FALLAS EN REDES EMPRESARIALES DE DATOS

POR:

ARTURO LEV SERVIN NIEMBRO

TESIS

Presentada al Programa de Graduados en Electrónica, Computación, Información y Comunicaciones.

Este trabajo es requisito parcial para obtener el grado de Maestro en Administración de las Telecomunicaciones

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

JULIO 2004

Dedicatoria

A Alejandra por su apoyo incondicional de principio a fin, su paciencia, sus consejos y su fuerza que me sirven de inspiración.
A Renée por su cariño y su sonrisa.

Gracias

Agradecimientos

A Alejandra y a Renée por su paciencia y su apoyo

A mis padres por darme la oportunidad de ser lo que soy.

A Sol por ser una gran hermana.

A Bill Gates ya que sin el este estudio no tendría razón de ser.

A Steve Jobs y a GNU por proveerme de herramientas eficientes para este proyecto.

A mi comité de tesis por el apoyo brindado para la realización de esta tesis.

Gracias

Resumen

Con el crecimiento del Internet en los últimos años el desarrollo de redes empresariales y redes públicas de datos ha crecido enormemente haciéndolas cada vez más grandes y complejas. Esto, aunado con la globalización de mercados y el incremento de dependencia de las aplicaciones en el buen funcionamiento de las redes de datos hace indispensable el tener buenos sistemas que permitan oportunamente informar a la administración de la red de posibles fallas de ésta para minimizar las fallas al usuario final.

Las redes han pasado de ser un valor agregado a ser un instrumento crítico para organizaciones gubernamentales, de negocios, educativas, etc. Al crecer las redes tiendes a volverse más complejas de administrar haciendo necesario el uso de herramientas inteligentes. Identificando esta necesidad este estudio tiene como objetivo diseñar e implementar un algoritmo que identifique las fallas de la red incluso antes de que estas sucedan. El estudio se basará en el supuesto que las redes en situaciones de operación normal tienen comportamientos previsibles y una falla o situación anormal tiende a modificarlos.

Tabla de Contenido

Lista de Gráficas

Lista de Tablas

Prefacio

Las redes modernas de datos han pasado de ser un valor agregado en las organizaciones a ser un recurso imprescindible y que no puede fallar. Las redes actuales de datos soportan una gran variedad de aplicaciones y muchas soportan sistemas de alta prioridad como hospitales o centrales eléctricas. Este nuevo cambio de paradigma donde la red paso de ser un lujo a un recurso que no puede dejar de operar ha obligado a la creación de mecanismos complejos. Entre estos mecanismos encontramos esquemas de respaldo donde coexisten 2 o más equipos con la misma función, enlaces de respaldo que conectan los mismos lugares pero con trayectorias físicas y lógicas diferentes y hasta centros de datos espejos con protocolos que aseguran cual de los dos sitios tiene mejor operación.

Estas redes complejas también resultan complejas de administrar y hacen imposible que puedan ser manejadas por una sola persona. Esto obliga al uso de herramientas computacionales y automatizadas que auxilien en la operación (Stallings 1996). Sin embargo la mayoría de estas herramientas pueden ser tan complejas como la red misma (Keffel 1992) y requieren de operadores altamente capacitados, además los sistemas actuales tienen la capacidad de recopilar una gran cantidad de datos haciendo muy complejo su análisis aún cuando esta información ha sido previamente sintetizada. Una solución factible es el uso de sistemas expertos o basados en el conocimiento que sean capaces de diagnosticar y encontrar fallas de una forma automática y sin la necesidad de la intervención humana.

El objetivo central de este estudio es el de buscar una forma sencilla de analizar la información que se puede obtener de la red para diagnosticar de forma sencilla y automática un problema incluso antes de que esto ocurra. El estudio estará basado en la tesis de que existen variables que pueden ser monitoreadas en la red y que éstas tienen un comportamiento sin grandes variaciones en periodos determinados de tiempo, y que la variación en este comportamiento es el resultado de un evento anormal en la red como una falla, un ataque o incluso un mal uso de la misma.

Para validar este estudio primero será necesario encontrar si realmente existe ese comportamiento sin variaciones en el tiempo el cual llamaremos patrón. Si ese patrón permanece constante en períodos determinados de tiempo será además necesario encontrar que si ante un evento externo como una falla estos patrones se modifican. Otro de los puntos críticos de este estudio será encontrar que variables son las que se requiere monitorear de la red. Existe un sin número de variables cada una de las cuales puede arrojar información importante y de acuerdo a ciertos ambientes de red, pero cuales son las variables que pueden darnos la información más exacta con el menor procesamiento, que variables modifican su patrón ante una falla y cuales no, cuales variables tiene un patrón y

cuales no son algunas de las preguntas que deberá resolver este estudio para cumplir con su objetivo.

Una vez validado la existencia de un patrón, su cambio ante factores externos como fallas y las variables que pueden ser analizadas será necesario proponer un algoritmo para analizar este patrón y calcular cuando se ha roto por una falla. El algoritmo propuesto debe cumplir con ciertas bases como tener fundamentos teóricos, fundamentos prácticos basados en la observación y en el análisis numérico de datos, debe ser matemáticamente sencillo para un implementación en software sencilla y finalmente debe ser confiable en los resultados obtenidos.

Una vez construido el algoritmo será necesario probar su validez. Esto se hará mediante la programación de un sistema en un lenguaje computacional que se adapte a las necesidades del estudio. Con ese sistema se tomaran muestras de las variables seleccionadas en redes de prueba y en redes de producción para posteriormente inducir fallas y comprobar la validez del algoritmo y del sistema.

Finalmente el proceso del diseño del algoritmos y del sistema será un sistema retroalimentado con el objetivo que las fallas iniciales del algoritmos sean corregidas para su mejora.

Capítulo 1

Situación problemática, problema y objetivo

Introducción

Este es el capitulo introductorio del estudio y en éste se hará una pequeña introducción al problema que se desea investigar y sus posibles soluciones. Finalizará este capítulo presentando el objetivo del estudio.

1.1. Situación Problemática

Con el crecimiento del Internet en los últimos años el desarrollo de redes empresariales y redes públicas de datos ha crecido enormemente haciéndolas cada vez más grandes y complejas. Tan sólo en Enero del 2002 Internet tenía cerca de 130 millones de usuarios, un 40% de crecimiento sobre los 97 millones que tenía en Diciembre de 2000 (C. Duffy 2002). En 1994 el número de usuarios conectados era sólo de 1,3 millones y la mayoría eran servidores de uso específico (Internet Software Consortium [ISC] 2002.). El número de servidores de web también se ha incrementado de 11.16 millones en Febrero de 2000 a 38.68 millones en hasta Febrero de 2002 (Netcraft 2002) así como el tráfico en el backbone de Internet en Estados Unidos medido desde los principales puntos de interconexión se ha movido de 5 a 55 petabytes[1] mensuales (C. Duffy 2002).

Todo esto nos da una idea de la importancia que esta red ha ido obteniendo desde sus inicios en la década de los 70s como una red académica llamada ARPANET y que posteriormente cambiaría su nombre a Internet abriendo sus puertas al mundo comercial el cual le permitió crecer a lo que conocemos hoy en día (Leiner et al. 2000).

Con la globalización de mercados y el incremento de dependencia de las aplicaciones en el buen funcionamiento de las redes de datos hace indispensable el tener buenos sistemas que permitan oportunamente informar a la administración de la red de posibles fallas de ésta para minimizar las fallas al usuario final. Las redes han pasado de ser un valor agregado a ser un instrumento crítico para organizaciones gubernamentales, de negocios, educativas, etc. Las redes cada vez soportan más aplicaciones y usuarios haciendo éstas cada vez más grandes y más complejas (Stallings 1996). Las redes de hoy en día no sólo sirven para interconectar dispositivos o simples usuarios, sino también sirven para permitir

[1] Petabyte: 2 a la 50th potencia (1,125,899,906,842,624) bytes. Un petabyte es igual a 1,024 terabytes. Webopedia (2002)

comerciar entre ellos. Tan sólo en el año 2000 en Estados Unidos las ganancias de reventa de productos por Internet creció de 17.8 a 44.5 billones de dólares (Prior 2001), las ganancias del mercado de ISPs en EU en 2002 se pronostica sea de 140 billones de dólares según Gartner Group (Carr et al. 1999) junto con un ganancias de 115.9 billones en el mercado de interconexión en 2001 a nivel mundial (Gifford 2001).

Algunas de las redes modernas soportan aplicaciones sumamente sensibles a las fallas. Algunos ejemplos son las redes que conectan los sistemas de los hospitales que controlan los dispositivos de monitoreo de pacientes. Las redes eléctricas actuales son controladas por una sistemas computacionales complejos conectados por redes de datos. Los gobiernos están instalando sistemas de información capaces de simplificar innumerables cantidades de trámites además de obtener todo tipo de información, todo soportado por las redes de cómputo. Tan sólo los nuevos paradigmas de intercambió de información que están surgiendo hoy en día y que el no contar más con ellos se hace casi inimaginable requieren de sistemas de redes totalmente resistentes a las fallas (Birman 2001)

El Internet inicialmente fue concebida como una red de colaboración basada en la confianza y nunca se pensó que llegaría a crecer al tamaño que tiene ahora y mucho menos que se utilizará para vender y comprar (Leiner et al. 2000). Con el crecimiento del Internet y la interacción de múltiples entidades y personas esa relación de confianza desapareció. Hoy en día, dos entidades quieren comunicarse entre sí, pero no se confía entre ellas (Clark 2001). Gran parte de esto es por el anonimato que se tiene en el Internet, esta desconfianza, en conjunto con lo costoso que podría resultar perder información hacen importante analizar el tráfico que corre sobre las redes empresariales conectadas a Internet.

Un aspecto muy importante a analizar cuando se habla de la importancia de tener los sistemas en operación es el costo de las fallas en la red. Por ejemplo; en una LAN con 200 usuarios y tres servidores asumimos que cada servidor y sus componentes tienen en promedio 98% de disponibilidad durante las horas de producción en cualquier año (10 horas por día, 5 días por semana). Esto no da una tiempo esperado de caída de 52 horas por año, multiplicado por los 3 servidores nos da 156 horas por año. Con 200 usuarios en la LAN, la pérdida máxima total en horas persona es de 31,200 horas por año. Si asumimos un salario promedio de $35,000 USD por empleado, con una carga de factor de beneficios del 25% el salario promedio con esto es ahora de $43,750 USD y $21.88 USD por hora, entonces $21.88 x 31,200 = $682,656 USD. Una estimación más realista es asumir un factor de carga de trabajo por empleado, este estimado indica el tiempo en que el empleado esta accediendo la LAN para efectos no triviales. Si se asume un 30% requerido entonces el costo de falla es del $204,749 USD (Anixter 2002 The Garner Group).

Si en lugar de hacer el análisis tomando en cuenta las fallas y lo hacemos tomando en cuenta la lentitud de la red, entonces en una compañía en donde sus empleados pierdan cada uno 5 minutos esperando a que se cargue la aplicación, esto nos da 21.67 horas por empleado cada año. Para el ejemplo anterior se traduce en una pérdida de $208.33 USD por empleado por año dando un total de pérdida de $20,833.33 USD de pérdida en la producción para la compañía. Pero el ejemplo no queda ahí, si calculamos una generación de ganancias por minuto de la compañía, esos 5 minutos se convierten en $2,083.33 USD de pérdida en las ganancias. Al año se pierden $104,166,67 USD en ganancias y $20,833 USD en producción, siendo la pérdida total de la compañía en $125,000 USD al año por tiempos lentos de la red (Anixter 2002)

Para satisfacer las necesidades de una red con una operación confiables que reflejen bajos costos de operación es necesario contar con sistemas eficientes de monitoreo que puedan recopilar información importante sobre el comportamiento de la red y la información que corre sobre ella (Stalings 1996). Esta información puede ser útil para planear nuevos crecimientos, para diagnosticar inseguridades, para prever problemas o simplemente para conocer patrones de comportamiento de los usuarios. Un ejemplo real de esto fue el crecimiento en el uso de software para intercambio de canciones en formato MP3 mediante servicios como Napster y Gnutella, este tipo de tráfico ha tenido un incremento desde que inicio su popularidad y este crecimiento pudo ser observado mediante los análisis de tráfico (Clark 2001). Las aplicaciones que propiciaron este movimiento llamado Peer-to-peer (P2P) se creyeron nuevas, sin embargo nos dice Andy Oram (2002) que si se observa el comportamiento de las redes este tipo de aplicaciones han estado ahí mucho tiempo. Esto nos da una pequeña idea de lo importante que es el análisis de las redes.

Una red grande no puede ser administrada por el esfuerzo de una sola persona. La complejidad de un sistema así hace obligatorio el uso de herramientas de administración automática de redes (Stallings 1996). Aunque construir una red para comunicar computadoras puede ser relativamente sencillo, en el momento de existir una falla existe la pregunta de donde está la falla, en el software de la computadora o en la red. Las redes modernas son una mezcla heterogénea de sistemas operativos, marcas de computadoras, servidores e incluso diferentes protocolos de red y esta heterogeneidad es una de las principales causas de falla (Kauffels 1992).Para resolver esta necesidad es necesario comprar software especializado el cual resulta muy costoso de implementar debido a su costo como software, al costo de la plataforma donde debe instalarse y a la complejidad de este proceso que requiere de personal altamente especializado. Una vez instalado resulta compleja su administración por la gran cantidad de funciones que este software puede realizar y que muchas de éstas no son prioritarias o necesarias para el administrador de la red.

Hoy en día los operadores juegan un rol muy importante en interpretar la información que se obtienes, además como avanza la tecnología, el nivel de

control automático de decisiones esta avanzado también. Esto además de hacer más eficiente la operación permite al personal desarrollarse en otras áreas de conocimiento (Hoske 1999).

Para desempeñar las tareas de administración de la red, es necesario contar con una sistema de administración de red. Este sistema se define como una colección de herramientas que en forma conjunta ofrecen una interfaz única para que el operador de la red pueda desempeñar sus tareas (Stallings 1996 y León-Garcia, Widjaja 2000). En las grandes redes corporativas de hoy en día no sólo es necesario un software de monitoreo de la red y que obtenga estadísticas de tráfico, si no que también es necesario de un software que automáticamente detecte modificaciones en los patrones de tráfico que puedan ser comportamientos anormales de la red resultado de fallas, malas configuraciones, uso excesivo de usuarios no autorizados o ataques de externos a la red.

Sin embargo estos sistemas tienden a ser muy complejos y requieren de habilidades especiales en los administradores de red además de que entre más funciones tengan será más la información que se genere haciendo cada vez más difícil su interpretación (Keffel 1992). Una posible solución es la aplicación de sistemas expertos o sistemas basados en el conocimiento. Sistemas de este tipo son un fenómeno en crecimiento para manejar la complejidad creciente de operación de las redes modernas. Los sistemas expertos podrían simular cargas en la red así como predecirlas

1.2. Problema

En los ambientes contemporáneos la atención a patrones de tráfico no está solo dedicado para el gobierno sino que también se extiende hacia los ambientes privados. El análisis de tráfico esta apareciendo en los ambientes corporativos y en los proveedores de telecomunicaciones para crear y administrar políticas de uso de los recursos. En los ISPs por ejemplo el tráfico puede ser observado para analizar cambios en los patrones de comportamiento y preferencias de los usuarios. Esto puede ayudar a predecir usos futuros de servidores, enlaces y recursos y planear anticipadamente su implementación (Clark 2001).

Con el crecimiento del Internet y de la importancia de la conectividad por la gran cantidad de aplicaciones que dependen de ella, el monitoreo del buen comportamiento de esta conectividad se ha vuelto imprescindible. Las variables básicas que hemos analizado y manipulado para analizar situaciones de seguridad, ancho de banda y disponibilidad son las mismas, sin embargo el cambio de paradigma es que el peso asignado a estas variables esta por cambiar. "Tenemos que tirar las ecuaciones actuales y reagrupar las variables en nuevas ecuaciones." (Oram 2002).

Para poder analizar estos patrones de tráfico y obtener la información es necesario implementar lo que se llama "Administración de redes". La administración de redes según Leon-Garcia y Widjaja (2000) incluye configurar, monitorear y posiblemente reconfigurar componentes de la red con la principal meta de proveer óptimo desempeño, tiempo de caída mínimo, seguridad y flexibilidad. La administración se lleva a cabo usando un sistema de administración de redes, este contiene un conjunto de herramientas de software diseñado para proveer una operación confiable de la red. Stallings (1996) define un sistema de administración de redes como la colección de herramientas de monitoreo y control de la red integradas para ofrecer una interfaz única y amigable para el operador de la red con un conjunto de comando para desempeñar la mayoría de las tareas de administración de la red. Además un sistema de administración de redes está diseñado para ver la red completa como una sola arquitectura unificada, con direcciones y etiquetas asignadas a cada punto y los atributos de cada elemento y enlace conocido por el sistema.

Stallings (1996) también define que un arquitectura de administración de redes debe colectar estadísticas en equipos de comunicaciones y relacionados; debe guardar las estadísticas localmente; debe responder a comandos para trasmitir información, cambiar parámetros y proveer información de estado todo hacia un centro de control. Finalmente debe poder generar tráfico artificialmente para efectuar pruebas.

Los software actuales de administración de redes están limitados en uso. Cuando se le configura para hacer tareas complejas estos programas tienden a generar una gran cantidad de información que se entrega sin aplicar filtros a los administradores. Esto requiere que los administradores estén muy bien capacitados para poder interpretar toda esta información, sin embargo la información muchas veces es demasiada y esto dificulta su interpretación. Los desarrolladores de software están buscando nuevas técnicas basadas en sistemas auto-reactivos y de inteligencia artificial para resolver este tipo de problemática (Kauffels 1992). En el monitoreo de las grandes redes corporativas basadas en el protocolo de TCP/IP existe un gran número de variables a monitorear por interfaz de equipo, un gran número de interfaces por equipo y finalmente un gran número de equipos. Esto hace que el análisis de la información de manera manual resulte compleja e ineficiente.

La administración de la red tiene como objetivo primordial mantener la operación normal de la red. Esto incluye funciones de monitoreo y control de la red, estas funciones pueden ser implementadas usando protocolos como el Simple Network Managment Protocol (SNMP) (Chen et al. 1994) o el Common Managment Information Protocol (CMIP) (León-Garcia, Widjaja 2000). El SNMP fue desarrollado en los inicios del Internet por el Internet Activities Board para la administración de redes TCP/IP y originalmente como una solución a corto plazo para la problemática del monitoreo de las redes (León-Garcia, Widjaja 2000).

Alternativamente también se desarrollo el CMIP como parte de los trabajos de la Organización Internacional de Estándares (OSI) y el Comité Consultivo Internacional de Telefonía y Telecomunicaciones (CCITT) (Stallings 1996). Este protocolo se desarrollo como una solución a largo plazo, sin embargo debido a la facilidad y a la velocidad en la que el SNMP fue primero implementado, además de algunas otras diferencias, el desarrollo del CMIP se considera separado del desarrollo del SNMP (León-Garcia, Widjaja 2000).

En las redes actuales de alta velocidad es importante que el manejo y administración de fallas sea pro-activo para detectar, diagnosticar y resolver los posibles problemas que pueden resultar en severas degradaciones del desempeño de la red. La administración pro-activa de fallas depende del monitoreo de la red para obtener los datos en los cuales se basen las decisiones de administración (Qiming, Shayman 2000). También con el crecimiento de la importancia de las redes en las organizaciones como una herramienta de bajo costo para las transacciones de negocios, mercadotecnia, e-commerce y soporte al cliente es necesario ofrecer herramientas para la administración de la seguridad. Sin embargo por la cantidad de información que puede llegar a generarse es necesario tomar acciones de forma automática con sistemas capaces de reconocer patrones específicos (Barrifu, Milano y Montanari 2001).

En el software comercial de monitoreo existen funciones que permiten agregar alarmas cuando una variable ha excedido algún valor mediante algunas reglas fijas predefinidas. Sin embargo la problemática con esta solución es que estos límites deben ser puestos manualmente por el administrador de la red en cada una de las interfaces monitoreadas y por cada una de las variables lo cual en redes muy grandes presenta una gran dificultad. Existe además el problema que la cantidad de información generada es demasiada haciendo la actividad muy compleja para los operadores de redes si esto se hace manualmente (Deri y Suin, 2000) Pero no sólo la ineficiencia del método de aplicación es el problema, sino que usualmente sólo se permite poner un límite por interfaz/variable monitoreada, lo cual se pretende probar es insuficiente.

Esta deficiencia se debe a que las redes corporativas no muestran un patrón fijo de comportamiento. Por ejemplo tienden a transmitir más en horas de oficina que fuera de ellas y más entre semana que los fines, esto sin contar que puede haber lugares de la red con ciertos límites de transmisión máxima diferentes a otros dependiendo del tipo de usuarios que existan. Estas variaciones de comportamiento hacen que límites fijos y únicos en una red corporativa no son eficientes (Deri y Suin 2000).

El software de monitoreo comercial y las herramientas como sniffers o probes son limitados en la cantidad de información que pueden desplegar (Deri y Suin 2000) y no cuenta con la capacidad de reconocer patrones dinámicos de comportamiento de variables de uso de la red lo cual es ineficiente para el

diagnóstico pro-activo de una red evitando tiempos nulos de respuesta ante situaciones de posible falla.

Haciendo referencia al trabajo de Geib y Goldman (2001) sobre Sistemas de Detección de Intrusos (IDS) podemos decir que para el monitoreo de la red y para detectar posibles intrusiones o fallas debidas a esto es necesario contar con sistemas que incorporen la inteligencia artificial como parte de un método pare reconocer y tomar acciones en caso de ataques de externos a la red.

1.3. Objetivo

Basados en la problemática de la inexistencia de sistemas de administración y monitoreo de redes capaces de reaccionar pro-activamente ante cambios en los patrones del tráfico de red debido a posibles fallas o intrusiones no autorizadas esta tesis tiene como objetivos los siguientes:

- Probar que el comportamiento de las variables monitoreadas tiene un patrón definido. Para este objetivo inicialmente es necesario definir las variables a analizar para posteriormente obtener información de su comportamiento en una red en producción. Estas variables se obtendrán en diferentes puntos de la red y con diferentes tipos de usuarios para simular la heterogeneidad de una red empresarial.
- Definir un algoritmo mediante una técnica matemática que permita prever de forma pro-activa un posible problema en la red debido a cambios en los patrones de tráfico usuales. El objetivo de este algoritmo será disminuir los tiempo de falla de la red.
- Desarrollar un sistema sencillo que utilice el algoritmo obtenido para identifica fallas en la red cuando las variables monitoreadas salgan de los patrones definidos.
- Probar que el algoritmo y el sistema pueden ser capaces de detectar diversos tipos de fallas en la red de forma pro-activa y con mínima configuración por parte de los administradores de red.

Capítulo 2

Marco Teórico y Modelo Conceptual

Introducción

En este capítulo se sentaran algunas bases conceptuales en las que estará cimentado el estudio. Cada uno de los conceptos teóricos requeridos por el estudio se abordarán en este capítulo de una manera introductoria para posteriormente tratarse más a detalle.

2.1. Marco Teórico

A partir de lo analizado en la definición del problema y de la situación problemática es necesario definir en el marco teórico tres puntos a analizar que son las variables a analizar, el patrón de tráfico y el algoritmo.

2.1.1 Modelo conceptual de administración de redes

Desde una perspectiva de los cuerpos de generación de estándares como la ITU, se desprende un modelo conceptual a partir del desarrollo de TMN (Telecommuncations Management Network) el cual se observa en la figura 2.1. En esta figura se observan componentes como la red de administración de telecomunicaciones encargada de proveer la interconexión entre los dispositivos de control como los NMS (Network Management Station). Se observa la red de telecomunicaciones encargada de conectar los dispositivos de telecomunicaciones (NE) los cuales pueden ser conmutadores LAN, enrutadores, estaciones de trabajo, concentradores, PBXs. Finalmente está la Red de Comunicación de Datos encargada de conectar los dispositivos administrados con los dispositivos de control (Subramanian 2000).

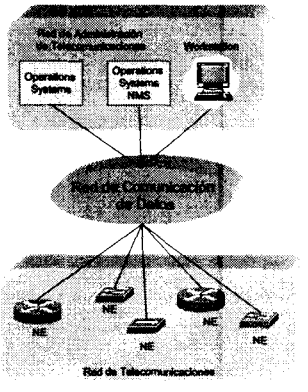


Figura 2.1

2.1.2 Variables a analizar

Para la administración y monitoreo de redes existen dos protocolos: el Common Managment Information Protocol (CIMP) y el Simple Network Managment Protocol (SNMP). El CIMP fue desarrollado por la ahora ITU-T como una solución a largo plazo para resolver las necesidades de administración de redes heterogéneas mientras que el SNMP fue desarrollo por el Internet Engineering Task Force (IETF) como una solución a corto plazo, más sencilla y únicamente para redes basadas en el protocolo de TCP/IP (León-Garcia, Widjaja 2000).

El SNMP consiste de tres componentes (León-Garcia, Widjaja 2000):

1) Un marco conceptual que define las reglas para describir el manejo de información, este se conoce como Estructura de Manejo de Información (Structure Managment Information SMI)
2) Una base de datos virtual conteniendo información acerca del dispositivo manejado, ésto se conoce como Base de Manejo de Información (Managment Information Base MIB)
3) Un protocolo para comunicación entre un manejador y un agente de un dispositivo administrado, este se conoce como el Protocolo Simple de Administración de Redes (Simple Network Managment Protocol SNMP)

Alternativamente Kauffels (1992) menciona dos componentes más, la Estación de la Administración de Red (Network Managment Station NMS) y los agentes. El NMS es un componente central usualmente una estación de trabajo o un servidor el cual provee al administrador con un visión general del estado de la red además de herramientas que facilitan su intervención en caso de fallas. El otro componente es el agente, el agente es un componente de software que reside en cada uno de los dispositivos administrados y cuya función es registrar y responde el estado de ciertos valores a monitorear.

En el artículo de Cabrera (et at al 2001) se menciona el interés por encontrar una forma pro-activa de identificar ataques distribuidos de negación de servicio (DDoS). "Ataques de DDoS envuelven el romper la seguridad de cientos de miles de máquinas sobre el Internet. Entonces el atacante instala software de DDoS en ellas, permitiéndole tomar el control de estas máquinas y enviar ataques coordinados a los sitios de las víctimas. Estos ataques típicamente se acaban el ancho de banda, la capacidad de procesamiento de enrutadores y los recursos de la red, haciendo que los sitios de las víctimas pierdan conectividad" (Todd 200), ejemplos de estos ataques fueron los sufridos por Ebay y Yahoo en Febrero de 2000 (Todd 2000). Para las propuesta y experimentación y experimentación de Cabrera (2001) hicieron uso de diversas variables contenidas sobre el árbol del MIB-II como *ipOutRequests*, *icmpInEchosReps*, *tcpinErrs* y *udpInErrors*. Estas variables como lo indica su estudió les permitió saber cuando una máquina era

atacada antes de que se negara el servicio de ésta. Este estudio nos resulta muy útil ya que nos demuestra la factibilidad de detectar anomalías en el comportamiento de la red, sin embargo es muy específico en el tema de la administración de la seguridad, mientras que nuestro estudio se tiene contemplado para integrar la administración de fallas, desempeño y seguridad.

Por otro lado en el artículo de Ji y Thottan (1998) se habla de la detección de anomalías en la red usando agentes distribuidos. En su artículo mencionan que no existe una variable única para detectar todas las manifestaciones de anomalías dentro de una red ya que cada una puede manifestarse diferente dependiendo de las circunstancias. Sin embargo es bueno reducir el universo del MIB a un grupo reducido para facilitar el desarrollo de su agente. El MIB mantiene 171 tipos de variables (Rose 1996). Estas caen dentro de otros grupos, de los cuales sólo algunos son relevantes para el tráfico de redes TCP/IP. También en estos grupos específicos, existen variables con información redundante. Bajo estos argumentos, Ji y Thottan (1998) encontraron cinco variables relevantes para su algoritmo: *ifIO*, *ifOO*, *ipIR*, *ipIDe* y *ipOR*.

Las variables descritas anteriormente son suficientes para iniciar el desarrollo del experimento que llevará a concluir con la implementación de una herramienta basada en un algoritmo de predicción que pueda anticipar fallas o comportamientos anormales dentro de la red.

2.1.3 Patrón de Tráfico

Las funciones desempeñadas por un sistema de administración de redes pueden ser categorizadas en las siguientes cinco áreas de acuerdo a León-Garcia y Widjaja (2000):

- Administración de fallas: Se refiere a la detección, aislamiento y resolución de problemas de red. Debido a que una falla puede causar que una parte o toda la red falle, la administración de fallas es un medio para mejorar la confiabilidad de la red.
- Administración de la configuración: Se refiere al proceso de iniciar la configuración de un dispositivo dentro de la red y ajustarlo en respuesta a cambios requeridos por la red. Esta es una de las funciones más importantes de la administración de redes ya que una configuración no óptima puede hacer que la red no funcione a toda su capacidad o hacer que no funcione nada.
- Administración de contabilización: Esta se refiere al rastreo del uso de los recursos de la red. Por ejemplo puede monitorearse la carga generada a la red por un usuario en cierto momento del día. Alternativamente, se puede analizar el nivel de tráfico que pasa por un puerto en particular de un dispositivo dentro de la red.

- Administración del desempeño: Envuelve el monitoreo de la utilización de la red, tiempo de respuesta entre dos punto de la red y otros tipos de mediciones en diversos puntos. La administración de desempeño se usa para mejorar los tiempos de respuesta. Algunos ejemplos son monitorear la utilización de un segmento de Ethernet o los puertos de un switch para analizar si es tráfico ha pasado un nivel crítico que no permita un correcto funcionamiento.
- Administración de la seguridad: Se refiere al proceso de hacer la red más segura. Este proceso envuelve el manejo de procesos de seguridad como los servicios de control de acceso, autenticación, confidencialidad e integridad. En este aspecto se puede agregar de acuerdo a Barrifu, Milano y Montanari (2001) que la administración de la seguridad ha tomado una gran importancia debido a que las redes manejan cada vez más información sensible y que su pérdida o modificación no autorizada puede significar grandes cantidades de dinero. También mencionan que la administración de la seguridad debe defenderse contra accesos no autorizados y negaciones de servicio. Para esto es necesario analizar una gran cantidad de información de forma automática y esto puede hacerse mediante sistemas expertos.

Nuestro estudio estará basado en una integración de una herramienta para soportar la administración del fallas, la administración de desempeño y la administración de la seguridad de forma integrar y pro-activa. Para ello será necesario recolectar la información de los valores de las diferentes variables que se consideran importantes para tratar de saber cuando algo no está funcionando correctamente dentro de la red. Al comportamiento de estas variables a lo largo del tiempo es lo que llamaremos "Patrón de Tráfico".

El patrón de tráfico de estas variables monitoreadas debe tener un comportamiento cíclico a lo largo del tiempo repitiendo su comportamiento en cierto período de tiempo. Uno de los objetivos intermedios del estudio será determinar si existe ese comportamiento y en cuanto tiempo se repite.

2.1.4 Algoritmo

La palabra algoritmo es difícil de definir si nos basamos puramente en el significado de diccionario ya que no fue hasta 1957 cuando apareció en el diccionario "Webster's New World Diccionary" y es una derivación o mal formación de la palabra aritmética (Knuth 1993). Para la Enciclopedia Británica (2002) un algoritmo es un procedimiento sistemático que produce en un número finito de pasos una respuesta a una pregunta o la solución a un problema. Además afirma que el nombre deriva de la traducción al latín de la palabra "Algoritmi de numero Indorum" del matemático musulmán del siglo 9º al-Khwarizmi.

Los procesos de toma de decisiones que se toman dentro de una empresa incluyen la administración de las comunicaciones en una red, esto es obtener, analizar y tomar decisiones en base a la información. Este proceso puede demandar una gran cantidad de recursos y generalmente es un proceso intensivo del cual depende la organización. Algunos ejemplos de estos procesos son la detección automática de fallas en la red, reorientación del tráfico dentro de la red, re-configuración en línea, análisis de la seguridad, análisis del desempeño de la red, detección de intrusos, contabilización de usuarios, etc. (Ericson, Lisa Ericson y Minoli 1989)

Debido a la complejidad de este tipo de procesos y a la gran cantidad de información que se debe analizar es necesario automatizarlos (Barrifu, Milano y Montanari 2001) mediante alguna técnica de Inteligencia Artificial (AI). Derek (1998) Define la Inteligencia Artificial como "el campo de la ciencia computacional que intenta construir mecanismos computacionales para resolver actividades que requieren inteligencia cuando son hechas por humanos."

Una forma es hacerlo mediante sistemas expertos. "Los sistemas expertos son programas computacionales diseñados para simular el proceso de razonar de humanos expertos en un área" (Ericson, Lisa Ericson y Minoli 1989). Los procesos que deben hacer los sistemas expertos son aprender y razonar, siendo el aprender uno de los mas complejos de implementar. En el caso del monitoreo y análisis de tráfico de la red para evitar falla se requiere de una aplicación capaz de tomar decisiones en tiempo real (Braden 1988). Un sistema experto de tiempo real utiliza la información de forma óptima para hacer decisiones, además si los hechos cambia, también lo hacen las decisiones (Ricketts 1988).

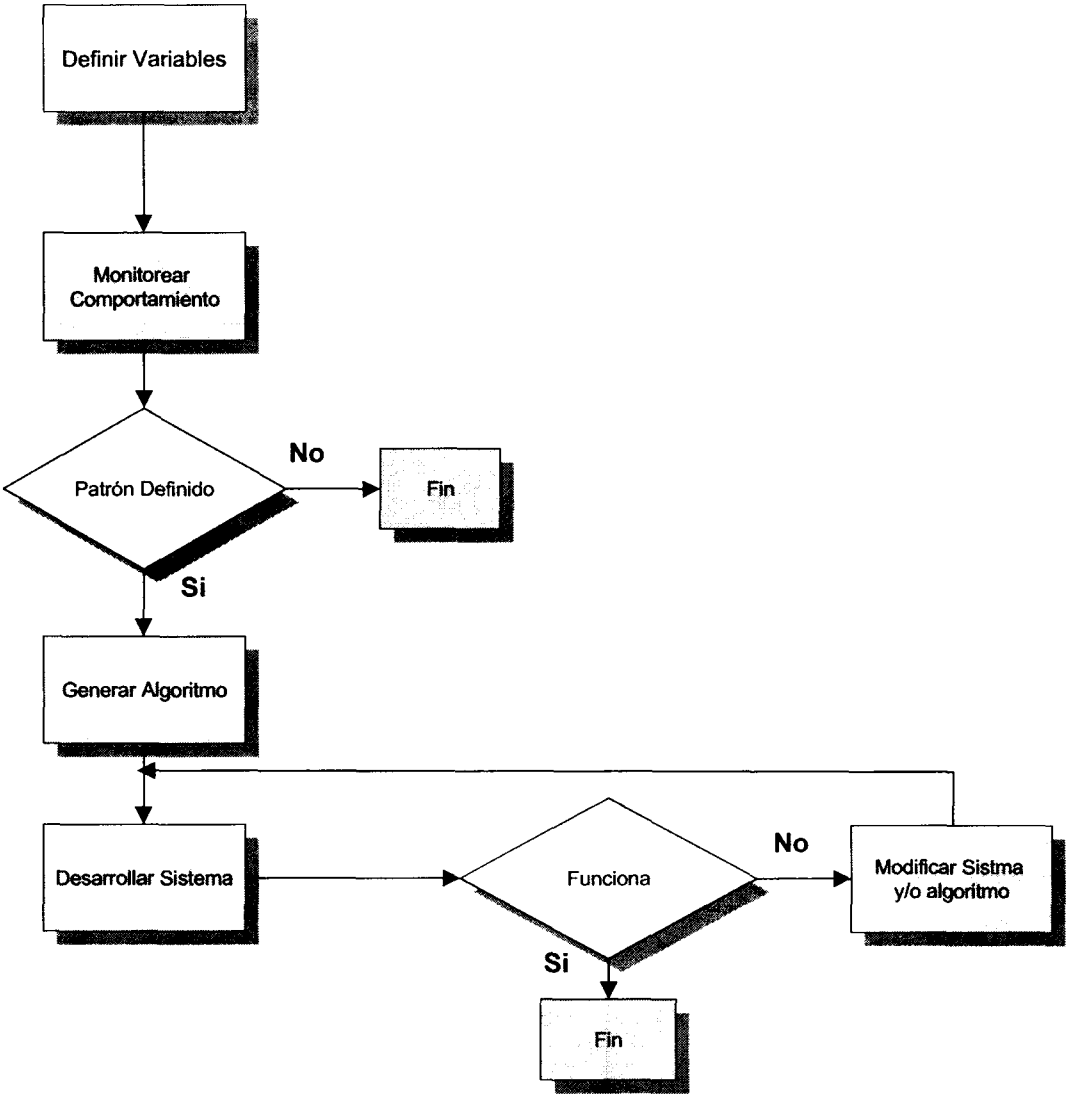
En este estudio se desea un algoritmo pueda ser usado en un sistema experto que tenga la habilidad de predecir cuando hay probabilidad de un problema en la red ocasionado por una falla, un incremento en el uso de ésta o en una ataque a la seguridad de algún elemento de la red. El algoritmo estará basado en la existencia de un comportamiento cíclicos en las variables observadas.

Una vez determinados los ciclos de repetición de las variables, será necesario mediante una técnica matemática obtener una ecuación que refleje el comportamiento de las variables a lo largo del tiempo.

2.2. Modelo Conceptual del Estudio

Este estudio esta conformado de varias etapas de trabajo de investigación, experimentación y desarrollo de sistemas. La figura 2.2 muestra un diagrama de flujo de las principales actividades las cuales se detallan a continuación.

Figura 2.2



Definir Variables: Antes de iniciar el estudio es necesario hacer una investigación bibliográfica y experimental de las posibles variables a monitorear que pueden resultar útiles para la identificación de problemas en la red ya sea por fallas de dispositivos o por ataques maliciosos.

Monitorear Comportamiento: Uno de los argumentos que soportan este estudio es el supuesto que las redes tienen comportamiento repetitivos en el tiempo y que este comportamiento puede ser predicho, una variación en este

comportamiento puede ser tomado como una posible falla o como el inicio de una. Antes de continuar con el estudio es necesario comprobar que realmente se tiene un patrón de comportamiento periódico. Esto se hará monitoreando las variables definidas en una red real. Para esta parte de la investigación se estará realizando un trabajo exploratorio de campo utilizando herramientas de administración de redes.

Patrón Definido: Si el resultado de monitorear las variables definidas en un red real es el comportamiento esperado se continuará con el estudio, si no es así se finalizará este concluyendo que las redes de datos actuales no tienen comportamientos predecibles.

Generar Algoritmo: Una vez que se tienen los patrones de comportamiento se puede diseñar un algoritmo. Para esta etapa del estudio de hará una investigación bibliográfica para el diseño inicial del algoritmo.

Desarrollar Sistema: Una vez que se tiene el algoritmo a utilizarse, se programa un sistema que lo ejecute. Esta etapa del estudio se apoyará en investigaciones bibliográficas para definir las herramientas de programación a utilizar y para el desarrollo del mismo.

Funcionamiento: Una vez desarrollada una gran parte del sistema será necesario probar la validez de éste y del algoritmo que lo soporta. En caso de que el desarrollo y/o el algoritmo requiera una modificación se hará en la siguiente etapa, si los experimentos demuestran un comportamiento aceptable se procederá a las conclusiones del estudio. Esta etapa del estudio estará apoyada en gran parte en trabajo de experimentación.

Modificar Sistema o Algoritmo: En esta etapa se modificará el sistema o el algoritmo dependiendo de la retroalimentación arrojada por la etapa anterior. Esta etapa junto con la etapa de Funcionamiento son iterativas entre ellas para asegurar que el resultado final sea el adecuado y se deberá ejecutar tantas veces sea necesario.

Fin: En esta etapa se generarán las conclusiones finales del estudio y del desarrollo del sistema.

Capítulo 3

Método

Introducción

Para todo estudio es necesario definir inicialmente el tipo de investigación que se hará. En este capítulo se define el tipo de investigación así como la forma en que esta se desarrollará.

3.1 Método

3.1.1 Tipo de Estudio

Este estudio consta de diferentes etapas en su desarrollo, cada una de las cuales utiliza un medio de investigación diferente. Como se vio en el capítulo anterior, el estudio cuenta con las siguientes etapas:

- Definición de Variables
- Obtención y validación del Patrón de Comportamiento
- Generación del Algoritmo
- Desarrollo del Sistema
- Validación

Para la "Definición de Variables" el estudio se apoyará en una investigación bibliográfica. Para la "Obtención y validación del Patrón de Comportamiento" se hará mediante la implementación de una herramienta de administración y monitoreo de redes, este tipo de investigación esta basada en cuasi experimentos ya que no se tiene un control de todas variables externas y el experimento no es en una ambiente controlado de laboratorio sino se hace en una red con tráfico y usuarios reales. La etapa de "Generación del Algoritmo" estará basada en los datos obtenidos y en investigación bibliográfica. Para el "Desarrollo de Sistema" no se estará haciendo ninguna investigación salvo la bibliográfica necesaria para la programación de éste. Finalmente la etapa de "Validación" es totalmente experimental, en esta etapa se realizarán diversos experimentos para asegurar el correcto funcionamiento del algoritmo y la correcta implementación del sistema. En algunos experimentos se utilizarán datos provenientes de ambientes y usuarios reales donde no se tiene todo el control de las variables, en otros experimentos se hará bajo ambientes de laboratorio donde se tiene el control de todas las variables.

En las etapas del proceso de investigación donde se requiera realizar experimentos, éstos se harán obteniendo datos de las variables definidas cada 5

minutos, si es necesario además 24 horas al día, 7 días a la semana. Con esto se tendrán muestras diarias que pueden ser comparadas entre sí y no solo obtener un patrón de comportamiento diario, sino también obtener un patrón semanal o mensual.

3.1.2 Situación inicial

La situación inicial que se tiene es un ambiente de red del cual no se puede predecir cuando existe la posibilidad de una falla debido a un mal funcionamiento de la red, a exceso de transmisión de información de un usuario, debido a un ataque de negación de servicio, como resultado de una explotación de una vulnerabilidad de seguridad por un usuario malicios o por cualquier causa anormal del comportamiento de algún dispositivo conectado a la red y que pueda causar una falla o una degradación en el servicio de ésta.

Aunque existen herramientas que permiten activar alarmas cuando ciertos límites en ciertas variables se han sobrepasado, estas herramientas no son adaptables a ambientes dinámicos de red y cada límite debe ser configurado previamente por el administrador de la red (Kauffels 1992). Esta configuración manual puede hacerse a todos los objetos monitoreados asignando el mismo valor a todos, lo cual es lo más sencillo pero también lo más ineficiente ya que algunos dispositivos u objetos pueden presentar un comportamiento particular en ambientes normales diferente a los otros dispositivos. También es posible en algunas herramientas asignar límites diferentes a los objetos, sin embargo esto debe ser manual por cada uno.

En el supuesto que se hace de que las redes tienen un comportamiento periódico y predecible, también se asume que este comportamiento es variable dependiendo de la hora del día, el día de la semana e incluso en el mes que se esté monitoreando. Es por ello que un análisis de la red de formal fija como los límites de alarmas del software tradicional no permite reaccionar ante cambios en un ambiente dinámico como las redes actuales.

Otro inconveniente del software actual es que genera demasiada información imposible de analizar sin herramientas especiales. Si se desea obtener un detalle de comportamiento más fino del que se obtiene por un simple sistema de contabilización y la red esta conformada por muchos dispositivos y muchas variables a analizar, el proceso se convierte en difícil y complicado para procesar la información de forma manual. Esto hace imprescindible la utilización de herramientas automáticas de procesamiento de datos (Barufi, Milano y Montanari 2001).

3.1.3 Manipulación de variables

El primer paso es obtener un comportamiento general de las variables seleccionadas y a partir de éste generar un algoritmo. Se propone que una vez que se obtenga una cantidad de información adecuada, se podrá hacer un análisis estadístico de estos datos con la finalidad de encontrar las relaciones entre el comportamiento de unas variables con respecto a otras variables en situaciones de operación normal de la red. El objetivo es encontrar un modelo matemático que explique el comportamiento de la red. Este modelo matemático puede ser tan complejo como un conjunto de ecuaciones o tan simple como el promedio de los valores obtenidos.

3.1.4 Situación final

Una vez obtenido el modelo matemático del comportamiento de la red en situaciones normales es necesario crear un mecanismo basado en software que tome los datos de las variables monitoreadas en los agentes y lo procese de acuerdo a un algoritmo basado en el modelo matemático. El mecanismo de petición de información puede ser la misma estación monitora NMS que se usó para recopilar los datos para la formulación del modelo.

El procesamiento de los datos en el algoritmo debe dar como resultado una alarma si el modelo detecta que la red se está comportando de forma diferente a la habitual o una simple confirmación si el comportamiento es normal.

Este mecanismo se probará junto con el algoritmo haciendo simulaciones de comportamiento normal y anormal de la red. El resultado deberá ser enviar la confirmación o una alarma según sea el tipo de simulacro.

Capítulo 4

Análisis de protocolos y herramientas de monitoreo

Introducción

Este capítulo tiene como objetivo analizar la arquitectura de monitoreo recomendada por el ISO-OSI conocida como TMN, los protocolos disponibles, las aplicaciones y herramientas actuales para el monitoreo de redes.

Los protocolos que se analizarán son SNMP, RMON, CMIP y NetFlow; de cada protocolo se analizará su funcionamiento y sus componentes. Las herramientas que se analizarán son Cricket, RRTool, HPOV, NetSNMP y herramientas para el análisis de flujos. Para las herramientas el análisis consistirá en identificar sus funciones, los problemas que pueden resolver, su origen y sus relaciones con los protocolos analizados.

Finalmente en base a un criterio basado en las necesidades del estudio, se escogerán los protocolos y las herramientas que puedan ser útiles para el desarrollo de éste. El criterio de selección además deberá incluir la factibilidad técnica y económica de la implementación de la herramienta.

4.1. Modelo TMN

Para resolver los problemas relacionados con la variedad de los protocolos de comunicaciones que pueden existir en los proveedores de servicios de telecomunicaciones y la administración de los dispositivos en la red sin comprometer la operación de estos, en 1988 la ITU inicia la definición de una arquitectura de administración de telecomunicaciones llamada Telecommuncations Management Network (TMN) la cual se ve terminada en 1992. Esta arquitectura fue definida para soportar un modelo de administración centralizado o distribuido (Raman 1999).

4.1.1 Arquitectura de TMN

La arquitectura de TMN puede ser visto o dividido en tres modelos diferentes (Chen, Kong 2000):

Modelo de arquitectura funcional: Define los componentes de TMN y las interfaces entre estos componentes

Modelo de arquitectura física: Identifica los componentes físicos de TMN y la conectividad entre estos componentes

Modelo de arquitectura de información: Provee una metodología orientada a objetos de redes de telecomunicaciones y servicios y a las funciones de administración.

El modelo de arquitectura funcional describe los diferentes componentes funcionales de TMN, las interrelaciones entre los componentes se conocen como puntos de referencia. A continuación se describen los componentes de TMN (Chen, Kong 2000):

Workstation Function (WSF): Soporta la traslación requerida para presentar la información de los sistemas de administración a un usuario humano y para trasladar las requisiciones del usuario a las representaciones de las entidades TMN (Raman 1999).

Operations Systems Functions (OSF): Soporta el procesamiento de la información de administración para monitorear, coordinar y controlar operaciones de telecomunicaciones (Chen, Kong 2000). Las actividades del OSF son obtener información sobre alarmas de estado de una entidad administrada, llevar a cabo las actividades de administración de la información tomada de los dispositivos administrados como por ejemplo hacer la correlación de las causas de una falla y dirigir a las entidades administradas para llevar a cabo acciones correctivas (Raman 1999).

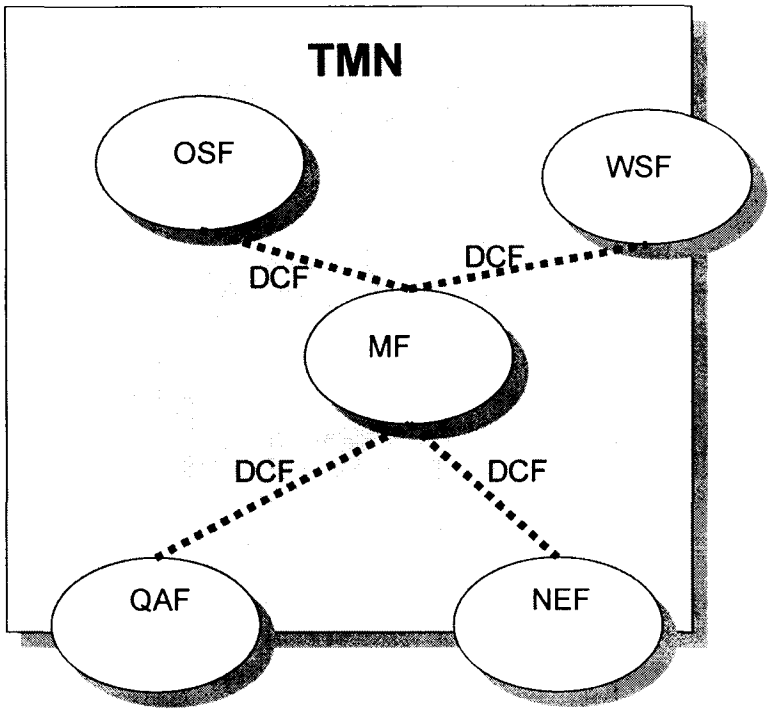
Network Element Function (NEF): Este componente funcional representa las capacidades de administración soportadas por los dispositvos de red.

Mediaton Function (MF): Este componente funciona facilita el paso de información entre el OSF y el NEF o QAF cuando puntos diferentes de referencia internos son usados. Provee soporte para guardar, adaptar, filtrar, definir límites y condensar información.

Q Adapter Function (QAF): Cuando la red o los dispositivos de ella soportan TMN, el OSF usa NEF para acceder a esta funcionalidad. Si estos dispositivos no soportan TMN, OSF require de una interfaz para acceder estos dispositivos no-TMN. El QAF el cual provee la traducción de información y protocolos requerida por el OSF.

La figura 4.1 muestra la arquitectura funcional, sus componentes y su interconexión. Se agrega además el concepto de Data communication function (DCF) la cual esta encargada de interconectar los componentes, las líneas punteadas que lo identifican sirven para denotar que no es parte de un bloque de función de la arquitectura, pero en caso de una implementación física es necesario considerarlo. También se notan algunos bloques que se encuentran fuera de TMN, esto es porque estas funciones requieren de interacción con otros dispositivos o protocolos fuera de la definición de TMN (Raman 1999).

Figura 4.1



El propósito de la arquitectura física es identificar y correlacionar las entidades físicas con los componentes funcionales, los componentes de esta arquitectura incluyen (Chen, Kong 2000):

Data Communication Network (DCN): Esta es la red física de comunicaciones que soporta la interconexión de todos los dispositivos en un sistema TMN.

Network Elements (NE): Son dispositivos de telecomunicaciones en el ambiente TMN. La función de NEF es comúnmente soportada por estos elementos. En caso de no hacerlo es necesario que exista un QAF que haga la interacción con este dispositivo.

Mediation Device (MD): Este es requerido entre dos componentes internos de TMN si diferentes interfaces Q son usadas. Este proveerá funciones de mediación que permita que la información se traduzca.

Q Adapter: Este dispositivo permite a dispositivos no-TMN ser parte de un sistema TMN. Este soporta las funciones de adaptadores Q. En muchos casos es una implementación de software que soporta esta traducción

Operation Systems (OS): Es un software usado para monitorear, coordinar y controlar las operaciones de telecomunicaciones y administración.

Workstations (WS): Estas son usadas por los operadores para acceder al ambiente de administración de TMN, estos soportan WFS.

La arquitectura de información esta basada en como estructurar la información que es manejada entre los dispositivos. La información que se maneja es usada para monitorear y para controlar recursos (Raman 1999).

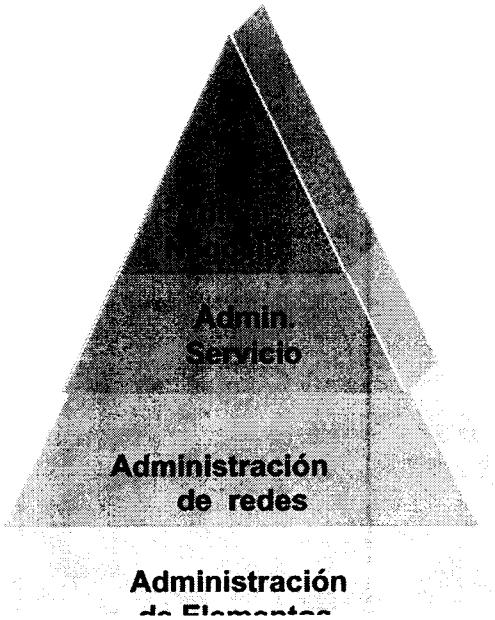
4.1.2 Niveles de Administración de TMN

Desde un punto de vista operacional y práctico TMN debe ser visto de forma diferente y catalogarlo en varios dominios cada uno con sus propios objetivos y sus propios requerimientos. Esta categorización tiene como resultado el dividir el ambiente de administración en varios niveles. Estos niveles se observan en la figura 4.1 (Chen, Kong 2000)

La capa de administración de negocio tiene la responsabilidad del sistema. En este nivel cae el proceso de toma de decisiones, mantenimiento de agregación de datos, soporte técnico para administración de presupuestos y soporte para administración de fuerza de trabajo.

La capa de administración de servicio tiene como responsabilidad cumplir con los acuerdos de servicio con los clientes. Sus procesos son el control y manejo de interfaces para los clientes de las proveedores de servicio, el manejo de la interacción con los clientes, manejo de la interacción entre servicios y mantener la calidad de los datos.

Figura 4.2



La capa de administración de red es responsable de la administración de la red con el soporte de la capa de administración de elementos. Sus responsabilidades son controlar y coordinar todos los dispositivos de red; proveer, terminar o modificar las capacidades de red de los clientes; mantener las capacidades de la red; mantener datos de la red e interactuar con la capa de servicios en desempeño, uso y disponibilidad de la red.

Finalmente la capa de administración de servicios se encarga de manejar cada elemento de la red de forma individual o de grupo. Sus funciones son controlar y coordinar un grupo de elementos de red de forma individual o colectiva, mantener estadísticas, logs y otros datos de los elementos.

4.2 Network Managment System (NMS)

Como se vio anteriormente (Kauffels 1992) el NMS es una herramienta automatizada que ayuda al personal de administración de la red ha desempeñar mejor sus tareas. Entre sus funciones están las de recibir y almacenar la información de monitoreo como lo son las de las variables de los agentes SNMP (Subramanian 2000).

El NMS esta compuesto de los siguientes elementos: Hardware, Sistema Operativo, Servicios de Aplicación de Core, Servicios comunes de SNMP, Servicios específicos de NMS de fabricante. A continuación se muestra una tabla que ejemplifica cada uno de los elementos de un NMS con aplicaciones conocidas en el mundo real (Subramanian 2000).

Componente	Servicio	Ejemplo
Hardaware	Procesador Monitor Comunicaciones	Sun HP 9000 Basado en Intel
Sistema Operativo	Servicios de Sistema Operativo	Solaris Linux BSD MS Windows NT
Aplicaciones de servicios de core	Diplay Interfaz Gráfica Base de Datos Genración de reportes Servicios de comunicación	OpenView SuNet Manager

Servicios Comunes de SNMP	Mensajes de SNMP versión 1,2 y 3 Manejo de MIBs Aplicaciones básicas de SNMP	SNMPc Net-SNMP HPOV node manager
Servicios de NMS específicos de fabricante	Manejo de MIB Aplicaciones de SNMP	Cisco Works Transcend Spectrum Element Manager

Las funciones requeridas por un NMS incluyen (Chen, Kong 2000):

- Representación de la red: Usando un modelo estándar
- Administración de conexión de fin a fin: Incluye ruteo entre dominios y configuración de equipo
- Administración de Fallas: Las fallas reportadas por el equipo administrado deben ser traducidas en el modelo general de la red y propagadas a niveles superiores.

4.3 Protocolos

4.3.1 SNMP

El Simple Network Managment Protocol (SNMP) fue desarrollado por el IETF como el protocolo para monitorear y administrar dispositivos bajo redes TCP/IP. Hasta el momento existen tres versiones del protocolo, el primer desarrollo e implementación conocido como SNMPv1 y definido en el RFC 1157, una segunda versión que no es estándar de Internet conocida como SNMPv2 y definida en los RFC 1901 y RFC 3417. Finalmente la versión actual conocida como SNMPv3 y definida en los RFC 3417, RFC 3412 y RFC 3414 además del estándar 62 (STD0062) de Internet (Harrigton et at 2002).

Como lo definido en el estándar de Internet STD0062 (et at Harrington 2002), un sistema de SNMP tiene estos tres elementos:

1) Muchos nodos, cada uno con una entidad de SNMP que contiene programación para responder a comandos y para originar notificaciones para las aplicaciones quienes tiene acceso a los instrumentos de administración. Estos nodos comúnmente se les llama "Agentes".
2) Al menos una entidad de SNMP que contenga generadores de comandos y/o aplicaciones para la recepción de notificaciones. Esta entidad comúnmente se le llama "Administrador" o NMS.

3) Un protocolo de administración, usado para transportar información de administración entre las entidades de SNMP.

Para SNMPv1-3, éste consiste de tres componentes (León-Garcia, Widjaja 2000):

4) Un marco conceptual que define las reglas para describir el manejo de información, este se conoce como Estructura de Manejo de Información (Structure Managment Information SMI). A partir de la versión 2 de SNMP, se utiliza la SMIv2.
5) Una base de datos virtual conteniendo información acerca del dispositivo manejado, ésto se conoce como Base de Manejo de Información (Managment Information Base MIB)
6) Un protocolo para comunicación entre un manejador y un agente de un dispositivo administrado, este se conoce como el Protocolo Simple de Administración de Redes (Simple Network Managment Protocol SNMP)

Un sistema basado en SNMP soporta tres importantes tipos de comando: GET, SET y EVENT. El comando GET permite al NMS solicitar información a un agente sobre una variable u objetos contenidos dentro de un MIB; el comando SET permite al NMS escribir o modificar en el agente un valor de una variable u objeto dentro de un MIB. EVENT permite al agente enviar información al NMS como forma de una alarma no requerida, esto es útil en situaciones de falla donde es necesario informar al administrador de forma inmediata que ha sucedido una falla (Kauffels 1992). Estos comandos se traducen en las siguientes operaciones: Get-request, Get-next-request, Get-response, Set-request y trap (Xiao 2002).

Cuando inicia la operación de SNMP, esta inicia con la operación de "Get-request" del proceso administrador al proceso agente el cual contesta con un "Get-response". El comando "Get-next-request" es muy similar al "Get-request" con la diferencia del que el Objeto Identificador siempre es el siguiente en el árbol del MIB. Por ejemplo si el Identificador de Objeto anterior es ifInOctets (1.3.6.1.2.1.2.2.1.10) como puede verse en la figura 4.4, al efectuar un "Get-next-request", el objeto que se pedirá es ifInUcastPkts (1.3.6.1.2.1.2.2.1.11) (Subramanian 2000). El comando "Set-request" es utilizado para asignar un valor en el agente. Cuando un agente recibe el "Set-request" altera el valor actual del Identificador de Objeto enviado con el valor ligado al objeto. El trap es un mensaje enviado por el agente al administrador para informar sobre ciertas condiciones en el agente o cambio en éstas (Analyser 2001).

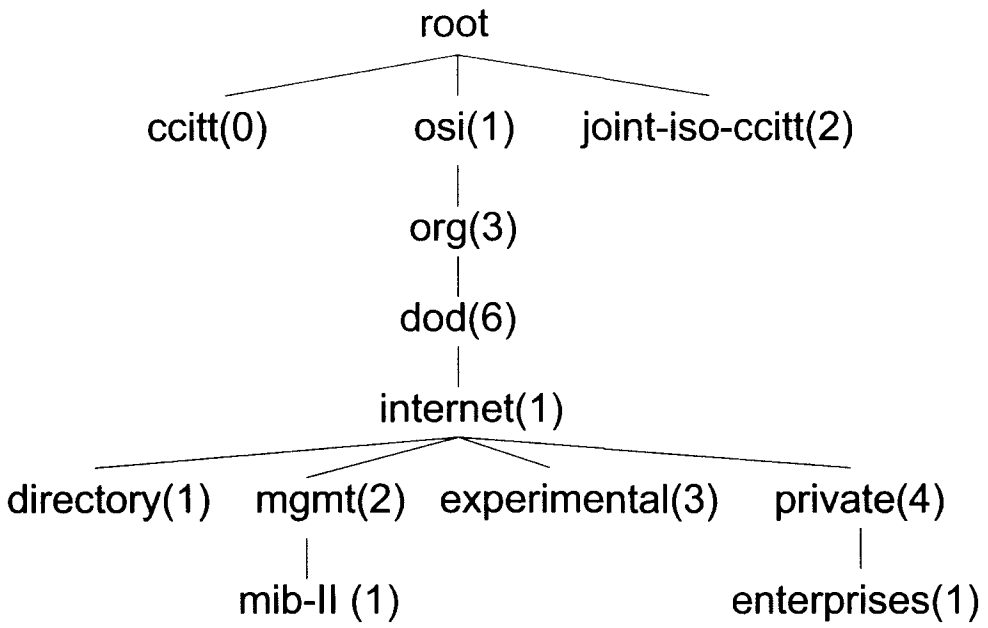
La SMI es una base de datos pública, libre, estándar y universal que especifica un objeto o variable MIB (Wolcott 2000). Para SNMPv2 y SNMPv3 se utiliza la SMIv2, esta nueva versión tiene como ventajas que maneja contadores más grandes, es más clara y más consistente y tiene un mejor manejo de las operaciones (Case 2002).

Gracias al SMI, la estructura del MIB es jerárquica en forma de árbol. Las hojas del árbol son los objetos o variables que se administrarán (Stallings 1997). De ésta forma cada variable tiene un identificador jerárquico que en los agentes de SNMP es una cadena de números decimales separados por puntos y cada número tiene una correspondencia en nombre para un mejor entendimiento por parte de los humanos, a este identificador se la llama Identificador de Objeto (Object Identifier OID) (Wolcott 2000).

Un ejemplo de la estructura jerárquica del MIB puede observarse en el figura 4.3 y 4.4. En la figura 4.3 se observa el árbol jerárquico iniciando por la raíz (root). A continuación se describen los niveles del árbol MIB (León-García, Widjaja 2000):

- La rama de directory(1) esta reservada para uso futuro.
- La rama de mgmt(2) es identificada para objetos estándares que estan registrados por el Internet Assigned Numbers Authority (IANA).
- La rama de experimental(3) es para objetos que estan bajo experimentación en grupos del IETF. Si el objeto se convierte en estándar, entonces debe moverse a la rama de mgmt(2)
- La rama de private(4) es para objetos definidos por una sola entidad, usualmente vendedores de equipo. Este además tiene una rama llamada enterprise(1), que aloja a las compañías que registran sus objetos de redes.
- La rama de security(5) es para objetos relacionados con seguridad.
- La rama de snmpv2(6) esta reservada para propósitos de depuración del SNMPv2.

Figura 4.3



(León-Garcia, Widjaja 2000)

En la figura 4.4 se observa el árbol para una variable a monitorear que posiblemente se use mucho, ifInOctets. Esta variable forma esta conformado de la siguiente manera que se le conoce como el Identificador de Objeto (OID):

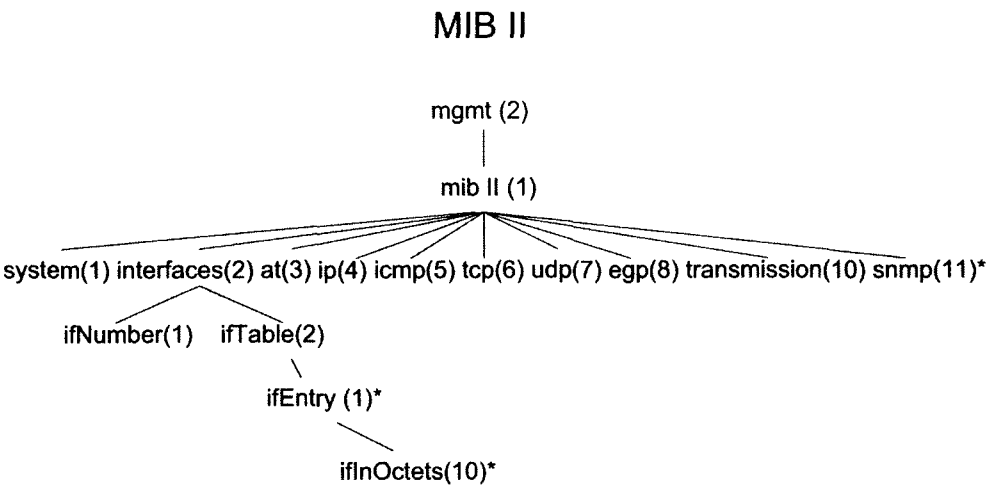
1.3.6.1.2.1.2.2.1.10

En lenguaje más humano se lee como:

osi.org.dod.internet.mgmt.mibii.interfaces.ifTable.ifEntry.ifInOctets

Este tipo de cadenas jerárquicas de identificación son las que permiten al software de administración de redes y a los dispositivos agentes la posibilidad de comunicarse entre sí e intercambiar información. Además, cada dispositivo enumera las interfaces u objetos que se le pueden monitorear. De esta forma, cuando un software de monitoreo desea acceder al valor de los contadores para cierta variable en un dispositivo remoto solo tendrá que solicitar la información identificando con el OID la variable de la que desea conocer el valor y el número de identificador de interfaz u objeto interno del dispositivo.

Figura 4.4



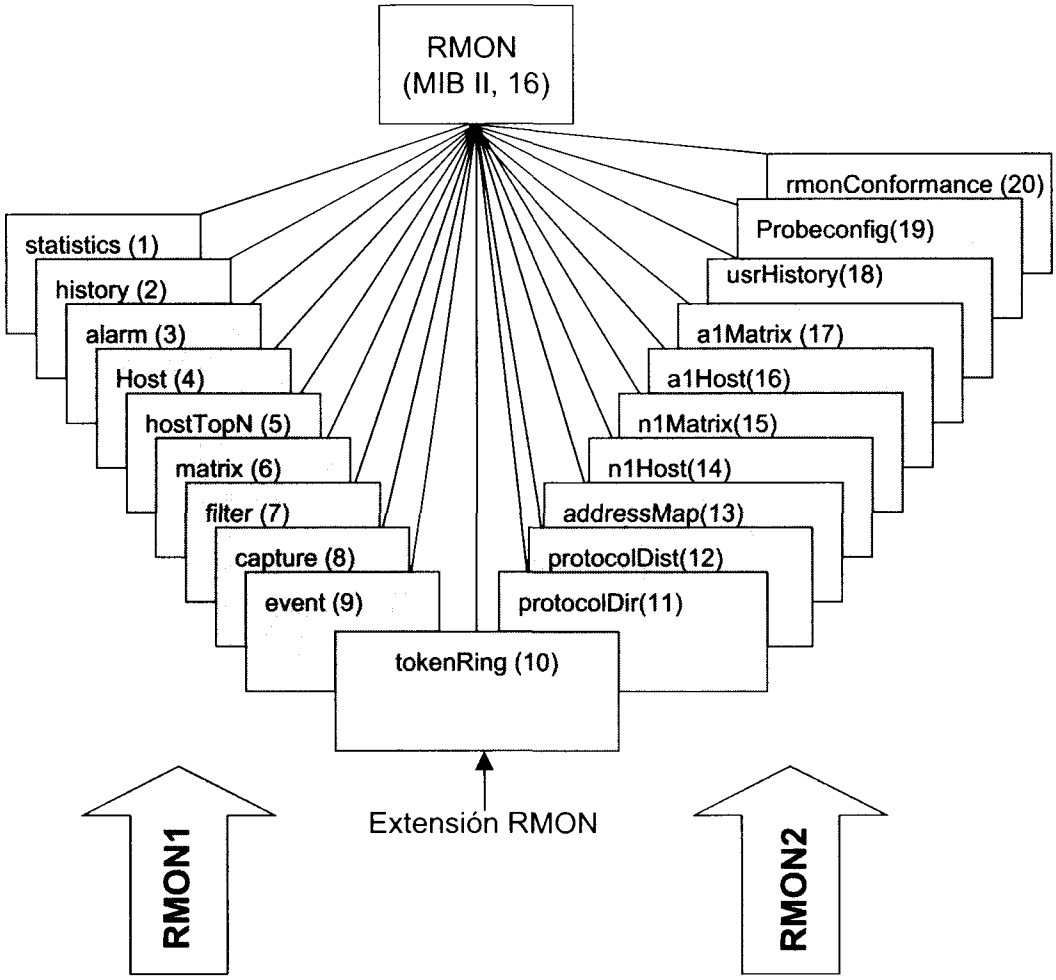
* Se han suprimido algunas entradas para facilitar el despliegue

4.3.2 RMON

Este protocolo significa Remote Network Monitoring, actualmente existen dos versiones, RMON1 y RMON2, cada una con diferentes capacidades que se explicarán posteriormente. Los dispositivos con RMON habilitado ejecutan un monitoreo pasivo de la red "sniffeando", analizando y guardando el comportamiento del tráfico dentro de la red. A esta acción de monitoreo se le llama "probing", es por ellos que a los dispositivos con RMON comúnmente se les llama "probes" (Subramanian 2000).

RMON1 fue desarrollado para su uso en Ethernet y está definido en el RFC 1513. Aunque es muy útil tiene la restricción de que sólo analiza información de capa 2. En la figura 4.5 puede verse el árbol de RMON (grupo 16 de MIB-II) en el SMIv2 y sus respectivos grupos.

Figura 4.5



A continuación se da una descripción de los grupos de RMON1:

Statistics: Provee estadísticas de nivel de enlace

History: Colecta información estadística de forma periódica y la guarda para su envío posterior.

Alarm: Genera eventos cuando la muestra de datos sobrepasa los límites establecidos.

Host: Guarda datos estadísticas basados en hosts

Host Top N: Calcula el Top N respecto a host en la categoría especificada

Matrix: Guarda información estadística basada en el tráfico entre dos hosts

Filter: Ejecuta funciones de filtrado que habilita la captura de los parámetros deseados.

Packet Capture: Provee captura de paquetes
Event: Controla la generación de eventos y notificaciones
Token Ring: Información específica para Token Ring

La arquitectura e RMON2 es igual a la de RMON1 con la diferencia de que RMON2 puede ver información relativa a los protocolos de capa 3 como IP. El árbol de RMON2 puede verse también en la figura 4.5. Los grupos (11 al 20) se describen a continuación:

Protocol directory: Inventario de protocolos
Protocol distribution: Estadísticas en octetos y paquetes
Address map: Resolución de MAC-address a direcciones de capa 3
Network layer host: Tráfico de datos de cada host y a cada host
Network layer matriz: Tráfico de datos por cada par de hosts
Application layer host: Trafico de Datos por protocolo y de cada host y a cada host.
Application layer matrix: Tráfico de datos por protocolo por cada par de hosts.
User history collection: Datos históricos por usuario, alarmas y datos estadísticos.
Probe configuration: Configuración de parámetros de probe
RMON conformance: Compatibilidad de RMON2 MIB y compatibilidad de grupos.

4.3.3 Common Management Information Service (CMIS)/Common Management Information Protocol (CMIP)

La arquitectura y las funciones definidas para TMN y analizadas anteriormente requieren de protocolos para llevar a cabo las funciones requeridas. Common Management Information Service (CMIS) y Common Management Information Protocol (CMIP) son los vehículos para la convergencia de operaciones sobre objetos, notificaciones de objetos, resultados de errores en las operaciones y notificaciones sobre entidades (Chen, Kong 2000).

En la figura 4.5 puede observarse el modelo de funcionamiento de CMIS. Este modelo es muy similar al empleado por SNMP, por un lado existe un "Invoker" que genera una petición "Request" hacia un objeto "Performer", el cual tiene un agente (función NEF de TMN) que contesta la petición dependiendo del objeto que se le esté solicitando. Por otro lado, cuando un agente requiere hacer una notificación, éste, ahora "Invoker" generará una notificación la cual llegará a la estación monitora (WS en TMN) "Performer" la cual generará una confirmación "Acknowledgment" (Raman 1999).

El comportamiento anterior se lleva a cabo mediante las siguientes funciones (Chen, Kong 2000):

Operaciones sobre Objetos
m-create: Solicita la creación de un nuevo objeto y solicita confirmación.
m-delete: Solicita el borrado de un objeto y solicita confirmación.

Operaciones sobre atributos
m-get: Solicita los valores de los atributos de un objeto administrado y solicita confirmación.
m-set: Reemplaza los valores de los atributos de un objeto administrado, la confirmación es opcional.

Otras (Raman 1999)
m-event-report: Reporta la ocurrencia de un evento a otro sistema

Cada uno de los objetos controlados por un agente son guardados en una "Management Information Base" (MIB) el cual es un árbol estructurado y jerárquico conocido como árbol contenedor de objetos. El contenedor comienza con un nodo raíz y cada objeto es un nodo dentro del contenedor (Chen, Kong 2000). Este concepto de MIB es muy similar al MIB de SNMP.

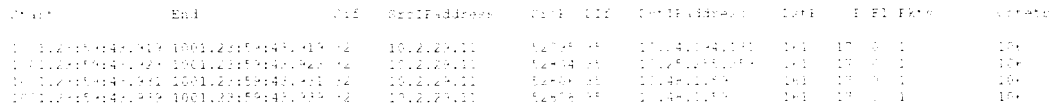
4.3.4 NetFlow

NetFlow es una tecnología desarrollada en 1996 por Darren Kerr y Barry en Cisco Systems e inicialmente ideada como una tecnología de conmutación rápida mediante el análisis de los paquetes de capa 3 y 4 para la creación de flujos de tráfico. Un flujo de tráfico es una secuencia de paquetes en una comunicación entre dos nodos en la cual se mantiene la información de dirección de IP fuente, dirección de IP destino, puerto fuente, puerto destino, tipo de protocolo de capa 3, TOS (ahora DSCP) y la interfaz de entrada al equipo de red. El día de hoy NetFlow es la principal tecnología de contabilización en las redes contestando las preguntas de quien, que, a donde, cuando del tráfico en la red (Cisco 2003)

Actualmente existen 5 versiones de NetFlow, cada uno con sus características y sus aplicaciones, cada una se describe a continuación:

Versión 1: La versión original de NetFlow.
Versión 5: Es la mas utilizada y común. En su registro maneja información de flujo como conteo de paquetes, conteo de bytes, dirección fuente, dirección destino, puerto fuente, puerto destino, interfaz de entrada, interfaz de salida, Type of Service, TCP Flags, Protocol, tiempo de inicio de flujo, tiempo de terminación de flujo, Next Hop Address, Sistema Autónomo de origen, Sistema Autónomo destino, máscara fuente,y máscara destino. Una ejemplo de flujos de NetFlow versión 5 puden verse en la Tabla 4.1.

Tabla 4.1



Datos Red ITESM

Versión 7: Similar a la versión 5 pero el registro no incluye información de AS, interfaz, TCP Flag y TOS ya que fue diseñada para ser utilizada en switches de core o distribución de las LANs. Tampoco soporta información de IP Multicast. Un ejemplo de flujos de versión 7 pueden verse en la Tabla 4.2.

Tabla 4.2

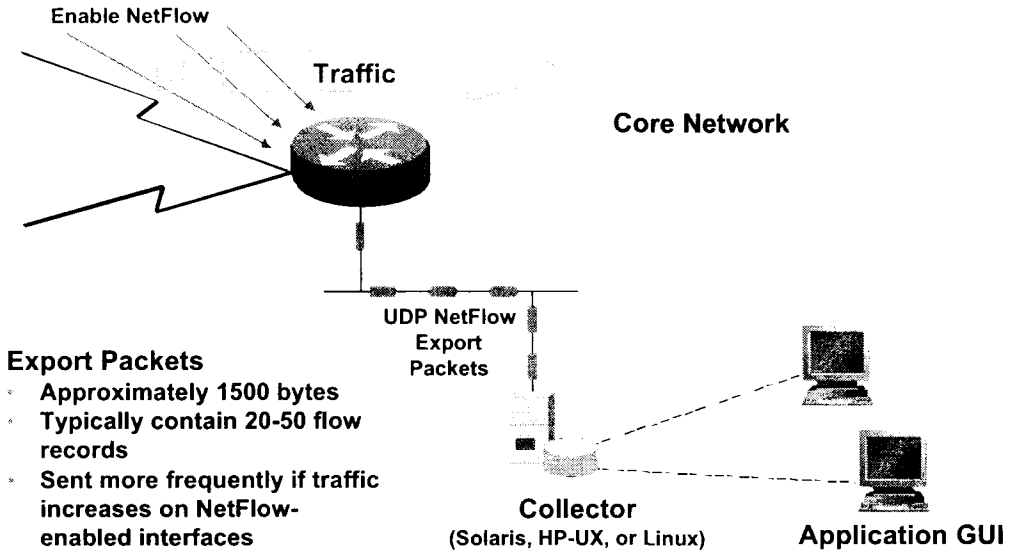
Start	End	Sif	SrcIPaddress	SrcP	DIf	DstIPaddress	DstP	P	Fl	Pkts	Octets
1004.00:16:17.400	1004.00:16:17.581	0	196.3.80.42	65423	0	10.2.25.72	80	6	0	5	808
1004.00:16:17.900	1004.00:16:17.900	0	205.158.174.201	53	0	10.7.2.2	53	17	0	1	152
1004.00:16:17.159	1004.00:16:17.160	0	200.34.0.50	1875	0	10.56.171.14	1494	17	0	1	404

Versión 8: Incluye una variedad de esquemas de agregación que permite la reducción de uso de recursos. El router sumariza información de los flujos como prefijo fuentes y destino, por ejemplo en lugar de enviar un registro de la transmisión de la IP 10.1.1.1 y otro de la 10.1.1.2 (suponiendo que están en la misma subred) solo enviará la información de los flujos sumarizados de ambas direcciones IP como si fuera un solo flujo.

Versión 9: Es la versión más flexible ya que usa un formato extensible y exportable que soporta la adición de información de nuevas tecnologías como MPLS, Multicast y BGP. Las versiones anteriores de NetFlow son inflexibles y no permiten agregar más información, asi que cuando una nueva tecnología surgía era necesario crear una nueva versión de NetFlow para adicionar nueva información y adecuar las aplicaciones para el soporte de los nuevos campos. Esta versión además ha sido aprobada para ser un estándar del IETF.

La contabilización se hace de entrada a las interfaces de los equipos de redes, por lo tanto se tiene información unidireccional de los flujos, por tanto si se desea tener información bidireccional es necesario activar NetFlow en las interfaces de entrada por donde circule el tráfico. La figura 4.6 muestra el comportamiento de NetFlow y como es generada la información.

Figura 4.6



Cisco Systems 2003

Cuando un flujo de tráfico entra por la interfaz de un equipo de red (PE) esta generará una contabilización de este flujo con los campos anteriormente descritos. Una vez que el flujo ha terminado y dependiendo del tipo de agregación que se haya configurado o que la versión de NetFlow soporte el equipo de red enviará esta información a un colector. El PE tiene entonces la función de creación de caché, sumarización o agregación de información y de exportar los datos. El colector recibe la información la cual además de colectarla la puede filtrar, agregar, guardar y manejar sistemas de archivos Finamente, mediante diversas aplicaciones se pueden presentar los datos al usuario. Como se indica en la figura, los paquetes de NetFlow miden 1500 bytes y contienen información de aproximadamente 20 o 30 registros y se envían mediante UDP.

Dependiendo de la capa de en la arquitectura de red donde se colecte información de NetFlow (Acceso, Distribución o Core) es el uso que se le puede dar a ésta Por ejemplo en la capa de acceso se puede utilizar para mitigar ataques de negación de servicio al detectar patrones anómalos de tráfico, monitoreo de direcciones de IP de usuarios, monitoreo de aplicaciones; en la capa de distribución se puede usar para monitoreo de acuerdos de interconexión, cobro por uso y cobro diferenciado en calidad de servicio, hora del día de la transmisión, tipo de aplicación, etc; en la capa de core se puede usar para ingeniería y análisis de tráfico.

4.4 Herramientas

4.4.1 RRDTool

RRDtool fue escrito por Tobias Oetiker junto con las contribuciones de mucha gente alrededor del mundo y se distribuye bajo la licencia GNU de fuente abierto. RRDtool se refiere a "Herramienta de Base de datos Round Robin" (Round Robin Database tool). El término "Round Robin" se refiere a que se utiliza una cantidad fija de datos y por consecuencia de un tamaño fijo. En la base de datos, existe un apuntador que se fija a un elemento de la base de datos, al escribir el siguiente elemento el apuntador se mueve, una vez que se han utilizado todos los elementos, estos se reciclan y se reutilizan, de esta forma la base de datos no crece en tamaño y por lo tanto no requiere de mantenimiento (Bogaerdt 1999).

RRD puede guardar y desplegar series de datos, por ejemplo ancho de banda en la red, temperatura en un cuarto de computadoras, carga promedio de un servidor, etc. Los datos son guardados de una forma muy compacta y pueden presetar gráficas muy útiles, puede ser usado con simples scripts en shell o Perl o por medio de "frontends" que saquen información de los dispositivos de red y que usen una interfaz amigable.

RRDTool es una herramienta muy poderosa que tiene algunas ventajas sobre otras bases de datos, por ejemplo: Además de guardar datos permite la creación de gráficas, lo cual lo hace una herramienta de "frontend", otras bases de datos solo almacenan información. Las bases de datos linealies ponen los datos al final, esto hace que entre más datos reciban, estos crecen, RRDtool es una base de datos cíclica, de esta manera cuando se alcanza el nivel máximo de datos, el siguiente dato se sobrescribe sobre el dato más antiguo. Otras bases de datos guardan los valores que reciben, RRDtool puede ser configurado para calcular el radio de cambio entre el valor actual y el anterior y guardar esta información. Finalmente otras bases de datos guardan valores cuando se le suplen, RRDtool está estructurada de tal forma que requiere de datos en intervalos específicos de tiempo, esto permite el cálculo de promedios en esos intervalos además de algunas otras operaciones (Patel 2003).

4.4.2 Cricket

Cricket es un sistema de alto desempeño extremadamente flexible para monitorear series de datos de monitoreo. Cricket se desarrollo expresamente para ayudar a los administradores de redes a visualizar y entender el tráfico dentro de sus redes, sin embargo puede ser usado para otras muchas aplicaciones en otras áreas.

Existen dos elementos funcionales en Cricket, un colector y un graficador. El colector corre desde un proceso ejecutado en un intervalo de tiempo definido y guarda los datos en una base de datos manejada por el RRD Tool. Posteriormente cuando se desean leer los datos esto se despliegan mediante una página basada en web. Cricket está escrito en Perl y se distribuye bajo la Licencia Pública General GNU.

Cricket fue desarrollado para su uso sobre el sistema operativo Solaris corriendo el servidor de web Apache, sin embargo se ha portador para ser ejecutado en otros sistemas operativos como Linux, variantes de BSD, HP-UX, Windows NT y/o Windows 2000. Cricket fue originalmente escrito por WebTV Networks Inc. y posteriormente publicado bajo la licencia de GNU (Sourceforge Cricket 2002).

El sistema de configuración de Cricket es jerárquico, un conjunto completo de archivos de configuración es llamado "config tree". Las configuraciones que van a ser usadas una y otra vez pueden ser guardadas en lo más alto del árbol, entonces ésta será usada en todos los objetos debajo. Información más específica puede ser guardada más cerca de los objetos. Los archivos son agrupados en directorios y posteriormente procesados en un orden predecible y de acuerdo a estos directorios. Mientras cada directorio es procesado, el estado del sistema es salvado y restaurado, de esta manera, los cambios hechos a un subdirectorio no afectan a otros subdirectorios.

4.4.3 UCDSNMP/Net SNMP

UCD-SNMP, un software de licencia de fuente abierto desarrollado inicialmente por la Universidad de Carnegie Mellon y la Universidad de California at Davis y posteriormente distribuido a través bajo la Licencia Pública General GNU. Se integro en el proyecto de SNMP de Sourceforge bajo el nombre de Net SNMP (Sourceforge Net SNMP 2002).

Net SNMP puede definirse como un conjunto de herramientas relacionadas al Simple Network Management Protocol que incluyen:

1) Una agente extensible: Permite instalar un agente de SNMP en una estación de trabajo y monitorear desde un NMS algunos objetos de MIB-II
2) Librerías SNMP: Todo el software necesario para el funcionamiento de SNMP.
3) Herramientas que solicitan información de agentes SNMP: Una serie de programas para solicitar información SNMP.
4) Herramientas para recibir y manejar traps de SNMP: Los traps que generan los agentes de SNMP son manejados por un programa llamado “snmptrapd”.

5) Una versión del comando "netstat" usando SNMP.
6) Un navegador de MIB en Perl y con interfaz GUI

De las herramientas que "pollean" los agentes MIB se tienen:

1) snmptranslate: Permite conocer el árbol MIB. E una herramienta muy poderosa que permite ver el árbol MIB desde la línea de comandos:

```
%snmptranslate .1.3.6.1.2.1.1.3.0 SNMPv2-MIB::sysUpTime.0
```

2) snmpget: Trae datos desde un agente, tiene como entrada el host, valores de serguridad y el OID:

```
%snmpget -c demopublic -v 2c test.net-snmp.org system.sysUpTime.0
system.sysUpTime.0 = Timeticks: (586731977) 67 days, 21:48:39.77
```

3) snmpgetnext: Es similar a snmpget, es usado para traer el siguiente OID en el árbol MIB. En lugar de regresar el dato que se pide, regresa el valor del siguiente OID en el árbol:

```
%snmpgetnext -v 2c -c demopublic test.net-snmp.org system.sysUpTime.0
system.sysContact.0 = Wes Hardaker wjhardaker@ucdavis.edu
```

4) snmpwalk: Ejecuta una serie de getnet de forma automática y se detiene cuando no existe un rango mayor en el árbol del OID que origninalmente se especificó. Es muy útil para traer toda la información guardada en un agente para un grupo de MIB:

```
%snmpwalk -v 2c -c demopublic test.net-snmp.org system system.sysDescr.0 = HP-UX
net-snmp B.10.20 A 9000/715
system.sysObjectID.0 = OID: enterprises.ucdavis.ucdSnmpAgent.hpux10
system.sysUpTime.0 = Timeticks: (586998396) 67 days, 22:33:03.96
system.sysContact.0 = Wes Hardaker wjhardaker@ucdavis.edu
system.sysName.0 = net-snmp
system.sysLocation.0 = UCDavis
continua …
```

5) snmptable: Despliega la tabla de SNMP en una forma sencilla de ver:

```
%snmptable -v 2c -c demopublic test.net-snmp.org sysORTable SNMP table:
system.sysORTable

sysORIndex
sysORID
sysORDescr  sysORUpTime
          1
.iso.org.dod.internet.snmpV2.snmpModules.snmpMIB
The Mib module for SNMPv2 entities. 0:0:00:00.00
          2
.iso.org.dod.internet.snmpV2.snmpModules.snmpVacmMIB.vacmMIBConform
ance.vacmMIBGroups.vacmBasicGroup
View-based Access Control Model for SNMP. 0:0:00:00.00
          3
.iso.org.dod.internet.snmpV2.snmpModules.snmpFrameworkMIB.snmpFrame
workMIBConformance.snmpFrameworkMIBCompliances.snmpFrameworkMIBComp
liance                                          The SNMP Management
Architecture MIB. 0:0:00:00.00
          4
```

```
.iso.org.dod.internet.snmpV2.snmpModules.snmpMPDMIB.snmpMPDMIBConfo
rmance.snmpMPDMIBCompliances.snmpMPDCompliance
The MIB for Message Processing and Dispatching. 0:0:00:00.00
          5
.iso.org.dod.internet.snmpV2.snmpModules.snmpUsmMIB.usmMIBConforman
ce.usmMIBCompliances.usmMIBCompliance The management information
definitions for the SNMP User-based Security Model. 0:0:00:00.00
```

6) snmpset: Es usado para modificar información en el host remoto. Para cada variable que se requiere modificar se necesita especificar un OID, el tipo de dato y el valor que se desea cambiar.

4.4.4 Network Node Manager de Hewlett-Packard

El Open View (HPOV) es la herramienta de administración de redes de Helett-Packard. HPOV esta conformado con un GUI el cual interactúa con la Administración de Aplicaciones mediante APIs. También mediante APIs, la Administración de aplicaciones se comunica con los Servicios de Administración; finalmente éstos se comunican con los objetos manejados. Cómo podrá observarse, HPOV fue constituido con la idea de ser un sistema abierto que pudiera interactuar con otros sistemas de administración de redes con los cuales pudiera complementar funciones (Subramanian 2000).

HPOV también tiene la capacidad de funcionar de forma distribuida como lo propone el modelo de TMN. Tanto en el funcionamiento centralizado como en el distribuido, el corazón del HPOV es el Postmaster el cual integra todos los servicio de administración, servicios de enrutamiento de mensajes, servicios de control de eventos y mensajes de alarmas. Este también maneja múltiples protocolos como SNMP y CMIP.

4.4.5 Herramientas para al análisis de flujos de tráfico

Para el análisis de flujos de tráficos provistos por un protocolo como NetFlow, existen una variedad de herramientas. Estas van desde las que reciben los datos provenientes de NetFlow y las guardan como cflowd, las que grafican la información como FlowScan o las que reciben, guardan y procesan la información como flow-tools.

Cflowd: Es una herramienta para analizar los flujos de NetFlow. La versión actual (2.0) incluye la colección de datos, el almacenamiento y módulos básicos para análisis a través de arts++. Sus componentes funcionales son cflowdmux e cual recibe la exportación de flujos a través de los dispositivos de red; cflowd toma los datos de memora y crea tablas de sumarizacion; cfdcollect es el colector central el cual recibe las instancias de cflowd y permite escribir los datos a

archivos tabulares en intervalos regulares produciendo series de datos; finalmente provee algunas utilerías (Caida, Cflowd 2002).

FlowScan: Analiza y reporta los datos exportados de NetFlow. Consiste en un grupo de scripts y módulos de Perl que ligados a un "engine" de colección (como cflowd), a una base de datos (Round Robin Database - RRD), y a una herramienta de visualización (RRDtool). FlowScan produce imágenes gráficas que proveen un análisis continuo y en tiempo real de los flujos que pasan por un equipo en la red (Caida, FlowScan 2003)

Flow-tools: Es una librería y una colección de programas que se usan para colectar, enviar, procesar y generar reportes de datos obtenidos por NetFlow. Las herramientas pueden ser usadas en conjunto en un servidor o de forma distribuida en múltiples servidores para implementaciones grandes. Las librerías de flow-tools proveen un API para el desarrollo de aplicaciones especializadas para exportaciones de NetFlows versiones 1,5,6 y 8. Interfaces de Perl y Pitón han sido desarrolladas e incluidas (Fullmer). Algunas de las herramientas que se incluyen son (existen más pero solo se incluyen las que se consideran más importantes)

1) flow-capture: Colecta información de NetFlow
2) flow-cat: Concatena y lee archivos de flujos
3) flow-print: Imprime información de archivos de flujos en formatos específicos
4) flow-report: Genera reportes predefinidos a partir de información de flujos
5) flow-filter: Filtra información de flujos de acuerdo a las reglas especificadas

4.5 Selección de Herramientas

La selección de las herramientas es uno de los procesos más críticos en este estudio porque las herramientas seleccionadas servirán para la obtención de la información necesaria para la generación del algoritmo y se espera que posteriormente éstas puedan servir como utilerías para el desarrollo del sistema que probará la veracidad de la tesis.

Además la selección deberá estar sustentada en las necesidades del estudio, el ambiente, y los recursos con los que se cuente. Entre las necesidades del estudio se tienen la recolección de la información que se definirá posteriormente a los agentes remotos mediante algún protocolo de monitoreo; almacenamiento de la información en una base de datos; una interfaz para visualizar la información de forma numérica y gráfica; un lenguaje de programación para tener versatilidad funcional para explotar la información, una red a la que se le puedan obtener datos reales de comportamiento y una red experimental con un ambiente controlado para las pruebas de concepto. Los recursos con los que se cuentan son una red basada en TCP/IP para la recolección de datos y para la ejecución de experimentos, conmutadores LAN

(switches) y enrutadores con agentes de SNMPv1-v3, switches LAN con probes de RMON1 compatible con los grupos alarms, events, history y statistics (Cisco Catalyst 2950) y una computadora x86 (PIII 128 MB RAM).

De los estatutos anteriores se elimina CIMP y RMON2, además se desea hacer el sistema que explote el algoritmo los lo suficiente abierto a diversas plataformas de hardware de equipo de redes también se elimina NetFlow, ya que este se soporta únicamente por enrutadores Cisco y Juniper. En protocolos, al final queda solamente SNMP, RMON1. Con el descarte anterior de protocolos, también se eliminan todas las herramientas de análisis de flujos, así mismo, al no usarse CIMP, HPOV deja de ser imprescindible por ser el único colector de datos que soporta los protocolos de TMN.

Otro de los recursos que hay que tomar en cuenta es la estación colectora de información (NMS). Para este estudio se cuenta con un equipo x86, para este equipo puede contar con los sistemas operativos Windows 2000, Linux, FreeBSD, NetBSD y Solaris para Intel. De esta lista se opta por sistemas operativos de tipo Unix ya que presentan un mejor desempeño que los basados en Windows. De estos OS tipo Unix, se elimina Solaris para Intel ya que tiene un costo, de los restantes, aunque se consideran los basados en BSD mucho más robustos y con menos vulnerabilidades, se opta por Linux por el conocimiento que se tiene sobre este. La distribución de Linux seleccionada es Red Hat 7.2[2].

Con esta última selección se elimina HPOV de la lista de herramientas ya que el Open View Network Node Manager solo corre en Solaris, Windows 2000 y HP-UX (HP 2003). Finalmente las herramientas que pueden utilizarse son RRDTool, Cricket y NetSNMP. Aunque puede pensarse que las opciones son limitadas, además de estas herramientas se analizaron algunas más que por su simpleza, complejidad o dificultad de implementación no se trataron en este documento. Algunos ejemplos son Multi Router Traffic Grapher MRTG, que aunque es una herramienta muy conocida y muy poderosa tiene problemas de escalabilidad (Sourceforge Cricket 2002). OpenNMS combina las funciones de Cricket y RRDtool junto con sistemas de alarmas, recepción de traps, escalamiento de reportes, etc., que lo hacen una excelente herramienta de monitoreo incluso competencia de HPOV, sin embargo tiene un número de funciones que no se utilizan en nuestro estudio, cuyo fin es además proveer un sistema sencillo que aproveche el algoritmo, esto con OpenNMS no podría ser posible.

Unas de las cualidades que no se ha mencionado de Cricket y RRDtool es que la primera esta escrita en Perl y la segunda tiene módulos de éste, lo cual permite hacer el desarrollo en este lenguaje. Si bien Perl no es tan poderoso como

---

[2] Al momento de iniciar el estudio esta era la versión más actual de RH, al terminar el estudio la versión más reciente fue la 9. Los cambios más significativos para el estudio entre las dos versiones son la eliminación de vulnerabilidades como las encontradas en OpenSSH.

C, si es un lenguaje muy flexible, con estructuras de datos y manejo de archivos adecuados para un sistema sencillo, además de que es mucho más sencillo de programar que en C. Cricket también incorpora sus propias funciones de SNMP, con lo cual no es necesario tener una herramienta extra como Net-SNMP para la obtención de los datos de los agentes. De cualquier forma se conservara Net-SNMP como una herramienta para hacer algunos muestreos sencillos y para verificar las configuraciones de los agentes.

Conclusiones

El sistema queda como sigue: Una estación con hardware Pentium III, sistema operativo Red Hat 7.2, Cricket 1.0.3 como herramienta de poleo de agentes y almacenamiento, RRDTool 1.0.37 como base de datos y UCD-SNMP (Net-SNMP) 4.2.3 como herramienta de poleo manual y verificación de SNMP en agentes. La primera tarea de este sistema será monitorear en los agentes las variables de MIB que se definirán en el siguiente capitulo con la finalidad de comprobar primero si existe un patrón repetitivo en el comportamiento de éstas, y de ser así, que los datos obtenidos sirvan para la formulación de un algoritmo.

Capítulo 5

Análisis de la problemática y de las posibles variables para la identificación de problemas

Introducción

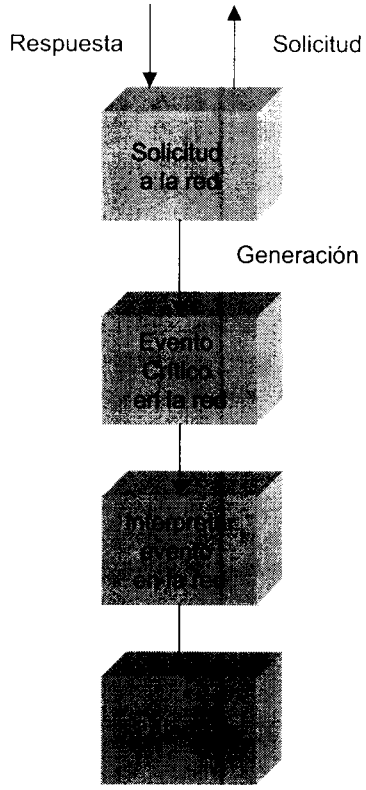
En este capítulo se hará un análisis del proceso para la administración de fallas y sus diferentes elementos. En este análisis se incluirán fallas comunes dentro de una red por diversos problemas y en diversas capas del modelo OSI. Se analizarán fallas de hardware y software que afecten las capas inferiores como la física y enlace de datos; para capas superiores se hará un análisis mas profundo incluyendo problemas y fallas causadas por vulnerabilidades de seguridad en los sistemas de cómputo actuales. Finalmente en el transcurso del capítulo se buscarán formas de detectar estas fallas mediante sistemas automáticos como el *polleo* de diversas variables en agentes SNMP.

5.1 Administración de fallas

De acuerdo a Held (1992) la administración de fallas es el proceso en el cual se detectan, se registran, se canalizan, se resuelven, se aíslan, se siguen y eventualmente se resuelven situaciones anormales a las condiciones de operación aceptable de la red. La administración de fallas envuelve prácticas importantes como la identificación de la ocurrencia de una falla en la red, aislar la causa de la falla y; si es posible la resolución de esta (Leinwand, Fang 1993).

Los beneficios de una administración de fallas es que incrementan la confiabilidad de la red cuando mediante esta administración se da a los ingenieros de redes una serie de herramientas y procedimientos para encontrar y resolver fallas. Antes de resolver un problema es necesario saber que este existe, una forma de conocer esto es recabando información de los dispositivos de redes, mediante esto puede conocerse mediante una análisis si un enlace esta fallando, si existe un sobre uso en algún sector de la red, etc. Leinwand y Fang (1993) proponen un modelo para identificar fallas en la red mediante el análisis de dispositivos, este modelo se puede observar en la figura 5.1.

Figura 5.1



En el diagrama de flujo de la figura 5.1 se solicita información a los dispositivos de redes. Estos contestan con los valores correspondientes los cuales son procesados para identificar si existe un evento crítico. Si existe una anormalidad, debe procesarse para identificar que tipo de anormalidad es para finalmente alertar al administrador. La situación anómala como se comentó anteriormente puede ser desde una falla de enlace fácilmente identificable, o hasta una cambio en el comportamiento en el uso del mismo por una situación adversa. Para este segundo caso es necesario contar con herramientas más inteligentes que puedan medir y comprar variables de uso de la red. A este análisis se le llama verificación de límites o "threshold setting". Held (1992) ejemplifica este problema en la práctica cuando un MODEM envía un nivel de señal más bajo de lo habitual pero más alto de lo necesario para identificar una pérdida en el servicio, sin embargo, eventualmente este comportamiento no sensible en la práctica puede llevar a una falla real, con un sistema de verificación de límites este problema se evita.

5.2 Protocolos de comunicación

5.2.1 Ethernet

Ethernet fue desarrollado a principios de los 70s por Xerox Corporation en su Centro de Desarrollo en Palo Alto (PARC) y fue la tecnología que sirvió como base para el estándar del IEEE conocido como IEEE 802.3 liberado a inicios de 1980. Posteriormente junto con Intel y Digital Corporation, Xerox formaría una alianza para desarrollar Ethernet Versión 2, la cual es compatible con el estándar. Actualmente estas dos tecnologías y sus derivados a 100 y 1000 Mbps dominan el mercado de redes locales (Cisco, 2002).

Ethernet usa una arquitectura de bus con acceso múltiple al medio. Para transmitir una estación escucha el medio (el bus) y cuando no existe una transmisión de otra estación la primera transmite. Debido al tiempo de propagación de las señales eléctricas en el medio puede ocurrir que dos estaciones no escuchen transmisión y decidan transmitir al mismo tiempo, cuando la señal se propaga y eventualmente choca, genera un ruido eléctrico que se le llama colisión (Stallings 1996). Para evitar volver a generar otra colisión las estaciones utilizan un algoritmo de retransmisión que les indica cuando volver a transmitir (Cisco, 2002). El proceso de escuchar a un medio de acceso múltiple y detectar colisiones se le llama CSMA/CD (Carrier Sense Multiple Access/Collision Detect) (Subramanian 2000).

Desde su creación, Ethernet ha evolucionado soportando diferentes medios físicos y velocidades de transmisión. La tabla 5.1 muestra estos cambios.

Tabla 5.1

Tipo	Familia	Distancia entre dispositivos	Medio Físico
10Base2	Ethernet	N/A	Coaxial delgado
10Base5	Ethernet	N/A	Coaxial grueso
10BaseT	Ethernet	100 mts	Cable torcido
10BaseF	Ethernet	2 km	Fibra óptica
100BaseT	Fast Ethernet	100 mts	Cable torcido
100BaseFx	Fast Ethernet	2 km	Fibra óptica
1000BaseT	Gigabit Ethernet	100 mts	Cable torcido
1000BaseSX	Gigabit Ethernet	500m, 2km	FO MM, FO SM
1000BaseLX	Gigabit Ethernet	1 km, 10 km	FO MM, FO SM
1000BaseLH	Gigabit Ethernet	70 Km	FO Single Mode

Ethernet trabaja en el nivel 2 del modelo OSI, en la red que analizaremos los equipos de red que utilizan esta tecnología son los conmutadores LAN y los enrutadores, sin embargo los que presentan un mejor escenario de análisis son los conmutadores LAN, ya que estos conectan a los usuarios y además permiten

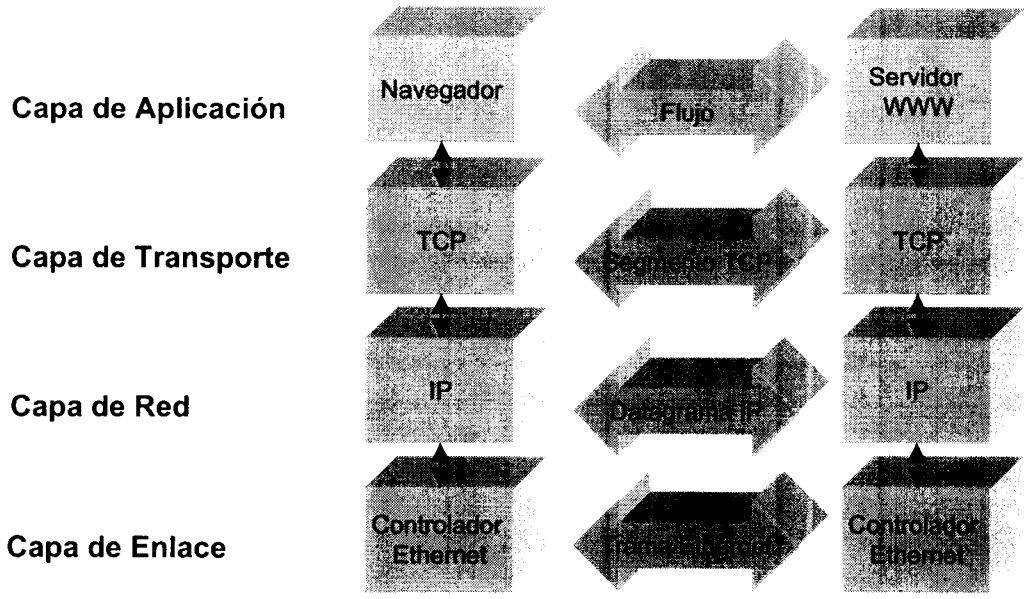
la interconexión con otros conmutadores y con los mismos enrutadores. En nuestro análisis se incluirán equipos con interfaces 100BaseTx, 100BaseFX, 1000BaseSX y 1000BaseLX.

5.2.2 Internet Protocol (IP)

El Internet Protocol fue diseñado para interconectar redes conmutadas de paquetes para la formación de redes a gran escala. IP es responsable de transmitir bloques de información y entregarlos a protocolos de capa superior para su procesamiento. Este protocolo surgió del desarrollo del Departamento de Defensa de Estados Unidos para formar una red de investigación militar llamada DARPA, la cual posteriormente se convertiría en el Internet que conocemos actualmente (Giles 1998).

IP es un protocolo definido en el nivel de red (3) del modelo OSI, sin embargo hace uso de protocolos de capa superior para completar el conjunto de protocolos que hace posible la comunicación uno a uno entre dos sistemas. La figura 5.2 muestra este diseño de capas, en la cual tenemos a la capa de red conformada por el protocolo de IP. Esta capa es la responsable de llevar los datos del sistema origen al sistema destino (Northcut, Novak 2001), IP se auxilia de protocolos de enrutamiento que permiten a los dispositivos de reenvío de paquetes conocidos como enrutadores conocer el siguiente nodo a donde se deben enviar los datos. Debido a este comportamiento de envío de punto a punto a IP se le conoce como un protocolo no orientado a conexión, también este comportamiento hace que IP deba tener mecanismos que aseguren que el paquete llegue satisfactoriamente a su destino.

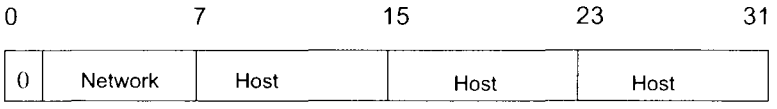
Figura 5.2



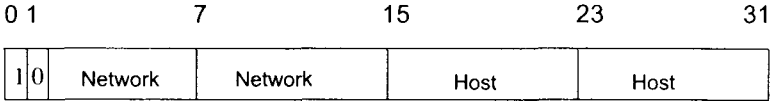
La siguiente capa es la de transporte, esta capa se mencionará a detalle más adelante pero cabe mencionar que esta se encarga de aspectos como el tipo de conexión (orientada a conexión o a no conexión), tamaños de bloques de transmisión, tasa de transferencia, detección de pérdida de paquetes y en general de la entrega satisfactoria de los datos. Finalmente la capa de aplicación es la capa de nivel superior que se encarga de procesar la información de acuerdo al tipo de aplicación, la cual puede ser correo electrónico, World Wide Web, transferencia de archivo o emulación de terminal entre otros.

Para realizar la comunicación entre dos sistemas IP utiliza direcciones como identificador. Estas direcciones están formadas por 4 octetos separados por un punto. Además la dirección está dividida en un identificador de red y un identificador de sistema, a los identificadores de red iguales se les conecta en el mismo segmento de broadcast o en la misma red (usualmente un segmento o un bus de Ethernet). El tamaño en bits o en octetos del identificador de red puede verse en la figura 5.3 (Halabi 1997)

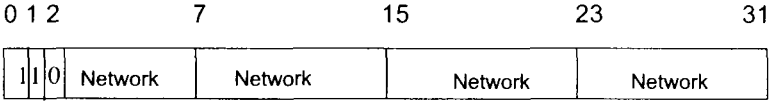
Figura 5.3



Clase A
Máscara Natural:
255.0.0.0



Clase B
Máscara Natural:
255.255.0.0



Clase C
Máscara Natural:
255.255.255.0

Debido a que redes tipo clase A o clase B permiten la creación de miles de sistemas en la misma red es necesario crear subredes, están se crean aplicando máscaras a las direcciones de IP, de esta forma una clase A puede ser dividida tan granularmente como una subred de 2 sistemas aplicando una máscara de 255.255.255.254 (o 11111111. 11111111. 11111111. 11111110).

Cuando dos sistemas están en dos redes diferentes (identificadores diferentes, por ejemplo 10.1.1.2 y 10.1.2.2 cada una con máscara de 255.255.255.0) se requiere de un dispositivo especializado para interconectarlas, estos dispositivos se llaman enrutadores. Los enrutadores están encargados de mover información de una red a otra. En redes complejas donde existen muchas redes y muchos enrutadores es necesario que existan protocolos de enrutamiento que permiten a los enrutadores conocer la información de las redes conectadas, ejemplos de protocolos de enrutamiento son Open Short Path First (OSPF), Routing Information Protocol (RIP) y Border Gateway Protocol (BGP) entre otros. Halabi define a los enrutadores como "dispositivos que dirigen tráfico entre sistemas. Los enrutadores construyen tablas que contienen una colección de información sobre todos los mejores caminos[3] a todos los destinos a los que saben llegar. Anuncian y reciben información de rutas de y para otros enrutadores. Esta información va a las tablas de enrutamiento".

Se puede hablar mucho de IP, de enrutadores y de protocolos de enrutamiento pero el estaría fuera del alcance de este estudio, así que nos

[3] El término en inglés es "path"

concretamos a proveer la información necesaria para comprender más adelante algunos de los problemas que pueden surgir en las redes debido a las carencias o simplemente por el comportamiento de los protocolos.

5.2.3 Transport Connection Procotol (TCP)

Uno de los protocolos superiores más importantes (si no es el más importante junto con IP) del conjunto de protocolos de IP es el TCP. TCP está ubicado en la capa 4 del nivel OSI y es un protocolo orientado a conexión. TCP provee además una conexión "full-duplex"[4], confirmación y control de flujo para las aplicaciones de protocolos superiores (Chappell, Farkas 1999). Giles además agrega la adaptación a la red como otra característica de TCP, esto es que TCP pueve variar la velocidad de transmisión adaptándose a ambientes de bajo y alto ancho de banda de forma automática.

TCP es el protocolo que asegura que la conexión sea confiable, para ello debe identificar la pérdida de datagramas y el cambio de orden en la entrega de éstos, los mecanismos que utiliza para llevar a cabo estas acciones son los siguientes (Northcut, Novak 2001):

Conexión TCP Exclusiva: Las conexiones que se establecen entre dos sistemas se llaman sesiones. Estas sesión es exclusiva y única. La negociación de esta sesión permite a los sistemas identificarla y seguir los datos que se transmiten entre ellos.

Número de secuencia de TCP: Proporcionan una secuencia cronológica de los datos enviados y recibidos por TCP. A cada datagrama enviado se le asigna un identificador que sirve para indiciar el orden y si estos llegan en un orden diferente TCP sabe como reorganizarlos para ser entregados a las aplicaciones de capas superiores.

Acuses de recibo: Se utilizan para informar al transmisor que se ha recibido información, se utilizan los números de secuencia como identificar. Si el transmisor no recibe información de recibido de ciertos identificadores asume que se han perdido y procederá a retransmitir.

Otros mecanismos de TCP son (Giles 1998):

Tamaño de ventana y "buffering": Cada sistema tiene un espacio de almacenamiento conocido como "buffer" el cual está listo para enviar y recibir datos. Esto permite a la red tomar y enviar datos mientras las aplicaciones están ocupadas en otras actividades. Para evitar el desbordamiento del buffer, TCP utiliza un mecanismo llamado "Tamaño de Ventana" en el cual se incluye en el

[4] Se transmite al mismo tiempo en que se recibe. En "half-duplex" sólo una de las estaciones puede transmitir mientras la otra escucha.

paquete de TCP y dice el tamaño del buffer. Cuando existe congestión en la red, TCP disminuye los tamaños de ventana adaptándose así a las características de la red.

Estimación del tiempo de retorno: Cuando un sistema envía información a otro sistema debe esperar cierto tiempo para escuchar la confirmación, si el paquete no regresa en cierto intervalo de tiempo, éste se considera perdido.

Para que dos sistemas comiencen una sesión de TCP es necesario que se establezca mediante un proceso llamado "three-way handshake" (Giles 1998). Este proceso se realiza de la siguiente forma (Northcut, Novak 2001, Giles 1998):

1) En este proceso el cliente envía un paquete de sincronización (con la bandera de SYN encendida) hacia el servidor. Además este paquete lleva el número de secuencia propuesto por el cliente.
2) Si el servidor acepta el servicio regresa al cliente una respuesta de sincronización proponiendo su propio número de secuencia y acuse de recibo del número de secuencia del cliente (banderas de SYN y ACK encendidas).
3) Para finalizar el cliente recibe el ACK y el SYN y si desea continuar levantando la conexión contesta con un acuse de recibido del número de secuencia del servidor.

Después de este proceso se estable la conexión o sesión de TCP. A partir de este punto todos los paquetes tendrán la bandera de ACK encendida y continuarán incrementando los números de secuencia. El ACK se hace del siguiente segmento que se espera.

Para finalizar la conexión existen dos métodos. El método elegante es cuando el cliente o servidor indica a su contraparte su deseo de terminar la sesión mediante un paquete de FIN. El receptor contesta con un acuse de recibo. Finalmente el iniciador del FIN al recibir el ACK contesta con un último acuse de recibo. El método brusco es cuando una de las contrapartes le envía un paquete de RESET al otro, este no requiere ACK y la sesión se termina de inmediato.

5.2.4 User Datagram Protocol (UDP)

Así como TCP es un protocolo superior de IP y orientado a conexión, UDP es un protocolo del conjunto de IP ubicado en la capa 4 del OSI y orientado a no conexión. Está definido en el RFC 768 y es usado por aplicaciones que sólo requieren un servicio sin conexión y bajo el mejor esfuerzo, es por ello que fue diseñado para transmitir información entre sistemas con un mecanismo de protocolo mínimo. Debido a esta simpleza, UDP no proporciona garantías de entrega ni protección ante duplicación de información. Sin embargo por esta simpleza es posible que aplicaciones como RTP (Real Time Protocol) usado en

aplicaciones multimedia tenga una entrega sencilla y pueda implementarse en dispositivos muy simples. Algunas aplicaciones que funcionan sobre UDP son: VoIP (Voz sobre IP), DNS (Domain Name Server) y SNMP (Simple Network Management Protocol). (Giles 1998).

No se debe pensar que UDP sea un protocolo que pierda paquetes o los desordene, simplemente no se hace responsable de ello y es necesario que las aplicaciones de capa superior se encarguen de esta labor (Northcut, Novak 2001).

5.2.5 Internet Control Message Protocol (ICMP)

IP o sus protocolos de capa superior no tienen un mecanismo por si mismos para reportar a la fuente de problemas en la capa de red, un protocolo que resuelve este problema es ICMP. ICMP provee información a la fuente de errores en la capa de red que ocasionan que el paquete no llegue a su destino final.

La principal razón de su creación fue el reporte de fallas de enrutamiento a la fuente, sin embargo desempeña otras acciones como paquetes de "echo" y "echo-reply" para probar conectividad de la fuente al destino; mensajes de "traceroute" o trazo de ruta para probar conectividad punto a punto de la fuente al destino; redirección de mensajes para estimular mejor y óptimo enrutamiento; mensajes de tiempo excedido que informan a la fuente que su datagrama ha excedido el tiempo de vida en la red; mensajes de anuncio de enrutadores y mensajes de solicitud de enrutador para determinar la dirección de enrutadores directamente conectados; y más recientemente ICMP provee un mecanismo para que los nodos en la red puedan descubrir sus máscaras de red (Chappell, Farkas 1999).

ICMP está en la misma capa de red que IP, está encapsulado dentro del datagrama de IP pero aun así se le considera al mismo nivel de red. Se difiere además de UDP y TCP ya que no tiene número de puertos (Northcut, Novak 2001).

5.3 Fallas comunes

5.3.1 Fallas físicas y de hardware

Las causas de fallas físicas en la red son muchas, muy variadas y en algunos casos muy dependientes de la tecnología del nivel físico y de enlace de datos. En este estudio nos concentraremos en las fallas físicas de equipo con interfaces Ethernet y en redes basadas en la familia Ethernet al ser una de las

redes más populares y además porque nuestro estudio estará basado en una red de este tipo.

Algunas de las causas más comunes, sus causas y sus síntomas son las siguientes (Cisco, Ethernet 2002):

Ruido excesivo: El ruido se presenta cuando existen errores de CRC pero no existen muchas colisiones, esto puede deberse a cables dañados, uso de 100BaseTx sobre cable de categoría menor a la 5, conectores mal hechos.

Colisiones excesivas: El número de colisiones debe ser un 0.1 % del total de tramas. Si existen muchas colisiones debe considerar conmutar la red, si es puerto de conmutador LAN poner en estado full-duplex. También es necesario checar por transivers mal conectados y/o usar una analizador de protocolos.

Excesivas tramas de "runt": en ambientes conmutados son causadas por colisiones. Si no es un ambiente conmutado estos pueden ser causados por tarjetas de red defectuosas o software dañado.

Colisiones tardías: Estas ocurren cuando se detecta una colisión después de haber transmitido los primeros 64 bytes. Esto puede ser causado por cables de longitud mayor a 100 mts o la existencia de muchos repetidores.

No hay integridad en el enlace: Ocasionada por ruido excesivo, cable cruzado en lugar de directo (o viceversa si se requiere), problemas de conectores o cableado.

5.3.2 Ataques de Negación de servicio

La definición más simple es que una ataque de negación de servicio es cualquier acción, iniciada por un humano o por algún otro medio que incapacite el software o el hardware de un nodo en la red y que le imposibilite prestar un servicio a los usuarios legítimos. Normalmente los ataques de negación de servicio son maliciosos (Anónimo Linux Security, 2000). En algunas ocasiones los ataques son ejecutados dentro de las mismas instalaciones y con los sistemas de la organización afectada (Northcut, Novak 2001).

Los ataques de negación de servicio puede ser la combinación de cinco posibles fallas en la seguridad (Rozenblit 2000):

- Inundación de la red
- Accesos no autorizados
- Corrupción de información de administración
- Usurpación de identidad
- Re-enrutamiento, cambio de rutas, borrados de mensajes

Los ataques de negación de servicio pueden ser contra un solo usuario o contra toda la red. En el caso de negación contra un solo usuario, el intruso niega el servicio de algún tipo a un usuario en alguna de dos formas. Una forma es interceptar la información del servicio al usuario de tal forma que ésta nunca llega a su destino. La segunda es acceder a la información de la administración de seguridad y corromperla de tal forma que niegue el acceso al servicio del usuario. En el caso de negación del servicio hacia toda la red, este puede ejecutarse cuando el intruso inunda el canal de comunicación con muchos mensajes basura. Inundando un servicio con información irrelevante puede causar que este servicio deje de "escuchar" y se detenga negando el acceso a toda la red. Otro ataque puede ser aquel donde el atacante envía información al servicio que parece ser verdadera, el procesamiento de esta información falsa puede causar una sobrecarga en el sistema del servicio terminando con la detención en la entrega de éste (Rozenblit 2000)

Los ataques de negación de servicio son un gran problema por dos razones. La primera es que estos ataques son rápidos y sencillos y generan un resultado inmediato y notable. Del texto de Linux Security podemos obtener varios tipos de ataques de negación de servicios como lo son el "Smurf", ataques de TCP SYN, ataques con "Spoofing", ataques que envían paquetes mal formados para causar fallas en stack de TCP/IP (Anónimo Linux Security 2000). Otros ataques similares son el echo-chargen que utiliza el servicio de echo de UDP, ping de la muerte que utiliza ICMP y paquetes fragmentados y ataques distribuidos de negación de servicio (DDOS Distributed Denial of Service) (Northcut, Novak 2001).

El ataque de smurf es uno de los más recientes y se llama así por el programa que dio su origen. Este tipo de ataque se caracteriza por el envío de una gran cantidad de paquetes de ICMP echo (ping) a una dirección de broadcast y con dirección fuente robada (spoofed) de la víctima. El hecho de que el paquete tenga como destino una dirección de broadcast ocasiona que la respuesta puede llegar a ser de cientos de sistemas en un solo segmento de red. Existe un primo del smurf llamado "fraggle" que en lugar de ICMP usa paquetes de echo de UDP (Huege 2000). El principal problema del Smurf es que consume el ancho de banda de los atacados, por un lado tenemos los sistemas que responden al ping que saturan su red y por otro lado la supuesta dirección origen a la cual los sistemas origen le llenan también su enlace. Si el ataque es por Internet y los originadores tienen un gran ancho de banda y la víctima no, esta segunda puede verse realmente en problemas (Northcut, Novak 2001).

El ataque de TCP SYN ocurre cuando se envían una gran cantidad de intentos de conexiones que no pueden ser completadas, esto trae como consecuencia que usuarios legítimos del servicio no puedan usarlo. En este tipo de ataque, el atacante inicia una sesión de TCP a un servicio, cuando el servidor contesta la requisición esperando un ACK, el atacante no lo envía. Esto ocasiona que el servidor utilice recursos en espera de una respuesta. Si esto se repite un gran número de veces a altas tasas de intentos de abrir conexión, pueden

ocasionar que el sistema sature los buffers se recepción de conexiones haciendo imposible para otros usuarios acceder el servicio (Cisco 2001). Este ataque se complica cuando el atacante utiliza una dirección de IP que no es la suya, esto ocasiona un doble ataque, la negación de servicio y la apariencia de ataque de alguien inocente (Ferguson y Senie 1998).

El ataque de echo-chargen es otro ejemplo de ataque de fuerza bruta que utiliza sitios protegidos con baja o nula seguridad. Este utiliza el servicio UDP en el puerto 19, el cual al recibir cualquier carácter, responderá con una cadena pseudoaleatoria de caracteres. El ping de la muerte utiliza paquetes mayores al máximo permitido, en este caso ICMP y 65,535 bytes. Sistemas que no pueden manejar la fragmentación son vulnerables a este tipo de ataques, como el paquete generalmente es mayor al permisible por el medio (Ethernet tiene un MTU de 1500 bytes) es necesario hacer fragmentación en los enrutadores de la red.

Los DDOS implican el esfuerzo de que el atacante domine o explote las vulnerabilidades de un gran número de sistemas en el Internet, comúnmente, explota vulnerabilidades muy conocidas y sistemas con baja protección una vez controlados todos estos sistemas, basta unos comandos muy sencillos para enviar la orden de ataque (Bennet 2000). Los ataques de DDOS utilizan algunos ataques ya mencionados como el Smurf, sin embargo en lugar de que la fuente del ataque sea única, se utilizan múltiples fuentes situadas en diversos lugares del Internet generando un ataque distribuido y simultáneo. La gravedad de este ataque es que inicialmente el efecto de negación se multiplica debido al número de sitios involucrados en el ataque, además, si un ataque normal de DOS es complejo de detener y rastrear, un DDOS multiplica este esfuerzo (Northcut, Novak 2001).

Otra fuente de DOS son los gusanos (worms) y los virus. La diferencia entre virus y gusano es que los gusanos pueden distribuirse a otras computadores por si mismos, mientras los virus requieren la ayuda de un externo, este puede ser una unidad de disco u otro programa (Boettger 2002). El primer gusano que afectó el Internet fue el Morris Worm, bautizado así por su autor Robert Morris. Este se liberó en Noviembre de 1988 y tomaba ventaja de un exploit en el sendmail de UNIX, para cuando el gusano fue aislado ya había afectado entre el 5 y el 10% de los sistemas de Internet con daños en alrededor de 98 millones de dólares (Boettger 2002). Aunque el gusano no hacia ningún daño, el tráfico generado por él fue capaz de afectar considerablemente la operación de todo el Internet infectando al 10% de los sistemas y causando daños por más de 100 millones de dólares (Travis et at 2003).

El 25 de enero de 2003 un nuevo virus fue introducido en el Internet global alrededor de las 12:30 A.M. (EST), el punto exacto de inserción permanece aún desconocido (Travis et at 2003). El virus se le conoció como SQLSlammer, W32.Slammer, y Sapphire Worm, y era un código que se propagaba automáticamente y que explota las vulnerabilidades descritas en VU#484891

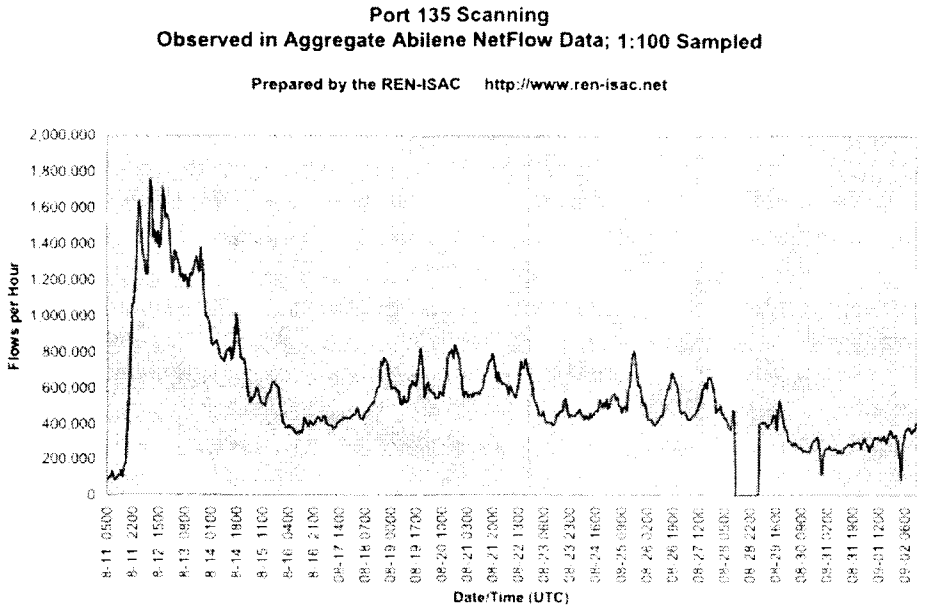
(CAN-2002-0649) del CERT. Esta vulnerabilidad permite la ejecución de código debido a una sobrecarga en asignaciones de memoria (stack buffer overflow) en un servidor SQL. Una vez que el servidor ha sido comprometido, el gusano se propaga a si mismo de manera automática. El gusano es básicamente un paquete de 376 bytes que se envía de manera aleatoria a direcciones de IP sobre el puerto 1434 de UDP. Si el paquete llega a una máquina con la vulnerabilidad, esta se infecta y comienza a tratar de infectar a otros sistemas (CERT CA-2003-04, 2003)

El SQL Slammer infectó a menos del .1 % de los sistemas en Internet pero el daño fue mucho más catastrófico que el Morris Worm. El SQL Slammer era capaz de infectar 25,000 direcciones por segundo con un solo paquete, aprovechando una vulnerabilidad en los sistemas Windows con SQL. El poder de infección fue tanto que fue capaz de alcanzar una magnitud global en menos de 8 minutos generado un tráfico aproximado de 1 Terabit/seg (Travis et at 2003).

En los estudios realizados por el Advanced Network Management Lab Cybersecurity Initiative de la Universidad de Indiana (Travis et at 2003) se puede notar los efectos devastadores de este gusano y en general de cualquier gusano que llegue a comprometer sistemas conectados a redes de alta velocidad. Es su estudio muestran como un servidor Pentium III de 1 Ghz con 256 MB de memoria fue capaz de generar 75 Mbps de tráfico y una tasa de direcciones para infección de 25,000 direcciones por segundo. Debido a que las direcciones destino seleccionadas por el gusano no diferenciaban bloques privados (10.0.0.0, 172.16.0.0, y 192.168.0.0) así como los bloques de multicast (224.0.0.0 – 239.255.255.255) esto causo otros daños colaterales. Estos daños colaterales fue sobrecargar enrutadores de IP Multicast y/o que usan protocolos de ruteo basados en IP-Multicast (OSPF, EIGRP, IS-IS).

El 16 de Julio de 2003 Microsoft publicó una vulnerabilidad que afectaba el funcionamiento de los Remote Procedure Call (RPC) y que podía causar que un atacante tomará control del sistema (Microsoft Security Bulletin MS03-026, 2003) explotando ésta, la vulnerabilidad se encontraba en sistemas Windows NT 4, Windows 2000 y Windows XP. El 11 de Agosto el CERT/CC anuncia su alerta CA-2003-20 sobre un gusano llamado W32/Blaster el cual explota la vulnerabilidad descrita por Microsoft un mes atrás. Según esta reporte, un sistema comprometido inicia la búsqueda de otros sistemas con la vulnerabilidad y al encontrarlo lo infecta. El ataque se ejecutaba usando los puerto TCP 135 y 445 usados por los sistemas para ejecutar RPCs (CERT/CC CA-2003-20). La figura 5.4 muestra el incremento en la actividad del puerto 135 de TCP en los enrutadores de la Red de Abilene de Internet2 en Estados Unidos.

Figura 5.4
Cortesía de REN-ISAC



El 18 de Agosto después de que el gusano atacó muchos sistemas personales en la red de Internet el Internet Storm Center detectó un incremento considerable en la actividad de ICMP específicamente en paquetes de *echo-request*. El Centro ligó el incrementó con el una gusano llamado Welchia o Nachi. Ese mismo día el CERT/CC así como otras agencias dedicadas a la protección contra virus anunciaron este gusano como una variante del W32/Blaster (CERT#33546, 2003; Symantec W32.Blaster.B.Worm, 2003). El CERT catalogó al gusano como de alto impacto debido al potencial de daño que tenía al generar paquetes de ICMP y causar una Negación de Servicio. Al día de hoy la actividad de W32/Blaster y sus variantes siguen existiendo como lo demuestra el reporte del CERT y las gráficas de la red Abilene específicamente en la actividad de ICMP como puede observarse en las gráficas de las figuras 5.5 y 5.6. En estas gráficas también puede observarse el incremento en la actividad de ICMP de unos cuantos paquetes a mies de paquetes después del 18 de Agosto de 2003.

Figura 5.5
Cortesía Abilene NOC

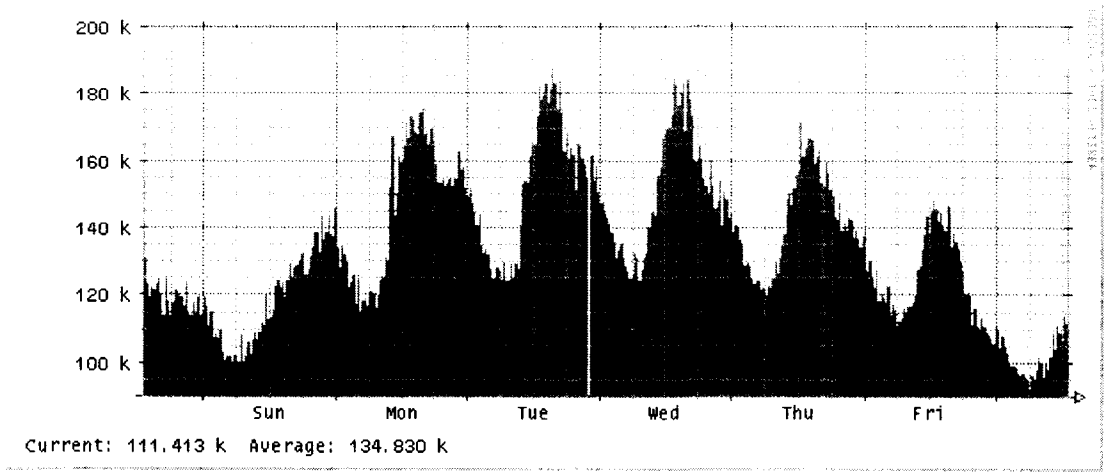
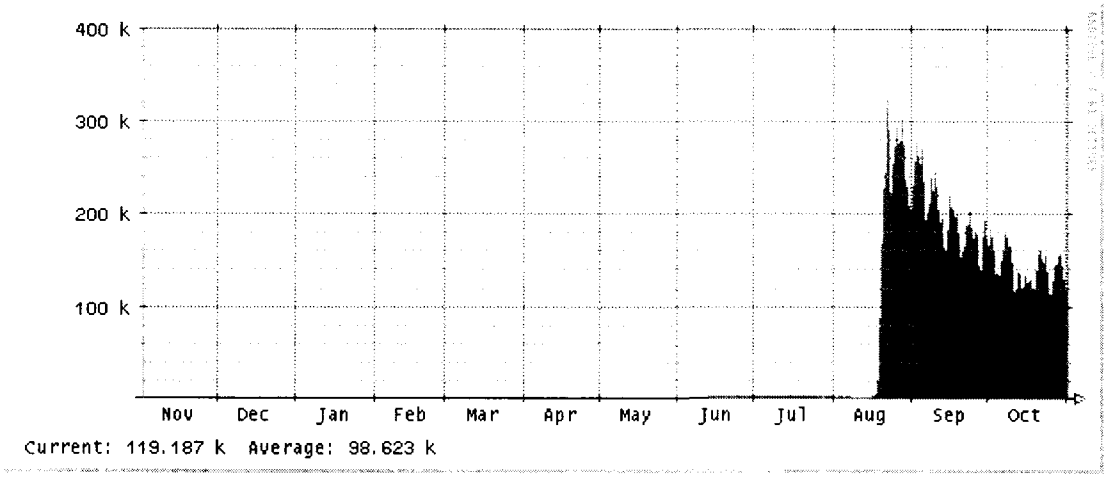


Figura 5.6
Cortesía Abilene NOC



Las figuras 5.7 y 5.8 muestran como la actividad del gusano W32/Blaster y sus variables afectaron el funcionamiento de un enrutador dentro de la red el Instituto Tecnológico de Monterrey. También puede observarse en la figura 5.9 como existe un patrón definido y posiblemente predecible antes del inicio de la actividad del gusano. Posterior al inicio de la actividad del gusano hasta su erradicación en el la red puede notarse como afecta significativamente la cantidad de memoria disponible en el enrutador.

Figura 5.7
Uso de memoria en patrón diario
Cortesía Telecomunicaciones y Redes Tecnológico de Monterrey

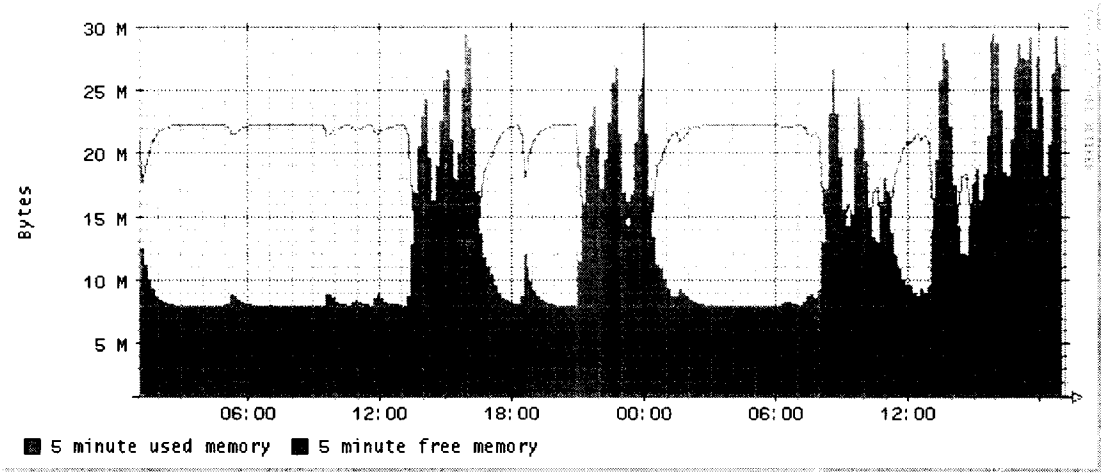


Figura 5.8
Uso de memoria en patrón semanal
Cortesía Telecomunicaciones y Redes Tecnológico de Monterrey

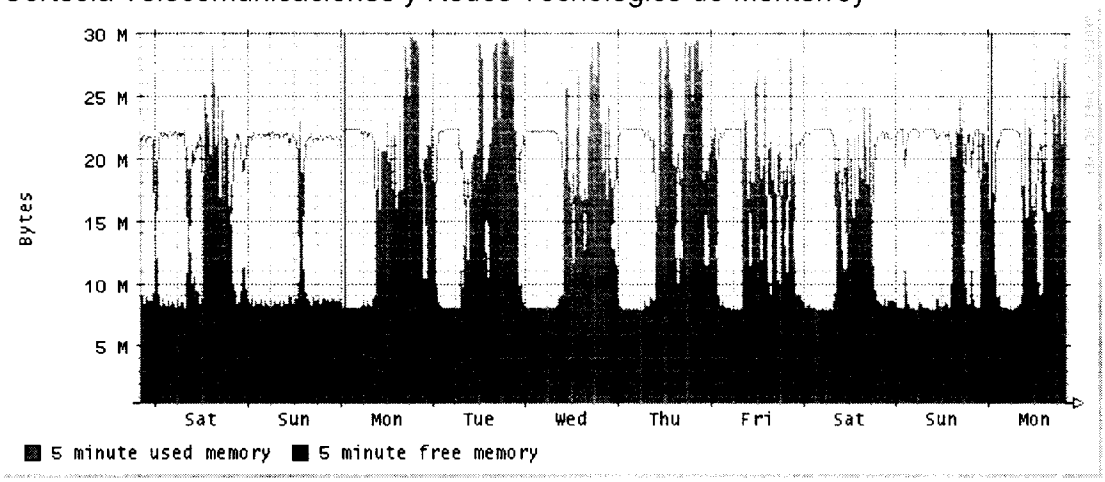
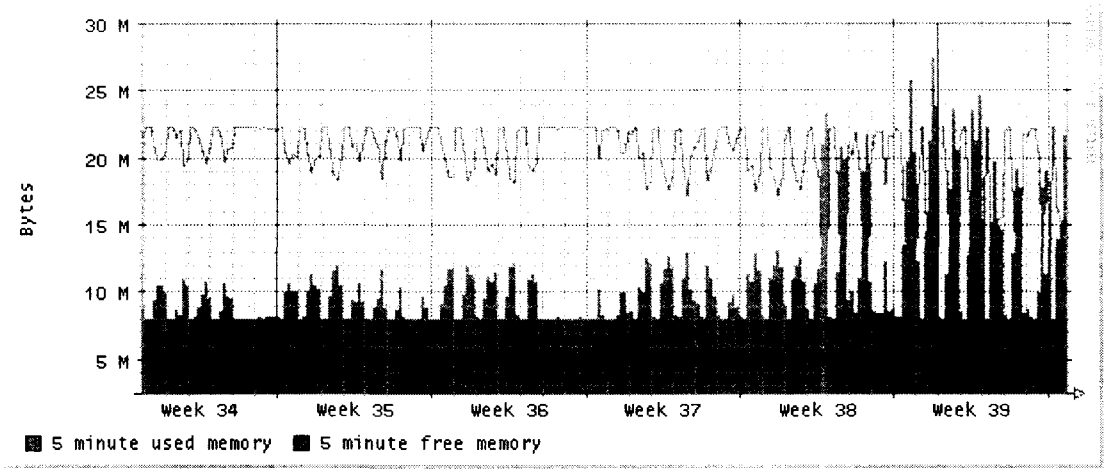


Figura 5.9
Uso de memoria en patrón mensual
Cortesía Telecomunicaciones y Redes Tecnológico de Monterrey



Esto son solo algunos ejemplos de actividad anormal dentro de las redes de datos y fuera del control de los administradores de la misma. También se ha visto que estas anormalidades se convierten rápidamente en un problema crítico que puede hacer fallar a todo una red corporativa con terribles consecuencias operacionales y económicas. Es por ello que se requiere de sistemas inteligentes que puedan identificar actividad anormal y candidata a convertirse en un problema crítico.

5.4 Definición de variables a Monitorear

En la definición del MIB-II (McCloghrie, Rose 1991) se encuentran diversas variables que pueden utilizarse para medir el desempeño y funcionamiento de la red. Los grupos en que se dividen son los siguientes:

- System
- Interfaces
- IP
- ICMP
- TCP
- UDP
- EGP
- Transmisión
- SNMP

Sin embargo, después de analizar las posibles fallas dentro de la red se considera que no todos estos grupos tienen variables que son útiles para nuestro estudio. Por ejemplo el grupo de EGP proporciona información sobre el protocolo de enrutamiento EGP, el cual esta en desuso en el Internet. Los grupos que ofrecen variables con datos interesantes para monitorear son *Interfaces*, *IP*, *ICMP* y *UDP*. De estos tres grupos, para el caso de conmutadores LAN de capa 2 sólo *Interfaces* es un grupo de donde se pueden obtener datos. Para obtener datos relevantes de los grupos de *IP*, *ICMP* y *UDP* se requiere de un dispositivo de capa 3 para procesar los paquetes que se transmiten en la red y contabilizar esto en las variables.

5.4.1 Interfaces

En el grupo de Interfaces, las variables interesantes a monitorear son:

ifInOctets, ifOutOctets: El número total de octetos recibidos y transmitidos respectivamente contando caracteres de tramas.
ifInUcastPkts, ifOutUcastPkts. El número total de paquetes de unicast recibidos por la sub-capa física para ser transmitidos a protocolos de capa superior y los paquetes de unicast que son enviados por capas de protocolos superiores para ser enviados a la sub-capa de red física.
ifInNUcastPkts, ifOutNUcastPkts: El número total de paquetes de no unicast recibidos por la sub-capa física para ser transmitidos a protocolos de capa superior y los paquetes de unicast que son enviados por capas de protocolos superiores para ser enviados a la sub-capa de red física.
ifInDiscards, ifOutDiscards: Número de paquetes de entrada y salida respectivamente que han sido descartados aunque no tuvieran errores. Una razón para ello es la liberación de buffers de memoria.
ifInErrors, ifOutErrors: El número de paquetes de entrada y salida respectivamente que no pudieron ser recibidos o enviados debidos a errores.
ifInUnknownProtos: El número de paquetes descartados por la interfaz de entrada debido a que el protocolo no es soportado o es desconocido.

5.4.2 IP

En el grupo de IP e ICMP que solo puede ser utilizado sobre equipos que procesen la capa de IP, las variables interesantes a monitorear son:

ipInReceives: El número total de datagramas de entrada recibidos en una interfaz. Incluye aquellos con error.
ipInHdrErrors: El número total de datagramas de entrada eliminados por errores en su encabezados de IP, incluye "checksum" erróneo, versión equivocada,

errores de formato, tiempo de vida excedido o errores en el procesamiento de opciones de IP.
ipInAddrErrors: El número total de datagramas de entrada descartados porque la dirección de IP en el encabezado de destino no es válida para esta entidad. Esto incluye direcciones 0.0.0.0 y clases no soportada (ej. Clase E). Para entidades que no son gateways este contador incluye aquellas direcciones que no son la dirección local.
ipForwDatagrams: El número total de datagramas de entrada en que esta entidad no fue el destino final de IP, como resultado se hizo un intento de encontrar una ruta para el paquete y reenviarlo.
ipInUnknownProtos: El número total de datagramas recibidos satisfactoriamente pero descartados por un error desconocido o por protocolo no soportado.
ipInDiscards/ipOutDiscards: El número total de datagramas de entrada/salida para los cuales no se encontró problemas para continuar su procesamiento pero que no fueron enviados, posiblemente por falta de buffers.
ipInDelivers: El número total de datagramas de entrada satisfactoriamente enviado a protocolos de IP (incluye ICMP).
ipOutNoRoutes: El número total de datagramas descartados porque no se pudo encontrar una ruta para transmitirlos a su destino.
ipReasmReqds: El número de fragmentos de paquetes de IP recibidos y que *requieren ser reensamblados en este entidad*.
ipReasmOKs: El número total de datagramas de IP satisfactoriamente reensamblados
ipReasmFails: El número total de datagramas de IP con detectados con falla para su reensamblaje.
ipFragOKs: El número total de datagramas de IP que han sido satisfactoriamente fragmentados por esta entidad.
ipFragFails: El número total de datagramas que han sido descartados porque requieren ser fragmentados en esta entidad pero que no han podido defragmentarse (Ej. Su bit de no fragmentar está en 1).
ipFragCreates: El número de datagramas de IP fragmentados que han sido generados por esta entidad.

5.4.2.1 ICMP

icmpInMsgs: El número total de mensajes de ICMP que esta entidad ha recibido. Este contador incluye aquellos contados por icmpInErrors.
icmpInErrors: El número de mensajes de ICMP que esta entidad ha recibido pero que se ha determinado que tienen errores específicos.
icmpInDestUnreachs: El número de mensajes de ICMP recibidos de Destination Unreachable.
icmpInTimeExcds: El número de mensajes de ICMP recibidos de Tiempo Excedido
icmpInParmProbs: El número de mensajes de ICMP recibidos de Problemas de Parámetros.
icmpInRedirects: El número de mensajes de ICMP recibidos de ICMP Redirect

icmpInEchos: El número de mensajes de ICMP recibidos de ICMP Echo (request)
icmpInEchoReps: El número de mensajes de ICMP recibidos de ICMP Echo Reply. IcmpInTimestamps: El número de mensajes de ICMP recibidos de ICMP Timestamp (request).
icmpInTimestampReps: El número de mensajes de ICMP recibidos de ICMP Timestamp Reply.
icmpInAddrMasks: El número de mensajes de ICMP recibidos de ICMP Address Mask Request.
icmpInAddrMaskReps: El número de mensajes de ICMP recibidos de ICMP Address Mask Reply.
icmpOutMsgs : El número total de mensajes de ICMP que esta entidad trató de enviar. Este contador incluye todos aquellos contados por icmpOutErrors.
icmpOutErrors: El número de mensajes de ICMP que esta entidad no envió debido a que ICMP descubrió problemas como falta de buffers. Este valor no incluye los errores descubiertos fuera de la capa de ICMP como la inhabilidad de IP para enrutar un datagrama resultante.
icmpOutDestUnreachs: El número de mensajes de ICMP Destination Unreachable que esta entidad envió.
icmpOutTimeExcds: El número de mensajes de ICMP Time Exceeded messages sent."
icmpOutParmProbs El número de mensajes de ICMP Parameter Problem messages que esta entidad envió.
icmpOutSrcQuenchs El número de mensajes de ICMP Source Quench messages sent que esta entidad envió.
icmpOutRedirects: El número de mensajes de ICMP Redirect que esta entidad envió. Para una estación final este objeto es siempre cero porque las estaciones no reenvían *redirects*.
icmpOutEchos: El número de mensajes de ICMP Echo (request) que esta entidad envió.
icmpOutEchoReps: El número de mensajes de ICMP Echo Reply que esta entidad envió.
icmpOutTimestamps: El número de mensajes de ICMP Timestamp (request) que esta entidad envió.
icmpOutTimestampReps: El número de mensajes de ICMP Timestamp Reply que esta entidad envió.
icmpOutAddrMasks: El número de mensajes de ICMP Address Mask Request que esta entidad envió.
icmpOutAddrMaskReps: El número de mensajes de ICMP Address Mask Reply que esta entidad envió.

5.4.2.2 UDP

udpInDatagrams: El número de datagramas de UDP de entrada

udpNoPorts: El número de datagramas de UDP en los cuales no había una aplicación en el puerto destino.
udpInErrors: El número de datagramas de UDP que no pudieron ser entregados cualquier razón, excepto que no hubiera aplicación en el puerto destino.
udpOutDatagrams: El número de datagramas de UDP enviados desde esta entidad.
udpTable: Una tabla conteniendo toda la información de puertos abiertos de UDP
udpEntry: Información acerca de un escuchante de UDP en particular.

5.4.2.3 TCP

tcpActiveOpens: El número de veces que las conexiones de TCP han tenido una transición del estado SYN-SENT desde el estado de CLOSED. tcpPassiveOpens El número de veces que las conexiones de TCP han tenido una transición del estado SYN-RCVD desde el estado de
tcpAttemptFails El número de veces que las conexiones de TCP han tenido una transición del estado SYN-SENT o del estado SYN-RCVDdesde el estado de CLOSED, más el número de veces que las conexiones de TCP ha cambiado al estado de LISTEN desde el estado de SYN-RCVD
tcpEstabResets: El número de veces que las conexiones de TCP han tenido una transición al estado de CLOSED desde el estado de ESTABLISHED o del estado de CLOSE-WAIT.
tcpCurrEstab: El número de conexiones de TCP cuyo estado actual es ESTABLISHED o CLOSE- WAIT.
tcpInErrs: El número de segmentos recibidos con error (ej. Checksus de TCP equivocados)
tcpOutRsts: El número de segmentos de TCP conteniendo la bandera de RST.

5.4.3 Uso de Unidad Central de Procesamiento (CPU) en enrutadores Cisco

Las siguientes variables son parte del OLD-CISCO-CPU-MIB:

busyPer: El porcentaje de uso del CPU en los últimos 5 segundos.
avgBusy1: El porcentaje del promedio de uso en el último minuto
avgBusy5: El porcentaje del promedio de uso en los últimos cinco minutos

La siguiente variables es parte del CISCO-MEMORY-POOL-MIB:

ciscoMemoryPoolUtilization1Min: Esta es la utilización de los pools de memoria en los último minuto.
ciscoMemoryPoolUtilization5Min: Esta es la utilización de los pools de memoria en los últimos cinco minutos.

ciscoMemoryPoolUtilization10Min: Esta es la utilización de los pools de memoria en los últimos diez minutos.

Conclusiones

Todas estas variables (IF, IP, Cisco) se requerirá que se monitoreen durante un período de tiempo para verificar que tienen un comportamiento repetitivo y que se puede modelar un patrón en base a este comportamiento. Posteriormente en base a este análisis se deberá generar un algoritmo que de forma pro-activa detecte anormalidades en el patrón de tráfico.

Capítulo 6

Análisis de Variables y Diseño de algoritmo

Introducción

En el capítulo anterior se analizaron una serie de variables que pueden servir para detectar un comportamiento anormal en una red debido a una falla, a una Negación de Servicio, o a un gusano o virus. Para poder diseñar un algoritmo capaz de detectar esas anormalidades primero es necesario definir si las variables tienen un comportamiento predecible y que siga cierto patrón de comportamiento.

Para simplificar el análisis inicial se limitará al uso de dos variables en forma general, además se incluyen pequeños análisis de otras siete variables. Las variables que se usarán se describieron en el capítulo anterior y son las siguientes:

General
- ifInOctets, ifOutOctets
- busyPer
- avgBusy1
- avgBusy5
- ciscoMemoryPoolUtilization1Min
- ciscoMemoryPoolUtilization5Min
- ciscoMemoryPoolUtilization10Min

Experimentos Simple
- ipForwDatagrams
- icmpInMsgs, icmpOutMsgs
- icmpInEchos, icmpOutEchos
- icmpInEchoReps, icmpInEchoReps

Para la medición de estas variables se usará la herramienta Cricket descrita en el capítulo 4. Específicamente para el comportamiento de *ifInOctets*, *ifOutOctets* y uso de CPU las mediciones se hicieron usando redes en producción. La variedad de fuentes fue diversa y esta incluyó enlaces a Internet, enlaces WAN, uso de puertos LAN, enrutadores y conmutadores LAN.

Los experimentos simples se hicieron midiendo el comportamiento de las variables en una red experimental especialmente construida para este propósito. El diagrama de esta red puede observarse en la figura 6.13

6.1 Tráfico en bits de entrada y salida

De acuerdo a las observaciones de las variables *ifInOctets*, *ifOutOctets* en todo tipo de enlaces que incluye enlaces de Internet, WAN y LAN tiene un comportamiento normal en la utilización de éstos. Como se puede ver en las gráficas de las figuras 6.1, 6.2, 6.3 y 6.4 la utilización de los enlaces comienza a cierta hora del día a bajos niveles para ir subiendo hasta alcanzar su máximo, posteriormente irá disminuyendo hasta alcanzar el mínimo. Las horas en que los enlaces alcanzan su mínimo y máximo varían dependiendo el tipo de enlaces y el uso de este, pero en general se puede observar que el mínimo se alcanza por las madrugadas y el máximo alrededor del medio día.

A partir de las gráficas se puede observar que el comportamiento también varía de acuerdo al día de la semana. Como regla general se observa que durante los días laborales los enlaces se usan más que en los fines de semana. El uso en cada día es variable, en la figura 6.1 puede notarse un comportamiento uniforme, mientras que en la figura 6.4 el comportamiento es menos uniforme, esto se debe a que la figura 6.1 elimina las particularidades ya que muestra el comportamiento de muchos usuarios y no de uno solo como en la figura 6.4. Por otro lado el comportamiento de uso de Internet de un centro de datos es interesante ya que además de ser normal en su uso diario, también lo es en su uso semanal, ya que el sábado es el día de menor actividad y el lunes/martes el de mayor, esto puede deberse a un comportamiento particular de los usuarios para el uso de los recursos en este centro de datos. La gráfica 6.5 muestra este comportamiento normal.

Otro comportamiento interesante es que las gráficas de Internet de un enlace de usuarios y una de un centro de datos muestran un comportamiento totalmente contrario en cuanto a la dirección de los datos. Mientras que para el enlace a Internet de usuarios se nota que la mayor utilización es de entrada (figura 6.1) mientras que la utilización mayor de Internet de un centro de datos es de salida (figura 6.2). Esto se debe a que los usuarios consumen información (que entra) y los centros de datos producen información (que sale). Debido a este comportamiento diferente, puede resultar complejo el generar manualmente alarmas en los límites de utilización.

Figura 6.1
Enlace de Internet
Cortesía Telecomunicaciones y Redes Tecnológico de Monterrey

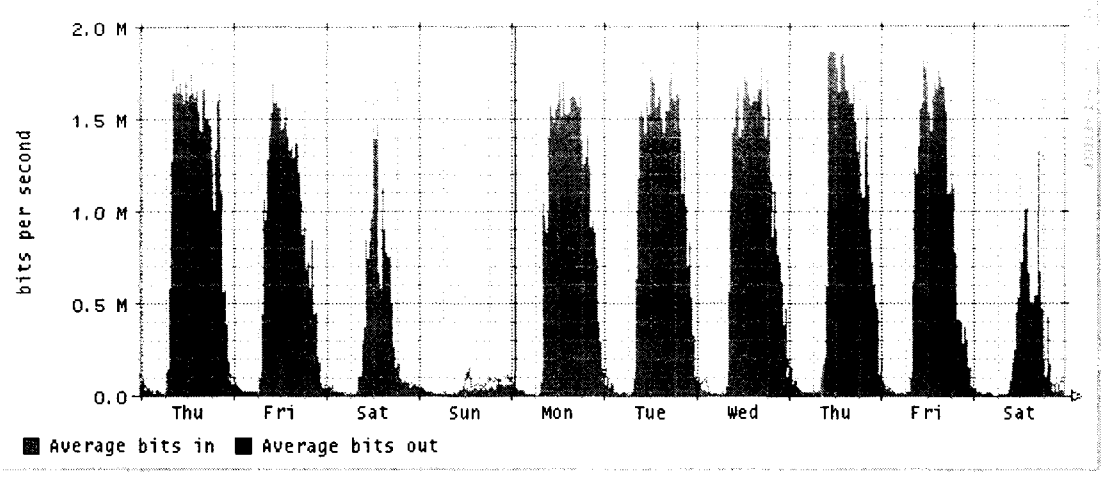


Figura 6.2
Enlace de Internet Centro de Datos
Cortesía Telecomunicaciones y Redes Tecnológico de Monterrey

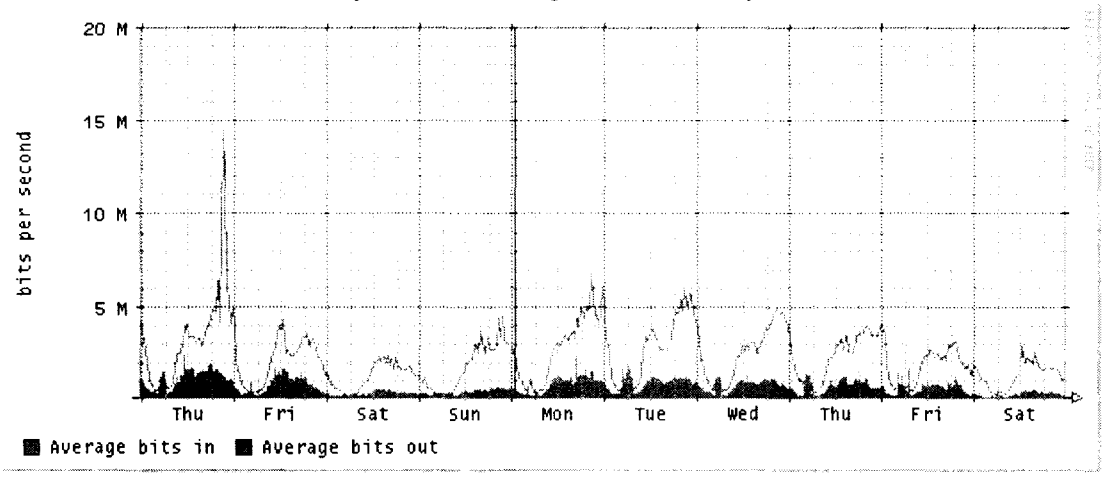


Figura 6.3
Enlace de LAN (servidor Web)
Cortesía Telecomunicaciones y Redes Tecnológico de Monterrey

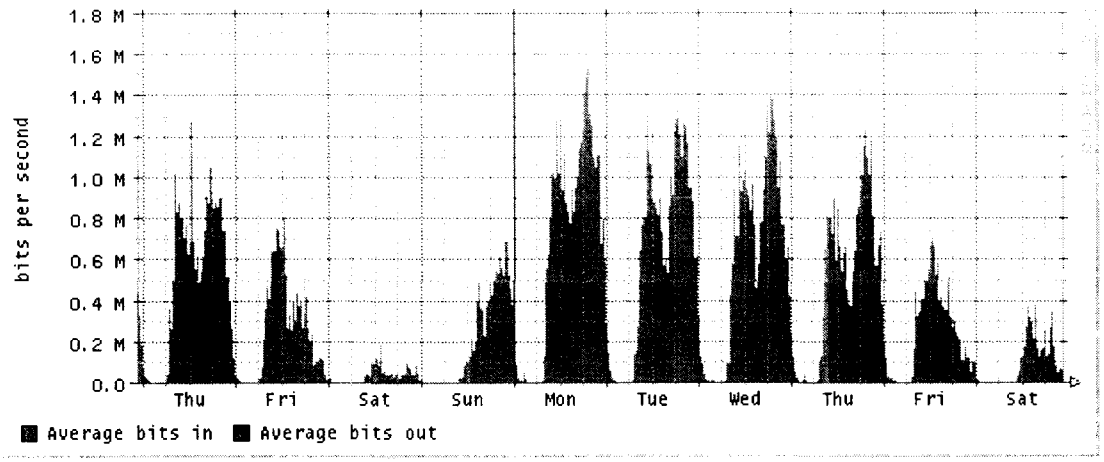


Figura 6.4
Enlace de LAN (usuario)

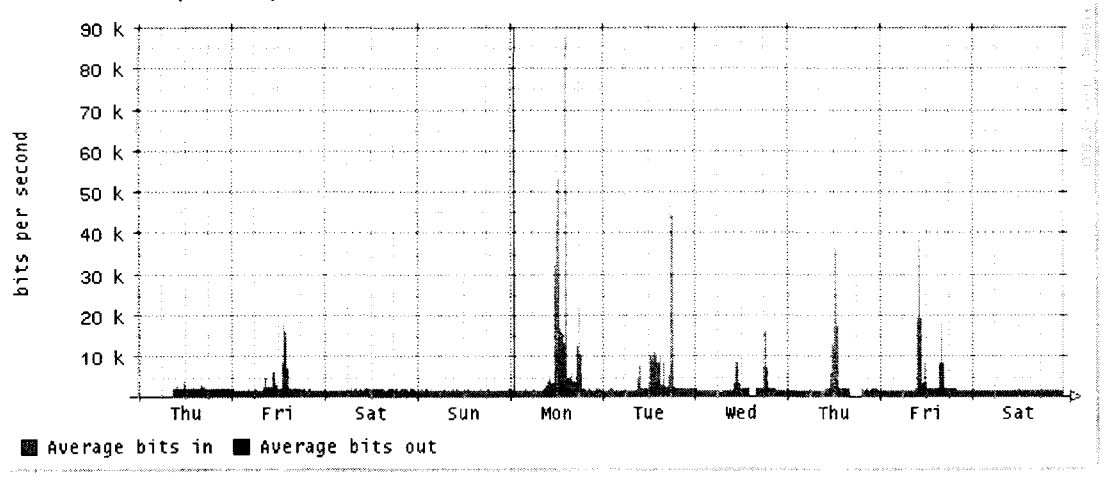
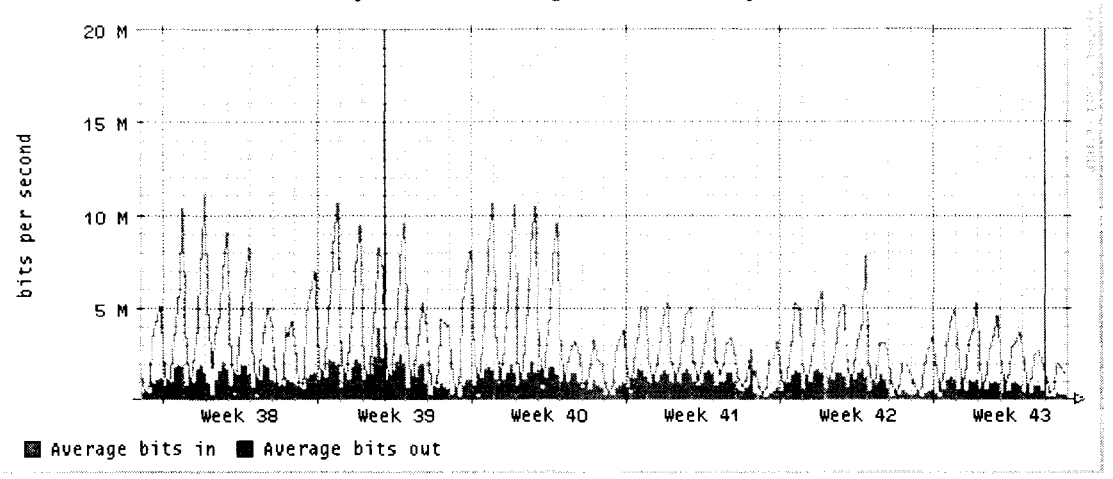


Figura 6.5
Enlace de Internet Centro de Datos
Cortesía Telecomunicaciones y Redes Tecnológico de Monterrey



La grafica de la figura 6.5 no representa al 100% un comportamiento similar en todas sus semanas, sin embargo si se compara por separado el uso de este enlace en las semanas 38, 39 y 40 y posteriormente se compara de la semana 41 a la 43, se nota que entre estos grupos si existe un comportamiento similar y predecible. El cambio entre la semana 40 y 41 deberá analizarse para ver si afecta en la predicción del tráfico y el por qué de este cambio. Sin embargo existen enlaces con comportamientos tan estables y predecibles que es posible notar esta similitud en un período de al menos un año como se muestra en las gráficas de las figuras 6.6 y 6.7.

Figura 6.6
Enlace de Internet
Cortesía Telecomunicaciones y Redes Tecnológico de Monterrey

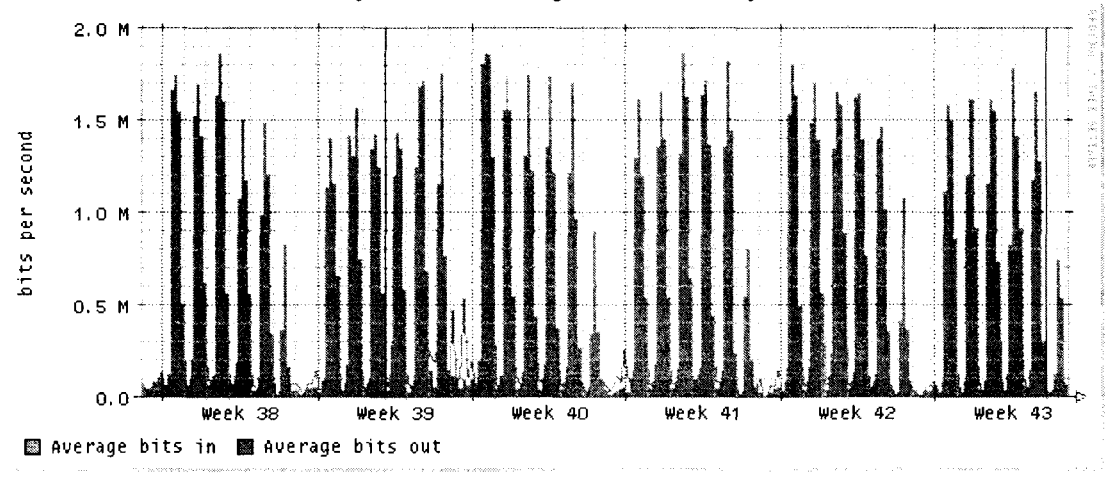
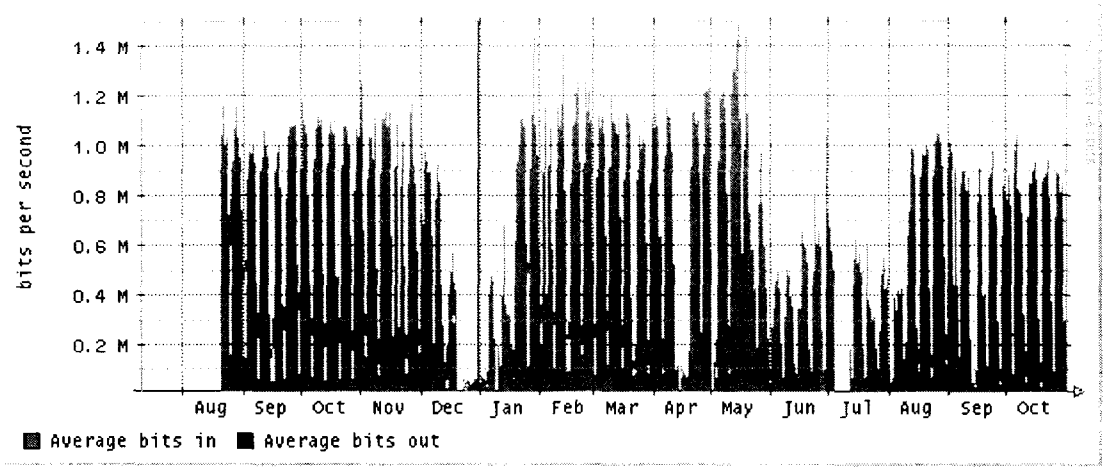


Figura 6.7
Enlace de Internet
Cortesía Telecomunicaciones y Redes Tecnológico de Monterrey



Como ha podido observarse el comportamiento de las variables *ifInOctets*, *ifOutOctets* en las redes bajo operación sin fallas y sin cambios en el ambiente puede ser normal y seguir un patrón determinado, incluso durante períodos largos de tiempo. También se puede concluir que se podría diseñar un algoritmo para predecir este comportamiento e identificar posibles anomalías en él.

6.2 Uso de CPU y Memoria

Al igual que se analizó el comportamiento de las variables de *ifInOctets*, *ifOutOctets* también se analizaron las variables *busyPer, avgBusy1, avgBusy5 y ciscoMemoryPoolUtilization5Min* (se decidió no incluir las variables *ciscoMemoryPoolUtilization1Min* y *ciscoMemoryPoolUtilization10Min* para simplificar la configuración de Cricket).

Como se observa en las gráficas de las figuras 6.8 a la 6.10 el comportamiento de las variables de uso de CPU tienen un comportamiento periódico y siguen un patrón determinado incluso en períodos largos de tiempo. Para la variable de memoria libre el comportamiento es prácticamente una constante con leves variaciones (figuras 6.11 y 6.12)

Figura 6.8
Enlace de Internet
Cortesía Telecomunicaciones y Redes Tecnológico de Monterrey

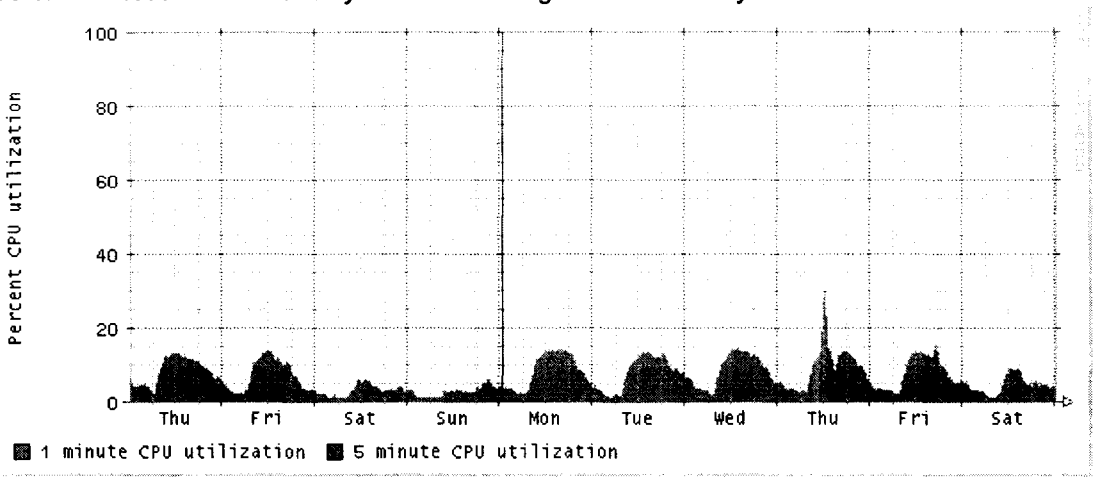


Figura 6.9
Enlace de Internet
Cortesía Telecomunicaciones y Redes Tecnológico de Monterrey

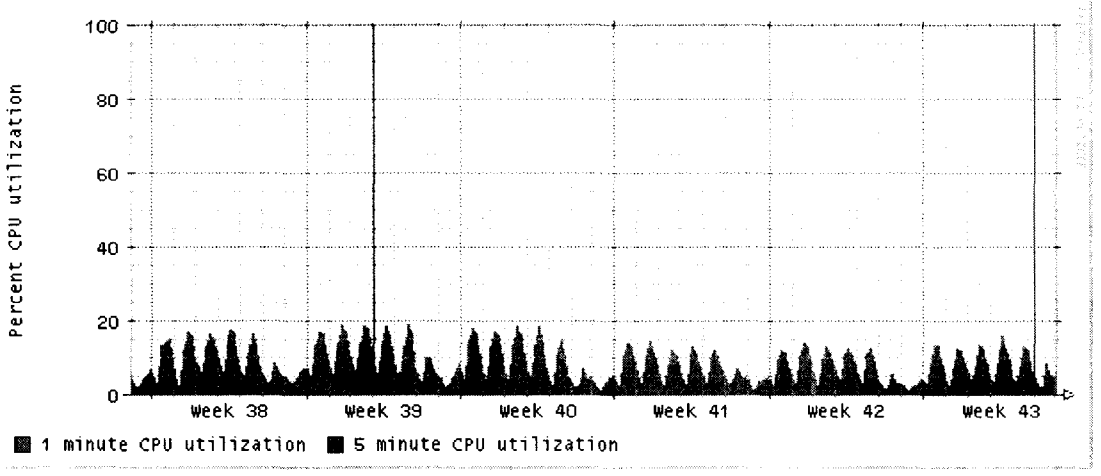


Figura 6.10
Enlace de Internet
Cortesía Telecomunicaciones y Redes Tecnológico de Monterrey

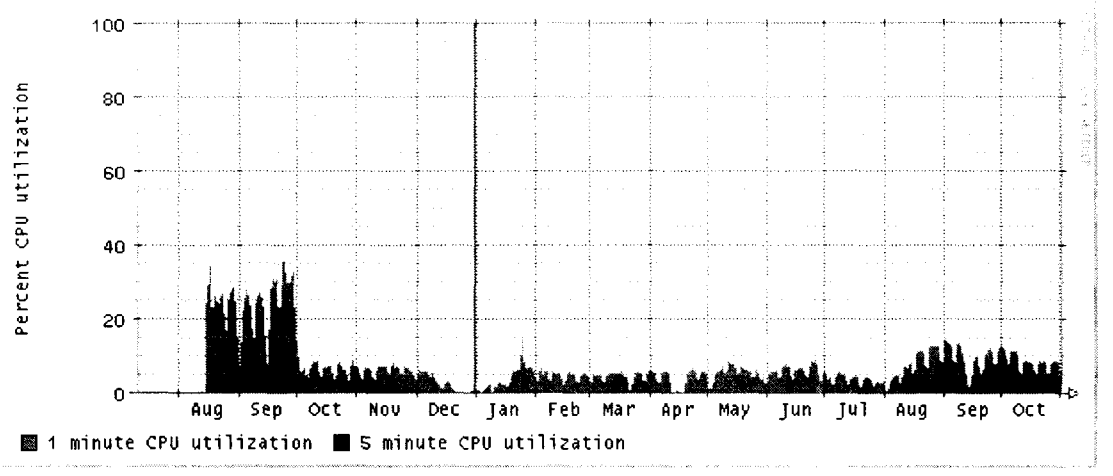


Figura 6.11
Enlace de Internet
Cortesía Telecomunicaciones y Redes Tecnológico de Monterrey

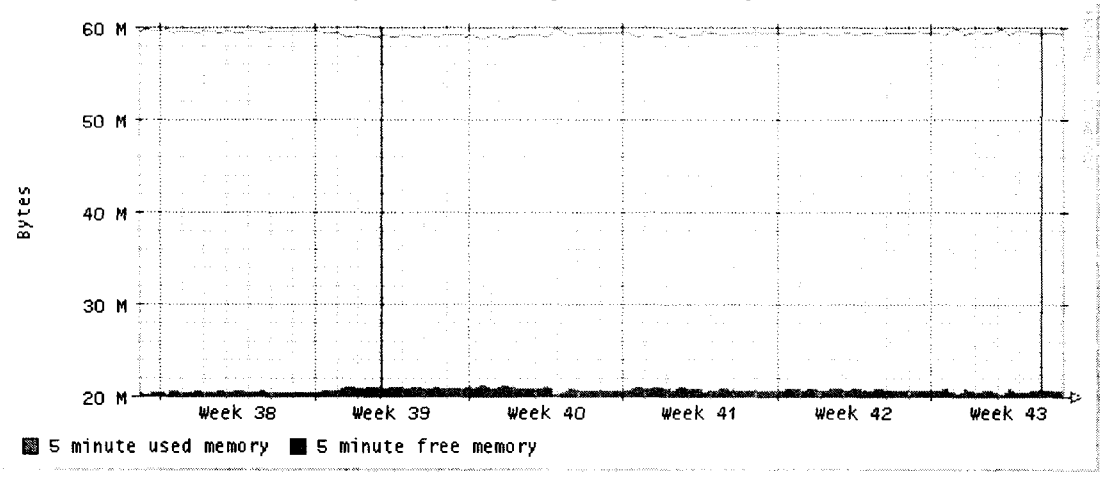
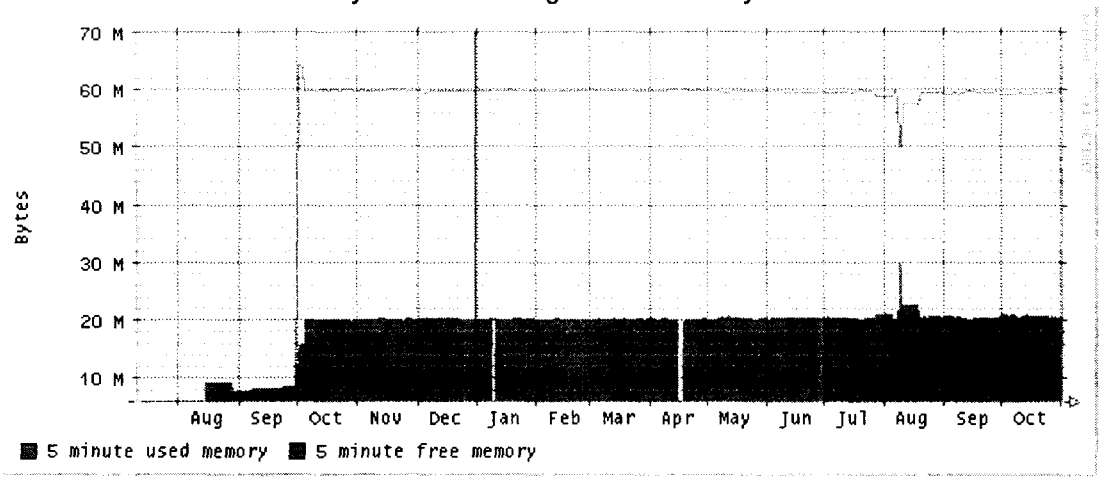


Figura 6.12
Enlace de Internet
Cortesía Telecomunicaciones y Redes Tecnológico de Monterrey

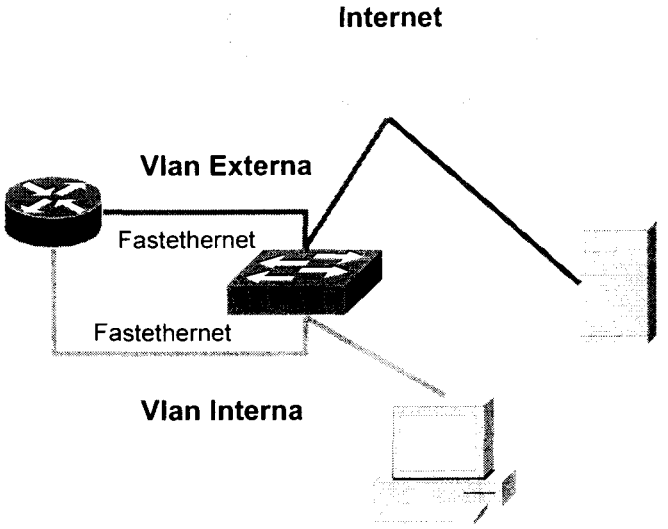


Para las variables de uso de CPU se concluye lo mismo que para las variables de utilización de enlace, esto es en pocas palabras que existe un comportamiento normal y periódico que podría ser moldeado y predecible mediante un algoritmo. Para la variable de utilización de memoria se detecta un comportamiento constante que no varía en dependiendo de la hora o día de la semana, es por ello que para encontrar anomalías que afecten esta variable solo es necesario una simple alarma de límite como la existente en todos los sistemas de monitoreo analizados y no se cree necesario un sistema más inteligente como el requerido para las variables de utilización de enlace y CPU.

6.3 IP

Para el comportamiento de las variables en el MIB-II bajo la categoría de IP e ICMP no se tienen datos de redes de producción, excepto aquellos tomados de redes como la de Abilene. Para la toma de muestras de estas variables se construyó una red experimenta cuyo diagrama se observa en la figura 6.13.

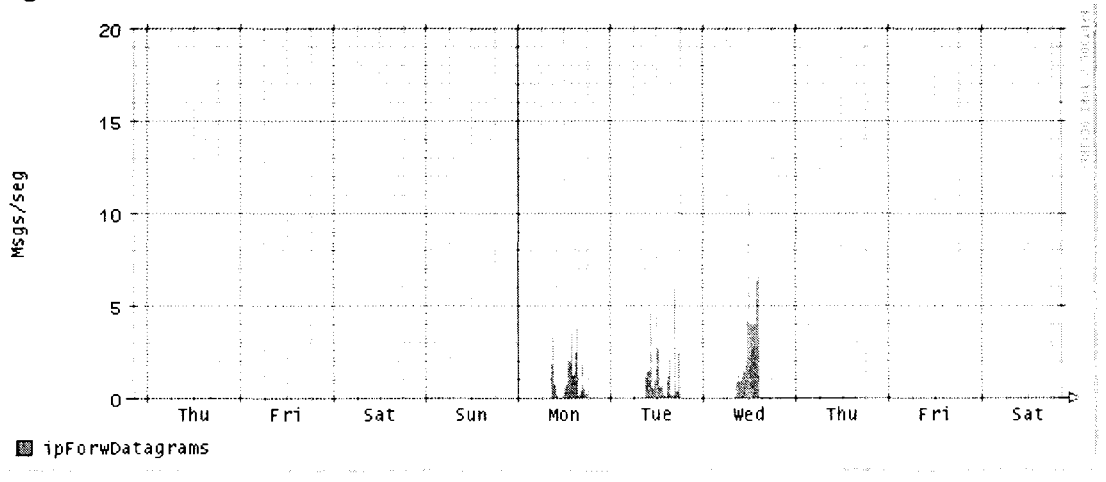
Figura 6.13
Diagrama de Red Experimental



6.3.1 IP

De las variables bajo el grupo IP solo se analizó ipForwDatagrams. En la gráfica 6.14 se ven las muestras tomadas de la red experimental. Aunque la distribución normal no es tan visible ya que solo se muestreo el comportamiento de un sistema.

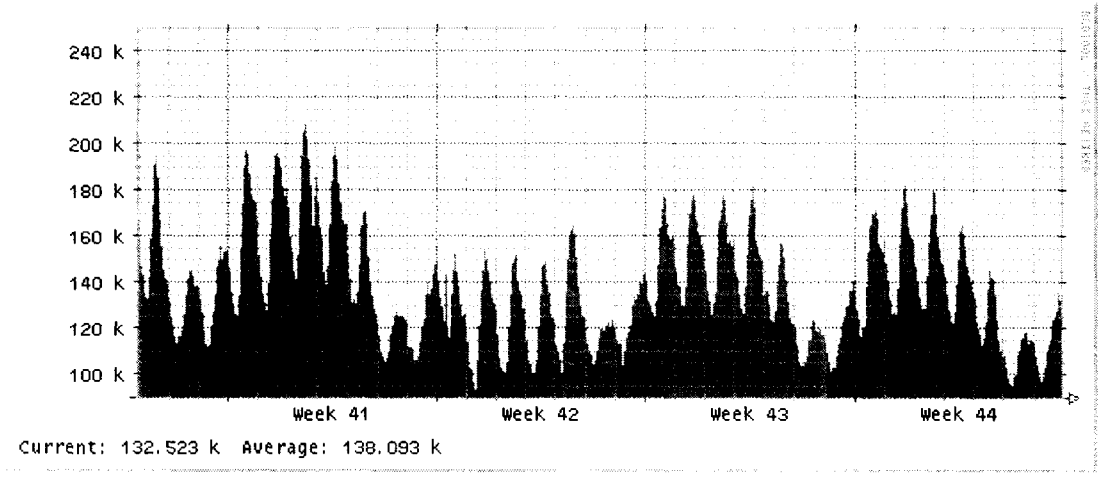
Figura 6.14



6.3.2 ICMP

Para el grupo ICMP de MIB-II se analizaron las variables icmpInMsgs, icmpOutMsgs, icmpInEchos, icmpOutEchos, icmpInEchoReps e icmpInEchoReps por considerase las variables más interesantes. La figura 6.15 muestra los paquetes de icmp-echo-request que atraviesan la red de Abilene en Estados Unidos. Como puede observarse, a pesar de ser el resultado de una gran cantidad de sistemas infectados con el gusano W32/Blaster las gráficas muestran un comportamiento normal y periódico.

Figura 6.15
Cortesía de REN-ISAC



Las gráficas de las figuras 6.16, 6.17 y 6.18 muestran el comportamiento de ICMP en la red experimental. La figura 6.16 muestra todos los mensajes de ICMP de todo tipo (echo-reques, echo-reply, unrechables, etc) tanto de entrada como de salida procesados por la red experimental. La gráfica 6.17 muestra los mensajes de ICMP echo-request de entrada y salida, la gráfica 6.18 muestra los mensajes de ICMP echo-reply de entrada y salida.

Figura 6.16

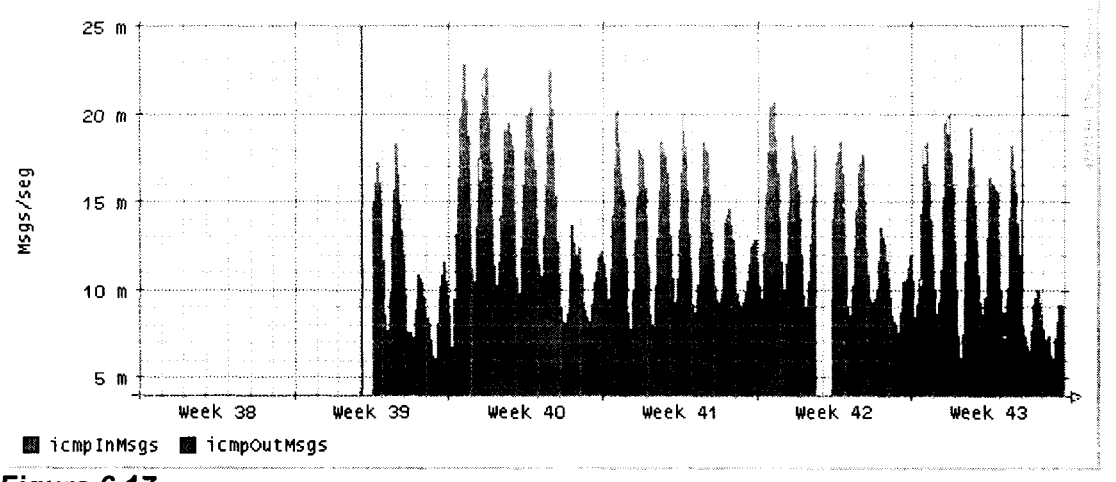


Figura 6.17

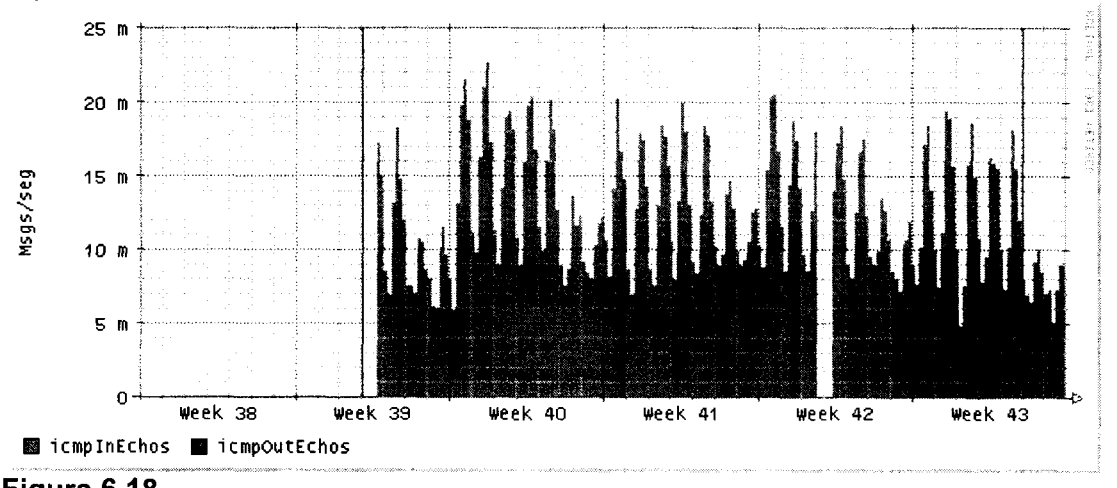
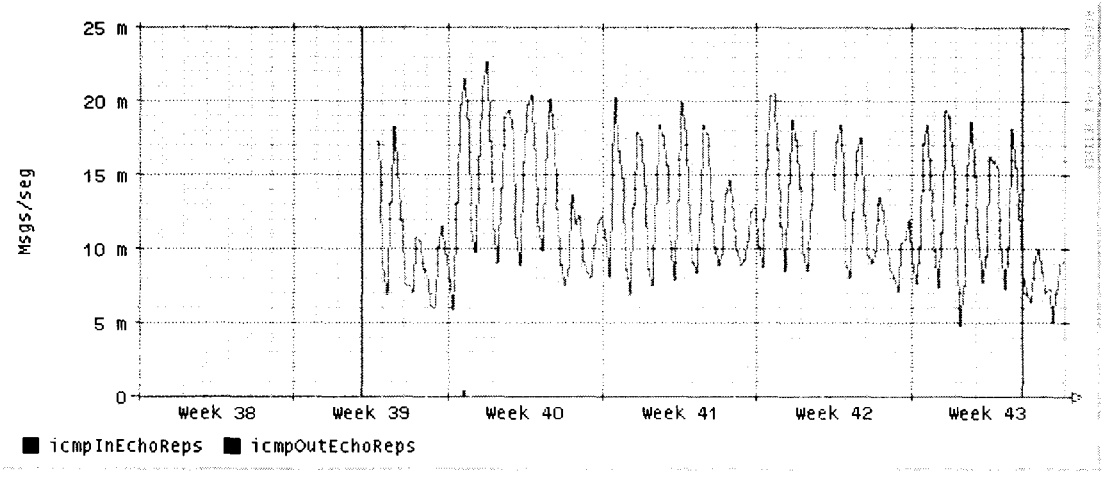


Figura 6.18



Como se puede observarse el comportamiento de las variables es normal y periódico. Existe una mayor actividad de ciertas variables debido a la existencia de sistemas infectados con el gusano W32/Blaster.

Conclusiones

En general en todas las variables observadas y en todos los escenarios se encuentra que estas se comportan de forma periódica y siguen un patrón determinado en ciertos períodos de tiempo. Esto lleva a que puede ser factible diseñar un algoritmo y posteriormente un sistema capaz de identificar que este patrón se rompe. Esta ruptura del patrón normal de comportamiento indica una anomalía en el sistema la cual bien podría ser causada por una falla.

Capítulo 7

Construcción del Algoritmo

Introducción

El objetivo de este estudio es identificar si existe un comportamiento periódico y predecible de algunas variables de comportamiento que pueden ser medidas en los equipos de redes para que a partir de ello pudiese generarse un algoritmo y un sistema capaz de identificar anomalías y fallas en la red de manera proactiva y automática.

El comportamiento se ha identificado y el siguiente paso es el diseño del algoritmo que sea capaz de identificar anomalías en el comportamiento de las variables monitoreadas. En este capítulo se analizarán herramientas estadísticas y de métodos numéricos que puedan auxiliar en el diseño del algoritmo. Posteriormente diseñado un algoritmo con base teórica y en la observación se ejecutarán una serie de pruebas o experimentos para comprobar la eficacia del algoritmo, si se comprueba que este no cumple con los objetivos de diseño se modificará cuantas veces sea necesario hasta tener un resultado satisfactorio.

Para los experimentos y la comprobación del algoritmo se ha diseñado un programa que se describirá en el capítulo 8; los experimentos y sus resultados se detallarán en el capítulo 9.

7.1 Bases del Algoritmo

De acuerdo al modelo conceptual del estudio del capitulo 2 (ver figura 2.2), hasta este momento hemos ya definido las variables, las hemos monitoreado y hemos encontrado que siguen un patrón definido y periódico. El siguiente paso es como definir un algoritmo que pueda identificar variaciones en el patrón de tráfico como resultado de una falla o una anomalía no benéfica en el servicio de red.

El algoritmo resultante deberá ser probado mediante su programación en un sistema computacional. Si el resultado no es el esperado se deberá modificar el algoritmo y repetir el proceso de verificación. Este proceso de retroalimentación debe ser repetido cuantas veces sea necesario hasta obtener los resultados esperados. El algoritmo resultante debe cumplir con lo siguiente:

- Tener fundamentos teóricos

- Tener fundamentos prácticos basados en la observación y en el análisis numérico de datos
- Debe ser matemáticamente sencillo para un implementación en software sencilla
- Debe ser confiable en los resultados obtenidos

Para auxiliar en la definición del algoritmo se analizarán algunas bases teóricas de métodos numéricos, probabilidad y estadística. De acuerdo a Mason y Lind (1995) se define:

Promedio: Valor que representa un conjunto de datos. Señala un centro de los valores.
Media Aritmética: Medida de tendencia central que se obtiene de la suma de todos los valores dividida entre el número total de valores.

$$\bar{X} = \frac{\sum X}{n}$$

Dispersión: Indica que tan separados están los datos unos de otros. Mediante una medida de dispersión es posible evaluar la confiabilidad del promedio que se está utilizando.
Desviación media: Conocida también como desviación promedio, mide el promedio en donde los valores de una población, o muestra, varían con respecto a su media. En otras palabras es la media aritmética de los valores absolutos de las desviaciones con respecto a la media aritmética.

$$D.M. = \frac{\sum |X - \bar{X}|}{n}$$

Variancia: Media aritmética de las desviaciones cuadráticas con respecto a la media
Desviación estándar: Raíz cuadrada de la variancia
Variancia poblacional:

$$\sigma^2 = \frac{\sum (X - \mu)^2}{N}$$

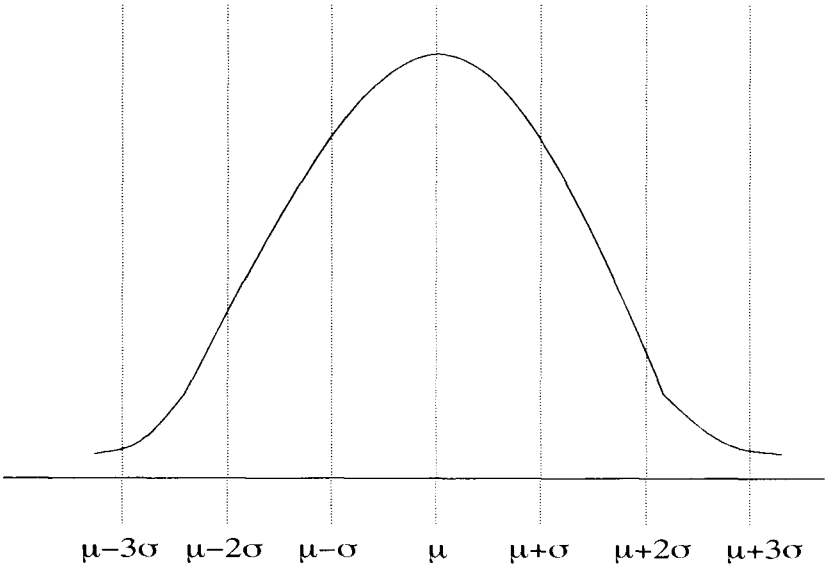
Desviación estándar poblacional:

$$\sigma = \sqrt{\frac{\sum (X - \mu)^2}{N}}$$

Teorema de Chebyshev: El matemático ruso P.L. Chebyshev (o Tchebycheff) desarrolló un teorema que permite determinar la proporción mínima de los valores que se encuentran dentro de un número específico de desviaciones estándares con respecto a la media. El teorema dice "Para un conjunto cualquiera de

observaciones la proporción mínima de los valores que se encuentran dentro de *k* desviaciones estándares desde la media es al menos $1 - 1/k^2$, en donde *k* es una constante menor que 1.
Regla empírica o regla normal: Para la distribución de frecuencias simétrica de campana, aproximadamente el 68% de las observaciones se encontrará a más y menos una desviación estándar de la media; aproximadamente 95% de las observaciones se encontrarán a más y menos dos desviaciones estándares desde la media; y prácticamente todas las observaciones (99.7%) se encontrarán a más y menos tres desviaciones estándares desde la media. Esto puede observarse en la figura 7.1

Figura 7.1



Con estas bases teóricas y con la información recabada se procederá al diseño del algoritmo.

7.2 Diseño Inicial del Algoritmo

De acuerdo a lo concluido con la observación de las diferentes variables que afectan el tráfico en la red se concluyó en el capítulo anterior lo siguiente:

"En general en todas las variables observadas y en todos los escenarios se encuentra que estas se comportan de forma periódica y siguen un patrón determinado en ciertos períodos de tiempo. Esto lleva a que puede ser factible

diseñar un algoritmo y posteriormente un sistema capaz de identificar que este patrón se rompe. Esta ruptura del patrón normal de comportamiento indica una anomalía en el sistema la cual bien podría ser causada por una falla".

Se ha detectado que el comportamiento periódico es similar de un día de la semana (ej. Lunes semana 1) contra otro día de la misma semana (ej. Miércoles semana 1) o similar de un día de una semana (ej. Lunes semana 1) contra el mismo día de otra semana (ej. Lunes semana 4). Para analizar la dispersión y que tan similares son ambas opciones se hizo un análisis de 4 semanas. Los resultados gráficos pueden verse en las figuras de la 7.2 a la 7.5.

Figura 7.2

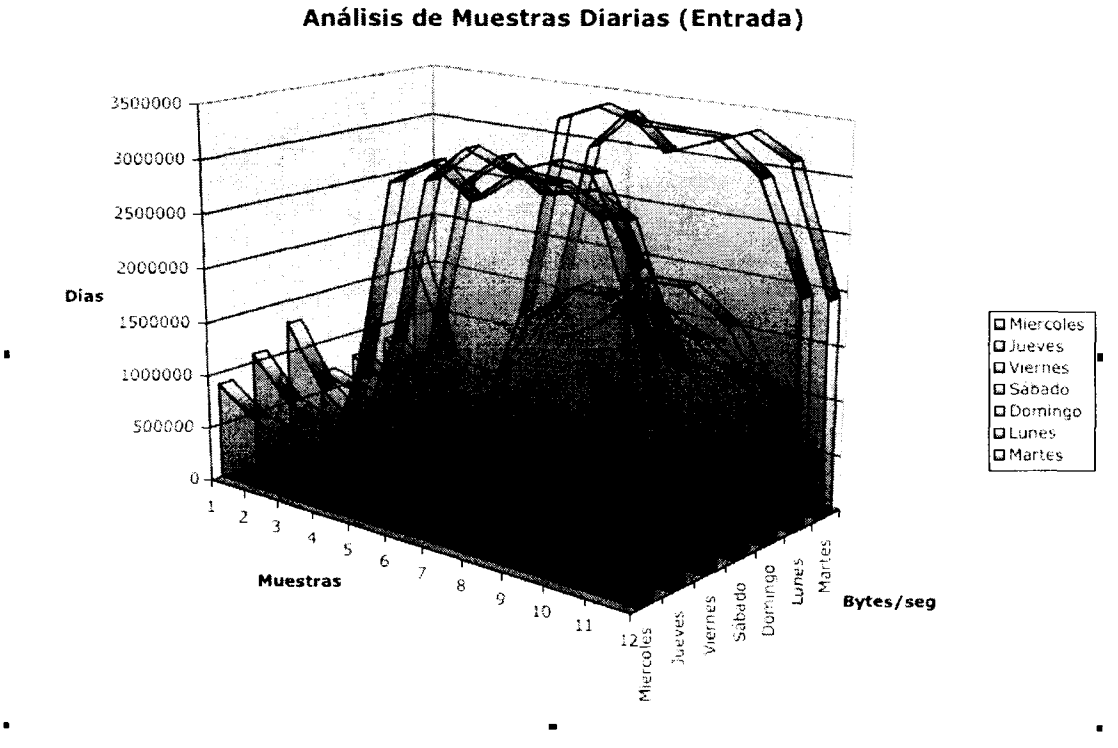


Figura 7.3

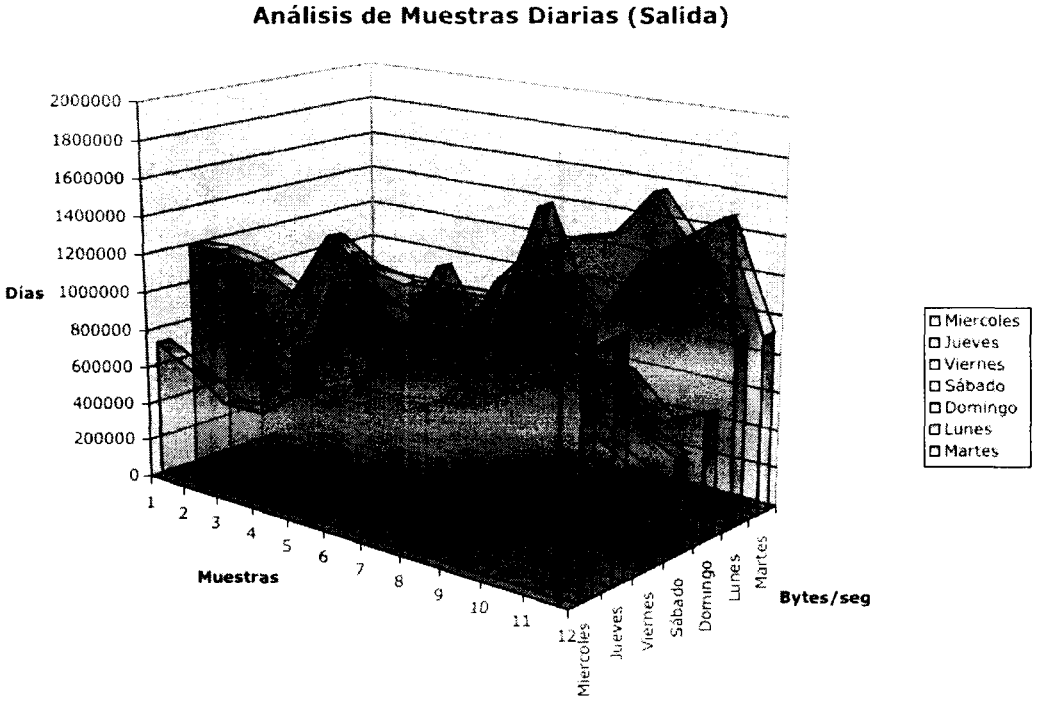


Figura 7.4

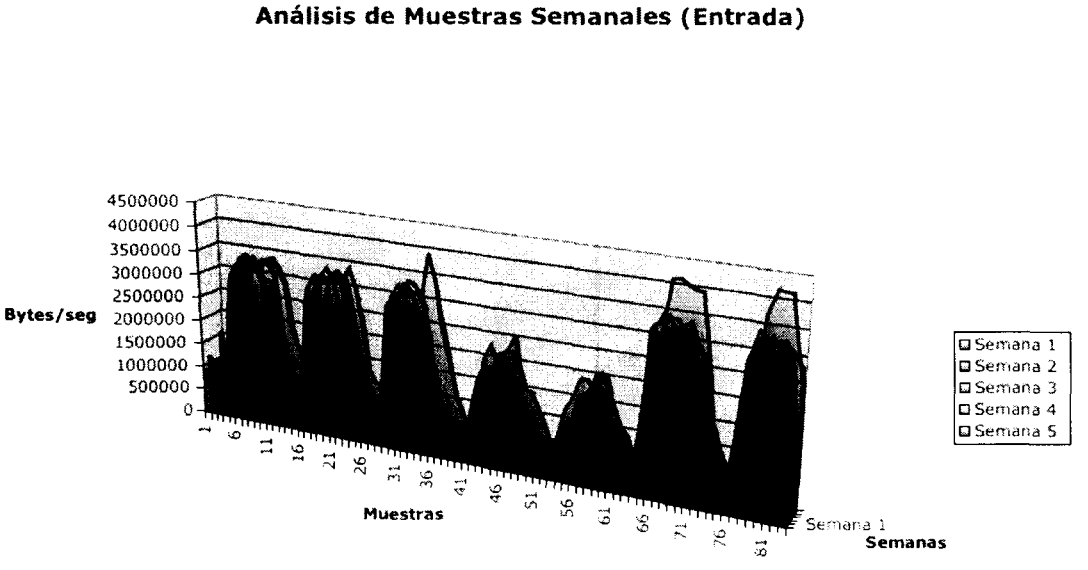
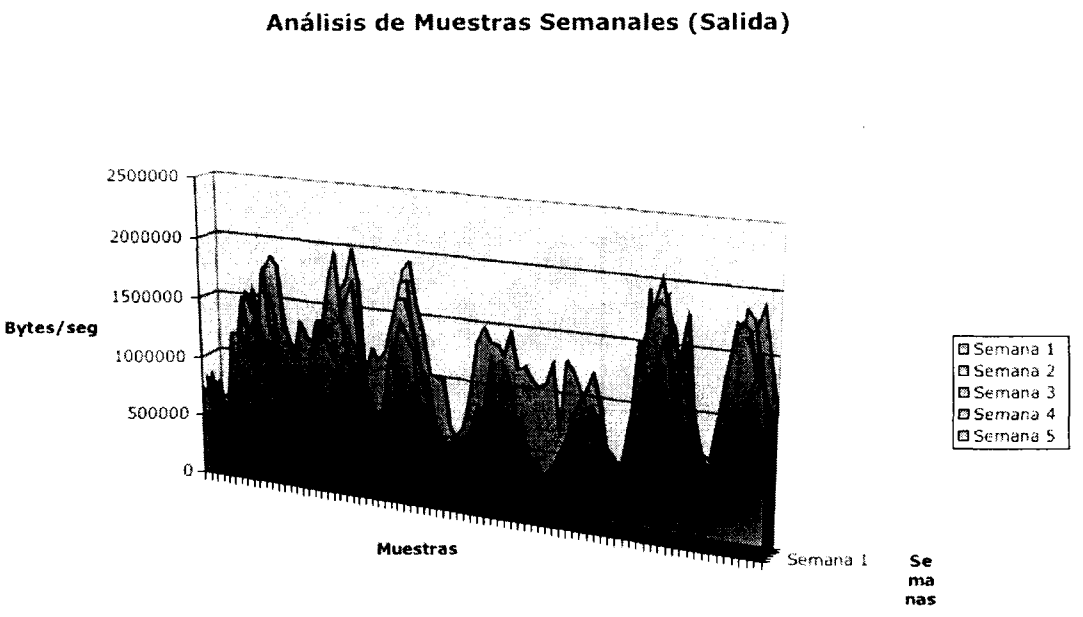


Figura 7.5



La primera opción es atractiva ya que es necesario analizar menos semanas para tener una gran cantidad de datos, sin embargo se debe analizar matemáticamente la dispersión ya que visualmente resulta grande. La segunda opción por observación tiene menos dispersión pero es necesario analizar y guardar los datos de más semanas. La dispersión representada por la desviación estándar de las 4 gráficas se muestra en la tabla 7.1. En esta tabla se muestra claramente que las comparaciones entre semanas resultan con menos dispersión que las comparaciones de un día a otro de la misma semana.

Tabla 7.1

	Entrada	Salida
Diario	477820.795	322771.272
Semanal	229729.193	230882.633

De lo anterior se decide usar las medidas semanales como base para la generación del algoritmo. También en la práctica se observo que el software de monitoreo obtiene el valor de una variable cada 5 minutos pero no es capaz de conservar este valor por más de 36 horas debido a su funcionamiento de base de datos circular. Sin embargo si podemos obtener el valor promedio de las mediciones en períodos de dos horas. Con esto se obtienen 12 muestras diarias y

84 muestras semanales por variables, se puede obtener cualquier número de semanas. El algoritmos se construye de la siguiente forma:

Para obtener los datos de comparación:

- Se toman muestras de la variables y se promedian cada 2 horas para obtener 12 muestras diarias.
- Se agrupan las muestras por semana y se tienen 84 muestras por semana. Cada muestra recibe un número de secuencia de 1 a 84. A este se le llamara muestra número de secuencia o (MNS)
- Se obtiene el promedio y la desviación estándar de cada MNS
- Se obtiene el valor de comparación superior para el número de secuencia (VCNS+) a partir del promedio del MNS más dos veces la desviación estándar y el valor de comparación inferior para el número de secuencia (VCNS-) a partir del promedio del MNS menos dos veces la desviación estándar.

Para la comparación de datos

- Se toma el valor de la variable y se compara con sus correspondientes VCNS. Si el valor de la variable es superior al VCNS+ o inferior al VCNS- indica que ha ocurrido una violación a los patrones de tráfico
- Se ejecuta una acción de alarma que puede ser registrar el incidente, enviar un correo electrónico, generar un trap de SNMP, ejecutar un proceso, todos los anteriores o una combinación de éstos.

Con esto queda construido el algoritmo cumpliendo con los puntos de diseño mencionados al principio del capítulo excepto el de ser confiable. Este punto aún no puede ser aún comprobado y será necesario hacer experimentación para validar esta característica.

7.3 Primera Iteración de Diseño

El diseño descrito en el punto anterior como se verá en el capítulo 8 se programó en un sistema para posteriormente hacer la captura de información. A partir del "Experimento 1" que se verá con más a detalle en el capítulo 9 se hicieron algunas observaciones y modificaciones en el algoritmo. En este experimento se encontró que aunque el algoritmo había probado parte de su funcionamiento al encontrar una anomalía en la red donde se estaba probando, existían defectos en la captura de información que estaban arrojando que los resultados de las comparaciones no fueran totalmente confiables y exactos.

Se detectó que los errores provenían del software del monitoreo y a su inestabilidad como sistema lo que hacía que hubiera muestras vacías. La modificación al algoritmo fue sustituir esas muestras vacías por el último valor muestreado.

La primera iteración del algoritmo quedó como:

- Basado en el algoritmo original.
- Sustituir el valor no muestreado por error en la aplicación por el último valor muestreado.

Este nuevo algoritmo fue la base del "Experimento 2". El resultado del experimento 2 no fue el esperado al tener una mejoría en la confiabilidad y exactitud del algoritmo. Las pruebas numéricas mostraron que la cantidad de posibles falsas alarmas fue incluso mayor que cuando no se sustituían los valores no muestreados.

Se regresó al algoritmo inicial para los experimentos futuros. Además se cambio el muestreo a un servidor con el mismo software de recopilación de información pero con mayor capacidad y bajo un esquema de mayor confiabilidad.

7.4 Segunda Iteración de diseño

Para el "Experimento 3" se utilizó nuevamente el algoritmo original y fue muy similar al "Experimento 1" con la diferencia de que los espacios vacíos por las muestras no tomadas ya no existieron. Al no contener errores y pesar de presentar una mejoría en relación al "Experimento 1" no se pudo comprobar la confiabilidad y exactitud del algoritmo ya que aún existían un gran número de alarmas.

Para encontrar una explicación a todo esto, se hizo un análisis más amplio encontrándose dos razones posibles al comportamiento encontrado. Una que no se comenta en el "Experimento 3" a profundidad pero que si se analiza en el "Experimento 4" que es que una gran parte de las alarmas se da porque el área entre el VCNS+ y el VCNS- es muy pequeña, esto hace que la probabilidad de que exista una alarma al caer el dato fuera de estos límites sea muy grande. La otra razón que se analiza a detalle en forma numérica en el "Experimento 3" es que existen épocas del año en donde el tráfico cambia su patrón de comportamiento y se mueve verticalmente más o menos en relación a otro período. Un comportamiento general de este patrón de tráfico se muestra en la figura 7.6 y en la figura 7.7

Figura 7.6
Enlace a Internet Anual
Cortesía de Telecomunicaciones y Redes del Tecnológico de Monterrey

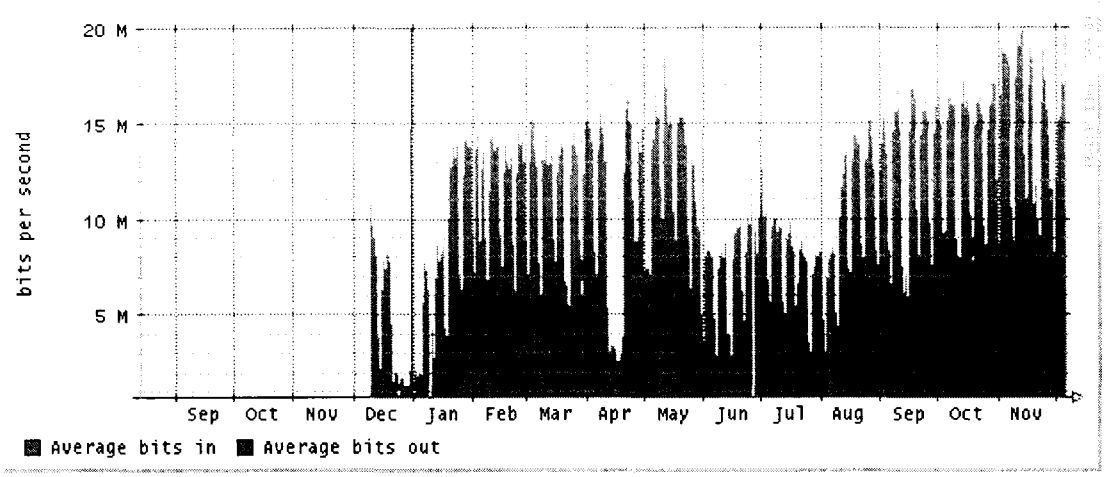
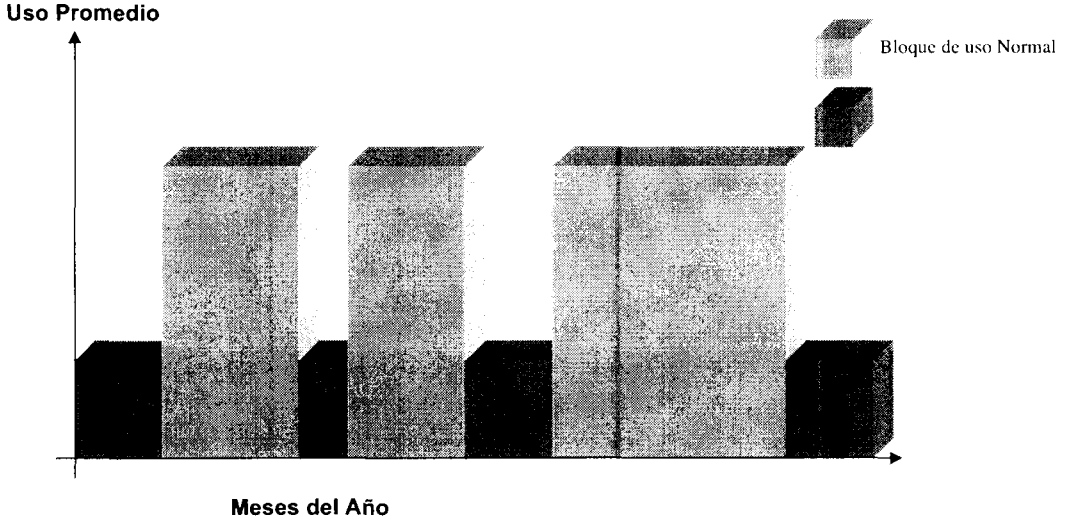


Figura 7.7



En la gráfica de la figura 7.6 se puede observar el comportamiento a largo plazo de un enlace. Como se ve, existen épocas del año donde el comportamiento del enlace es muy similar, pero también existen períodos muy diferentes (en este caso Diciembre-Enero, Abril, Junio y Julio). También por el tipo de organización donde se están analizando los datos se prevé que a partir de mediados de Agosto hasta mediados de Noviembre, el enlace se comporte de una forma muy similar a los meses de Febrero-Marzo y Mayo. En la figura 7.7 se pueden observar dos

tipos de bloques, en gris-azul los bloques donde la demanda de uso del enlace es alta y en naranja los bloques donde la demanda es baja.

Para el "Experimento 3" los datos fuente para generar los datos de comparación VCNS+ y VCNS- se tomaron del final un período de alta utilización y del principio del período de baja utilización contiguo, además los datos con los que se compararon fueron de un período de baja utilización. Esto trajo como resultado un problema de incongruencia de datos entre la alta y la baja utilización al tener una alta dispersión y que los VCNS+ y los VCNS- fueran mucho mayores que lo que deberían con resultados como el que una muestra ni siquiera alcanzaba el valor del VCNS-.

El algoritmo resultante es similar al original con las siguientes modificaciones:

- Se elimina el VCNS-
- Se divide el año en dos Períodos de Uso (PU). El Período de Alto Uso (PAU) y el Período de Bajo Uso (PBU).
- Los datos muestras MNS para generar el VCNS+ deben pertenecer al mismo tipo de período (PAU o PBU)
- Los datos muestreados deben compararse con el VNCS+ correspondiente a su PU.

7.5 Algoritmo Final

El diseño de algoritmo definitivo se obtuvo a partir de los resultados obtenidos en el "Experimento 3" y a la comprobación que se obtuvo a partir del "Experimento 4". El algoritmo definitivo queda como sigue:

Períodos de Uso (PU)

- Se debe verificar que las MNS no pertenezcan a períodos de uso diferentes
- Para cada período identificado se deben generar sus propios datos de comparación

Para obtener los datos de comparación:

- Se toman muestras de la variables y se promedian cada 2 horas para obtener 12 muestras diarias.
- Se agrupan las muestras por semana y se tienen 84 muestras por semana. Cada muestra recibe un número de secuencia de 1 a 84. A este se le llamara muestra número de secuencia o (MNS)
- Se obtiene el promedio y la desviación estándar de cada MNS
- Se obtiene el valor de comparación superior para el número de secuencia (VCNS) a partir del promedio del MNS más dos veces la desviación estándar.

Para la comparación de datos

- Se toma el valor de la variable y se compara con el VCNS de su PU. Si el valor de la variable es superior al VCNS indica que ha ocurrido una violación a los patrones de tráfico
- Se ejecuta una acción de alarma que puede ser registrar el incidente, enviar un correo electrónico, generar un trap de SNMP, ejecutar un proceso, todos los anteriores o una combinación de éstos.

El diseño anterior se probó en el "Experimento 4" cuyos resultados fueron mucho mejores que el de los experimentos anteriores. Se puede definir que con el último diseño el algoritmo cumple con sus objetivos descritos en el principio de este capítulo, sin embargo como cualquier otra implementación se detectaron posibles mejoras que se comentaran en las conclusiones de este estudio y en el desarrollo del "Experimento 4" en el Capítulo 9.

Conclusiones

En este capítulo se hizo la investigación teórica para el diseño del algoritmo y en base a este se diseño un sistema para su comprobación. De los experimentos realizados se obtuvieron resultados que permitieron modificar el diseño del algoritmo para que este cumpliera con sus objetivos. Para el diseño final del algoritmo en este capítulo se apoyo con argumentos obtenidos a partir de investigación teórica, en un sistema de software desarrollado explícitamente para este estudio y en la experimentación práctica.

El resultado de este capítulo es satisfactorio con comentarios de mejoramiento para futuras modificaciones del algoritmo y del sistema que pueden surgir de futuros estudios tomando este como base.

Capítulo 8

Desarrollo del sistema para la verificación del algoritmo

Introducción

En este capítulo se analizan las herramientas y el lenguaje para la generación de los programas auxiliares para la verificación del algoritmo. Se explicará que lenguaje de programación se escogió y por qué. Se analizará el diseño de los programas así como su funcionamiento. Finalmente se explicará como funciona el sistema completo y como debe utilizarse.

8.1 Análisis del lenguaje de programación

Uno de los objetivos en el diseño del algoritmo es que fuese sencillo de implementar. Por esa misma razón se busca que la implementación del algoritmo en forma práctica sea sencilla también. Otro objetivo que se persigue en la *implementación es que el lenguaje de programación sea eficiente y que permita* que el algoritmo se ejecute en cualquier plataforma de sistema operativo con mínimas o ninguna modificación. Finalmente el lenguaje de programación debe aprovechar las salidas de algunas utilerías, los datos y módulos de la herramienta de Cricket.

En la Tabla 8.1 (Willcam 2003) se comparan diferentes lenguajes de programación. Perl y Java resaltan por su sencillez y poder comparados con C y C++.

Tabla 8.1

	Java	Perl	Smaltalk	C++	C	TCL	Shell(s)
Simple	√	√	√		*	√	*
Orientado a Objetos	√	√	√	*			
Robusto	√	√	√			√	√
Seguro	√	√	*			*	*
Portable	√	√	*	*	*	√	*
Dinámico	√	√	√			√	*
Neutral a Plataforma	√	√	*			*	*
Threads	√	√					
Colección de Basura	√	√	√				
Excepciones	√	√	√				
Interprete/Compilado	I/C	I/C	I/C	C	C	I	I
Desempeño/Velocidad	Alto	Alto	Medio	Alto	Alto	Bajo	Bajo

Característica Existente √
No totalmente implementado *

De las opciones anteriores se escogieron Java, Perl, C y Shell para ser los posibles lenguajes de programación. De la lista se eliminaron los shells ya que dependiendo del tipo de shell que se use y en que sistema operativo se use, es necesario hacer algunas modificaciones, además de que por interpretarse línea por línea no suelen ser tan eficientes (Willcam 2003). De las pocas ventajas que tienen los shells, es una integración sencilla de las herramientas y datos proporcionados por Cricket.

Aunque Cricket está programado en C y ofrece el código fuente para su modificación, por la complejidad del lenguaje se decidió eliminarlo. Entre Java y Perl se escogió Perl por varias razones, pero principalmente porque es un lenguaje más sencillo de aprender y porque Cricket maneja módulos específicos para Perl, lo cual hace la programación mucho mas sencilla.

Perl es un lenguaje de programación de alto nivel escrito por Larry Wall y otros cuantos miles. Deriva principalmente de C y toma influencias de sed, awk, el Shell de Unix y una docena de otros lenguajes. La manipulación de Perl de procesos, archivos y texto loa hace particularmente bueno para tareas que involucran prototipos rápidos, utilerías de sistemas, herramientas de software, acceso a bases de datos, redes y programación Web (Ashton 2002). Otras de las razones por las cuales se escogió Perl fueron (Sys Admin 2003):

- Se compila en el momento de ejecutarse, por ello funciona sin problemas en diversas plataformas como Unix, Windows, NT, Macs, DOS, Plan 9, OS/2, VMS, and AmigaOS.
- Es colaborativo, el CPAN (Comprehensive Perl Archive Network) contiene cientos de programas y utilerías que pueden usarse para el desarrollo de sistemas.
- Es gratis, se distribuye a través de la licencia GNU
- El intérprete de Perl está escrito en C y una década de optimizaciones han resultado en un ejecutable rápido. El compilador de Perl puede convertir un programa a C o un bytecode interno para mayor velocidad, y un puente de Perl a C llamado XS permite pasar de C a Perl y viceversa.

8.2 Perl y los módulos de RRDTool

Para obtener los valores de la base de datos de RRDTool (Round Robin Data Base) que son obtenidos a través de Cricket existe un módulo en Perl con el cual es posible extraer estos valores de forma nativa a Perl, esto es análogo a una subrutina en C. El módulo es llamado a través del comando "use" de Perl con el nombre de RRDs: "`use RRDs;`"

Existen varias funciones, algunas se describen a continuación:

RRDs::last regresa un único valor INTEGER que representa el valor del último tiempo de actualización
RRDs::fetch es la función más compleja en cuanto a la devolución de valores se refiere. Existen 4 valores, 2 enteros normales, un apuntador a un arreglo y un apuntador a un arreglo de apuntadores.

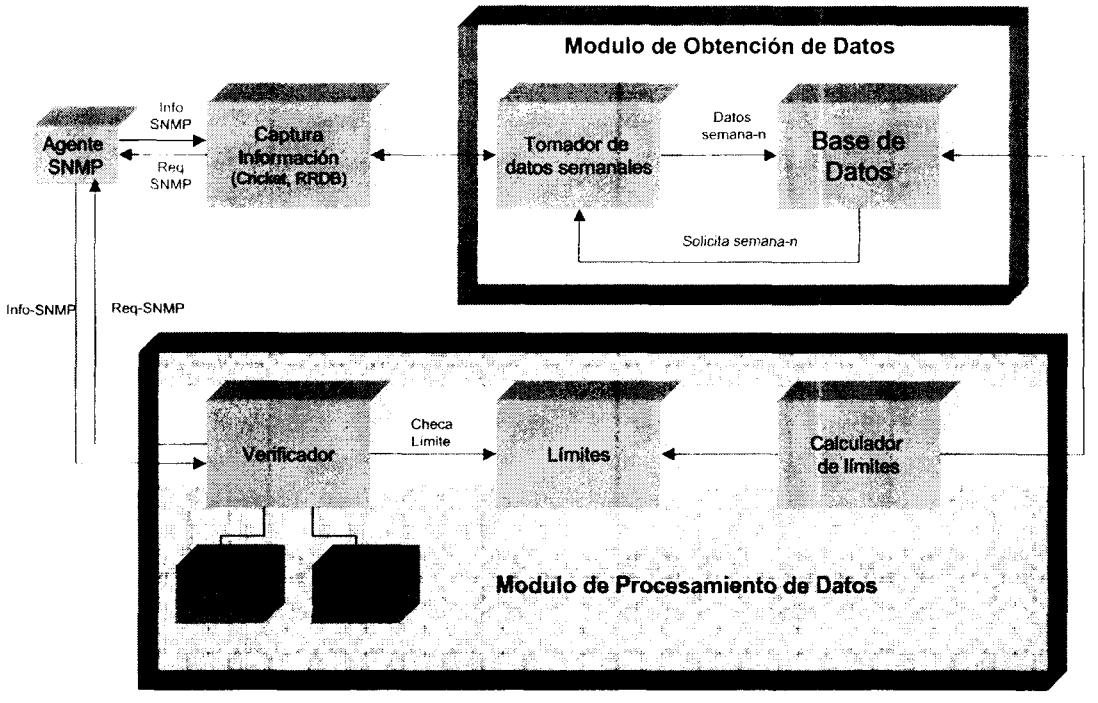
RRDs::error. Esta función debe ser invocada después de alguna otra función para detectar un error. Si la función devuelve un valor igual a cero la función *anteriormente invocada no tuvo ningún error.*

Estas funciones se utilizarán en el código que se generará para probar el algoritmo.

8.3 Componentes del sistema

El sistema se ha diseñado como un sistema modular donde cada módulo tiene una función en específico y la salida de uno sirve como la entrada de otro y cada módulo se ejecuta secuencialmente. La figura 8.1 muestra todos los módulos y su relación uno con otro.

Figura 8.1



Existen dos grandes módulos funcionales, el Módulo de Obtención de Datos y el Módulo de Procesamiento de Datos, cada uno formado de al menos dos sub-módulos.

El Módulo de Obtención de datos tiene como función servir de interfaz entre los datos almacenados en Cricket y las funciones de cálculo de valores de comparación superior e inferior. Este módulo esta formado por la subrutina que recoge semanalmente los valores almacenados en Cricket y los guarda posteriormente en un nuevo formato el cual no pierde detalles de la información almacenada como lo hace Cricket en su mecanismo de sumarización de datos.

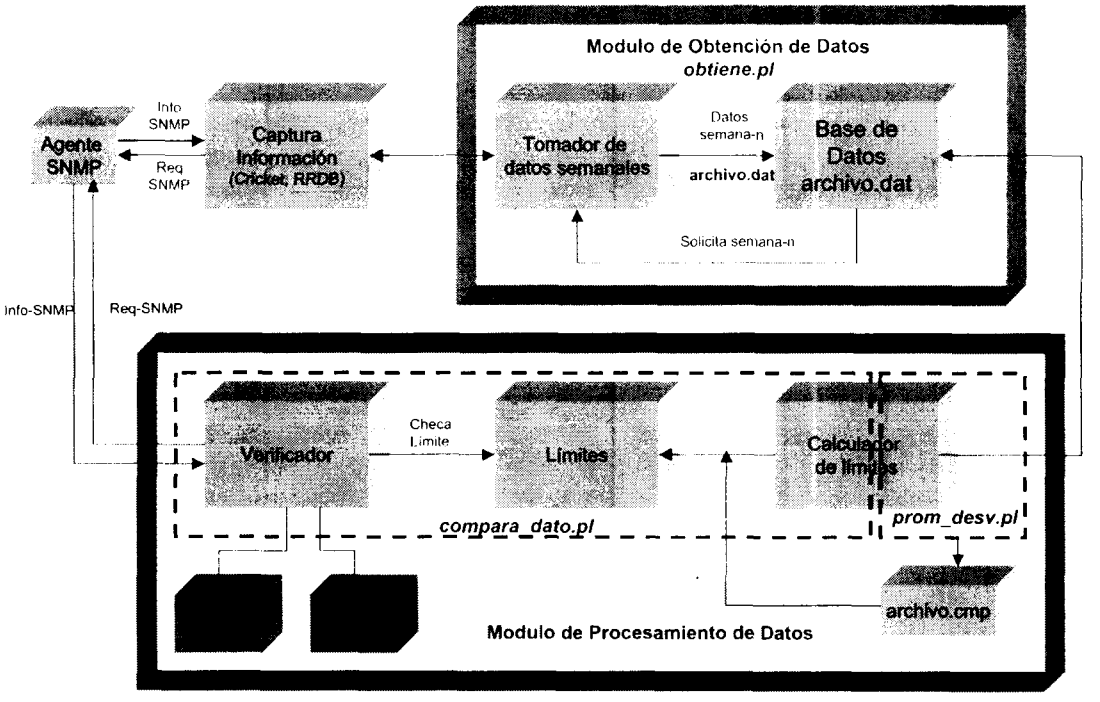
Módulo de Procesamiento de Datos está encargado de obtener los valores de comparación superior e inferior y de verificar que el dato muestreado no sobrepase su valor correspondiente de límite. Este módulo esta compuesto de dos submódulos que ejecutan por separado las dos acciones de cálculo y comparación de límites.

8.4 Módulos de Código

Ya se ha analizado el modelo funcional del sistema y sus diferentes componentes. Para la implementación de este modelo en software se dividieron aun más las funciones de cada módulos en piezas de código. El resultado en código y sus interacciones se muestran en la figura 8.2.

Todo el sistema esta basado en tres programas; "obtiene.pl", "prom_desv.pl" y "compara_dato.pl". Como puede observarse "obtiene.pl" tiene todas las funciones del módulo de Obtención de Datos. La función de cálculo de límites del módulo de Procesamiento de Datos se dividió en dos, la primer parte del cálculo la efectúa el programa de "prom_dev.pl" mientras que el proceso final del cálculo de límites lo efectúa "compara_dato.pl", el cual al mismo tiempo efectúa las funciones de comparación y generación de alarma. Las funciones y operación de cada uno de los programas se describe a continuación.

Figura 8.2



8.4.1 "obtiene.pl"

Nombre

obtiene.pl

Parámetros de entrada

<archivo_entrada> Archivo de base de datos rrd. Este archivo contiene los datos que se obtendrán para generar el arreglo de salida

<día> Día a partir del cual se obtendrán los datos

<mes> Mes a partir del cual se obtendrán los datos

Parámetros de Salida

<archivo.dat> Archivo con los datos de salida. El formato es:
tiempo:dato_entrada:dato_salida

Formato de ejecución

./obtiene.pl <archivo_entrada> <día> <mes>

Descripción

Este programa tiene como objetivo obtener la información proveniente de la base de datos del objeto en RRDTOOL para posteriormente vaciarla en un archivo <file>.dat, donde <file> es el nombre del archivo del objeto en RRDTOOL.

La información que se obtiene es de forma semanal, en períodos de 7,200 segundos (2 horas) y el primer dato corresponde a un lunes a las 1:00 hrs. En total se obtienen 84 muestras de datos de entrada y 84 muestras de datos de salida. El archivo resultante es la entrada del archivo de procesamiento de datos para cálculos estadísticos.

Este programa también puede concatenar los valores del mismo objeto de RRDTOOL de semanas posteriores obteniéndose una base de datos mucho mayor proporcionando más información a los programas de procesamiento de datos.

El programa valida que el dato de entrada corresponda a un día lunes, que el período de tiempo entre datos sea de 7,200 segundos y que la concatenación de datos actual no tenga una fecha posterior a la última concatenación del archivo destino.

8.4.2 "prom_desv.pl"

Nombre

prom_dst.pl

Parámetros de entrada

<archivo.dat> Archivo que contiene los datos de las semanas muestreadas. El archivo de entrada debe tener 84 muestras separadas por períodos de 7,200 segundos. El formato es:

tiempo:dato_entrada:dato_salida

Parámetros de salida

<archivo.cmp> Este archivo contiene el promedio y la desviación estándar de cada uno de los escalones de entrada y salida. Cada escalón tiene una separación de 7,200. En una semana deben existir 84 muestras. El formato del archivo de salida es:

promedio_in:desv_st_in:promedio_out:desv_st_out

Formato de Sipnosis

./prom_dst.pl <archivo_entrada>

Descripción

Este programa obtiene el promedio y la desviación estándar de las n muestras de cada escalón de 7,200 segundos de una semana. En total se procesan 168 x n muestras, 168 es igual a 2 x 84 donde 84 son el número de escalones de 7,200 segundos en una semana. N es igual al número de semanas muestreadas. El número de semana depende de cuantas veces se haya procesado el archivo de entrada con la función "obtiene.pl".

El programa procesa secuencialmente cada uno de los escalones, primero obtiene el promedio y lo guarda en los arreglos @promedio_in y @promedio_out. Posteriormente procesa secuencialmente cada uno de los escalones para obtener la varianza y la desviación estándar. La desviación estándar se guarda en los arreglos @desv_st_in y @desv_st_out.

Finalmente se escribe el contenido de los arreglos a un archivo de salida.

8.4.3 "compara_datos.pl"

Nombre

compara_datos.pl

Parámetros de entrada

<archivo.rrd> Archivo RRDTool donde se encuentra el último dato muestreado. Implícitamente se recibe el valor de entrada y de salida del objeto monitoreado.

<archivo.cmp> Archivo con las datos estadísticos de los días de la semana.

Parámetros de salida

Se genera una alarma si el límite superior o inferior fue sobrepasado. La alarma es de la forma:

fecha:% TH salida sobrepasado: <valor_sobrepasado> : th_up_out: <valor_límite_superior> th_dw_out: <valor_límite_inferior> -> <archivo.cmp>

Sipnosis

./compara <archivo.rrd> < archivo.cmp>

Descripción

El programa abre el archivo que contienen los datos de comparación y los guarda en un arreglo. El arreglo se descompone en los diferentes datos que contiene el archivo tanto para entrada como para salida. Se lee el último dato de la base de datos RRDTool, se obtiene el valor de entrada (valor_in), salida (valor_out) y la hora (tiempo) en que estos valores fueron obtenidos.

De acuerdo a la hora en que el valor se obtuvo, se calcula la posición en los arreglos en la que se encuentran los datos con que se deben comparar. Se obtiene los valores de límite (th_in_up, th_in_dwn, th_out_up, th_out_dwn) mediante la siguiente forma:

Límite_sup/inf = promedio (+)(-) (2 * desviación estándar)

Se comparan los valores obtenidos (valor_in, valor_out) con sus respectivos límites, si se sobrepasan los límites se emite una alarma al archivo de salida alarma.txt

8.5 Código Fuente

El código fuente de cada uno de los programas se encuentra en el Anexo 1 de esta investigación.

8.6 Funcionamiento del Sistema

El sistema requiera la ejecución de los tres programas en forma secuencial. El primer paso y más complejo por los cuidados que se tiene que tener para no introducir datos erróneas en la base de datos es obtener suficiente información en la base de datos de Cricket para que estos sirvan como datos fuente. Aunque el mínimo tiempo de recolección es una semana, el programa "obtiene.pl" no es capaz en esta versión de obtener datos hasta después de 3 semanas de su obtención, esto se debe a una complicación en el manejo de la función RRDs::fetch, la cual no pudo ser manipulada como se deseaba. Se recomienda correr semanalmente el programa obtiene.pl en al menos 4 semanas para que existan datos suficientes para calcular los límites de forma adecuada. En esta versión "obtiene.pl" no valida que la entrada para la obtención de datos sea compatible con los datos actuales que existen en la base de datos, es por ello que una entrada de datos con fecha de obtención anterior pueden quedar antes que datos de fechas de obtención más reciente. Para no corromper la base de datos es necesario verificar que el último de la base de datos sea más reciente que el que se va a ingresar. Por este hecho también es necesario que siempre se

agreguen datos del mismo día de la semana y de la misma hora, se recomienda usar el lunes a las 0:00 hrs.

Cuando se tiene suficientes datos (al menos 4 semanas) se ejecuta "prom_dev.pl" el cual generará los datos de entrada para calcular los límites por período de 2 horas.

Finalmente se debe poner en el cron del servidor la ejecución del programa "compara_datos.pl". Se recomienda que se ejecute 1 o 2 minutos después de la ejecución de Cricket. Posterior a esto no es necesario ejecutar ningún programa y el software se maneja solo. En caso de existir un problema se generará una entrada en el archivo de alarmas.txt. Se puede hacer otro programa que haga interfaz con alarmas.txt para una generación de alarmas meas dinámica, por ejemplo cuando exista una nueva alarma genere un proceso que envíe un correo electrónico. Cuando se obtengan más semanas de datos se recomienda correr el proceso de carga a la base de datos mediante "obtiene.pl" y posteriormente recalcular los datos para los límites mediante "prom_dev.pl".

A continuación se muestra una sumarización del proceso:

1. Configurar Cricket para capturar información
2. Después de tres semanas empezar a cargar datos semanalmente. Se recomienda que el primer dato de la semana sea un lunes a las 0:00 hrs.
3. Cargar suficientes semanas, al menos 4
4. Al tener suficientes semanas ejecutar "prom_dev.pl" con sus entradas correspondientes
5. Poner en el cron del servidor el programa "compara_datos.pl" con sus entradas correspondientes
6. Si se desean agregar meas datos semanales repetir del punto 2 al 4.

Conclusiones

El código generado para la validación del algoritmo y de la investigación resultó muy provechoso ya que sin el hubiera sido muy complicado automatizar la captura, procesamiento y comparación de los datos obtenidos. Después de terminado el sistema se encontraron puntos de mejora para una nueva versión, entre las mejoras que pueden hacerse son:

- Dividir la ejecución de actividades en más programas para que se adecuen al modelo funcional propuesto
- Mejorar la entrada de datos en los programas
- Generar un programa maestro que se encargue del procesamiento de todos los programas como uno solo

- El programa "obtiene.pl" debe obtener datos a la semana de su obtención como mínimo. Actualmente solamente lo hace después de tres semanas de su obtención.
- Validar la entrada de datos del programa 'obtiene.pl" con lo que actualmente se tiene en la base de datos para no corromperla.
- Generar salidas gráficas

Capítulo 9

Experimentos para la validación del algoritmo

Introducción

En este capítulo se analizarán a detalle los diferentes experimentos realizados para la validación del algoritmo y en su defecto para el perfeccionamiento de éste. Se analizan 4 experimentos generales los cuales fueron el resultado de otro número mayor de experimentos que no se mencionan aquí ya que fueron parte del perfeccionamiento del experimento o porque el resultado no era concluyente en ningún aspecto o no mostraba una nueva aportación como conclusión.

Los experimentos que se tratan en este capítulo fueron realizados en ambientes controlados y en ambientes en producción de tal forma de poder comprobar con una mayor certeza la validez del algoritmo diseñado y del sistema programado para su implementación.

9.1 Experimento 1

9.1.1 Objetivo

El objetivo del experimento 1 es comprobar la correcta implementación del proceso de comparación de los datos monitoreados con los límites calculados, verificar la cantidad de falsas alarmas que pueden obtenerse por no contar con una cantidad suficiente de datos y que el algoritmo con el cual se generan los datos de comparación arroje resultados que permitan una generación de alarmas precisa.

9.1.2 Proceso

Para el experimento 1 se midió durante 6 semanas la utilización en bits sobre segundo de un segmento de red en la interfaz del enrutador de la LAN. Se obtuvieron 3 archivos de salida con los datos de comparación.

vlan101_0.cmp: Contiene los datos de comparación de las primeras 4 semanas
vlan101_1.cmp: Contiene los datos de comparación de las primeras 5 semanas

vlan101_2.cmp: Contiene los datos de comparación de las 6 semanas completas.

Se configuró el proceso de comparación de tal manera que los datos obtenidos por el sistema de monitoreo en tiempo real se compararan con los archivos anteriores. El programa utilizado para obtener los datos del NMS y compararlos fue compara.pl. Si un valor monitoreado sobrepasa el límite superior (VCNS+) o inferior (VCNS-) se genera una entrada en el archivo de alarmas (alarmas.txt)

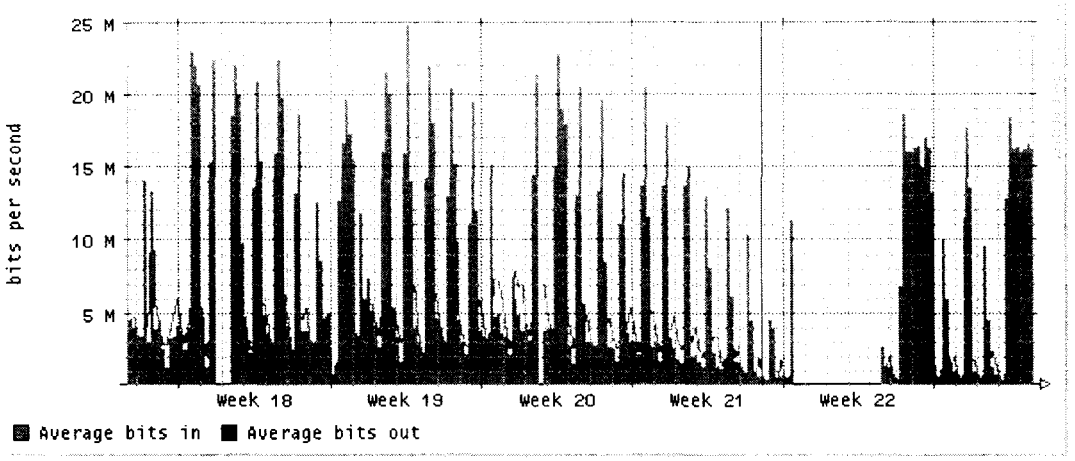
Desde el momento de activarse el proceso se comenzaron a generar alarmas de los tres archivos tanto de los límites superiores como los inferiores pero con una mayor razón en los datos correspondientes a la entrada de tráfico a la interfaz. Se dejó correr la prueba hasta el día siguiente. Las alarmas continuaron, inicialmente se consideró un error en la implementación del experimento o errores en los datos fuentes con los que se generaron los archivos de comparación (.cmp). No se encontró ningún error así que se procedió a analizar los datos que se estaban obteniendo. Las alarmas generadas fueron:

Archivos con límite de entrada sobrepasado:
vlan101_0.cmp
vlan101_1.cmp
vlan101_2.cmp

Archivos con límite de salida sobrepasado:
vlan101_0.cmp
vlan101_1.cmp
vlan101_2.cmp

Se encontró que existía mucho tráfico de entrada a la interfaz del enrutador, es decir que existía un sistema generando tráfico. Para verificar si este tráfico era normal se analizó el histórico de la interfaz encontrándose un patrón muy estable en las 6 semanas anteriores pero que se había modificado considerablemente en los últimos 2 días. Esto se puede visualizar en la gráfica de la figura 9.1.

Figura 9.1
Cortesía Telecomunicaciones y Redes del Tecnológico de Monterrey



Este comportamiento se analizó con la administración de la LAN y se encontró sistema que tenía un proceso en mal funcionamiento el cual generaba paquetes aleatorias a grandes razones de transferencia. A continuación se muestran algunos datos del problema en el cual un sistema con una dirección falsa (152.163.208.57) se encontraba enviando paquetes de TCP a otro sistema dentro de la red (10.17.21.159). La información proviene de la tabla de NetFlow arrojada por el enrutador de la LAN.

```
#sh ip cac flow | inc 10.17.21.159
SourceInt     SourceHost       DestInt       DestHost        Prot SP   DP    Packets
Vl100         152.163.208.57   Vl101         10.17.21.159    06 0050 1324    2
Vl100         152.163.208.57   Vl101         10.17.21.159    06 0050 1325    2
Vl100         152.163.208.57   Vl101         10.17.21.159    06 0050 1326    3
Vl100         152.163.208.57   Vl101         10.17.21.159    06 0050 1327    3
Vl100         152.163.208.57   Vl101         10.17.21.159    06 0050 1320    3
Vl100         152.163.208.57   Vl101         10.17.21.159    06 0050 1321    2
Vl100         152.163.208.57   Vl101         10.17.21.159    06 0050 1322    1
Vl100         152.163.208.57   Vl101         10.17.21.159    06 0050 1323    2
Vl100         152.163.208.57   Vl101         10.17.21.159    06 0050 132C    3
Vl100         152.163.208.57   Vl101         10.17.21.159    06 0050 132D    3
Vl100         152.163.208.57   Vl101         10.17.21.159    06 0050 132E    4
Vl100         152.163.208.57   Vl101         10.17.21.159    06 0050 132F    3
Vl100         152.163.208.57   Vl101         10.17.21.159    06 0050 1328    2
Vl100         152.163.208.57   Vl101         10.17.21.159    06 0050 1329    3
Vl100         152.163.208.57   Vl101         10.17.21.159    06 0050 132A    3
Vl100         152.163.208.57   Vl101         10.17.21.159    06 0050 132B    3
Vl100         152.163.208.57   Vl101         10.17.21.159    06 0050 1334    1
Vl100         152.163.208.57   Vl101         10.17.21.159    06 0050 1335    3
```

Una vez resuelto el problema el número de alarmas disminuyó considerablemente pero aun existían alarmas sin que éstas debieran de generarse. Aunque las nuevas alarmas provenían en general de los mismos 6 archivos, se observó que éstas provenían principalmente de los siguientes:

Archivos con límite de entrada sobrepasado:
vlan101_0.cmp 1680 alarmas
vlan101_1.cmp 1534 alarmas

Archivos con límite de salida sobrepasado:
vlan101_0.cmp 2000 alarmas
vlan101_1.cmp 1951 alarmas

Para "vlan101_2.cmp" se obtuvieron 1163 alarmas para el límite de entrada y 1424 para el límite de salida. Aunque estos datos no son 100% concluyentes, si se puede observar que entre más datos se tengan, es menor la cantidad de posibles falsas alarmas. Sin embargo la cantidad de falsas alarmas aún en el archivo de muestras de 6 semanas es muy alto, por esta razón se procedió a analizar los datos fuentes encontrándose que en todos los archivos de datos existían entradas de 0 bytes tanto de entrada como salida. Estos "huecos" de datos son debido a que la plataforma de monitoreo no siempre estaba capturando datos. Por este hecho, los resultados pueden no tener una alta exactitud como la encontrada en este experimento.

9.1.3 Resultados

De este experimento se puede concluir lo siguiente:

- Entre más semanas muestreadas se tengan (en general más datos) es menor la probabilidad de que existan falsas alarmas.
- No se ha encontrado un número mínimo de semanas que se deban muestrearse para tener bajas probabilidades de falsas alarmas.
- No se ha probado que funcione el algoritmo de comparación.
- No se ha probado que no funcione el algoritmo de comparación.
- El proceso de comparación y generación de alarmas funciona.
- Se deben obtener datos más precisos y sin "huecos" para verificar el funcionamiento del algoritmo que genera los datos de comparación.

9.2 Experimento 2

9.2.1 Objetivo

Comprobar si se disminuye el error generado por los intervalos de tiempo en los que no se obtuvieron valores de monitoreo mediante la substitución de estos valores

9.2.2 Proceso

A partir de los datos del experimento 1 se generó una nueva base de datos en la cual se sustituyeron todos los valores erróneamente cuantificados en cero por el monitoreo cuando éste no puede acceder a los equipos de red. El valor de cero se sustituyó con el último valor válido obtenido, de esta forma la siguiente tabla de ejemplo se sustituye de la siguiente manera:

Tabla 9.1

Datos Actuales con error

Tiempo	Entrada	Salida
1051020000:	1121539.8:	400180.7
1051027200:	2558062.0:	677210.5
1051034400:	0.0:	0.0
1051041600:	0.0:	0.0
1051048800:	0.0:	0.0

Datos modificados

Tiempo	Entrada	Salida
1051020000:	1121539.8:	400180.7
1051027200:	2558062.0:	677210.5
1051034400:	2558062.0:	677210.5
1051041600:	2558062.0:	677210.5
1051048800:	2558062.0:	677210.5

Una vez sustituidos todos los valores en cero, se genera nuevamente la tabla de promedios y desviaciones estándares de entrada y de salida. Este mismo procedimiento se siguió para las bases de datos de 4 y 6 semanas.

Al igual que en el experimento 1, estos valores de promedio y desviaciones estándar sirvieron como base comparativa de los valores obtenidos por el monitoreo. Además, para comparar el comportamiento de los datos modificados, contra los no modificados, el archivo de alarmas se generó en base a los archivos originales y los modificados. Finalmente se analizó el número de alarmas generado y sí estas realmente se generaban por un problema dentro de la red.

En un período de 2821 muestreos se obtuvo el siguiente número de alarmas:

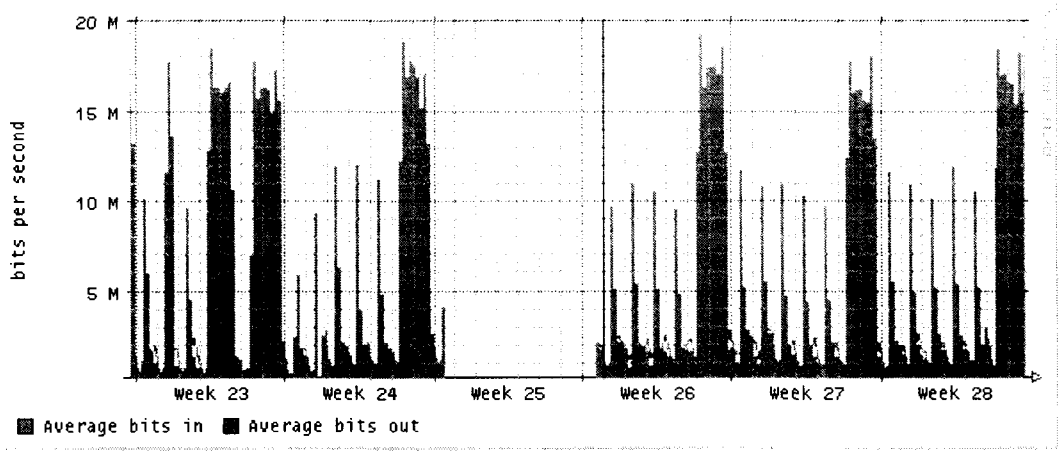
Tabla 9.2

	Orig. Ent.	Mod. Ent.	Orig.Sal.	Mod. Sal.
5 semanas	1585	1894	1931	2242
6 semanas	1046	1651	1177	2244

La gráfica de la figura 9.2 siguiente muestra los datos que generaron las alarmas, los datos corresponden a la semana 27 y 28.

Figura 9.2

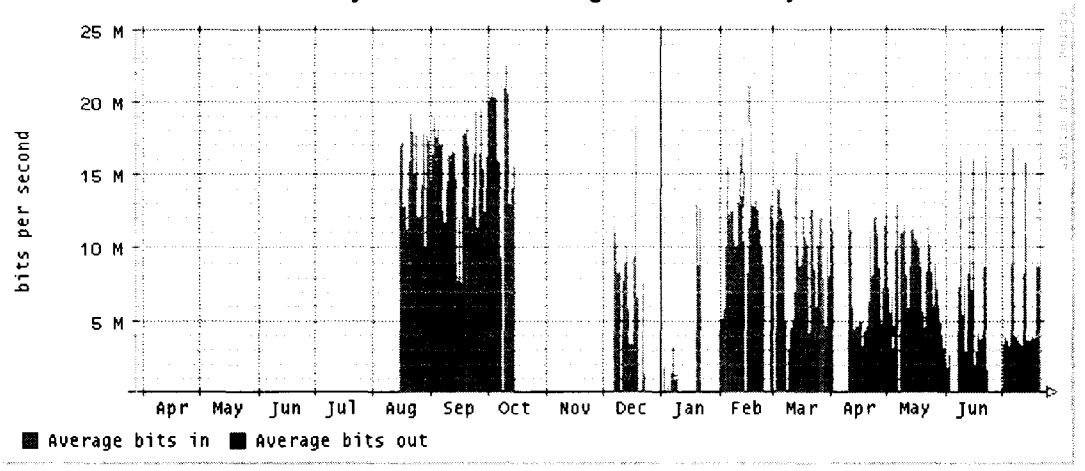
Cortesía Telecomunicaciones y Redes del Tecnológico de Monterrey



Los datos utilizados para generar los archivos generadores de promedios y desviaciones estándares se observan en la siguiente (Figura 9.3) gráfica a partir de los datos del mes de junio y mediados de mayo.

Figura 9.3

Cortesía Telecomunicaciones y Redes del Tecnológico de Monterrey



Como puede observarse a partir de la tabla, el modificar los archivos de datos con valores ficticios aparentemente no ayuda a mejorar la confiabilidad del sistema, sin embargo, si se analizan los datos fuentes del mes de mayo/junio contra los datos de julio, si existen variaciones en el comportamiento del enlace. Este comportamiento fue analizado y no se encontraron problemas, pero consultando con la administración de la red se obtuvo que en esos meses existe una carga menor de usuarios aunada a un cambio en la función del equipo monitoreado.

9.2.3 Resultados

El experimento número 2 no resultó ser totalmente concluyente en cuanto a la necesidad de sanar los huecos de datos del monitoreo pero si muestra una tendencia a que el sistema genera demasiadas alarmas. Sin embargo no se puede concluir que estas alarmas sean por datos corruptos para la generación de los archivos para comparación, por un ambiente muy dinámico de red que no tiene patrones bien definidos o porque realmente exista esa gran cantidad de problemas dentro de la red.

9.3 Experimento 3

9.3.1 Objetivo

Comprobar el correcto funcionamiento del sistema mediante el análisis de una base de datos en producción de un enlace también de producción.

9.3.2 Proceso

Se hará la captura de información de un enlace de Internet de producción con la finalidad de no perder datos como sucede cuando se utiliza un NMS experimental que no cuenta con los requerimientos de alta disponibilidad necesarios para obtener resultados para comprobar el correcto funcionamiento del sistema.

El enlace monitoreado fue un enlace E3 de 34 Mbps para uso de servicio de Internet. El período de toma de muestras para generar los archivos de comparación fue de mayo 18 a julio 13 (con la suspensión de la última semana de junio debido a problemas eléctricos que afectaron el monitoreo) dando un total de 7 semanas.

La tabla 9.3 muestra el número de alarmas generadas:

	Entrada	Salida
4 semanas	820	752
6 semanas	922	607

Tabla 9.3

Como puede observarse el número de alarmas siguió siendo alto. Existen varias razones para ésto. Los datos para generar los valores de límite pertenecen al período de mediados de mayo y principios de junio, el experimento se realizó a finales de julio. Como puede observarse en la gráfica de la figura 9.4, los valores de mayo son muy diferentes a los valores de junio, esto da como resultado una mezcla de datos de límite no confiable. Inclusive si las muestras hubiesen sido tomadas en períodos congruentes en cuanto a datos, el experimento tampoco hubiera generado resultados confiables en los límites inferiores dado que el comportamiento de julio tiene datos mucho menores al del resto del año.

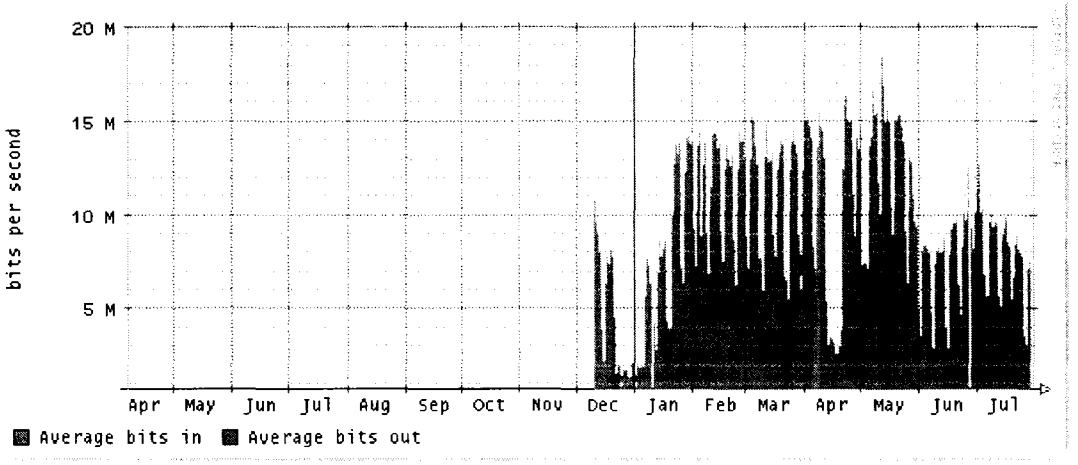
La tabla 9.4 muestra los mismo valores de la tabla 1 pero separando las alarmas generadas por la violación de un límite superior o inferior.

	Entrada Up	Salida Up	Entrada dwn	Salida dwn	Total Ent	Total Sal.
4 Muestras	504	389	316	363	820	752
6 Muestras	501	430	421	177	922	607

Tabla 9.4

Figura 9.4

Cortesía de Telecomunicaciones y Redes del Tecnológico de Monterrey



9.3.3 Conclusiones

Este experimento arrojó varias modificaciones a la implementación del sistema entre las cuales se tienen dos categorías; conceptuales y prácticas.

En la categoría de conceptuales, se engloban algunas sugerencias o mejoras en cuanto a la filosofía de funcionamiento del sistema. Entre estas modificaciones se tienen:

- Monitorear solo los límites superiores y no los inferiores. Normalmente los que arrojan información sobre posibles problemas son los límites superiores y no los inferiores.
- Preparar al sistema para dividir el año en regiones o bloques. Existen bloques de comportamiento normal y bloques de comportamiento fuera de lo normal, esto por la operación particular de la organización donde se efectúan los experimentos. Se propone solo usar como fuente de datos los bloques de comportamiento normal. Esto se analizó más a

profundidad en el capítulo 7 en la iteración número 2 del diseño del algoritmo.

En la categoría de prácticas se tiene:

- Para analizar más fácilmente la información y para preparar el sistema para sólo validar los límites superiores se modifica el método de registrar las alarmas añadiendo si el límite sobrepasado es superior o inferior.
- Se modifica el sistema para solo analizar los límites superiores
- Se modifica el método de registro de alarmas para facilitar el análisis de información.

9.4 Experimento 4

9.4.1 Objetivo

Observar el comportamiento del software en un ambiente práctico y con los valores adecuados eliminando datos que estuviesen fuera del comportamiento normal del tráfico.

9.4.2 Proceso

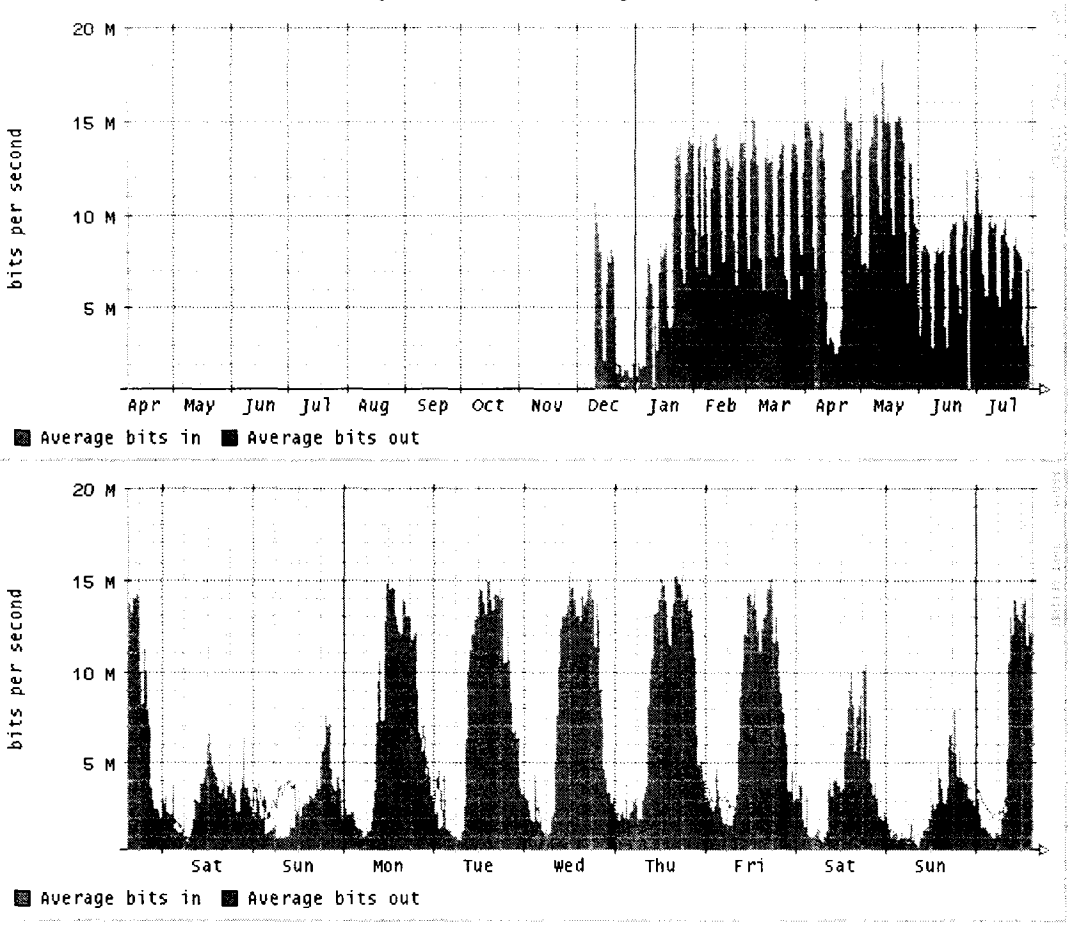
El proceso es similar al del experimento 3 con las siguientes modificaciones:

- Se tomaron como fuente los datos correspondientes al período de tiempo con un comportamiento similar al que se espera en el experimento.
- Se elimina la alarma por violación de los límites inferiores

La gráfica de la figura 9.5 muestra la utilización del enlace durante el período de muestreo.

Figura 9.5

Cortesía de Telecomunicaciones y Redes del Tecnológico de Monterrey



La figura 9.6 muestra los límites de entrada y la gráfica de la figura 9.7 muestra los límites de salida.

Figura 9.6

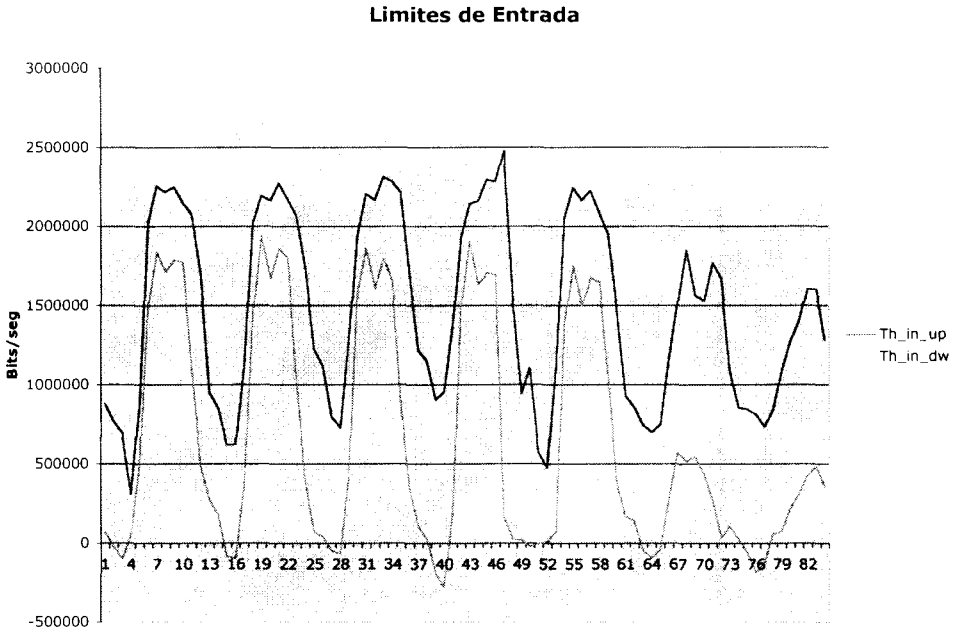
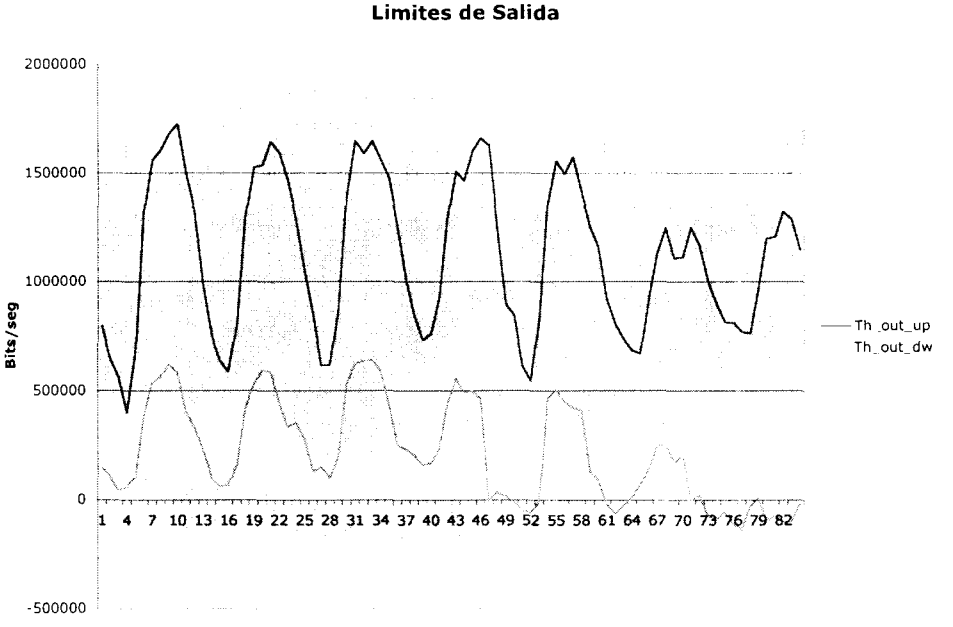
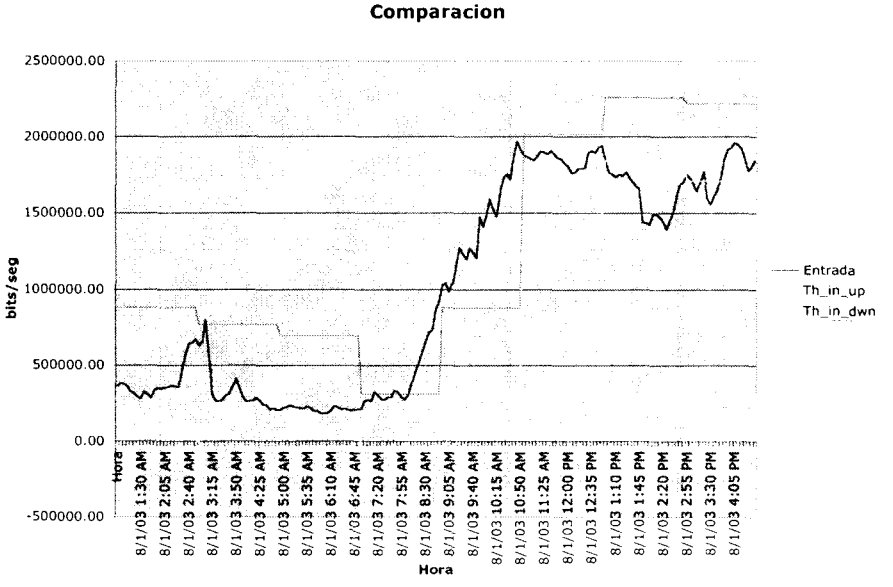


Figura 9.7



Se tomaron los datos de los límites superior e inferior y se graficaron contra los datos muestreados durante uno de los días del experimento obteniéndose la gráfica de la figura 9.8. Los puntos donde se interceptan los límites con el valor muestreado son donde se genera una alarma. A partir de este experimento solo se registran los valores que sobrepasan los límites superiores.

Figura 9.8



A partir de los datos generados por el archivo de alarmas se puede obtener el valor de la muestra que causo la alarma, el tipo de dato (entrada o salida), la hora y día de la alarma, y finalmente los valores de límite superior e inferior. Mediante el cálculo de los límites superiores e inferiores con una segunda herramienta se comprobó que las alarmas generadas fueron legítimas y el software esta funcionando bien. Sin embargo aún se recibe un alto número de alarmas sin encontrar una causa clara de falla.

9.4.3 Conclusiones

A partir del análisis de los valores de límite superior e inferior como los mostrados en la gráfica de la figura 2 se puede observar que los valores válidos posibles son pocos debido a que la diferencia entre el valor de límite superior y el inferior es pequeña, por tanto la probabilidad de que un valor caiga entre los límites es pequeña. Es por ello que en las pruebas anteriores existían muchos eventos de alarma. Con esto se comprueba que el eliminar las alarmas de los límite inferiores como se había propuesto no añade un efecto negativo.

Mediante el análisis visual de las gráfica de las figuras 1,2,3 y 4; y *mediante* la comparación matemática de los valores muestreados y sus correspondientes valores de límite se comprueba que el procedimiento usado es el adecuado

Conclusiones

Se ha comprobado el funcionamiento del algoritmo propuesto y la implementación de éste en software. Se han encontrado también mejoras en el mismo algoritmo para hacerlo más eficiente y más exacto y en el software para hacerlo más funcional y amigable; todas estas conclusiones serán parte de la conclusión de este estudio.

Conclusiones

Como mencionando en diferentes partes de este estudio el objetivo fue diseñar un algoritmo y un sistema capaces de identificar fallas en la red antes de que estas ocurran o con un período de tiempo corto al momento de su ocurrencia. Este algoritmo y este sistema debían ser sencillos de programar e implementar además de que debían resultar confiables.

El objetivo se cumplió con el diseño del algoritmo y del sistema aunque al final del estudio se han detectado algunas mejoras para hacerlos más exactos y más amigables. A continuación se mostrarán las conclusiones con las que se ha llegado con este estudio seguido al final con las recomendaciones que pueden hacerse para mejorar el sistema y el mismo algoritmo en posibles estudios futuros.

De acuerdo a las observaciones se ha determinado que para el tipo de red y variables analizadas se presenta un comportamiento periódico, es decir que ciertas variables medidas en la red presentan un valor similar en períodos de tiempo diferentes pero de la misma duración. Los períodos de tiempo que se analizaron fueron 24 horas, una semana, mensuales y anuales. Se determinó que los períodos de 24 horas son muy similares entre sí, sin embargo existen períodos que presentan variaciones muy grandes con respecto a otros, por ejemplo entre un día miércoles y domingo podría haber un comportamiento similar pero con valores con una alta dispersión entre sí. Si se analiza un período de una semana se puede encontrar que los valores entre períodos son prácticamente idénticos para algunas variables. De estas observaciones se ha determinado que el valor mínimo para un período de observación debe ser de una semana, de lo contrario los valores medidos reflejarán la advertencia de fallas cuando estas no existen.

Para los valores mensuales también se encontró un comportamiento interesante, se determinó que existen meses en los que los valores son muy parecidos entre sí, sin embargo existen meses donde los valores son mucho menores que en un mes promedio. Este comportamiento es debido a los períodos de trabajo de la red de la organización analizada. Para este caso se determinó que el año debe ser divido en períodos compatibles de tiempo. Los rangos de cada período pueden ser analizados en el capítulo 7 a detalle. Para períodos anuales solo pudo compararse los datos de dos años, encontrándose que fueron muy similares en comportamiento, así que los datos obtenidos en cierto año pueden ser utilizados para analizar variables de años posteriores.

El estudio mostró que existen variables que son más sensibles a cambiar su comportamiento cuando existe una falla, incluso cualquier tipo de falla o estado anormal de la red, como por ejemplo la cantidad de bits de salido o entrada por una interfaz. Hay variables que sólo cambian de comportamiento cuando existe un cierto tipo de anomalía en la red, tal es el caso del número de paquetes de ICMP o

el porcentaje de uso de procesador. También se encontró que hay variables que sólo cambian su comportamiento ante fallas críticas en la red o que incluso nunca lo cambian o no tienen un comportamiento periódico, tal es el caso de uso de memoria en equipos enrutadores.

En cuanto al sistema que se diseño para comprobar la eficiencia del algoritmo se encontró que pueden realizarse algunas mejoras. Entre las mejoras que pueden hacerse es simplificar la operación de este mediante algún proceso automático que genere todos los subprocesos que actualmente deben de ejecutarse. Actualmente la generación de varias proceso pueden propiciar algunos errores en la generación de los valores de comparación necesarios para el funcionamiento del sistema. Otra mejora del sistema tiene que ver con la protección del mismo ante la entrada de valores erróneos, el sistema actual no valida que los datos pertenezcan a períodos de tiempo incompatibles entre sí o que sean de variables diferentes.

Como último punto de mejora se encontró que al generar la documentación del estudio la información gráfica fue muy útil para hacer conclusiones, se determinó que si el sistema proporcionará información gráfica además de numérica podría ser de mayor utilidad para el análisis visual con propósitos de revisión rápida del comportamiento de la red. Un ejemplo de esto es comparar el valor de límite de la variable contra el valor actual de la variable en la red.

Este estudio ha demostrado que es posible analizar la red de una forma eficiente y proactiva. Aunque tanto el sistema como el algoritmo requieren mayor estudio para incrementar su eficiencia se considera que la continuación de este estudio o su uso como base para otros estudios pueden arrojar una forma de administrar redes grandes y complejas con poco personal altamente capacitado.

Bibliografía

Anónimo. Sin Fecha. Documentación de Cricket. [En Línea]. Disponible: http://cricket.sourceforge.net/ [Agosto 2002]

Anónimo. Sin Fecha. Documentación de Net SNMP. Disponible: http://net-snmp.sourceforge.net/ [Agosto de 2002]

Anónimo. Maximun Linux Security, A Hackers Guide to Protecting your Linux Server and Workstation. SAMS 2000

Anónimo. Sin Fecha "Willcam's Language Feature Simple Comparison". [En Línea]. Disponible: http://www.willcam.com/perl_vs_java.html [Noviembre 2003]

Abilene NOC. Abilene ICMP Stats. Actualización diaria. [En línea] Disponible: http://stryper.uits.iu.edu/icmp/ [Noviembre 2003]

Anixter. "The Hidden Cost Guide, Bussines Performance and Efficiency". 2002

Ashton Elaine, Hietaniemi Jarkko. CPAN Frequently Asked Questions. 2002 [En línea] Disponible: http://www.cpan.org/misc/cpan-faq.html#What_is_Perl [Noviembre 2003]

Baruffi Rosy, Milano Michela, Montanari Rebecca. "Planning for Security Managment". IEEE Intelligent Systems. Enero-Febrero 2001. Pag. 74-80

Bennett Todd Febrero 2000. Distributed Denial of Service Attacks [En línea] Disponible: http://www.opensourcefirewall.com/ddos_whitepaper_copy.html [Abril 2002]

Birman, Kenneth. Building Secure And Reliable Network Applications. Greenwich : Manning, 1996

Bogaerdt Alex van den. Octubre 1999. RRDTUTORIAL. [En línea]. Disponible: http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/ [Agosto 2002]

Braden Robert. "A Pseudo-Machine for Packet Monitoring and Statistics". Expert Systems Applications in Integrated Network Managment. Editado por Eric Ericson; Lisa Ericson; Daniel Minoli. Artech House 1989.

Caida. "cflowd: Traffic Flow Analysis Tool". Junio 2002. [En Línea]. Disponible: http://www.caida.org/tools/measurement/cflowd/index.xml [Septiembre 2003]

Caida. "FlowScan - Network Traffic Flow Visualization and Reporting Tool". [En línea]. Disponible: Octubre 2003.
http://www.caida.org/tools/utilities/flowscan/index.xml [Septiembre 2003]

Case Jeff. "SNMP Update". Sin Fecha. SNMP Research. [En Línea]. Disponible: http://www.nanog.org/mtg-0105/snmp.html [Septiembre 2003]

CERT/CC. "CERT Advisory CA-2003-04 MS-SQL Server Worm". Enero 25, 2003 [En Línea] Disponible: http://www.cert.org/advisories/CA-2003-04.html [Noviembre 2003]

CERT/CC. "CERT Advisory CA-2003-20 W32/Blaster worm". Agosto 11, 2003 [En Línea] Disponible: http://www.cert.org/advisories/CA-2003-20.html [Noviembre 2003]

Cisco Systems Documentation. "Defending Strategies to protect against TCP SYN Denial of Service Attacks". [En línea] Disponible: http://www.cisco.com/warp/public/707/4.html [Septiembre 2002]

Cisco Systems Documentation. "Toubleshooting Ethernet". Agosto 2003 [En línea] Disponible:
http://www.cisco.com/univercd/cc/td/doc/cisintwk/itg_v1/tr1904.htm [Septiembre 2002]

Cisco Systemas IOS Technologies. "Cisco IOS Netflow". 2002. [En línea] Disponible: http://www.cisco.com/warp/public/732/Tech/nmp/netflow/ [Septiembre 2003]

Chen Graham, Kong. Integrated telecommunications management solutions IEEE Press, 2000

Chen Jiann-Liang; Sun Hung-Fa; Tsai Che-Hsien; Wu Chun-Yuan. "A knowledge-based system for ATM network management". Humans, Information and Technology., 1994 IEEE International Conference on , Vol.: 1 , 1994. Pag.: 548 –552

Clark David D. M.I.T. Lab for Computer Science, 2001. Rethinking the design of the Internet: The end to end arguments vs. the brave new world [En línea] Disponible: http://www.ana.lcs.mit.edu/anaweb/PDF/Rethinking_2001.pdf [Marzo 2002]

Derek Partridge. ARTIFICIAL INTELLIGENCE and SOFTWARE ENGINEERING, Understanding the Promise of the Future. AMACOM (American Management Association) 1998

Deri, L.; Suin, S. "Effective traffic measurement using ntop". IEEE Communications Magazine , Vol.: 38 Issue: 5 , Mayo 2000, Pag. 138 –143

Duffy Marsan Carolyn "Is the Internet shrinking? Nonsense!" Network World. Enero 2002 Vol.19, Issue: 4, Página 76

Encliclopedia Británica en Línea. [En Línea] Disponible: http://www.search.eb.com/eb/article?eu=5785&tocid=0&query=algorithms [Abril 2002]

Ericson Eric; Ericson Lisa; Minoli Daniel. Expert Systems Applications in Integrated Network Managment. Artech House 1989

Ferguson P., Senie D. Request For Comments 2267 "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing". 1998

Fullmer Mark. "flow-tools". Sin fecha. [En línea] Disponible http://www.splintered.net/sw/flow-tools/docs/flow-tools.html

Gartner Group. Charles Carr, Kathryn Hale, Jennifer Song. North American ISP Market, 1999 [En línea] Disponible http://www.gartner.com/public/static/%20home/ggebiz.html [Agosto 1999]

Gartner Group. Betty Gifford, Eric Goodness. Febrero 2001. 2000 Network and Internet Connectivity Services Forecast and Market Share [En línea] Disponible: http://www.gartner.com/public/static/%20home/ggebiz.html [Marzo 2002]

Giles Roosevelt. "The Cisco CCIE Study Guide". McGraw-Hill 1998

Grant Ricketts. "How to do more in less time". Expert Systems Applications in Integrated Network Managment. Editado por Eric Ericson; Lisa Ericson; Daniel Minoli. Artech House 1989.

D. Harrington, R. Presuhn, B. Wijnen. STD0062, RFC3411 "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks". Diciembre 2002
Held, Gilbert. "Data communications testing and troubleshooting". Van Nostrand 1992, Segunda Edición

Hoske Mark T. "Distributed intelligence". Control Engineering.Octubre 1999. Vol. 46, Issue: 10, páginas 48-52

Huege Craig A. Febrero de 2000. The latest in denial of service attacks: "Smurfing" Description and information to minimize effects. [En línea]. Disponible: http://www.pentics.net/denial-of-service/white-papers/smurf.cgi [Septiembre 2002]

Internet Software Consortium. Sin Autor. Enero 2002. Internet Domain Survey, Jan 2002 [En línea]. Disponible: http://www.isc.org/ds/WWW-200201/index.html [Marzo 2002]

Internet Storm Center. Increase in ICMP scans. Agosto 2003. [En Línea]. Disponible http://isc.sans.org/diary.html?date=2003-08-18 [Noviembre 2003]

Kauffels Franz-Joachim,. Network Managment Problems, Standards and Strategies, 1992. Addison-Wesley 1992

Knuth Donald E.. The art of computer programming Vol.1 Fundamental Algorithms. Addison-Wesley 1973

Leiner Barry M., Cerf Vinton G., Clark David, Kahn Robert E., Kleinrock Leonard, Lynch Daniel C., Postel Jon, Roberts Larry G., Wolf Stephen. Agosto 2000. A Brief History of the Internet, version 3.31 [En línea].
Disponible: http://www.isoc.org/internet/history/brief.shtml [Marzo 2002]

Leon-Garcia, Alberto; Widjaja,Indra. Communication Network. McGraww Hill, 2000

Masonm Robert; Lind Douglas. Estadística para Administración y Economía. Alfaomega, 1995. Séptima Edición.

McCloghrie K., Rose M. Request for Comments 1213. Marzo1991

Microsoft Corporation. Microsoft Security Bulletin MS03-026, Buffer Overrun In RPC Interface Could Allow Code Execution (823980). Julio 16 2003. [En línea]. Disponible:
http://www.microsoft.com/technet/treeview/?url=/technet/security/bulletin/MS03-026.asp [Noviembre 2003]

Netcraft. Sin Autor. Febrero 2002. The Netcraft Web Server Survey [En Línea]. Disponible: http://www.netcraft.com/survey/ [Marzo 2002]

Northcutt, Sthepen; Novak Judy. Detección de Intrusos, Guía Avanzada. Prentince Hall, 2 edicion. 2001, Madrid España

Oram Andy. O'Reilly Febrero 2002. Scrambling the Equations: Potential Trends in Networking [En línea] Disponible:
http://www.oreillynet.com/pub/a/webservices/2002/02/12/oram.html [Marzo 2002]

Prior Molly "Outlook bullish for e-commerce sales growth" Dsn Retailing Today; Julio 2001; Vol. 40, Issue: 13, página 46

Qiming He; Shayman, M.A. "Using reinforcement learning for pro-active network fault management". Communication Technology Proceedings, 2000. WCC - ICCT 2000. International Conference on , Vol.: 1 , 2000 Pag.: 515 –521

Raman Lakshmi G. "Fundamentals of telecommunications network management". IEEE 1999

Research and Education Networking Information Sharing and Advanced Center http://www.ren-isac.net/

Rose M.T. The Simple Book: an Introducction to Internet Managment. Prentince Hall Series in innovative Technology. Segunda Edición 1996. Referencia de Marina Thottan y Chuanyi Ji. "Proactive Anomaly Detection Using Distributed Intelligent Agents". Septiembre 1998. IEEE Network. Vol.12 Pag. 21-27

Rozenblit Moshe. "Security for telecommunications network management" Institute of Electrical and Electronics Engineers, 2000

Schwartz, Randal. Learning Pearl. O'Reilly & Associates, Inc, 1er Edición. 1993

Stallings, William. Networking Standars, A guide to OSI, ISDN, LAN and MAN standards. Sexta Edición, 1996. Addison Wesley

Stallings, William. SNMP, SNMPv2 and RMON, Practical Network Managment. Addison Wesley 1997 Segunda Edición

Subramanian Mani. "Network management : principles and practice". Addison-Wesley, 2000

Symantec Security Response. "W32.Blaster.B.Worm". Agosto 13, 2003. [En Línea] Disponible: http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.b.worm.html [Noviembre 2003]

Sys Admin. "The Perl Journal". 2003. [En Línea] Disponible: http://www.sysadminmag.com/tpj/whatisperl.html [Noviembre 2003]

Thottan Marina y Ji Chuanyi "Proactive Anomaly Detection Using Distributed Intelligent Agents". Septiembre 1998. IEEE Network. Vol.12 Pag. 21-27

Travis Gregory; Balas Ed; Ripley David; Wallace Steven. "Analysis of the "SQL Slammer" worm and its effects on Indiana University and related institutions". Advanced Network Management Lab; Universidad de Indiana.

Wall, Larry; Schwartz, Randal. Programming Perl. O'Reilly & Associates, Inc, 1er Edición. 1992

Webopedia. [En línea] Disponible:
http://www.webopedia.com/ [Abril 2002]

Wolcott Kedron. "Exposing MIB data to a Web-based interface". Embedded Systems Programming. Abril 2000. Vol.3

Xiao Perry. Presentación de SNMP. Abril 2002

```
#!/usr/bin/perl

#obtiene.pl
#Version 2.1


# Modulo de RRD
use RRDs;

# Modulo para cambiar de tiempo "humano" a tiempo Unix
use Time::Local;


$file="file.rrd";
$database = $file;

# Obtiene el ultimo tiempo de update de la base de datos
  $lastupdate = RRDs::last $database;
  $ERROR = RRDs::error;
  print "Ultima actualizacion de base de datos: ";
  print scalar localtime($lastupdate);
  print " \n ";
  print "$ERROR \n";

print "Datos de inicio: \n";
  print "Dia: \n";
  $dia = <STDIN>;
  print "Mes: \n";
  $mes = <STDIN>;
  $min = 0;
  $hora = 1;
  $seg = 0;
  $y = 2003;
  # Debido a que la funcion toma enero como el mes 0, hay que restarle uno al me
s
  $mes = $mes-1;
  $time = timelocal($seg,$min,$hora,$dia,$mes,$y);
  $time_inicio = $time;
  $time_final = $time + (60 * 60 * 24 * 7) - (60 * 60 * 2);
  print "Tiempo de Inicio: $time_inicio \n";
  print "Fecha de Inicio: ";
  print scalar localtime($time_inicio);
  print "\n \n";
  print "Tiempo Final: $time_final \n";
  print "Fecha Final: ";
  print scalar localtime($time_final);
  print "\n \n";

# Inicia carga de datos a los arreglos de d_in y d_out
$inicio = $time_inicio;
$final = $time_final;
$cont = 1;

# fetch a la base de datos;
($start,$step,$names,$data) = RRDs::fetch $database, "AVERAGE", "--start", $inic
io, "--end", $final;
print "Start:       ", scalar localtime($start), " ($start)\n";
  print "Step size:   $step seconds\n";

 print "DS names:     ", join (", ", @$names)."\n";
 print "names:  @names[1] \n";
 print "Data points: ", $#$data + 1, "\n";
```

```
  print "Data:\n";

print "\n";
print "Numero de datos por linea? (datos = n+1, dn):\n";
$datos_linea = <STDIN>;

# Obtencion de los datos del arreglo de donde se recibieron;
  foreach my $line (@$data) {
#    print "  ", scalar localtime($start), " ($start) ";
    foreach my $val (@$line) {
  #    printf "%12.1f ", $val;
      @arreglo[$cont] = $val;
      @arr_time[$cont] = $start;
      $cont = ++$cont;
    }
    $cont2 = $cont - 1;
#    print "Contadr = $cont2 \n";
    $start += $step;
  }

#$pausa = <STDIN>;

# Imprimir datos
$cont = 0;
foreach $dato (@arreglo) {
  $cont = ++$cont;
#  printf "Contador: %f, Dato de Contador: %12.1f Tiempo: %f",$cont,@arreglo[$co
nt],@arr_time[$cont];
 # print "\n";

  }

# Se crea el nombre del archivo de salida
$file_out = "$file" . ".dat";
$num_datos = 84 * $datos_linea;

#print "numero de datos: $num_datos, datos por linea: $datos_linea \n";
print "archivo: $file_out \n";
#$pausa = <STDIN>;

open (SALIDA, ">>$file_out");

for ( $cont = 1 ; $cont < $num_datos ; $cont = $cont + $datos_linea )
  {
    print SALIDA "@arr_time[$cont]:";
    printf SALIDA "%12.1f:",@arreglo[$cont];
    printf SALIDA "%12.1f \n",@arreglo[$cont+1];
  }

close (SALIDA);
```

```
#!/usr/bin/perl

#prom_desv.pl
#version 1.1
#El archivo de entrada tiene el formato:
#tiempo:datos_entrada:datos_salida

$file = <STDIN>;

$cont = 1;
open(ARCHIVO, "$file");

@lines = <ARCHIVO>;

close (ARCHIVO);

foreach $line (@lines)
{
   (@time[$cont], @in[$cont], @out[$cont]) = split(/:/,$line);
   $cont ++;
}

$longitud = $cont - 1;
$muestras = $longitud / 84;
$num_datos = 84;

#Calculo del Promedio


 for ( $dato = 1 ; $dato <= $num_datos ; $dato++ )
  {
    for ( $num_subdato = 0 ; $num_subdato < $muestras ; $num_subdato++ )
    {
       $posicion = ($num_subdato * 84) + $dato;
       @promedio_in[$dato] = @promedio_in[$dato] + @in[$posicion];
       @promedio_out[$dato] = @promedio_out[$dato] + @out[$posicion];
    }

    @promedio_in[$dato] = @promedio_in[$dato] / $muestras;
    @promedio_out[$dato] = @promedio_out[$dato] / $muestras;
  }

#Calculo de la desviacion estandar
#Desviacion Estandar

#Variancia, se necesita para la DS


 for ( $dato = 1 ; $dato <= $num_datos ; $dato++ )
  {
    for ( $num_subdato = 0 ; $num_subdato < $muestras ; $num_subdato++ )
    {

       $posicion = ($num_subdato * 84) + $dato;
       @dif_in[$dato] = @in[$posicion] - @promedio_in[$dato];
       @cuad_dif_in[$dato] =  @dif_in[$dato] *  @dif_in[$dato];
       @suma_cuad_dinf_in[$dato] =  @suma_cuad_dinf_in[$dato] + @cuad_dif_in[$da
to];
       @dif_out[$dato] = @out[$posicion] - @promedio_out[$dato];
       @cuad_dif_out[$dato] =  @dif_out[$dato] *  @dif_out[$dato];
       @suma_cuad_dinf_out[$dato] =  @suma_cuad_dinf_out[$dato] + @cuad_dif_out[
```

```
$dato];
    }
    @var_in[$dato] = @suma_cuad_dinf_in[$dato] / $muestras;
    @var_out[$dato] = @suma_cuad_dinf_out[$dato] / $muestras;
    @desvst_in[$dato] = sqrt @var_in[$dato];
    @desvst_out[$dato] = sqrt @var_out[$dato];
  }

#Escribe archivo de salida como: $file.cmp (se elimina el .rrd.dat)
($fileroot) = split(/([-.])/,$file);
$fileout = $fileroot . ".cmp";
#El archivo de salida tiene el formato:
#promedio_entrada:desv_st_entrada:promedio_salida:desv_st_salida

open (ARCHIVO_SALIDA,">>$fileout");

for ( $dato = 1 ; $dato <= 84 ; $dato++ )
  {
    print ARCHIVO_SALIDA "@promedio_in[$dato]:@desvst_in[$dato]:@promedio_out[$d
ato]:@desvst_out[$dato] \n";
  }

close (ARCHIVO_SALIDA);

print "El archivo generado fue: $fileout \n";
```

```
#!/usr/bin/perl

#compara_dato.pl
#Version 3.0
#Esta version es identica a la version 2.1 sin checar limite inferior

# Modulo de RRD
use RRDs;

$comparadores_file = "/home/aservin/tesis/perl/testing/comparadores.txt";

open(ARCHIVO, "$comparadores_file");
 @files = <ARCHIVO>;
close (ARCHIVO);

foreach $file (@files)
{

 $file_db ="/home/aservin/tesis/perl/vlan101.rrd";

 # Obtiene el ultimo tiempo de update de la base de datos
  $lastupdate = RRDs::last $file_db;
  $ERROR = RRDs::error;
  #print "Ultima actualizacion de base de datos: ";
  #print scalar localtime($lastupdate);
  #print " \n ";
  #Checar como imprimir error solo si sucede
  #print "$ERROR \n";
  $final = $lastupdate;

 # fetch a la base de datos;
 ($start,$step,$names,$data) = RRDs::fetch $file_db, "AVERAGE", "--start", $last
update, "--end", $final;
 $cont = 1;

 # Obtencion de los datos del arreglo de donde se recibieron;
  foreach my $line (@$data) {
   # print "  ", scalar localtime($start), " ($start) ";
    foreach my $val (@$line) {
    #  printf "%12.1f ", $val;
      @arreglo[$cont] = $val;
      @arr_time[$cont] = $start;
      $cont = ++$cont;
    }
    #print "\n";
   }

  #print "Los datos buenos son: \n";
  #print "Valor de entrada: @arreglo[1], valor de salida: @arreglo[2], tiempo: $
start \n";


 $tiempo = $start;
 #$file = "vlan101.cmp";
 $file_al = "/home/aservin/tesis/perl/testing/alarmas.txt";
 $valor_in = @arreglo[1];
 $valor_out = @arreglo[2];
```

```
($sec,$min,$hora,$dia,$mes,$year,$wday,$yday,$isdst) = localtime($tiempo);

 #print scalar localtime($tiempo);
 #print " \n";
 #print "$wday \n";
 #print "hora $hora, minuto $min \n";

 $obset_diario = (60 * 60 * 24 );
 #print "Obset fijo $obset_diario \n";

 if ($wday == 0) {
  $wday = 6;
  }
  else {
   $wday = $wday - 1;
       }

  #print "nuevo dia $wday \n";
  $obset = $obset_diario * $wday;
  #print "Obset $obset \n";

  $tim_rel = $obset + ($hora * 3600) + ($min * 60) + $sec;
  #print "Tiempo relativo = $tim_rel \n";


  $step = int ($tim_rel / 7200);
  if ($obset == 0)
  { $step = 84;
  }

  #print "Paso $step \n";

 open(ARCHIVO, "$file");
  @lines = <ARCHIVO>;
 close (ARCHIVO);

 $cont = 1;
 foreach $line (@lines)
 {
   (@promedio_in[$cont], @desvst_in[$cont], @promedio_out[$cont], @desvst_out[$c
ont]) = split(/:/,$line);
   $cont ++;
 }

 #print "@promedio_in[$step], @desvst_in[$step], @promedio_out[$step], @desvst_o
ut[$step] \n";

 $th_in_up = @promedio_in[$step] + (2 * @desvst_in[$step]);
# $th_in_dwn =  @promedio_in[$step] - (2 * @desvst_in[$step]);
 $th_out_up = @promedio_out[$step] + (2 * @desvst_out[$step]);
# $th_out_dwn =  @promedio_out[$step] - (2 * @desvst_out[$step]);

 #print "Valores \n";
 #print "Valor de entrada: $valor_in, th-in-up: $th_in_up, th-in-down: $th_in_dw
n \n";
 #print "Valor de salida: $valor_out, th-out-up: $th_out_up, th-out-down: $th_ou
t_dwn \n";


if ($valor_in >= $th_in_up)
  { $alarma = "TH entrada sobrepasado";
```

```
      open (ALARMA, ">>$file_al");
        print ALARMA scalar localtime($tiempo);
        print ALARMA " ($step)";
        print ALARMA ":% $alarma: ";
        printf ALARMA "%9.1f ",$valor_in;
        printf ALARMA ": th_up_in:  %9.1f : th_dw_in: %9.1f :", $th_in_up,$th_in_
dwn;
        print ALARMA " -> $file";
      close (ALARMA);
    }
 if ($valor_out >= $th_out_up)
    { $alarma = "TH salida sobrepasado";
      open (ALARMA, ">>$file_al");
       print ALARMA scalar localtime($tiempo);
       print ALARMA " ($step)";
       print ALARMA ":% $alarma: ";
       printf ALARMA "%9.1f ",$valor_out;
       printf ALARMA ": th_up_out:  %9.1f : th_dw_out: %9.1f :", $th_out_up,$th_o
ut_dwn;
       print ALARMA " -> $file";
      close (ALARMA);
    }


}
```

```
#!/usr/bin/perl

#conv_tiempo.pl
#obtiene un tiempo estandar a partir de tiempo unix

print "Dame el tiempo en unix> ";
$tiempo = <STDIN>;
print "\n";

print scalar localtime($tiempo);
print "\n";
```

Anexo 2

Comprobación de Distribución Normal de frecuencias

Una de las suposiciones que se hace para el desarrollo del algoritmo es que la distribución de frecuencias de los datos que se toman para calcular los límites superiores e inferiores es normal. En el capítulo 7 se hace referencia al Teorema de Chebyshev como una de las bases del algoritmo ya que este utiliza el promedio más dos veces la desviación estándar de los datos para suponer que cualquier valor fuera de ese rango es una anomalía en el comportamiento de la variable analizada.

Para comprobar la distribución de frecuencias normal se graficó el histograma de frecuencias de algunos períodos de muestra. En la figura 1 se observa el histograma correspondiente al período de muestreo 1 (0-2 hrs de un lunes), la figura 2 muestra el período de muestreo 30 (12-14 hrs de un miércoles).

Figura 1

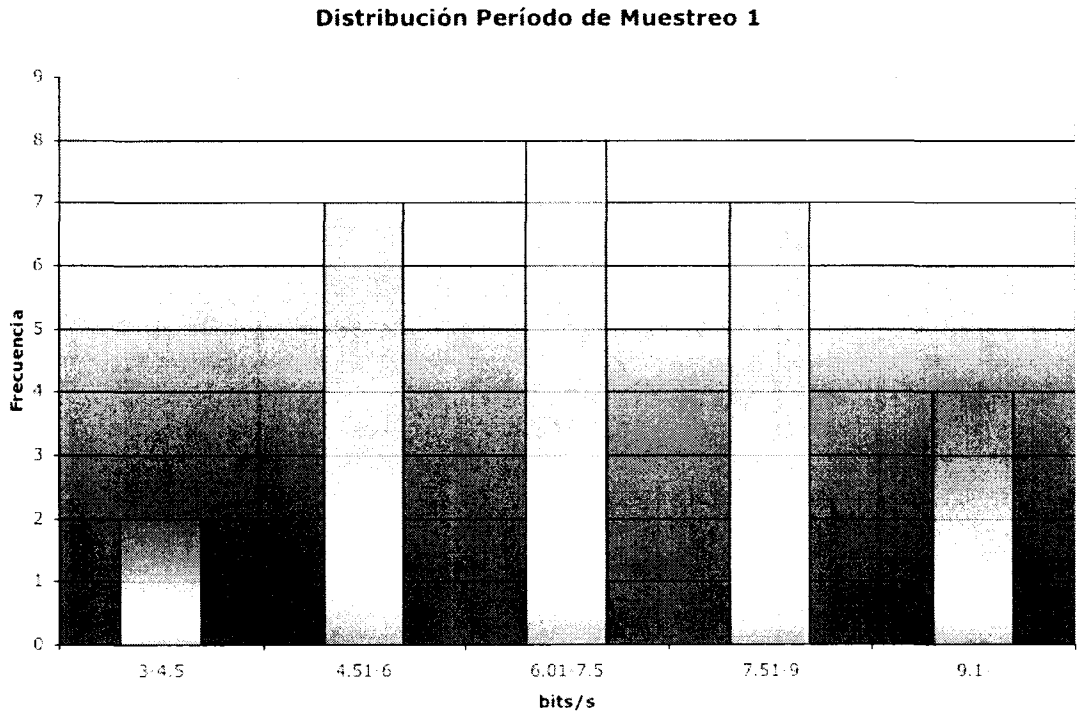
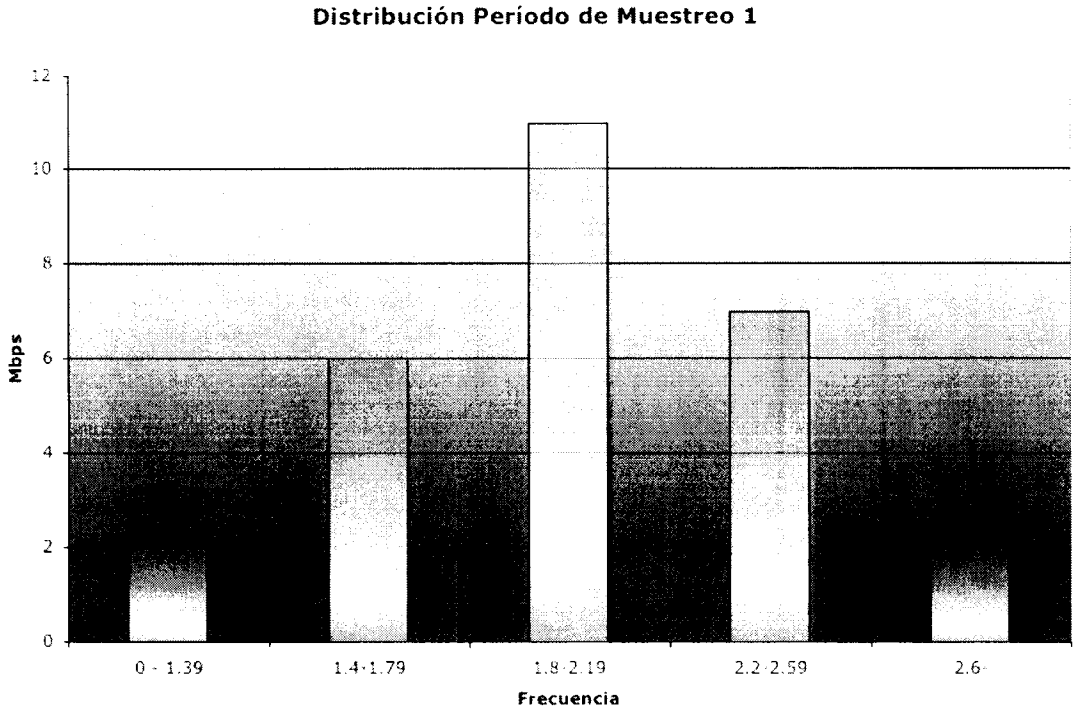


Figura 2



Por observación se asume que la distribución de frecuencias es normal. Para estos mismos datos se ha calculado el promedio y la desviación estándar. Para cada grupo se obtuvo el porcentaje de datos que cayó dentro de los rangos de $(\bar{x} \pm \sigma)$, $(\bar{x} \pm 2\sigma)$ y $(\bar{x} \pm 3\sigma)$. Los resultados se muestran en la tabla 1.

Tabla 1

Período	$(\bar{x} \pm \sigma)$	$(\bar{x} \pm 2\sigma)$	$(\bar{x} \pm 3\sigma)$
Período 1	81.48%	100%	100%
Período 30	66.67%	92.59%	100.00%

De acuerdo a la tabla anterior, si los datos se de la muestra del período de muestreo 1 se hubieran pasado por el algoritmo ninguna hubiera generado una alarma, mientras que para la muestra del período 30 solo un dato hubiera generado alarma. Este último comportamiento se espera que ocurra ya que de cumplirse una distribución completamente normal el mismo teorema de Chebyshev indica que un 5% de los datos podría generar una alarma sin que la

variable tuviese un comportamiento anormal. Esto dado a que el algoritmo usa $(\bar{x} \pm 2\sigma)$ como una referencia para saber si un dato monitoreado es o no admisible.