UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA

Monitor de Tráfico Ethernet
Netgraph

Memoria Presentada por:
Marcelo Maraboli Rosselott

Como requisito parcial
Para optar al título de Ingeniero Civil Electrónico
Mención en Computadores y Sistemas Digitales

Profesor Guía: Sr. Reinaldo Vallejos

Noviembre 1997

A mi Madre, Danilo, Andrea, Reinaldo y a la DCSC

por su constante e incondicional apoyo.

Indice

Figuras

Anexos

Resumen

El presente trabajo de título tiene por objetivo realizar la investigación, estudio, desarrollo e implementación de un monitor de tráfico Ethernet en tiempo real. Se entiende por monitor de tráfico a un programa que permite mostrar al usuario encargado de administrar una red de computadores, información relevante del tráfico que generan las aplicaciones que utilizan los usuarios de una red corporativa o una red de área local. En este escrito, se presenta todo el desarrollo del monitor de red implementado, una comparación objetiva con otros monitores, un caso ejemplo, proyecciones futuras, proyecciones comerciales y las conclusiones.

El programa desarrollado en este trabajo permite mostrar información cualitativa y cuantitativa del tráfico que fluye a través de la red de área local a la cual está conectado. Muestra gráficos que son generados dinámicamente a medida que se van capturando los datos de los paquetes de red. Estos gráficos son independientes y compatibles con todas las aplicaciones estándares de TCP/IP e IPX.

Este monitor enfoca de una manera nueva el monitoreo de una red Ethernet al mostrar cualitativamente y cuantitativamente el tráfico de un computador (IP en particular), de una subred IP, de un Port IP o de un Socket IPX en particular sin interferir con el desempeño normal de la red.

Por ende, Netgraph (nombre con que se bautizó este monitor) puede ser utilizado en la empresa, en la investigación académica y en la educación.

1.- Introducción

A medida que la masificación de computadores se ha ido desarrollando, las redes de computadores han adquirido una importancia a nivel mundial. Estas redes son muy beneficiosas para la comunicación entre las personas (Internet), la automatización de los Bancos (RedBanc), el cálculo distribuido de problemas matemáticos, la Tele Educación, la Tele Medicina y muchas otras aplicaciones.

Lo anterior resulta ser muy alentador y ventajoso, pero existen varios problemas no ocultos como son: la administración, la corrección oportuna ante fallas, la prevención de fallas, el monitoreo del tráfico, la distribución de la carga, etc.

Muchos de estos problemas son resueltos en forma heurística por los administradores de red, según las percepciones y experiencia históricas de su red en particular. Obviamente es necesario utilizar herramientas que permitan al administrador de red detectar problemas, cuantificarlos y solucionarlos. Para ello, existen muchos métodos que permiten resolver parcial o totalmente estos problemas.

Estos algoritmos son incorporados por los fabricantes de equipos de comunicaciones (necesarios en redes de computadores) para mejorar la eficiencia de la red, en los más variados casos. Los programas mencionados son utilizados por los administradores de red para realizar múltiples monitoreos, por ejemplo: el voltaje eléctrico (monitoreable a través

de una UPS; Uninterrupted Power System), la ventilación de una sala (temperatura), el ancho de banda de un enlace, el tráfico que genera una máquina en particular, etc.

El presente trabajo de titulación está dirigido a visualizar el tráfico de una red Ethernet, herramienta que permite al administrador de red detectar los problemas que se puedan presentar, esto le facilita las tareas de solucionar los problemas, así como proyectar y planificar las necesidades futuras de la red.

Las actividades que se pueden monitorear se resumen en los siguientes items:

- Funcionamiento de la red
- Servicios y/o aplicaciones más usadas por los usuarios
- Congestión
- Tráfico que genera una subred en particular.
- Tráfico que genera una máquina en particular.

Se eligió el estándar Ethernet porque en estos momentos es el protocolo que predomina en el mundo por su costo y facilidad de implementación y además porque alberga a las redes de tipo TCP/IP (usada para Internet o Intranet), Microsoft (Windows para trabajo en Grupo, Windows 95 y Windows NT), Novell (en todas sus versiones), AppelTalk y muchas otras.

El programa desarrollado en este trabajo permite ver, por cada aplicación o servicio (seleccionado por el usuario), un gráfico que muestra el tráfico en tiempo real (en bytes de data transferida entre un punto y otro) en páginas Web estándares. Los gráficos mencionados son generados en base al lenguaje Java, que también es estándar para la animación gráfica en ambiente de Hipertexto WWW.

Este Monitor o Graficador de Tráfico se bautizó con el nombre de **"Netgraph"** y permite mostrar en forma separada el tráfico de:

- Servicios IP (Ports) tales como WWW, SMTP, telnet, FTP, etc.
- Protocolos de Ethernet Versión II tales como IP, ARP, IPX/SPX
- Protocolo Ethernet IEEE 802.3
- Sockets IPX tales como SAP, RIP
- Direcciones IP (permite monitorear el tráfico de una IP en particular)
- Subredes IP (permite monitorear el tráfico de una Subred en particular)

La configuración de Netgraph permite al usuario elegir entre Ports, Protocolos y Sockets estándar (como los mencionados) y no estándar definidas por el usuario. Esto permite a los ingenieros utilizar Netgraph como una herramienta para el desarrollo de nuevos Servicios bajo el protocolo Ethernet.

En este escrito se comparan algunos monitores de red existentes con el desarrollado, según la clasificación del capítulo 3.1. Se incluye una descripción del Hardware usado junto con una descripción de cada programa y lenguaje de programación ocupado. Además se incorpora un manual de uso del Netgraph, detallando los cuidados requeridos para su utilización por parte de un administrador de red.

2.- El Estándar Ethernet

El estándar Ethernet es el protocolo de red más usado en el mundo y especifica el sistema de cableado y señalización (Capa 1 (Física) y Capa 2 (Data Link) del modelo OSI) de una red de computadores LAN.

El término "Ethernet" proviene de la antigua creencia de que no existe el vacío en el mundo, sino que él estaba lleno de una sustancia llamada éter ("ether" en inglés), que era la substancia que permitía conectar entre sí las cosas. Por este motivo, la red que une a computadores, por analogía, se bautizó como "Ethernet" [1].

Toda red Ethernet es una red del tipo Banda Base que ocupa código de línea Manchester (nivel físico) y por ello sólo un canal puede transmitir a la usando todo el ancho de banda del medio. Cuando dos o más estaciones transmiten al mismo tiempo ocurre una superposición de señales que interfieren todas las comunicaciones causando errores de transmisión hacia todas las estaciones [2].

Por esta razón, existe un protocolo (o algoritmo) que permite coordinar a todas las estaciones para que ocupen el canal en forma eficiente. Este protocolo se llama CSMA/CD [3], cuya sigla significa lo siguiente:

1) “Carrier Sense” (CS) significa "escuchar antes de transmitir", lo cual quiere decir que una estación debe detectar que el canal (ethernet) está en silencio antes de empezar a transmitir (no necesariamente al mismo tiempo).
2) “Multiple Access” (MA) significa que pueden haber múltiples estaciones conectadas a este canal, por lo que ellas "compiten" para ocuparlo.
3) “Collision Detection” (CD) significa que en caso de que ocurra una colisión, cada estación la detecta. Una colisión ocurre cuando 2 estaciones detectan silencio y ambas empiezan a transmitir.

Históricamente, las compañías Digital Equipment Corporation DEC, Intel y Xerox (formando el término DIX) desarrollaron la primera versión de Ethernet, que fue bautizada como ***Ethernet I*** y fue lanzada al mercado en 1980. En ese mismo año, la IEEE se reunió para estandarizar dicho protocolo. DIX lanzó en 1982 la versión de ***Ethernet II*** (que reemplazó a Ethernet I). En 1983, la compañía ***Novell***, con su producto Netware '86, lanzó su versión del protocolo en la que trabajaba la IEEE, que aún no estaba listo. Dos años más tarde, la IEEE finalmente estandarizó este protocolo, llamándolo ***IEEE 802.3***, que incluye el header LLC (Logical Link Control) del estándar IEEE 802.2, por lo que el protocolo utilizado por Novell no era estándar, vale decir, era de tipo propietario. Dado que la IEEE sacó su versión de Ethernet, y que existían protocolos que necesitaban trabajar con la versión Ethernet II, la IEEE lanzó la versión ***IEEE 802.3 SNAP*** que permite la compatibilidad entre IEEE 802.3 y Ethernet II [4].

En resumen, las versiones de Ethernet son:

Año	**Fabricante**	**Versión**
1980	DIX	Ethernet Versión I (versión experimental a 3 Mbps)
1982	DIX	Ethernet Versión II
1983	Novell	Novell (Raw 802.3)
1985	IEEE	IEEE 802.3
-	IEEE	IEEE 802.3 SNAP

Lo anterior muestra que no existe una sola versión de Ethernet y que además se requiere que distintas versiones coexistan en la misma red sin interferirse, lo que lo hace al sistema más complicado y versátil. La gran diferencia entre estos estándares es el formato del paquete.

En el Anexo Nº 1 se explica con detalle el formato de paquete de las cuatro versiones comerciales de Ethernet y además se incluye un gráfico comparativo [5][6].

Para distinguir los servicios que ocupan cada uno de estos formatos, existe un campo llamado Ether-Type que especifica si el paquete contiene IP, IPX, ARP, etc. En el Anexo Nº 2 se incluye el listado de Códigos Ether-Type [7].

Cabe destacar que el protocolo IPX puede utilizar como transporte a cualquiera de las 4 versiones de Ethernet (mayormente Novell Raw e IEEE 802.3), mientras que TCP/IP

puede ocupar como transporte a sólo en Versión II e IEEE 802.3 SNAP (en su mayoría sólo Versión II).

Netgraph puede monitorear los Servicios (Ether-Type) y los Ports (en caso de IP) de la versión Ethernet Versión II. En el caso de Ethernet versión IEEE 802.3, sólo es posible monitorear en forma aislada el tráfico IPX.

En el caso de las versiones Novell RAW e IEEE 802.3 SNAP, a Netgraph se le pueden hacer las modificaciones necesarias para poder monitorear este tráfico. En esta versión de Netgraph esto no se hizo, dado que ambos protocolos no son usados en un porcentaje significativo comparado con la Version II e IEEE 802.3. Cabe recordar que la versión Novell RAW ya no es usada por las versiones modernas de Novell (4.0 y superior) que usan por defecto IEEE 802.3 (se puede cambiar fácilmente a Version II). Por otro lado, la versión de Ethernet IEEE 802.3 SNAP, aunque puede transportar a casi todos los protocolos de capas superiores, es usado en la actualidad sólo por AppleTalk (estaciones Macintosh).

3.- Definición de Monitores de Tráfico

Un Monitor es un instrumento que entrega datos de algún tipo (numérico, visual, auditivo) de un proceso o fenómeno. De aquí que el término puede ser utilizado para monitores de computador o de vigilancia, hasta monitores de incendio y de intrusos (alarmas). El último caso, además de ser un monitor normal, entrega un dato específico cuando alguna variable ha sobrepasado algún umbral o situación prefijada. Un ejemplo aplicado a la Electrónica básica sería concebir a un osciloscopio o un Multímetro (tester) como un monitor de variables electrónicas que entregan datos numéricos o visuales.

Un Monitor de Tráfico en Redes de Computadores es un instrumento que entrega datos acerca de la red en la cual está conectada. Estos datos pueden ser entregados en diversas formas, dependiendo del fin con el cual el monitor fue diseñado.

A continuación se entrega una clasificación de los diversos tipos de monitores de red que existen tanto en el mercado como en Internet (shareware). Esto permite determinar las características más importantes de ellos y compararlos con el Monitor "Netgraph" desarrollado en la presente memoria. Cabe destacar que la clasificación no es estricta, por lo que un monitor puede cumplir múltiples características.

3.1.- Clasificación de Monitores

Los monitores de Red se pueden clasificar según los criterios de Objetivo, Reporte, Intrusividad, Operación y Protocolos que miden.

- **3.1.1.- Objetivo**

  - *Monitor de Estados (variables).*

    El objetivo es el de “avisar” situaciones de emergencia, como "caídas" de equipos o el sobrepaso de un umbral de alguna variable fijada por el administrador. Por ejemplo: La NO-RESPUESTA de un Router o la superación del umbral de 40% en el porcentaje de colisiones. Para ello se ocupa generalmente el estándar SNMP (Simple Network Management Protocol) que, como su nombre lo indica, está orientado a monitorear y configurar el equipamiento físico de una red (bridges, Routers, hubs y workstations).

    Dentro de los Monitores SNMP Comerciales se destacan HP OpenView, SUNNET Manager y Cisco Works. Cabe mencionar que el SNMP no sólo sirve para monitorear y/o configurar variables de los equipamientos de la red física sino que mostrar el tráfico histórico en forma gráfica acerca de lo se ha “traficado” por los equipos de la red, como routers, switches, etc.

- *Monitor de Tráfico.*

  El objetivo es el de registrar el tráfico de las redes, ya sea con fines estadísticos o de detección de congestión. Este tipo de monitor también puede tener "alarmas" y por ende, enviar una señal al administrador cuando una variable monitoreada haya excedido de un umbral prefijado (considerado como alarmante).

- **3.1.2.- Reporte**

  - *Histórico*

    Este tipo de monitor entrega informes o resúmenes de la actividad histórica de la red, ya sea por: hora, día, semana o mes. La actividad mencionada puede corresponder a tráfico, alarmas ocurridas, etc. El reporte generalmente consiste en archivos de tipo texto, dado que éste puede ser ingresado a otro programa para transformarlo a gráficos. Los últimos monitores de este tipo entregan sus reportes en formato HTML (WWW) y generan automáticamente los gráficos usando programas propios o del sistema. Un monitor histórico debe entregar la mayor cantidad de información posible, ya sea en forma de texto o de gráfico, ya que la información entregada puede ser ingresada a una base de datos.

- *Tiempo real*

  Un monitor de este tipo entrega datos de muy reciente ocurrencia, desde 1 segundo hasta 10 minutos, con el objeto de detectar y corregir los problemas cuanto antes. En algunos casos, como en las "alarmas" de umbrales, las variables son de tiempo corto y por lo tanto son inmediatamente avisadas. En otros casos, si la variable tiene un tiempo de "ejecución", como la transmisión de un paquete, se debe esperar la transmisión completa antes de poder tomar acciones. Por estas razones, un monitor de tiempo real no sólo abarca a los monitores "instantáneos", sino también a los que deben monitorear variables que requieren de un tiempo determinado de ejecución.

- **3.1.3.- Intrusividad**

  - *Intrusivo*

    Es aquel monitor que interviene en el proceso o fenómeno a monitorear, vale decir, actúa como agente activo. En este caso, el monitor afecta la medición, haciéndola no confiable o no representativa del proceso. Por ejemplo, un monitor de tráfico local se considera intrusivo si ocupa la red para hacer mediciones. Sin embargo, si un monitor ocupa la red para obtener datos, no significa que su medición sea intrusiva, ya que esto depende del proceso que desee medir. Por ejemplo: si el monitor debe medir el tráfico que es cursado por un router y

consulta a éste último por esa información (usando la red local), su medición no es intrusiva.

- *No Intrusivo*

Un monitor es no intrusivo si no interviene el proceso o fenómeno a monitorear, vale decir, si actúa como agente pasivo. Por ejemplo: un monitor de tráfico local no es intrusivo si no utiliza la red para medir su tráfico local. Si el monitor es un monitor del estado de un proceso en UNIX, el monitor en cuestión no debe interactuar con tal proceso, porque afectaría su ejecución normal.

- **3.1.4.- Operación**

Esta clasificación se refiere a la operación del monitor, en cuanto a dónde despliega sus datos y dónde es configurado por un administrador.

- *Local*

Si la operación del monitor es local, entonces significa que muestra los datos en el mismo lugar de donde los obtiene. En el caso de una alarma de una casa, la alarma es configurada por el dueño en la casa misma y la alarma suena en la misma casa en caso de la entrada de un intruso,. En el caso de un monitor de tráfico, esto significa que los datos se despliegan en el mismo computador que obtiene los datos.

- *Remota*

  La operación de este tipo de monitores se realiza en forma remota. En el caso de la alarma de casa, la operación se realiza desde una Empresa dedicada. En caso de la entrada de un intruso, la alarma avisa en las dependencias de la Empresa y no en la casa en cuestión. En el caso de un monitor de tráfico, esto significa que los datos son desplegados en otro computador distinto al que obtiene los datos. Lo anterior requiere ocupar la red para comunicar estos 2 computadores, por lo que dependiendo del tipo de medición, será también clasificado como intrusivo o no intrusivo.

- **3.1.5.- Protocolos**

  Esta clasificación permite destacar los tipos de protocolos de redes con que el monitor puede trabajar.

  La clasificación contempla mencionar la compatibilidad con los siguientes protocolos de red:

  Las 4 versiones de Ethernet; Versión II, Novell RAW, IEEE 802.3 y SNAP.

  TCP/IP; estándar para aplicaciones Internet (Intranet).

  IPX; estándar para redes Novell

  NETbeui (NETbios); estándar para redes Microsoft.

4.- Monitores Existentes

A continuación se analizarán tres tipos de monitores existentes. El primero de ellos corresponde a los Compatibles con el Protocolo SNMP (Comerciales en su mayoría) y los dos restantes son monitores de dominio público que se encuentran disponibles en Internet. De estos últimos, uno representa a aquellos que poseen interface texto y el otro a los que tienen interface gráfica.

De estos monitores se incluye una descripción, la clasificación (según el punto anterior) y se muestra además sus ventajas y desventajas comparativas. Posteriormente, se entrega la misma información del monitor desarrollado “Netgraph”, para luego entregar un cuadro comparativo entre ellos

4.1.- Compatibles con SNMP (Simple Network Management Protocol).

El SNMP es un protocolo ampliamente conocido y muy utilizado en ambientes donde la red es muy grande (30 hubs, 20 Routers, 20 Servidores, etc.) y además donde la estabilidad física de la Red es vital (bancos, edificios corporativos, grandes empresas).

El SNMP consta de 3 elementos: los “Agents”, el “Manager” y los MIB (Management Information Base). Los Agents son programas que son instalados en Routers, Hubs, Estaciones de Trabajo, etc., que permiten una interfaz entre el protocolo SNMP y la configuración local del equipo. El “Manager” es el programa central que consulta (o configura) las variables de cada nodo, interactuando con el “Agent” respectivo. El MIB es el "árbol" de especificaciones de la red, donde la raíz del árbol contiene las variables más globales de lo que sucede en la red y las "hojas" del árbol corresponden a la información detallada de cada nodo en la red (los Agents). El MIB es conceptual y es mostrado por el Manager y éste debe reunir la información de los Agents [8].

Lo anterior se puede visualizar en el siguiente esquema:

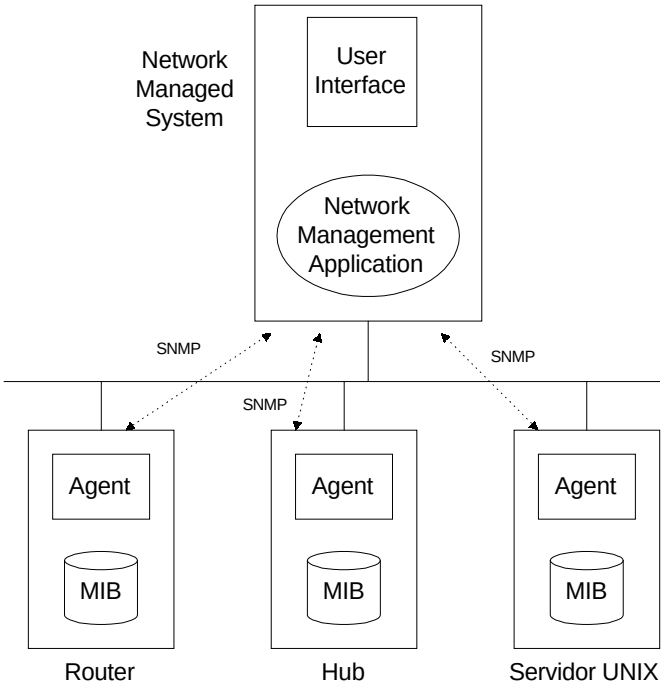


Figura Nº1 Diagrama de funcionamiento del protocolo SNMP

Existen cinco tipos de paquetes SNMP llamados PDU (Protocol Data Unit): dos para leer datos de los nodos, dos para configurar datos de los nodos y uno para los TRAP (umbrales prefijados que activan una señal de alarma), que usa el Agent para enviar eventos (programados por el administrador) como la "inicialización de un nodo" o la falla de una estación, etc.

Así, si el administrador desea saber el valor de una variable de un Router, usa SNMP para enviar un PDU a ese nodo. El agente (Agent) del Router busca en su MIB el valor actual de la variable y envía el valor en otro PDU al Manager

Lo que se puede hacer en cada nodo depende del agente respectivo, vale decir, se puede programar enteramente un Router vía SNMP, consultar sus estadísticas de tráfico, se puede administrar un nodo UNIX (crear cuentas, verificar espacio en disco,etc.) vía SNMP sólo "leyendo" y "escribiendo" variables. Las ventajas del SNMP es su fácil uso y el hecho de que permite manejar los nodos físicos de la red en forma centralizada (remota).

Los Managers que se destacan son en su mayoría versiones comerciales, como HP Open View, SUNNET Manager, CISCO Works, IBM NetView. También existen otros de dominio público, como XSNMP, AARNet Traffic Monitoring para UNIX; NetGuardian y SNMPMan para Windows.

En el Anexo Nº 3 se incluye una breve descripción de HP Openview y NetGuardian.

A partir de la clasificación del punto 3.1, podemos clasificar a los monitores SNMP según:

- ***Objetivo*:** Principalmente Monitor de Estados (o variables), ya que no están orientados a ser monitores de tráfico.
- ***Reporte*:** Histórico en su mayoría. En el caso de las alarmas o TRAP son de "Tiempo Real".
- ***Intrusividad*:** Se consideran Intrusivos, ya que utilizan bastante la red para obtener las variables. Se ha demostrado que la eficiencia (throughput) de la red baja cuando se utiliza SNMP.
- ***Operación*:** Claramente Remota.
- ***Protocolos:*** El SNMP es principalmente una aplicación TCP/IP, por lo que la gran mayoría de los Manager y Agents de la red son TCP/IP. En general, cubre todos los tipos de redes Microsoft, TCP/IP y Novell. En el caso de las versiones Ethernet, tanto el Manager como el Agent son construídos de acuerdo al dispositivo a manejar.

Las ventajas y desventajas que se presentan a continuación son establecidas con respecto a los otros monitores explicados.

Ventajas:

- Utilizan un protocolo estándar que permite integrar varios Agents de distintas marcas al Manager, compatibilizando el manejo centralizado de las redes.
- La plataforma de trabajo para el Administrador de la Red puede ser una estación UNIX o un simple PC con Windows 95.
- Constituye una plataforma de trabajo muy simple para la administración, donde entrega parámetros globales y específicos según la necesidad del administrador.

Desventajas:

- Es claramente un monitor de estados, por lo que sólo puede medir variables globales de tráfico según la información entregada por el Agent respectivo, el cual no está destinado a medir en tiempo real ninguna clase de tráfico.
- La mayor cantidad de reportes que entrega son históricos, por lo que no sirve para monitorear transiciones y/o fluctuaciones del tráfico mientras ocurren. Sólo es posible enterarse después.
- Dado que los Agents y los MIBS vienen del fabricante y el Manager sólo agrupa y despliega la información entregada, el SNMP no es una plataforma para poder medir variables de Protocolos y estándares en desarrollo o experimentación. El investigador debe programar su propio Agent y su propio MIB.

4.2.- NOCOL (Network Operations Center OnLine)

Este Software es un paquete de herramientas para redes TCP/IP que se ejecuta en ambiente UNIX y que entrega la información en formato texto. Está compuesto por varios módulos que entregan su información al "Manager" llamado "Netconsole" [9]. Los módulos son:

- IP ICMP (IP PING o (multiping))
- OSI ping
- RPC post mapper
- Ethernet load (BW & packets/s) "Etherload".
- TCP port reachability "tpmon".
- UNIX host (discos, memoria, swap, carga, uts, collisiones)
- SNMP (RMON, Cisco Router, terminal server)
- TCP data Throughput
- DNS (servidor de nombres)
- SNMP traps (alarmas umbrales)
- Otros, como Novell Service Monitor, Appletalk Route Monitor, etc.

Una de las ventajas comparativas de este programa es que incluye una librería de funciones (para “perl”) llamada ‘perlnocol’ que permite desarrollar nuevos módulos que se agregan fácilmente al Netconsole.

La mayoría de estos módulos son rutinas que contactan al host y prueban si el servicio que proveen está funcionando como es debido, de lo contrario, arroja mensajes CRITICOS.

Los monitores más relevantes de NOCOL para este estudio son***: Etherload*** y ***Tpmon***.

4.2.1.- Etherload

Este monitor entrega a Netconsole la utilización de Ancho de Banda (BW) y Paquetes por Segundo (packets/s) de una red Ethernet. Si cualquiera de estos parámetros excede un umbral (fijado por el usuario), envía un mensaje de aviso (según lo programado por el usuario) a Netconsole.

La siguiente imagen muestra a Netconsole entregando la información recibida por Etherload.

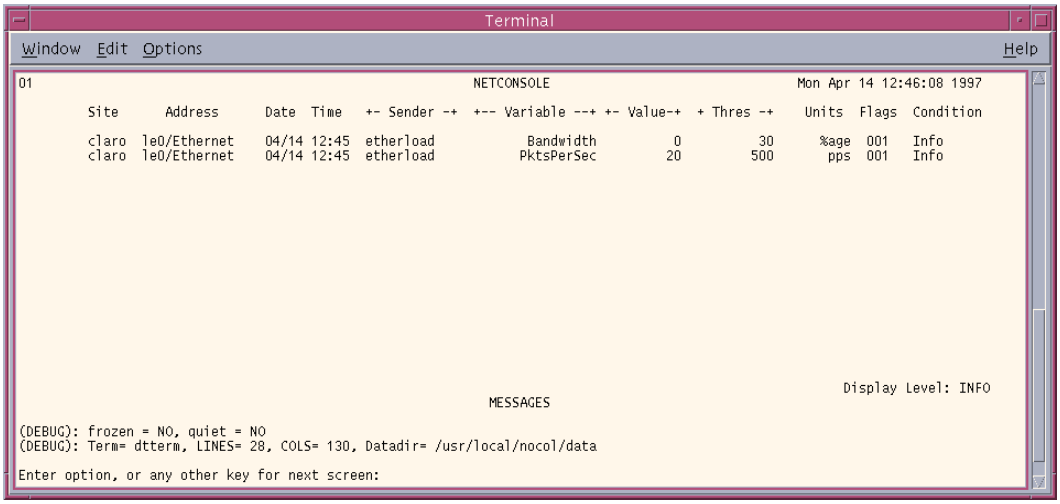


Figura Nº2 Netconsole recibe información de Etherload.

La configuración de Etherload se realiza a través de un archivo llamado etherload.conf,que en el caso mostrado contiene lo siguiente:

```
## $Id: etherload-confg,v 1.2 1994/06/13 20:09:11 vikas Exp $
##
# This is the NOCOL config file for measuring ethernet load.
#
# Bandwidth is represented in percentage of total.
# Packets per sec is a variable depending on the pkt size:
#
#       Pkt size        Max PPS theoretical on ethernet
#       ---------------------------------------------
#         64            14880
#         128           8445
#         256           4528
#         512           1024
#         1518          812
##
#
#
SLEEPSECS       120

#
SCANSECS        15

##
# Device names are specfied as:
#       le0, le1, ie0, fddi0, bf0       On SunOS & Solaris2.x
#       pf0, pf1, pf2                   On Ultrix & OSF
#       et0, ec0, fxp0, enp0, ipg0      On SGI
#       /dev/bpf0, /dev/bpf1            On BSDI
##
device le0
bw      30  50  70
pps     500 1000 1500
# LOS 3 PARAMETROS SON UMBRALES PARA: WARNING, ERROR, CRITICAL RESPECTIVAMENTE,
# TANTO PARA LA "CARGA" O BW COMO PARA LOS PPS
device le1
bw      25  55 75
pps     600 1100 1550
####
```

4.2.2.- Tpmon

Es un módulo que mide la transferencia efectiva (throughput) en Kbps desde el nodo monitor (generalmente el que tenga el Netconsole ejecutándose) a una lista de nodos predeterminada. El programa básicamente se conecta a estos nodos y envía una serie de paquetes de gran tamaño por un período corto de tiempo, con lo que calcula el "throughput" efectivo entre los dos nodos.

El módulo además de entregar esta información en forma **periódica**, puede programar una alarma que avise a Netconsole con un mensaje de AVISO ("INFO", "WARNING", "ERROR" o "CRITICAL") si el "throughput" baja de un cierto umbral.

En la siguiente figura se muestra a Netconsole con Etherload y Tpmon activos con cada una de sus señales de "INFO", "WARNING", "ERROR".

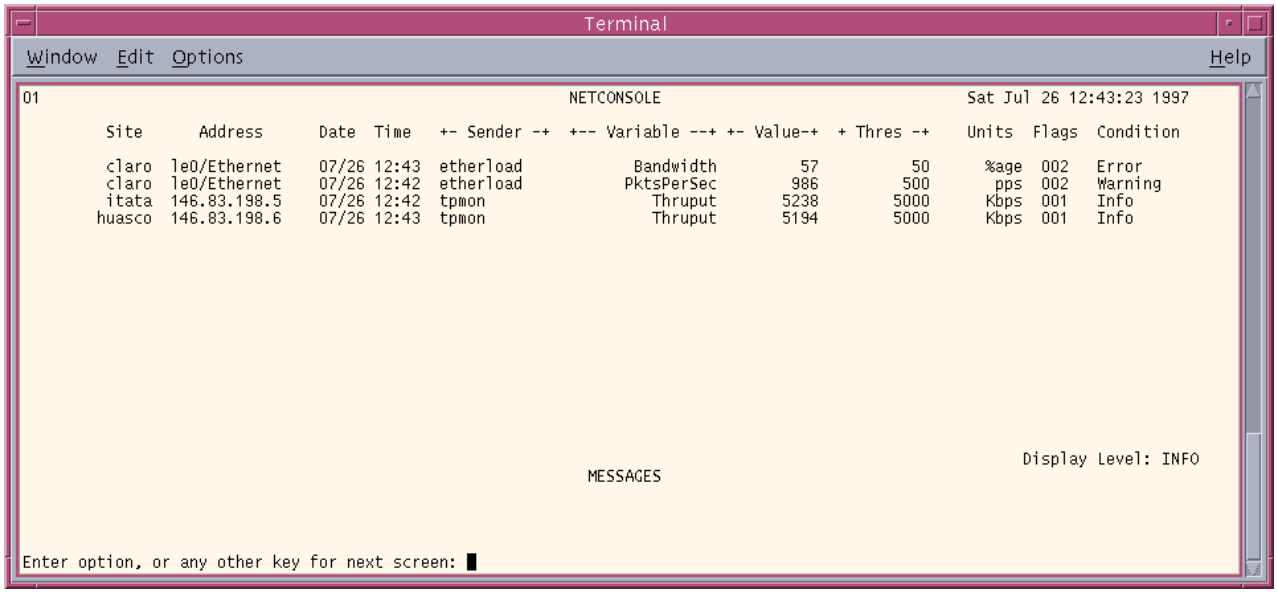


Figura Nº3 Etherload con condición de WARNING

Al bajar del nivel 5000 Kbps (5 Mbps) de uno de los nodos, se muestra la condición de “WARNING” como en la figura siguiente:

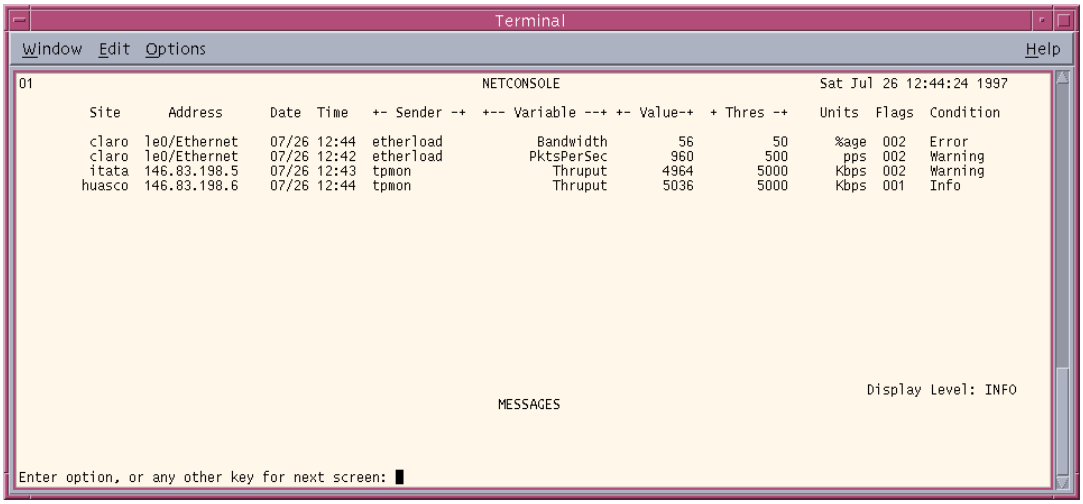


Figura Nº4 Etherload y Tpmon con condición de WARNING

El archivo de configuración de Tpmon es muy simple:

```
# $Id: tpmon-confg,v 1.2 1992/06/12 21:13:38 aggarwal Exp $
# Config file for 'tpmon' - throughput monitor
#
# The program connects to the discard port (port 9) of the listed machines
# and transfers data for 30 seconds to measure the thruput. It then sleeps
# for POLLINTERVAL seconds.
# Sleep for 2 hours between polls = 7200
POLLINTERVAL 5

# If the thruput (in bits per second) falls below the threshold, it raises
# the severity to WARNING level. you can use the 'tptest' program to see
# the typical thruput to a site.
#
# <hostname> <ip address> <threshold in Kbps> [TEST]
#
itata        146.83.198.5 5000
huasco       146.83.198.6 5000
```

A continuación se presentan las características de **Etherload** y **Tpmon** según la clasificación 3.1:

- ***Objetivo*:** de Estados; presentando variables globales como Ancho de Banda (BW) y la Transferencia Efectiva (Throughput).
- ***Reporte*:** Tiempo real
- ***Intrusividad*:** ambos son totalmente Intrusivos, ya que transfieren por la red que desean medir las variables de Transferencia Efectiva y Ancho de Banda.
- ***Operación*:** Local (o Remota utilizando el despliegue remoto del UNIX)
- ***Protocolos:*** Ethernet versión II e IEEE 802.3. En capas superiores es principalmente TCP/IP. Algunos módulos son IPX (Novell).

Ventajas

- El NOCOL realiza monitoreo de Servicios, Estados (SNMP) y variables globales de tráfico.
- Entrega valores de Ancho de Banda (BW), Transferencia Efectiva y Paquetes por Segundo (Packets/s).
- Permite desarrollar e integrar nuevos módulos.

Desventajas

- Es monitor de tráfico simple, no se sabe qué o quienes causan el tráfico.
- Totalmente orientado a texto (Interface de usuario no amistosa).
- La configuración de este monitor se basa en la edición de archivos texto y ejecutar nuevamente el programa.

4.3.- Netman (Network Monitoring and visualisation tools)

Netman [10] es un conjunto de herramientas para monitoreo y visualización de redes. Consta de seis programas para Sistemas Operativos UNIX, que son fundamentalmente gráficos. Ellos son: **Etherman, InterMan, Packetman, Loadman, Geotraceman y Analyser**. Sólo los primeros tres están disponibles en plataformas SUN Solaris, dado que son los más importantes. Aunque los últimos tres no son de importancia para este análisis, se incluye una breve descripción:

Loadman: Muestra gráficamente la "carga" o utilización de la red usando un algoritmo desarrollado por la DEC.

Geotraceman: Muestra gráficamente la "ruta" entre el nodo donde se ejecuta y otro nodo de la Internet. Es la versión gráfica del popular "traceroute".

Analyser: es una herramienta que recomienda la forma de "particionar" una red LAN, basándose en criterios como: Número de computadores clientes, Número de servidores, grupos de usuarios de alta demanda, etc.

De mayor importancia resulta el estudio de los primeros tres monitores dado que entregan información poco vista en otros monitores.

4.3.1.- Etherman

Etherman es un programa que despliega visualmente las comunicaciones **Ethernet** punto a punto en tiempo real, como se ve en la figura adjunta. Dado que trabaja en la capa 2 del modelo TCP/IP (Ethernet), este programa permite ver los protocolos que trabajan sobre Ethernet y por ende las máquinas son identificadas por su Ethernet Address.

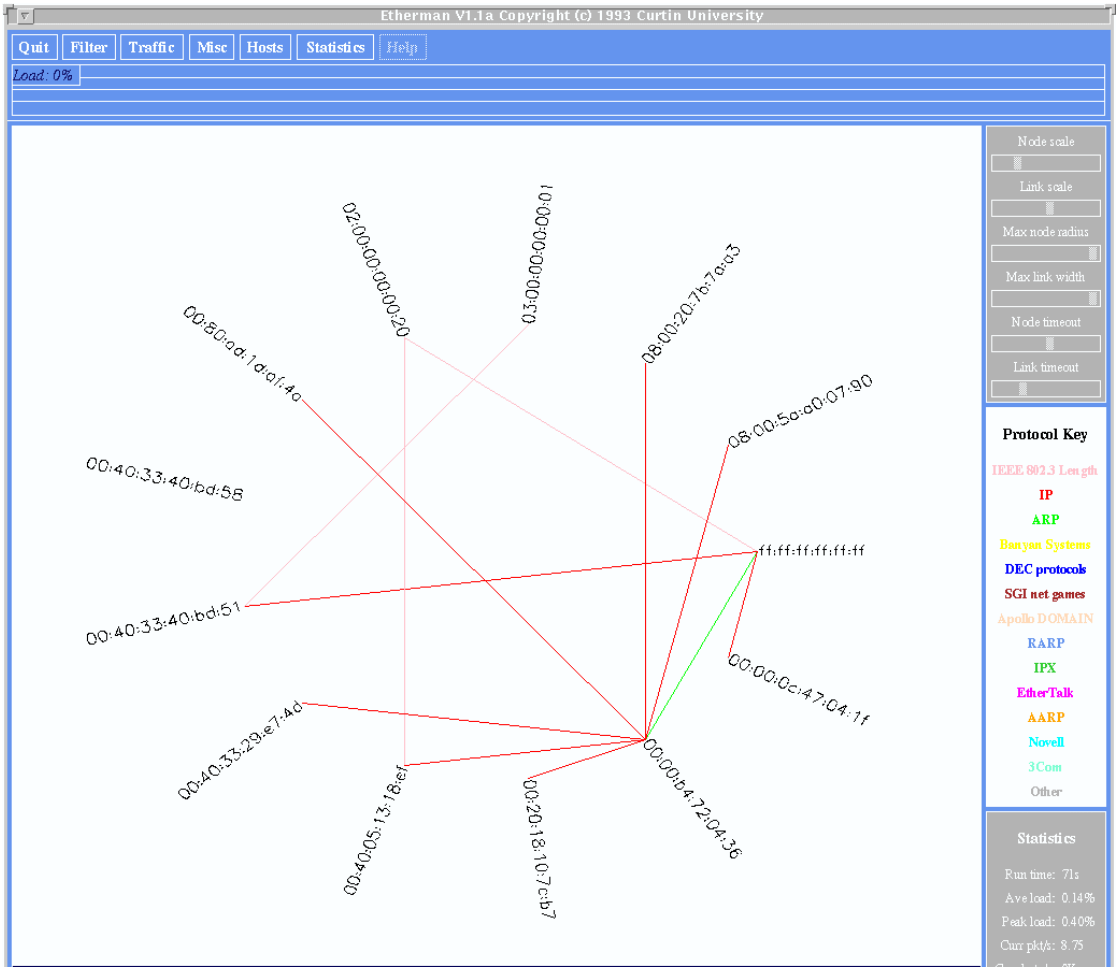


Figura Nº5 Etherman

Su clasificación corresponde a la siguiente:

- ***Objetivo*:** Monitor de Tráfico (o Conexiones), dado que al aumentar el tráfico de una conexión, el grosor de la línea aumenta (cualitativamente).
- ***Reporte*:** Tiempo real
- ***Intrusividad*:** No Intrusivo
- ***Operación*:** Local (o Remota utilizando el despliegue remoto del UNIX).
- ***Protocolos:*** Ethernet Version II e IEEE 802.3. En capas superiores contiene todos los protocolos que ocupan estas dos versiones de Ethernet.

Ventajas

- Muestra las conexiones Ethernet punto a punto
- Se puede capturar/imprimir la pantalla

Desventajas

- El tráfico se ve en forma cualitativa y no cuantitativa.
- El tráfico mostrado sólo corresponde al tráfico de la subred en el cual está conectado este monitor.
- No realiza registro de información o reportes históricos en forma automática.
- En una conexión punto a punto, se muestra el protocolo con más tráfico, aunque exista otro protocolo entre esos mismos 2 puntos con menos tráfico.
- Operación remota sólo en otros equipos compatibles con X11 (UNIX).

4.3.2.- Interman

Programa muy similar a Etherman, pero en vez de mostrar conexiones Ethernet punto a punto, muestra las conexiones TCP/IP de Redes a Redes, indicando qué máquina de una red se comunica con cuál de otra red.

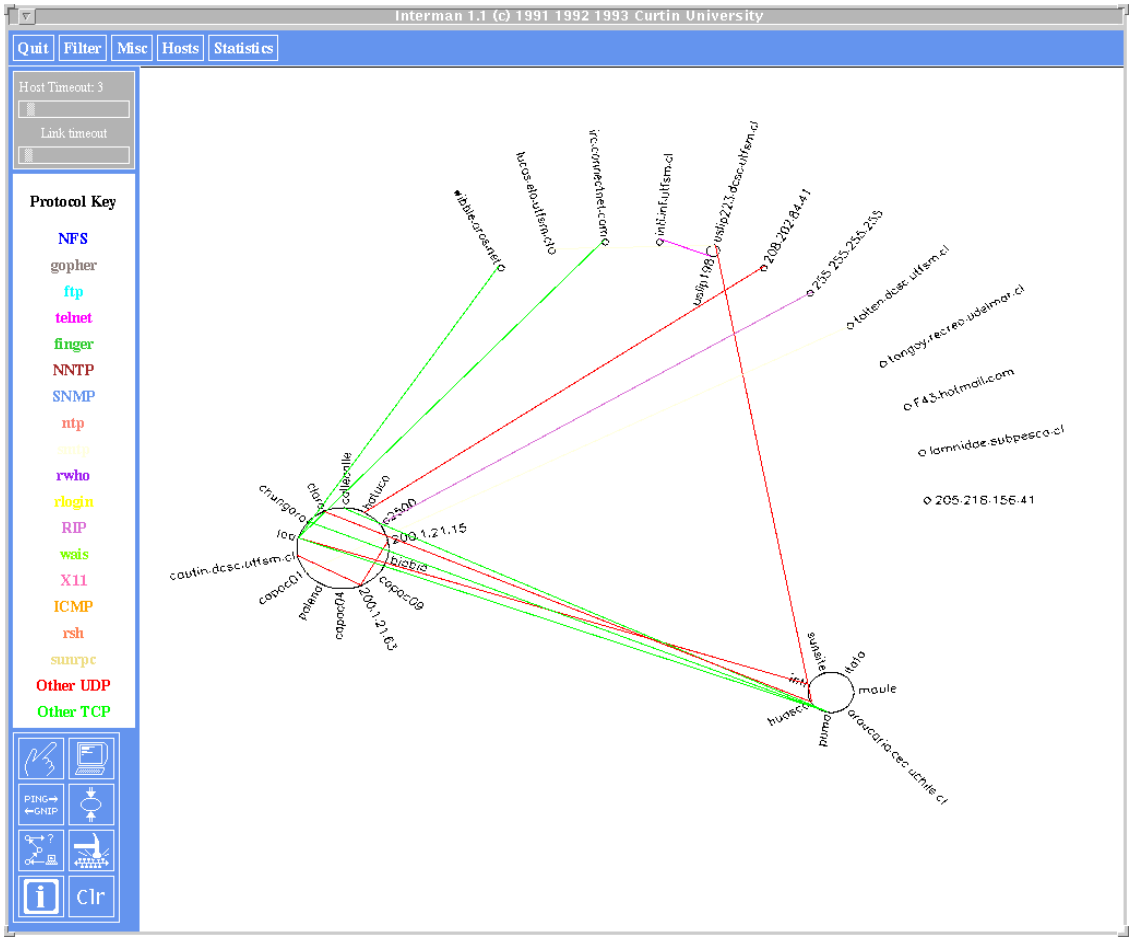


Figura Nº6 Interman

A diferencia con Etherman, este programa despliega el nombre de las máquinas en vez de su dirección Ethernet. Esto se debe a que Interman trabaja con conecciones IP, siendo TCP o UDP

Características

- ***Objetivo*:** Monitor de Tráfico (o Conexiones), dado que al aumentar el tráfico de una conexión, el grosor de la línea aumenta.
- ***Reporte*:** Monitor en Tiempo real
- ***Intrusividad*:** No intrusivo
- ***Operación*:** Local (o Remota utilizando el despliegue remoto del UNIX).
- ***Protocolos:*** Provee sólo de información de Servicios IP, sean TCP o UDP.

Ventajas

- Muestra las conexiones actuales entre nodos y redes.
- Se puede capturar/Imprimir el despliegue de conexiones (pantalla).

Desventajas

- Sólo muestra conexiones y no muestra cantidad de tráfico (ni cualitativo ni cuantitativo).
- No realiza registro de información o reportes históricos en forma automática
- En una conexión punto a punto, se muestra el protocolo con más tráfico, aunque exista otro protocolo entre esos mismos 2 puntos con menos tráfico.
- Operación remota sólo en otros equipos compatibles con X11 (UNIX).

4.3.3.- Packetman

Este no es un monitor de red propiamente tal, sino un analizador de paquetes Ethernet capturados en forma no-Intrusiva.

Para ello, la tarjeta de red actúa en modo "*Promiscuo*", en que permite al driver de la tarjeta de red escuchar todos los paquetes de la red aunque no sean dirigidos a su interface (y que tampoco sean un broadcast). Así puede escuchar todos los paquetes de la red Ethernet.

Esta herramienta se utiliza en la detección y análisis de fallas en las comunicaciones y protocolos a nivel del paquete, pero también puede ser usado en forma ilícita para espiar conexiones ajenas.

En el siguiente ejemplo, Packetman divide la ventana en tres secciones. La primera de ellas corresponde a todos los paquetes capturados en un intervalo predefinido de tiempo, donde aparecen los encabezados de los paquetes.

La segunda sección corresponde al detalle del encabezado del paquete especificado en la primera sección. En este caso, se muestran los encabezados Ethernet e IP de un paquete TCP/IP.

En la tercera sección se muestra la data real transportada.

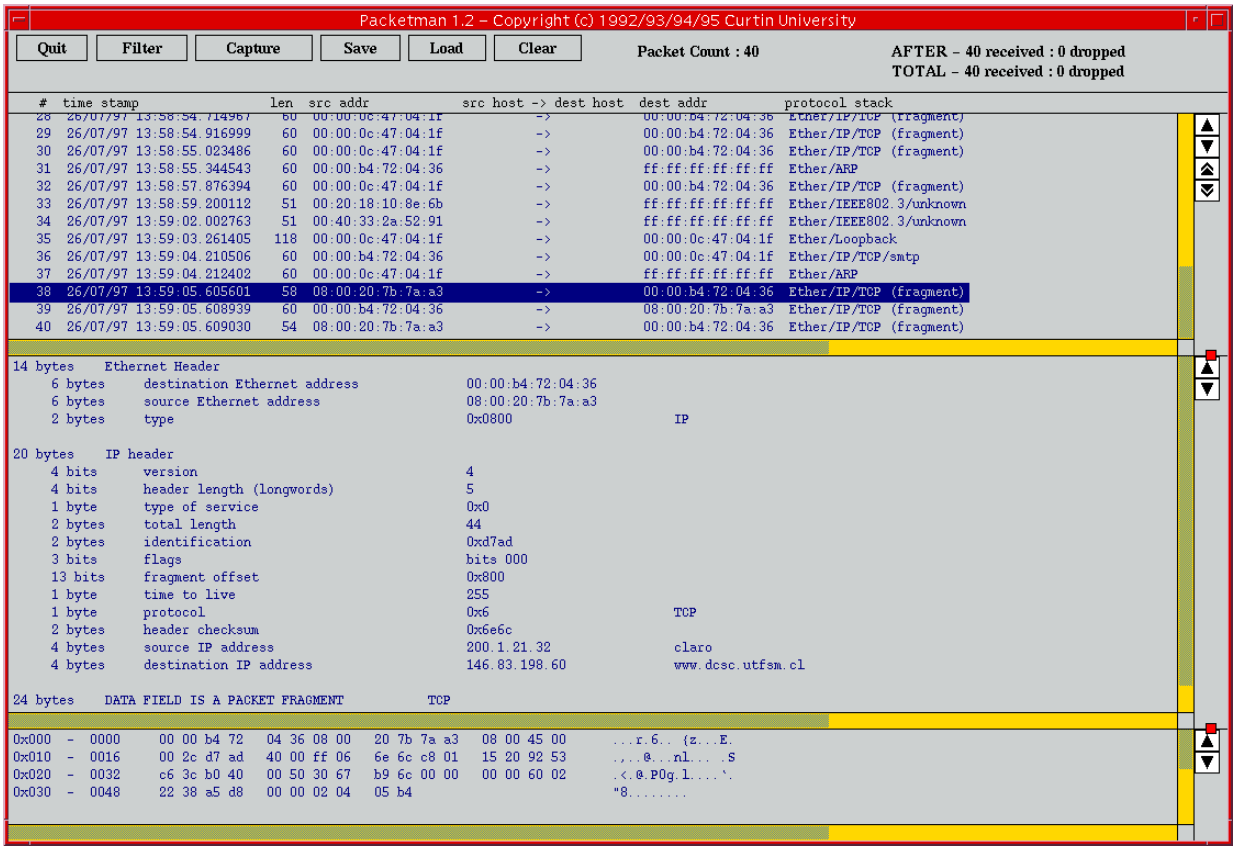


Figura Nº7 Packetman

5.- Monitor desarrollado (Netgraph)

Este monitor está orientado directamente a ser un Monitor de Tráfico en un ambiente muy amistoso para el usuario, que resuelve varias interrogantes que otros monitores no responden, presentando además la opción de ser utilizado para monitorear nuevas aplicaciones que se estén desarrollando.

Algunas de estas inquietudes son:

- ¿Cuánto tráfico genera un cierta dirección IP por esta subred?
- ¿Cuánto tráfico genera otra subred por esta subred?
- ¿Cuáles son los servicios IP que más ocupan el Ancho de Banda de la red?
- ¿Qué protocolos dentro de Ethernet Version II o IEEE 802.3 trafican más que otros?
- Mostrar cualitativamente y cuantitativamente el tráfico de un IP, de un Servicio IP o de una Subred IP.

Netgraph responde a estas inquietudes que no se pueden responder vía SNMP u otros programas. Estos programas tienen otras finalidades que generalmente complementan a Netgraph.

Netgraph es configurable a través del ambiente Web, donde el administrador selecciona qué servicios desea monitorear y luego los visualiza en forma gráfica en tiempo real, siendo necesario ejecutar a mano un sólo comando para que el sistema empiece a funcionar. Lo anterior se incorporó debido a la seguridad obligatoria que presenta Netgraph, permitiendo sólo al usuario autorizado a ejecutar este programa.

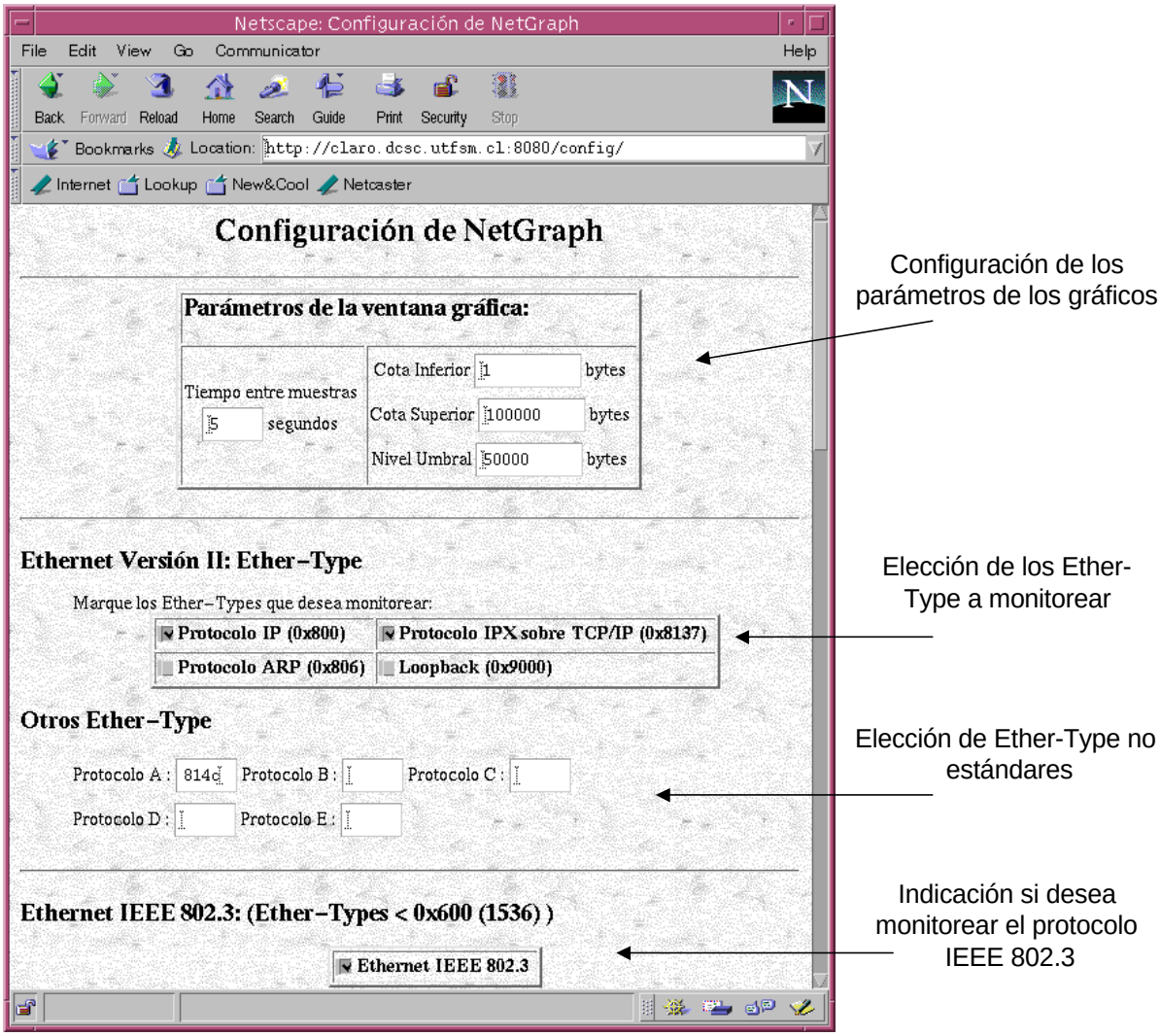
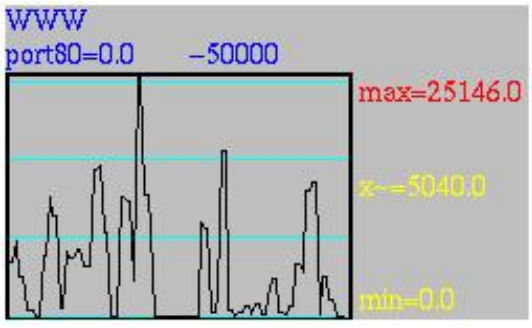


Figura Nº8 Configuración de Netgraph

Una de sus ventajas es que puede monitorear cualquier servicio que ocupe el estándar Ethernet Versión II, como son TCP/IP, UDP/IP, ARP, IPX (Novell) e incluso SNMP, además de los que aún no son estándar, lo que lo hace ideal para monitorear tráfico de protocolos en desarrollo. También permite monitorear el Tráfico completo del estándar IEEE 802.3. Las otras 2 versiones de Ethernet no es posible monitorearlas en esta versión de Netgraph.

Netgraph permite monitorear en forma independiente el tráfico de:

- Servicios IP (Ports) tales como WWW, SMTP, telnet, FTP, etc.
- Protocolos de Ethernet Versión II tales como IP, ARP, IPX/SPX
- Protocolo Ethernet IEEE 802.3
- Sockets IPX tales como SAP, RIP
- Direcciones IP (permite monitorear el tráfico de una dirección IP en particular)
- Subredes IP (permite monitorear el tráfico de una Subred en particular)



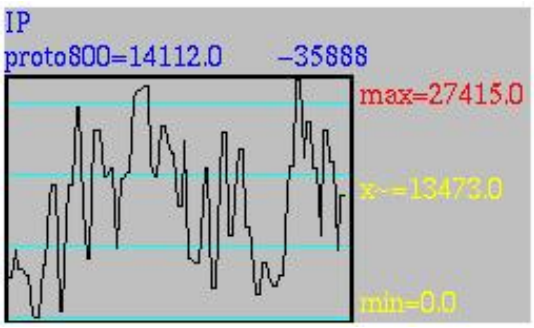


Figura Nº 9 Monitoreo del Port 80 (servicio Web) Figura Nº 10 Monitoreo del Protocolo IP

Características

- ***Objetivo*:** Monitor de Tráfico.
- ***Reporte*:** Tiempo Real e Histórico.
- ***Intrusividad*:** No Intrusivo. En el caso de operación remota, sólo afecta la medición del tráfico del Protocolo IP, ya que se transmite la información por un Port distinto a los estándares (usualmente medidos).
- ***Operación*:** Local (o Remota vía Web).
- ***Protocolos:*** Ethernet Versión II e IEEE 802.3. En capas superiores, es compatible con TCP/IP, IPX y sus servicios individuales.

Ventajas

- Muestra el tráfico de una Red separado por servicios.
- Entrega información en forma cualitativa y cuantitativa.
- Se puede capturar el despliegue para luego imprimir.
- Ambiente Web gráfico (fácil manejo)
- Operación Remota desde cualquier computador con Web browser Gráfico.
- Operación Remota no "contamina" medición de protocolos específicos (se transmite por otro port).
- Lleva un registro histórico (en archivos predeterminados)
- Permite monitoreo de nuevos protocolos en desarrollo.

Desventajas

- No muestra conexiones punto a punto
- No separa por servicios el tráfico de una máquina (una IP) en particular.

6.- Comparación entre los Monitores de Red descritos.

El motivo de esta comparación es el de caracterizar cada uno de los monitores descritos en los puntos 4 y 5 para concluir en qué casos es conveniente ocupar uno o una combinación de ellos. Lo anterior apunta a que no existe un monitor que realice todas las funciones deseadas, sino que para una finalidad específica existe un monitor adecuado. La razón de ello es que dependiendo exactamente de la necesidad del administrador de red o de un investigador existirá la forma de visualizar y desglosar la información en forma adecuada.

El siguiente cuadro resume las características principales de estos monitores.

Cuadro Nº 11 Cuadro Comparativo entre los monitores de red

	Objetivo		**Reporte**		**Intrusividad**		**Operación**		**Protocolos**
	Estado	Tráfico	Histórico	Tiempo Real	Sí	No	Local	Remota	
SNMP	√		√		√			√	Depende del producto. La mayoría es TCP/IP
Etherload y TPmon		√		√	√		√		V. II e IEEE 802.3 TCP/IP e IPX
Etherman		√		√		√	√		V. II e IEEE 802.3 TCP/IP e IPX
Interman		√		√		√	√		TCP/IP
Netgraph		√	√	√		√	√	√	V. II e IEEE 802.3 TCP/IP e IPX

Al hacer una comparación inicial, se puede concluir que Netgraph cumple las características ideales para un monitor de tráfico, lo cual es verdad. Sin embargo, esta conclusión depende de la información que se desea rescatar de un monitor. Por ejemplo si se desea obtener información sobre las conexiones punto a punto de la red, entonces lo más probable es que sean de mayor utilidad los programas Etherman e Interman.

Por esta razón es que se entrega el siguiente resumen de los monitores indicando el área donde son mejor aprovechados.

SNMP: Configuración y monitoreo de equipamiento físico de una red de computadores, como son los router, switch , hubs, servidores, UPS, etc.

Etherload y Tpmon: Medición de variables globales de tráfico, como son Ancho de Banda (BW), Throughput (bits por segundos) y Paquetes por segundo (pps).

Etherman e Interman: Su especialidad es mostrar cualitativamente las conexiones lógicas que existen en una red local o entre redes.

Netgraph: Monitoreo lógico en tiempo real de todos los servicios (en forma independiente) TCP/IP, IPX, Ethernet versión II y global de IEEE 802.3

7.- Descripción de Netgraph

A continuación se procede a describir a Netgraph en tres partes: primero se dará a conocer un diagrama funcional explicando cada una de las etapas o bloques; segundo, se explicará cada lenguaje ocupado y dónde éste fue aplicado; y tercero se explicará cada programa desarrollado cuyos listados se encuentran en los anexos adjuntos. Además se indica como cuarto punto el hardware necesario para usar Netgraph.

7.1- Diagrama funcional

El diagrama funcional (Figura Nº12) muestra las etapas por las cuales fluyen los datos desde la tarjeta de Red Ethernet hasta el gráfico en tiempo real. A continuación se presenta una breve descripción de las etapas:

Etapa 1: El usuario selecciona en páginas Web qué servicios desea monitorear. En esta etapa se ocuparon los lenguajes HTML, Javascript y C (en los Common Gateway Interface, CGI, que son programas en lenguaje C que permiten la comunicación entre solicitudes Web <FORM> y algún proceso en el servidor Web, como una base de datos, etc.).

Etapa 2: Los paquetes de la red son capturados y filtrados según la Etapa 1. Esta es la etapa más importante y fue desarrollada en lenguaje C para UNIX.

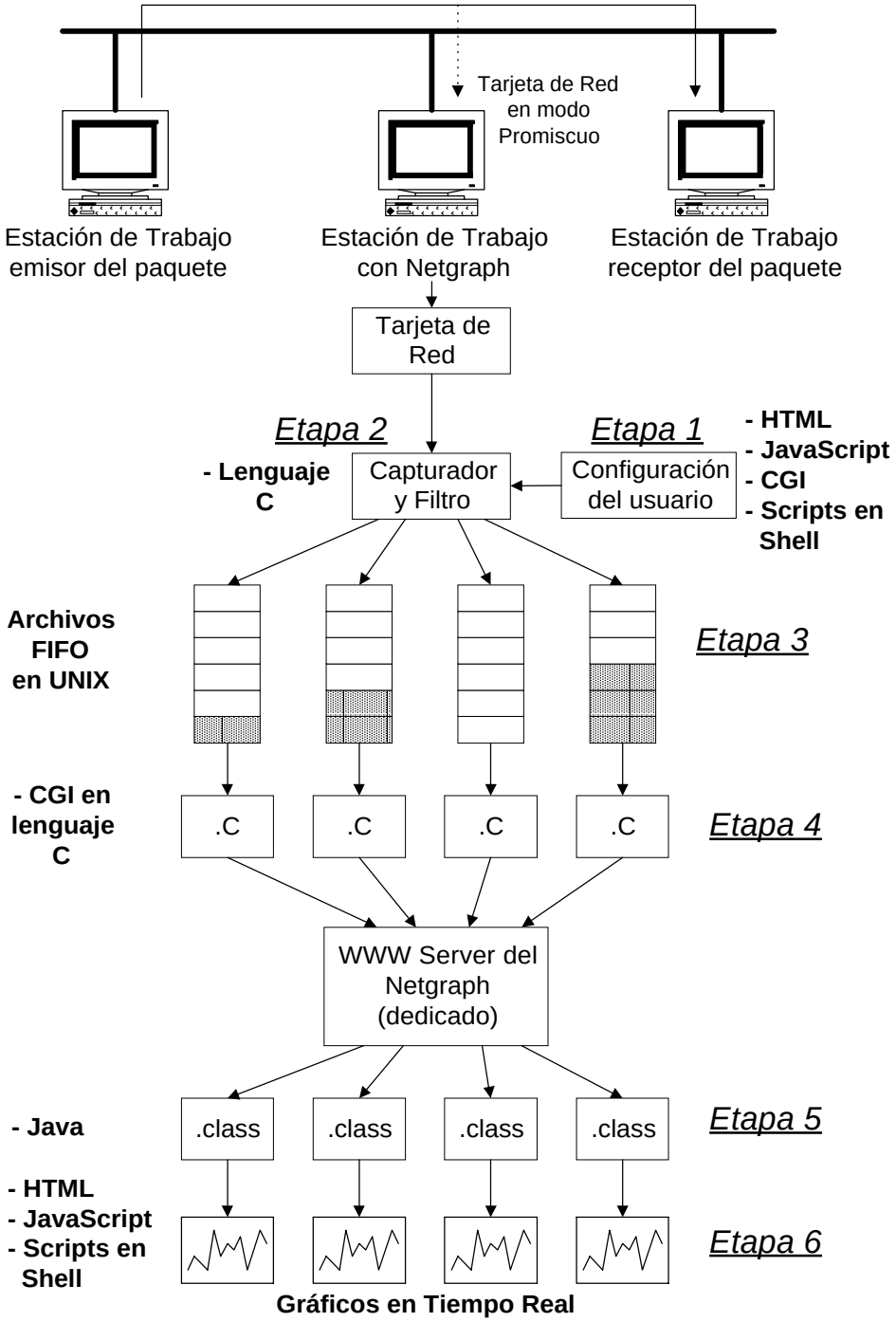
Etapa 3: La información es almacenada en archivos FIFO para ser rescatados por otra aplicación. Aquí se ocupa lenguaje C, utilizando una ventaja del Sistema Operativo UNIX, que es la existencia de archivos de tipo FIFO (First In First Out), los cuales son ideales para que una aplicación introduzca información y otra aplicación la consuma.

Etapa 4: Los Programas CGI leen periódicamente los archivos FIFO para obtener información. Estos programas, como ya se mencionó, están programados en lenguaje C para UNIX, los que utilizan la definición del estándar Common Gateway Interface que comunica información obtenida vía Web con programas en C o Perl para UNIX.

Etapa 5: El programa en Java (lenguaje explicado en el punto 7.2) ejecuta un programa CGI de la etapa anterior y recibe los datos

Etapa 6: El programa en Java grafíca los datos entregados por los programas CGI.

Figura Nº 12 Diagrama Funcional del Netgraph



Para un entendimiento más preciso, en forma de ejemplo se explicarán los pasos que debe seguir un paquete capturado:

Etapa 1: Suponga que el usuario desea monitorear la actividad del servicio TELNET en su red local y que ésta sea graficada con un tiempo entre muestras de 5 segundos, por lo que configura a Netgraph según el ejemplo en el Capítulo 8.

Etapa 2: El paquete es capturado y sólo se lee el encabezado, que para efectos del ejemplo será el siguiente:

```
845579273.547304 0800 76 200.1.20.13. 23 > 200.1.21.84. 1024 : tcp 22 (DF)
```

Su explicación es la siguiente:

`845579273.547304` : Tiempo en que el paquete fue leído (segundos desde 1 de Enero 1970 {sistema como trabaja el Sistema Operativo UNIX})

`800` : Protocolo (Ether-Type) de capa superior que ocupa a Ethernet

`76` : Largo del paquete

`200.1.20.13` : IP de una de las estaciones.

`23` : Port por la cual se emitió el paquete

`>` : Dirección del paquete

`200.1.21.84` : IP de la otra estación.

`1024` : Port por la cual se recibió el paquete

`tcp 22 (DF)` : Información adicional rescatable del header TCP/IP

Se puede deducir entonces que el paquete, de 76 bytes de longitud, viajaba desde la IP 200.1.20.13 hacia la IP 200.1.21.84 y que corresponde a un paquete del servicio TELNET (Port 23)

El paquete es descartado si no corresponde a ningún servicio a monitorear. En este caso corresponde a un servicio monitoreado (Port 23) y por ende su largo (76 bytes) es sumado con los otros largos de paquetes leidos durante la ventana de 5 segundos, según lo configurado.

Etapa 3: Cada 5 segundos, es almacenado en el archivo FIFO correspondiente al servicio TELNET el valor de largos de paquete acumulado, en este ejemplo suponga 1579 bytes. Dentro de este valor se incluyen los 76 bytes del paquete mencionado.

Etapa 4: El CGI que se dedica a comunicar el Applet Java (explicado en el punto 7.2) con el archivo FIFO (de TELNET), lee el valor (1579 bytes) que se encuentra en el archivo FIFO cada 5 segundos (sincronizado con la captura).

Etapa 5: El Applet Java ejecuta sólo la primera vez el CGI (a través de un servidor Web) y luego mantiene un canal abierto de flujo de datos. El Applet dedicado a graficar el servicio TELNET recibe el valor de 1579 bytes

Etapa 6: El mismo Applet Java de la Etapa 5, grafíca el valor recibido, preocupándose de ajustar márgenes, calcular el promedio de los datos en ventana, etc.

7.2- Lenguajes de Programación ocupados

En el desarrollo del Netgraph se ocuparon cinco lenguajes de programación estándares, que en conjunto permitieron comunicar las diferentes etapas del programa, empezando desde la obtención de datos de la tarjeta de red hasta la presentación de gráficos en tiempo real en un Web browser. A continuación se menciona el lenguaje ocupado indicando dónde éste fue aplicado y las razones de elegir este lenguaje, si es que había otra alternativa.

- **Lenguaje C:**

Este lenguaje se ocupó en tres oportunidades. La primera, en los CGI (Common Gateway Interface) que permite realizar acciones con datos ingresados por el usuario en las páginas Web de la configuración del Netgraph (Etapa 1). La segunda vez, fue en la captura, filtraje y almacenamiento de la data monitoreada (Etapa 2). Y la Tercera oportunidad fue en los CGI que comunican a los Applets con los archivos FIFO (Etapa 4). En este lenguaje se basó el grueso del desarrollo de Netgraph y por ende de la presente memoria.

En estas etapas se pueden haber ocupado otros lenguajes de programación del UNIX, como son el PERL o el Shell, pero ellos no son portables fácilmente a otras plataformas como lo es el lenguaje C, que con ligeros ajustes puede ejecutarse en otros Sistemas Operativos UNIX o PC.

- **Programación en Shell:**

  Programar Shell en UNIX es equivalente a programar en Batch (.BAT) en DOS en un PC, vale decir, son scripts que permiten realizar varias funciones con un solo comando. Específicamente se ocuparon en los Shell "sh" y "bash".

  Este lenguaje se ocupó en todas las etapas del Netgraph, desde la configuración inicial hasta algunas etapas de la presentación de la data monitoreada.

- **HTML (Hyper Text Markup Language) [11]:**

  El HTML es el lenguaje estándar para la creación de páginas Web, por lo que fue necesario trabajar con él en las etapas de visualización del Netgraph, tanto en la configuración realizada por el usuario (Etapa 1), como en la presentación de la data monitoreada (Etapa 6).

- **Javascript [12]:**

  Este lenguaje es un lenguaje de script, o sea, no es necesario compilarlo, sino que es interpretado por el Browser. Su sintaxis es una mezcla de C y Shell, y se inserta dentro del código HTML. Las funciones de Javascript se aplican al código HTML y a la página en sí, realizando diversas funciones, como son la validación de data ingresada, animación de páginas, etc.

Al igual que el lenguaje HTML, éste fue ocupado en casi todas las páginas Web que contiene Netgraph, para realizar funciones que el lenguaje HTML por sí solo no puede hacer. En este caso, Javascript realiza funciones de validación de la data ingresada por el usuario en estas páginas, es decir, este lenguaje permitió advertir al usuario de no dejar ciertos campos sin valor, validar la data ingresada y además modificar la página Web de la configuración del Netgraph mientras el usuario ingresa la data.

- **Java [13]:**

Este lenguaje es uno de los más recientes en ser liberado. Se trata de un nuevo concepto en Internet, que permite al programador generar programas en Java (lenguaje muy similar al C++, pero con algunas restricciones como por ejemplo: no puede tener acceso a archivos, sin punteros, etc.), ser precompilados e insertados en páginas Web. El usuario lector de esta página recibe el código precompilado y el browser lo termina de compilar en tiempo real en su computador.

Esto tiene dos grandes ventajas: 1) El programa se ejecuta en la máquina del usuario, no sobrecargando al servidor Web; 2) El programa es independiente de la plataforma en que se lea, vale decir, estos programas se ejecutan indistintamente en PC, Macintosh o UNIX.

Por otro lado, se necesitaba desplegar gráficos en tiempo real, a diferencia de otros programas que obtienen la data, generan el gráfico y lo despliegan como GIF y repiten el ciclo cada x segundos. El lenguaje que puede realizar esta función es únicamente JAVA. Por ello, Java fue ocupado en la etapa final del Netgraph, donde se muestra la data en tiempo real y que permite mostrar la capacidad del Netgraph (Etapa 6)

7.3- Programas que componen a Netgraph

Netgraph fue concebido inicialmente por la necesidad de ver en tiempo real los paquetes que circulan por la red local. Para ello, no se inició la creación de Netgraph desde cero, vale decir, programando la tarjeta de red en UNIX, etc., sino que se utilizaron dos programas disponibles en Internet que fueron altamente modificados y re-compilados para servir exactamente a este propósito.

El primero de ellos, es “tcpdump” de la Universidad de Berkeley [14] que rescata los headers Ethernet de la red local y los muestra en pantalla. El segundo de ellos es un demostrativo de graficador de datos en lenguaje Java que provee la Compañía SUN Microsystems como muestra de las capacidades de Java.

El resto de las funciones como son la configuración, filtraje, sincronización, automatización y control fueron íntegramente desarrollados como parte de la presente memoria.

Tanto el desarrollo de programas nuevos como la modificación de los dos existentes, requirió de un conocimiento detallado de lenguaje C estándar, del Sistema Operativo UNIX, estructura de datos y de todos los lenguajes mencionados en el punto 7.2.

- **7.3.1.- Programas ya existentes**

El primero de ellos es el “tcpdump” [15] que como su nombre lo indica, arroja, sin procesar, los headers de los paquetes Ethernet Versión II e IEEE 802.3 a la pantalla del usuario en un cierto formato predeterminado. Los fuentes en lenguaje C fueron modificados para que sirvieran el propósito exacto del Netgraph, por lo que el Netgraph incluye su propia versión del “tcpdump”.

El segundo programa es un graficador de datos en lenguaje Java, que se extrajo de un ejemplo que permite mostrar las fluctuaciones de los precios de acciones de una Bolsa de Acciones, llamado "Financial Portfolio (Stock Demo)" [16]. Este programa también fue modificado adaptándolo a la variable “byte” en vez de acciones que contienen valores negativos (fluctuaciones) y fracciones. Varios otros aspectos de presentación fueron modificados, como son el tamaño de ventana, letras, colores, tiempos de sincronización, manejo de parámetros entre páginas Web y el Applet (programa en Java).

7.3.2.- Programas desarrollados

Los programas desarrollados para construir Netgraph son los programas que se ocupan en cada etapa del flujo de datos del diagrama 7.1 y además los necesarios para que el usuario realice la configuración inicial del Netgraph (la instalación).

El listado de ellos se entrega a continuación y luego para cada uno de ellos se hará una descripción:

Configuración Inicial del Netgraph	configure.c
Capturador y Filtro	netgraph.c
CGI de Configuración	step1.c, step2.c
CGI (entre Java y FIFO)	applet-cgi.c
CGI (Salida de monitoreo)	stop.c
HTML y Javascript	.html y .js

7.3.2.1.- Configure.c (Anexo Nº 4)

Este programa se ejecuta la primera vez que se desea ejecutar el Netgraph. Su función es la de configurar todos los scripts y programas para que trabaje con la máquina en que se ha instalado el Netgraph, vale decir, rescata el "hostname" de la máquina y configura el servidor Web del Netgraph.

7.3.2.2.- Netgraph.c (Anexo Nº 5)

En este programa se encuentra el corazón del programa Netgraph, donde se realizan las siguientes funciones:

- Lectura de la configuración hecha vía páginas Web (por el usuario) de los servicios a monitorear.
- Lectura de los encabezados de todos los paquetes que viajan por la red.
- Filtración de los paquetes que se desean monitorear y guarda su largo en bytes.
- Creación de un archivo FIFO por cada servicio a monitorear e ingresa a él la sumatoria de los largos de paquete (acumulados en el tiempo entre muestras configurado por el usuario).

7.3.2.3.- Step1.c (Anexo Nº 6)

La configuración de Netgraph consta de dos pasos: *primero* se debe indicar todos los servicios a monitorear y *segundo* se indican los datos específicos de cada monitoreo (tamaño de ventana, etc.). Luego se visualizan los gráficos en tiempo real.

Step1 recibe todos los datos correspondientes al primer paso de configuración de Netgraph, los procesa y genera la segunda página (paso) de configuración.

Dado que este programa rescata datos de páginas Web, los procesa y genera otra página Web, es un programa que cumple con el estándar de CGI (Common Gateway Interface).

7.3.2.4.- Step2.c (Anexo Nº 7)

Este programa realiza una función muy similar a la de Step1.c. Recibe los datos indicados en el segundo paso de configuración, los procesa y luego genera la página Web donde se encuentran los gráficos en tiempo real (que ejecutan los programas en Java). Por la misma razón mencionada en Step1.c, Step2.c también es un programa CGI estándar.

7.3.2.5.- Stop.c (Anexo Nº 8)

Stop se ejecuta cuando se presiona sobre el botón de EXIT para salir del monitoreo de los servicios. Su función principal es el de cerrar todas las aplicaciones Java y terminar la ejecución del programa "capturador" de datos Netgraph.c, para que finalice todo el motor del Netgraph. Este programa además ejecuta un programa en Javascript que realiza una espera automática de cinco segundos.

7.3.2.6.- Páginas Web (HTML)

Esta es una de las funciones importantes del Netgraph, la interface visual con el usuario. Se desarrollaron en total 5 páginas Web, en que 3 de ellas son construidas dinámicamente, vale decir, son construidas dependiendo de los parámetros que haya ingresado el usuario, y por ende más difíciles de construir.

7.3.2.7.- Javascript (Anexo Nº 9)

Existen dos páginas Web en los que Javascript es fundamental. La primera de ellas es la primera página de configuración de Netgraph, donde el usuario escoge los servicios a

monitorear. Este programa debe realizar todas las funciones de validación de datos numéricos, campos requeridos que no deben dejarse en blanco y la “animación” que ocurre cuando se ingresan números IP o redes IP a monitorear. Este último aspecto es todo un desafío, ya que se modifica en la misma página “listas” que normalmente son fijas, sin ejecutar programas externos. Esto demuestra las capacidades del lenguaje Javascript que han sido incorporadas en el desarrollo de Netgraph.

El segundo programa en Javascript se ocupa una vez que se desea cortar el monitoreo de los servicios y se ejecuta el botón de “EXIT” donde se espera cinco segundos para volver a la página inicial. Esta espera automática se realiza con Javascript y su finalidad es la de cerrar apropiadamente todos los archivos FIFO y otros antes de volver a empezar otro monitoreo.

7.3.2.5.- applet-cgi.c (Anexo Nº 10)

Este programa CGI es ejecutado por el applet Java (graficador). Su función principal es la obtener los datos de los archivos FIFO y alimentar al applet Java que los grafica. Además, debe llevar en forma sincronizada la obtención de data con la alimentación.

Estas funciones no pueden haber sido incorporadas al programa en Java, porque éste no tiene la autorización de ocupar recursos del sistema en el cual es llamado, incluyendo lecturas a disco, por lo que el applet Java debe ocupar a un intermediario que sí está autorizado para leer información de disco. Esta restricción es del lenguaje Java, no de la programación, constituyendo una de las medidas de seguridad impuestas al lenguaje.

7.4.- Elementos de Hardware y Software requeridos.

Los elementos de Hardware requeridos como mínimo por Netgraph son:

Una estación de Trabajo SUN

modelo SPARCstation 4

32 Mb en RAM

Sistema Operativo Solaris 2.5

10 Mb espacio en Disco Duro.

Tarjeta de Red Ethernet 10BaseT

Netgraph ha sido probado exitosamente en esta máquina y a medida que Netgraph corra en máquinas de mejor rendimiento, la rapidez de procesamiento aumentará. Esto se ha corroborado instalando Netgraph en una SUN Ultra 1/170 de 128 Mb en RAM, Solaris 2.5.1. Cabe destacar entonces, que el requerimiento es simple y no necesita de Hardware especial (tarjetas de red, conectores o interfaces) para poder instalar Netgraph.

Los elementos de Software requeridos como mínimo por Netgraph son:

Ambiente Gráfico UNIX (Openwin, Common Desktop Interface CDE, X11)

Netscape Web Browser 3.0 (Navigator 3.0) [17]

Netgraph incluye su propio Web Server que, con un sólo comando, el administrador lo puede configurar de manera completa y además no interrumpe el posible Web Server convencional (Port 80) que está instalado en la máquina monitora.

8.- Manual de Uso del Netgraph

Este Manual de Uso está dirigido a un administrador de redes que está familiarizado con instalaciones de programas en UNIX y entiende, en forma general, los protocolos que desea monitorear. La primera parte muestra gráficamente el proceso de instalación de Netgraph y la segunda parte muestra una sesión de ejemplo.

8.1.- Instalación.

Netgraph debe ser ejecutado por una cuenta especial de nombre “netweb” que pertenece al grupo “monitor”. Esto se debe a los permisos especiales de superusuario “root” que tiene este programa, por lo que se debe crear una cuenta aparte para su utilización.

Paso 1) Crear la cuenta “netweb” (UID=90) y grupo “monitor” (UID=90)

root @ sun:~$ passmgmt –a –u 90 –g 90 –h /export/home/netweb –s /usr/local/bin/bash netweb

root @ sun:~$ passwd netweb (escoger password)

Paso2) copiar y descomprimir Netgraph.

root @ sun:~$ cd /export/home/netweb

root @ sun: /export/home/netweb$ cp /install/netgraph-v1.0.tar.gz .

root @ sun: /export/home/netweb$ tar xvf netgraph-v1.0.tar.gz

root @ sun: /export/home/netweb$ chown root:other ./tcpdump

Paso3) Ingresar a la máquina como usuario “netweb” y ejecutar “Configure”.

netweb @ sun:~$ ./Configure

Este programa configurará todos los parámetros de su máquina para que trabaje con Netgraph. Al final, le pedirá que ingrese una password que será la password de la página Web inicial como protección para que nadie ingrese a Netgraph vía la red, excepto este usuario “netweb”. La pantalla será como la siguiente:

Figura Nº 13 Configure

Paso4) Se debe habilitar el servicio de Web que trae el Netgraph en el port 8080.

netweb @ sun:~$ www.start

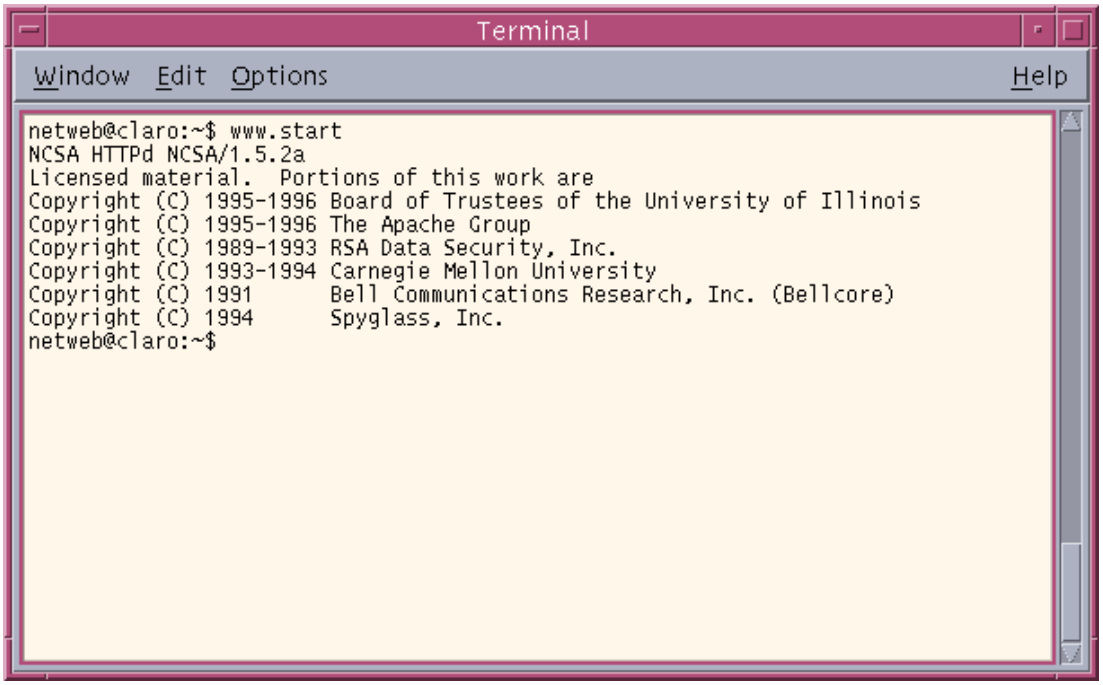


Figura Nº14 www.start

Paso5) Ejecutar Netscape y entrar a la página http://maquina:8080, .donde "maquina" es el hostname de su máquina SUN. En el recuadro anterior "maquina" corresponde a "claro".

8.2.- Sesión de Ejemplo.

A continuación se muestra paso a paso un ejemplo de uso del Netgraph.

8.2.1 Reporte en Tiempo Real.

1.- Al ejecutar el Netscape e ingresar a http://maquina:8080 (habiendo ejecutado previamente el comando www.start), se verá un recuadro que le pedirá un login y una password. Se debe ingresar "netweb" y la password que se ingresó en el momento de ejecutar el comando "Configure".

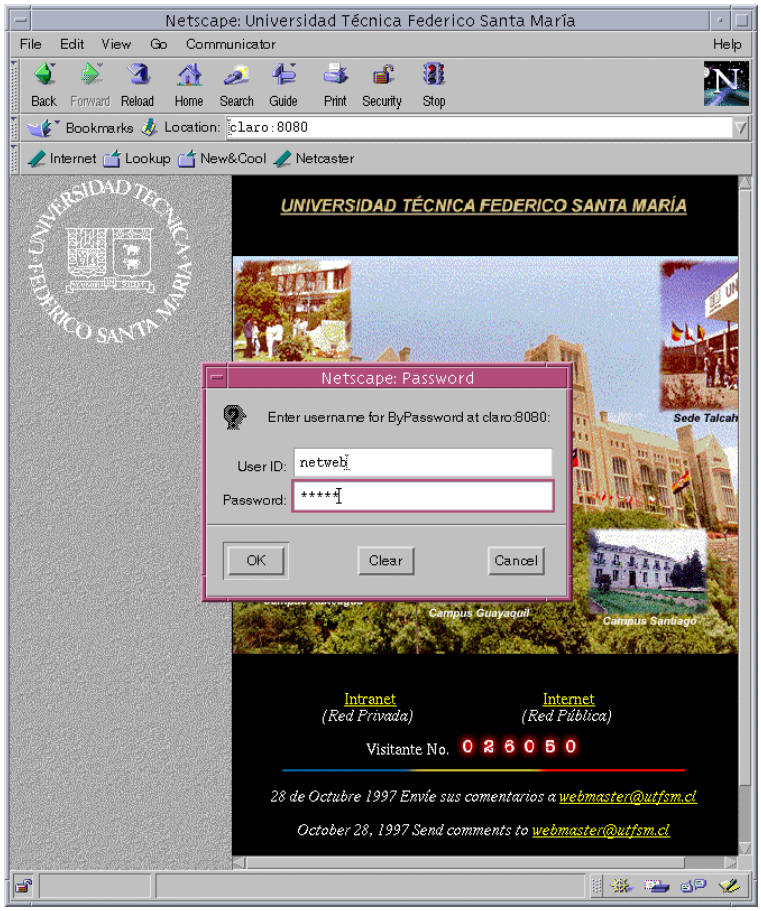
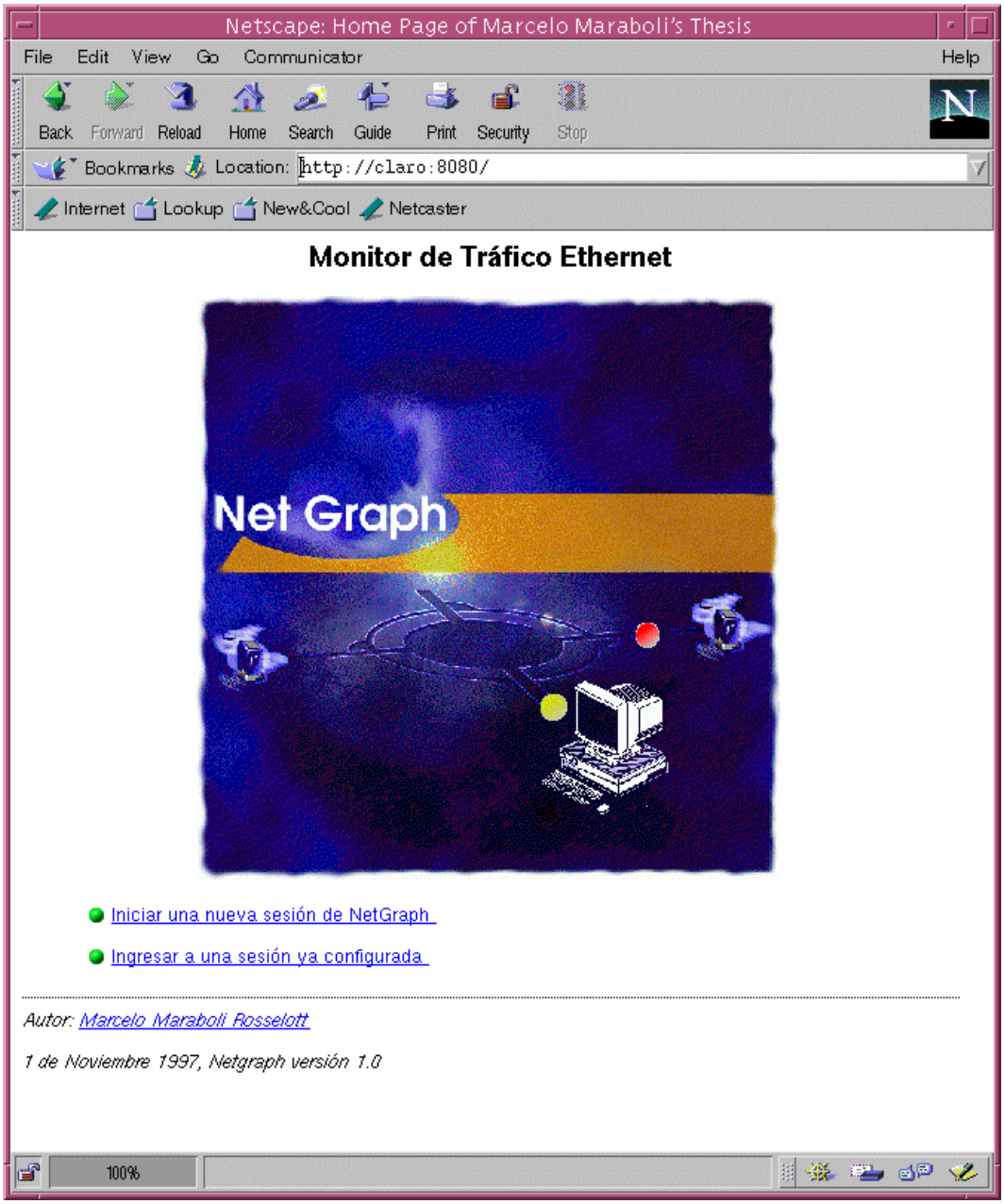


Figura Nº 15 Ingreso al Netgraph

2.- Una vez validada la password, se muestra la pantalla de presentación de Netgraph en que el usuario puede iniciar una nueva configuración de Netgraph o ingresar a la configuración de la sesión anterior.

Para poder ingresar directamente al Netgraph, ya debe haber configurado una sesión previamente (se toma la última).

Figura Nº 16 Página inicial de Netgraph



3.- Al ingresar a Nueva Sesión, se ingresa a la página donde se permite modificar los parámetros y seleccionar los servicios a monitorear. Esta página es larga por lo que se presentará de sección en sección.

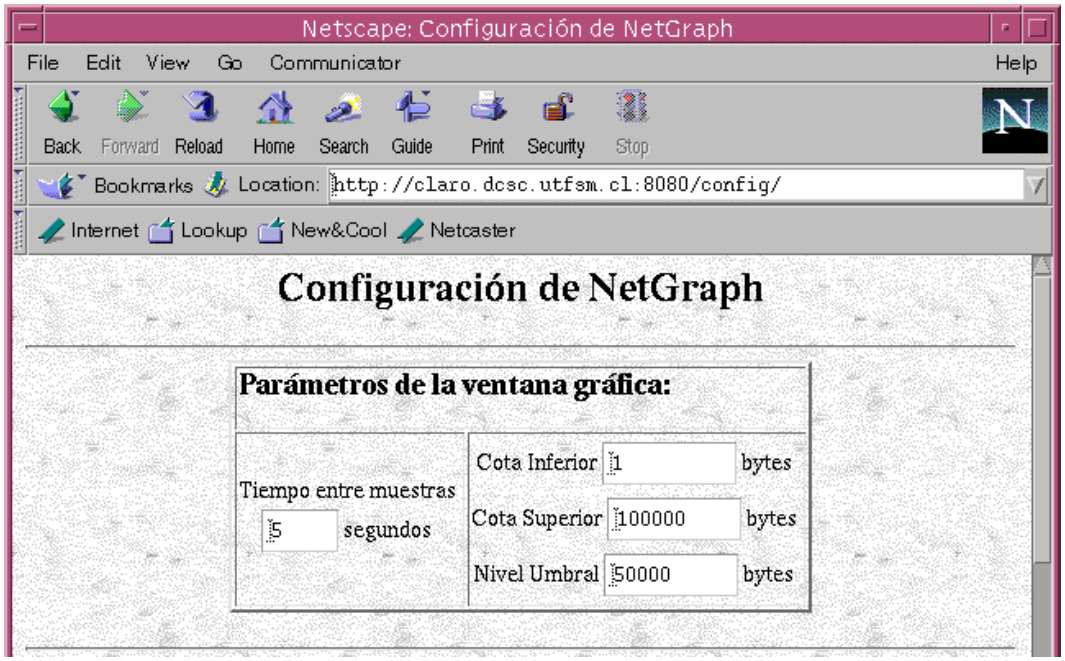


Figura Nº 17 Parámetros de la ventana gráfica.

Parámetros de la ventana gráfica:

Tiempo entre muestras (segundos): Este número indica cada cuántos segundos se debe mostrar un nuevo dato en los gráficos de cada servicio. Este parámetro es común para todos los servicios que se escojan y el dato mostrado corresponde a los largos de paquetes acumulados entre una muestra y la siguiente. En todos los gráficos, la cantidad de datos desplegados es 60, y como el tiempo entre muestras es por defecto de 5 segundos, el gráfico total muestra 5 minutos de tráfico histórico.

Cota Inferior: Corresponde al valor inicial del valor mínimo del eje “X” (bytes) del gráfico. Dado que los gráficos se ajustan dinámicamente, este valor es sólo el valor inicial. Su valor por defecto es 1 byte.

Cota Superior: Corresponde al valor inicial del valor máximo del eje “X” (bytes) del gráfico. Dado que los gráficos se ajustan dinámicamente, este valor es sólo el valor inicial. Su valor por defecto es 100 000 bytes.

Ethernet Versión II: Ether-Type

Corresponde a la sección donde se especifica los servicios de la capa tres (Capa de Red) del modelo ISO/OSI se desean monitorear. Los que se muestran son los más ocuapdos: IP, IPX, ARP y Loopback. Como se muestra, se pueden incluir hasta cinco Ether-Types adicionales.

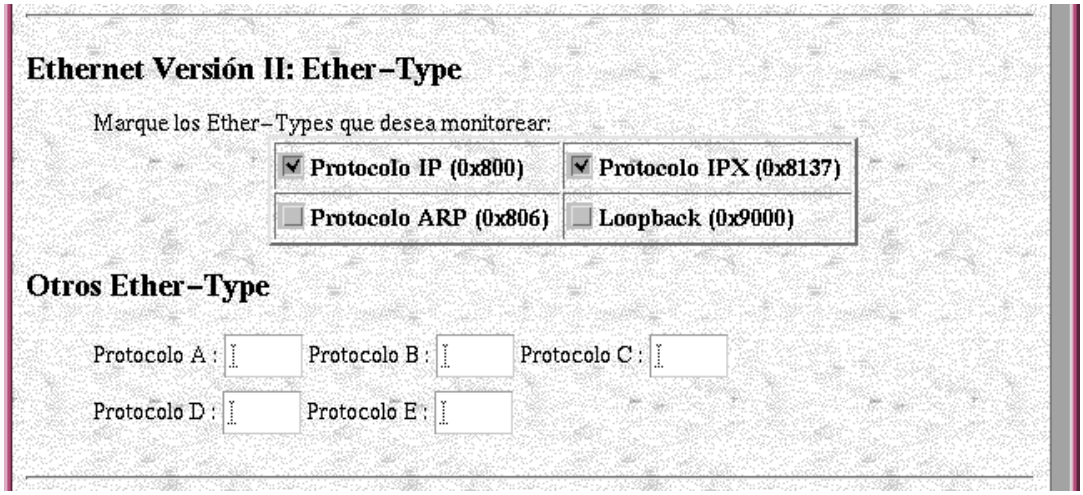


Figura Nº 18 Ether Type

Ethernet IEEE 802.3: (Ether-Types < 0x600 (1536))

En esta sección se puede seleccionar si se desea monitorear el tráfico de todo el protocolo IEEE 802.3, independiente del protocolo de capa superior que lo este ocupando.

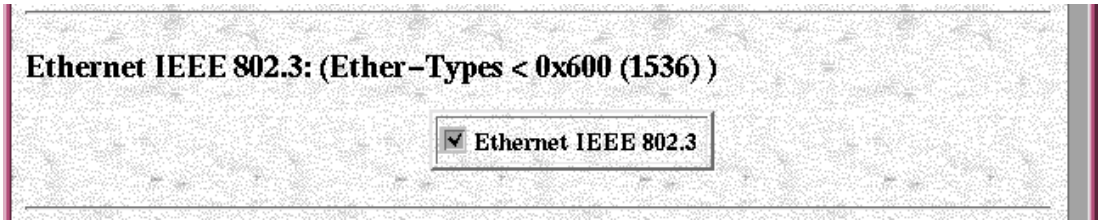


Figura Nº 19 IEEE 802.3

TCP/IP y UDP/IP Ports.

Corresponde seleccionar todos los servicios TCP y UDP que se desean monitorear. Aparecen seleccionados por defecto los Ports: 20, 23, 25 y 80. Se pueden además incluir otros cinco ports no estándar.

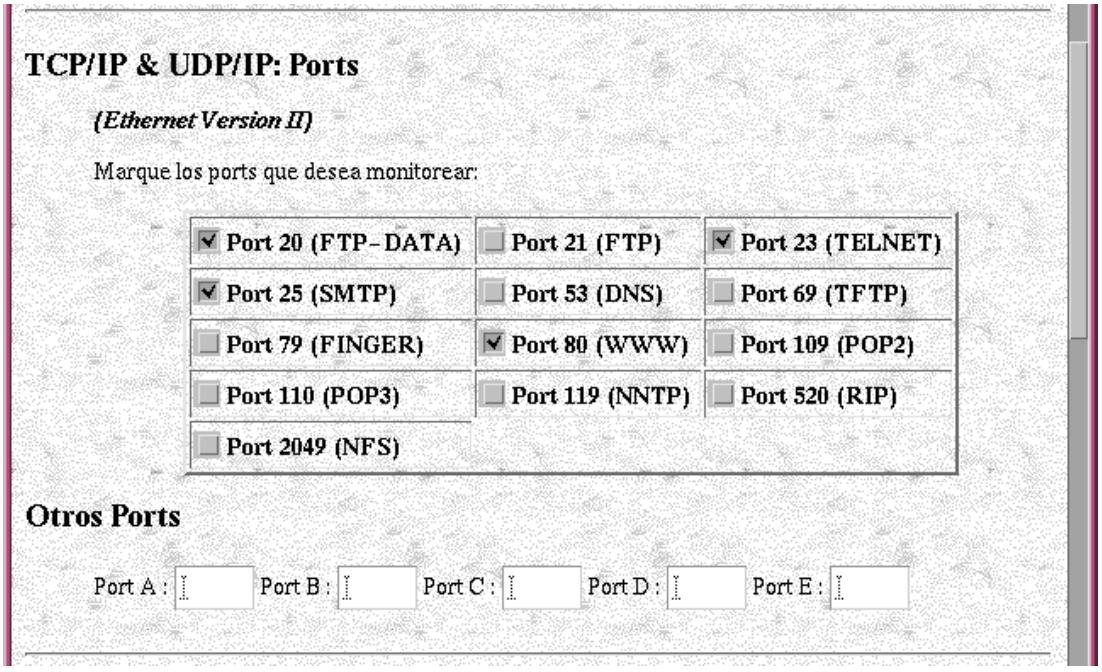


Figura Nº 20 Elección de puertos TCP/IP y UDP/IP

Sockets IPX:

Corresponde seleccionar los Servicios IPX (Sockets) que se desean monitorear. Estos sockets son los transportados por la norma IEEE 802.3 y no aparece ninguno seleccionado por defecto. Como los demás items, también se pueden incluir cinco no estándar.

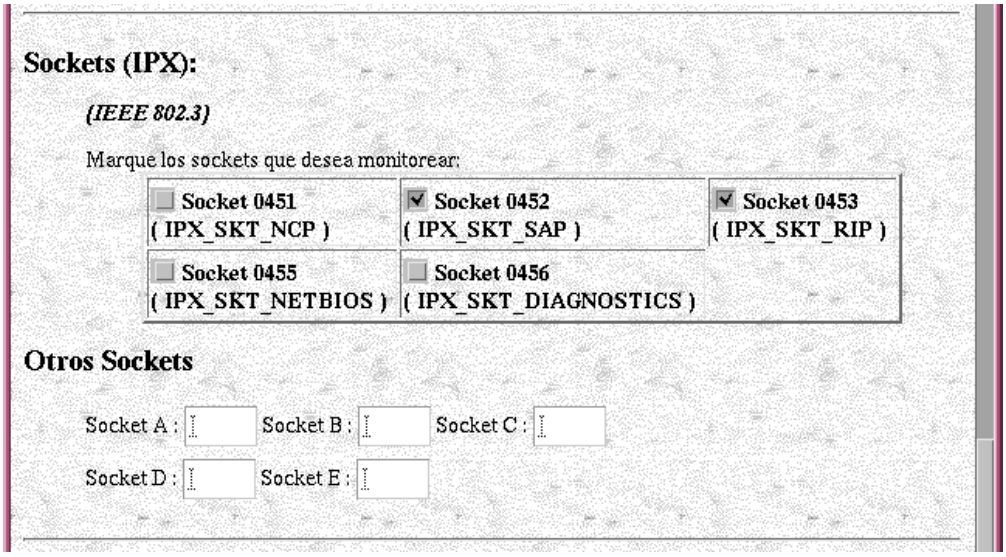


Figura Nº 21 Elección de Sockets IPX

Direcciones IP

En esta sección se ingresan las direcciones IP a monitorear. Para ello, se ingresa el número IP en el casillero de la izquierda y luego se agrega a la lista pulsando la flecha "→".

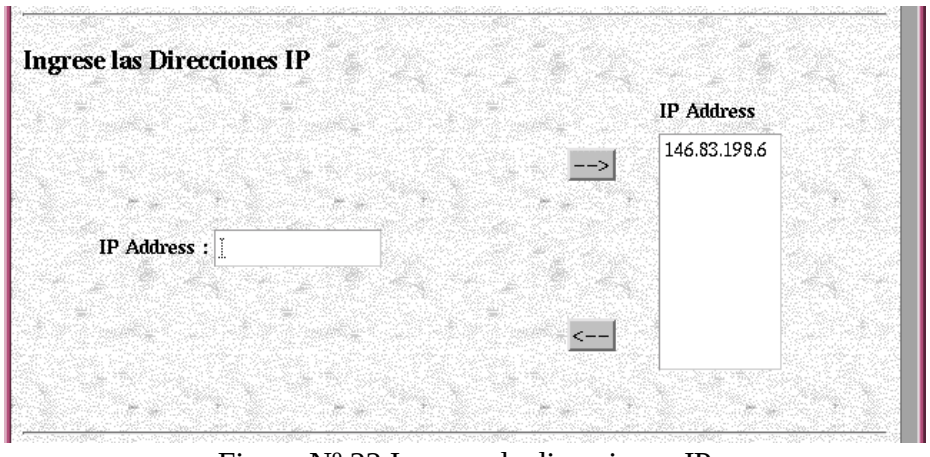


Figura Nº 22 Ingreso de direcciones IP

Subredes IP

Al igual que la sección de Direcciones IP, en esta sección se ingresan las redes (red y máscara) en el lado izquierdo y luego se pulsa sobre el botón "→".

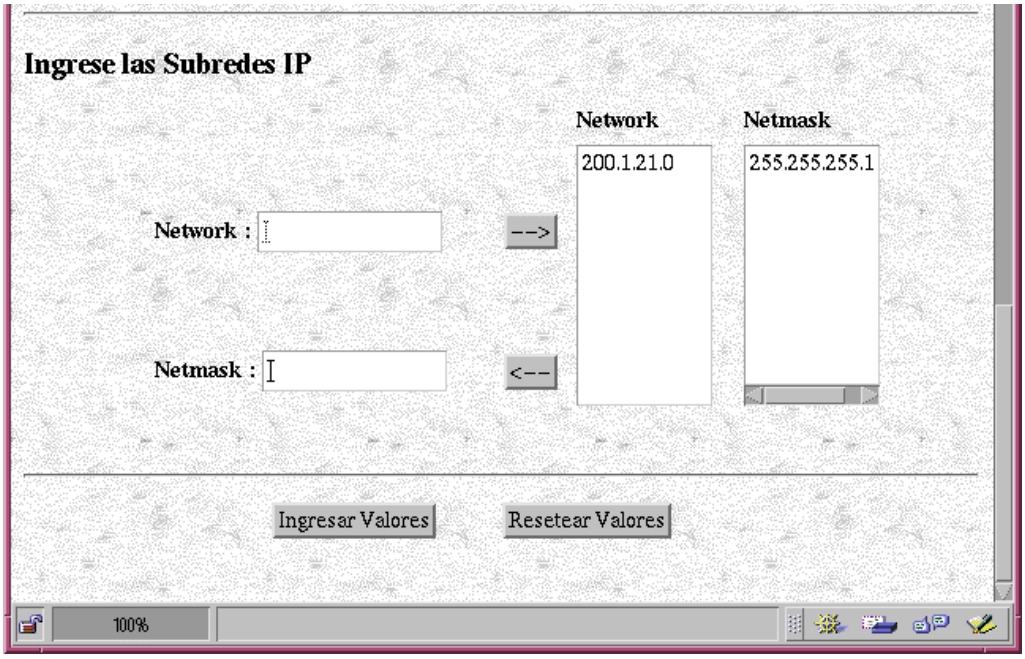


Figura Nº 23 Ingreso de subredes IP

En ambos casos, el botón "←" sirve para editar alguna de las opciones (IP o redes) ya ingresadas a la lista, por lo que es necesario seleccionar el ítem a editar y luego pulsar el botón de "←".

4.- Una vez lista la configuración de los servicios a monitorear, se debe presionar sobre el botón de "Ingresar valores" para seguir con la configuración y luego el monitoreo. El botón de "Resetear valores" permite borrar todas las opciones marcadas en esta sesión con lo que los parámetros vuelven a su estado inicial.

Al presionar sobre “Ingresar Valores”, se obtiene la siguiente pantalla:

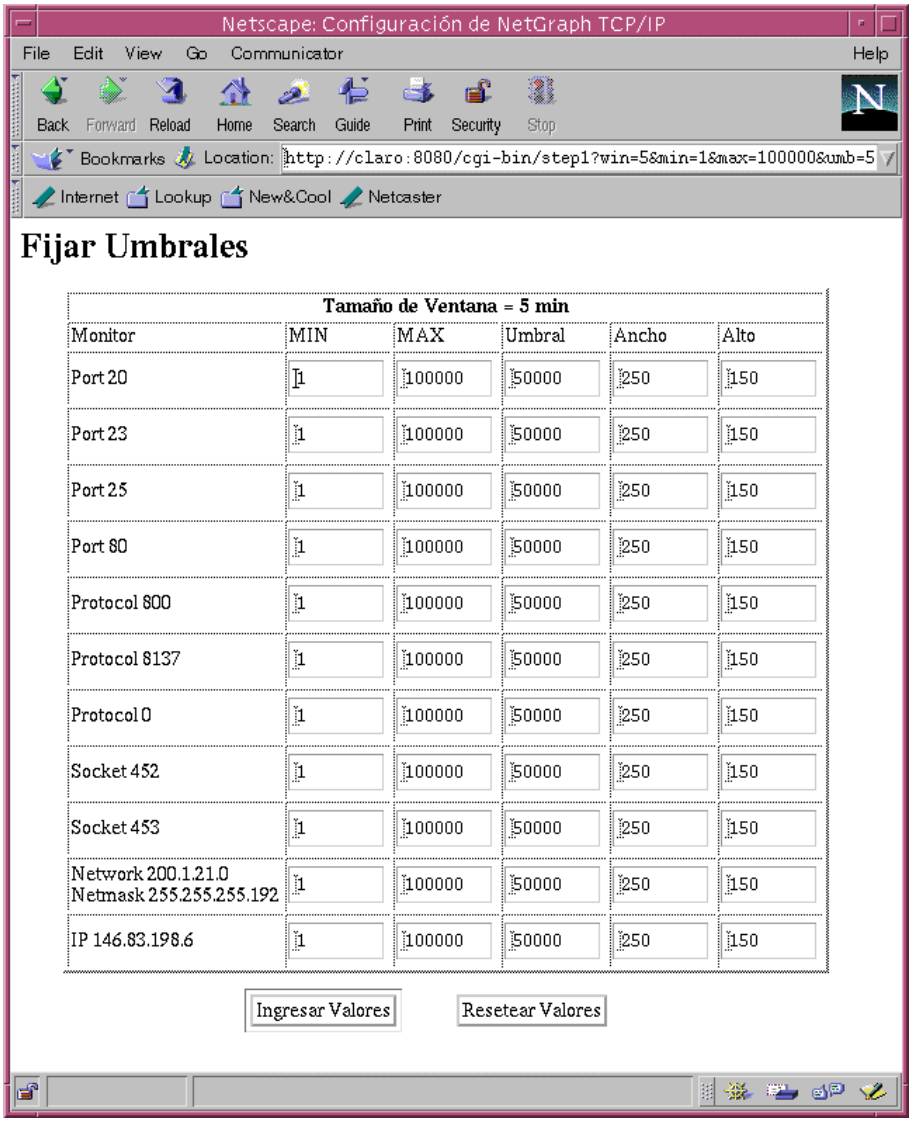


Figura Nº 24 Fijar umbrales

En esta tabla se puede modificar los “Parámetros de la ventana gráfica” en forma individual, permitiendo un control más fino de cada ventana de un servicio a monitorear.

5.- Una vez completadas las modificaciones, se presiona sobre “Ingresar valores” donde se llega a la siguiente ventana:

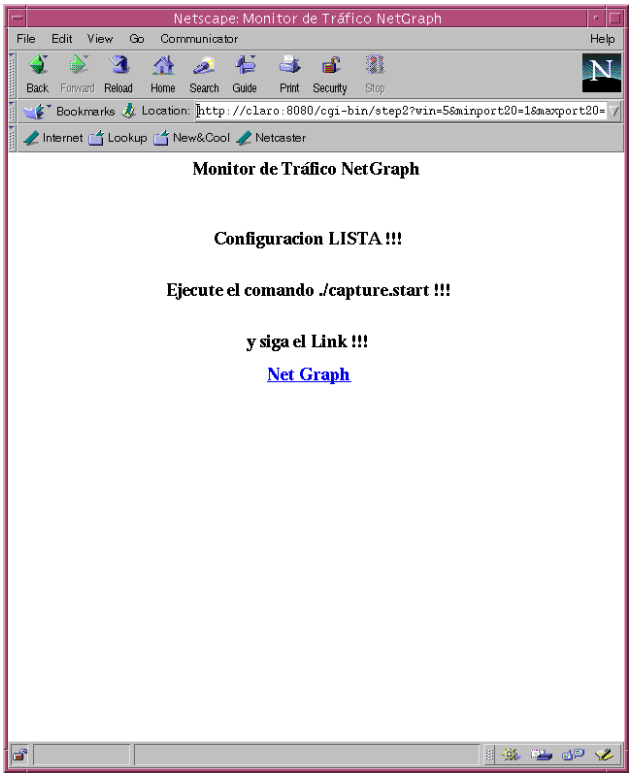


Figura Nº25 Configuración lista

6.- En este momento se requiere de la intervención directa del el usuario “netweb”, donde debe ejecutar a mano el comando indicado.

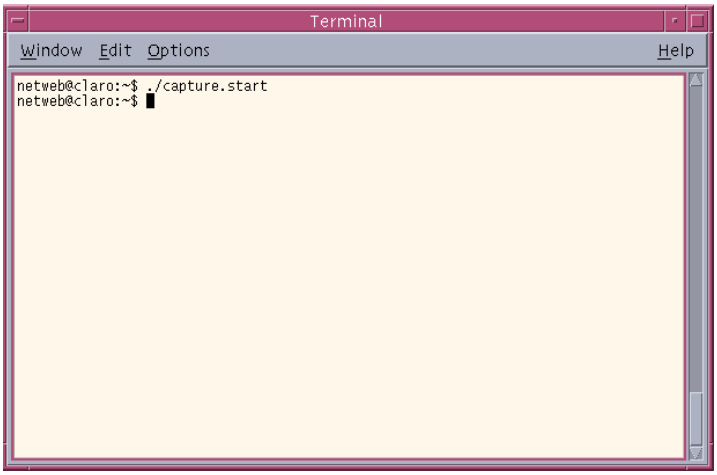


Figura Nº 26 Ejecución de capture.start

Lo anterior es requisito debido a razones de seguridad como: usuarios remotos no autorizados a usar Netgraph en esa máquina específica, etc. La razón de la doble “palabra clave”, una para ingresar a la página y otra para activar Netgraph, es para permitir que existan varios Operadores que puedan realizar monitoreo, pero dependiendo de las claves entregadas puede éste correr en distintas máquinas.

7.- En la siguiente sección se muestra el resultado de estas configuraciones una vez que el usuario netweb ha ingresado el comando “capture.start” y presionado sobre el link “Netgraph”. El resultado es el siguiente:

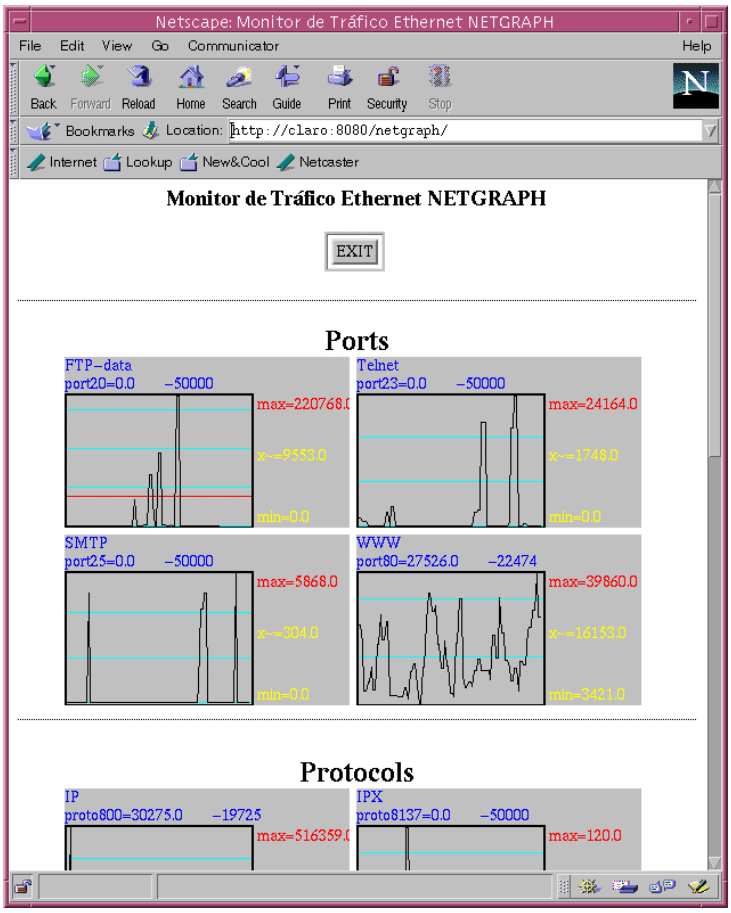


Figura Nº 27 Netgraph con varios monitores

A continuación se explica el significado de los elementos de cada ventana gráfica.

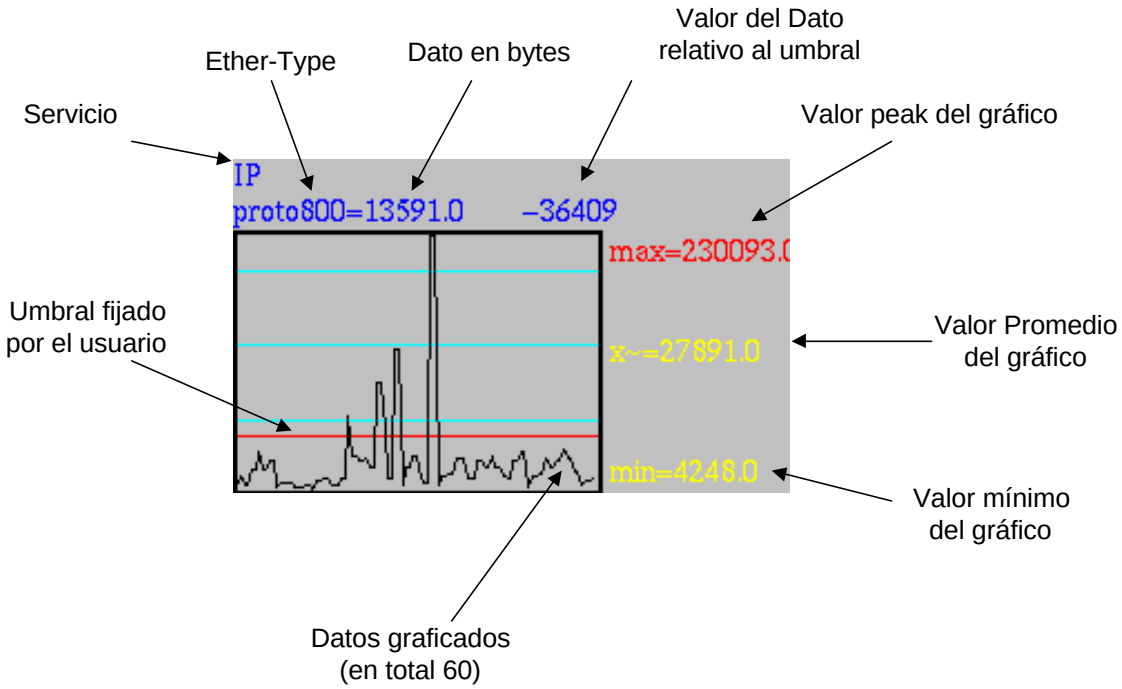


Figura Nº 28 Descripción de los gráficos

Se ilustra el ejemplo para el monitoreo del protocolo IP especificando un umbral de 50.000 bytes. Para el caso de las ventanas gráficas de las redes, se especifica la Red y la máscara de Red. En el caso de los IP, sólo se especifica el IP. El resto de los datos y simbología es común para todos los servicios.

Al hacer una análisis de cada sección, se pueden apreciar los servicios que se han indicado para el monitoreo.

EXIT

Una vez finalizado el monitoreo, se DEBE presionar el botón de EXIT, de lo contrario el Netgraph seguirá capturando data.

Figura Nº 29 Salida de Netgraph

Ports:

En este ejemplo, se muestra el tráfico de los servicios de: FTP, Telnet, SMTP (email) y Web; cada uno con sus respectivos datos individuales.

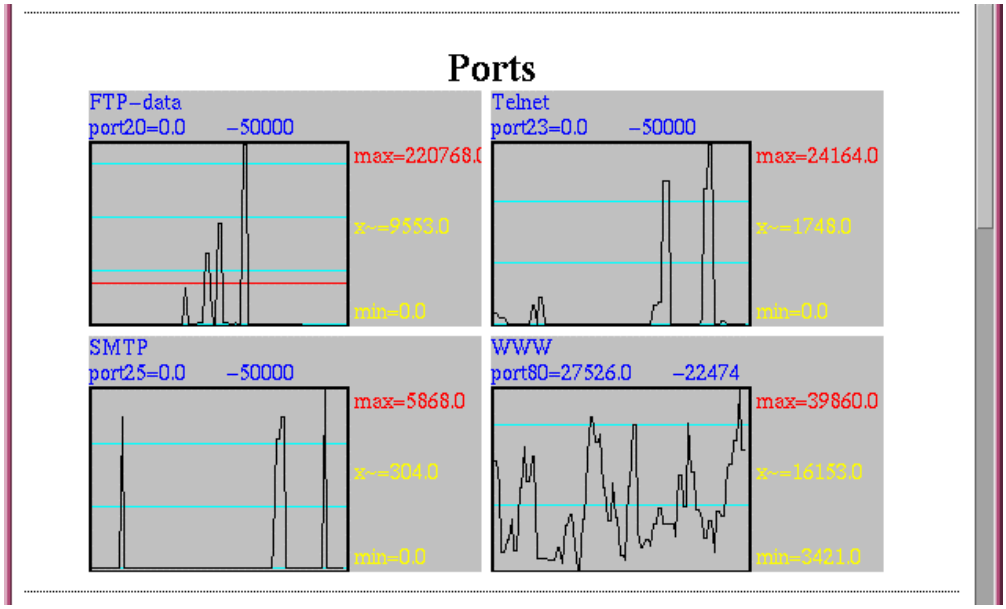


Figura Nº 30 Monitores de puertos IP

Protocolos:

Se muestran los protocolos: IP, IPX sobre Ethernet Versión II e IEEE 802.3

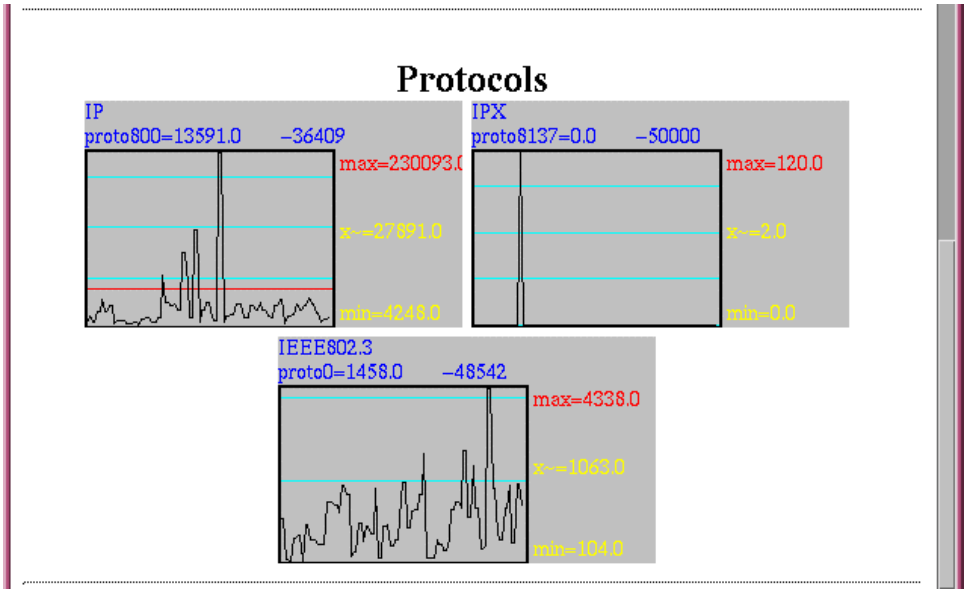


Figura Nº31 Monitores de Protocolos

Sockets:

En estos dos recuadros se muestran los Sockets IPX (IEEE 802.3) de los Servicios SAP (Service Advertisement Protocol) y RIP (Routing Protocol).

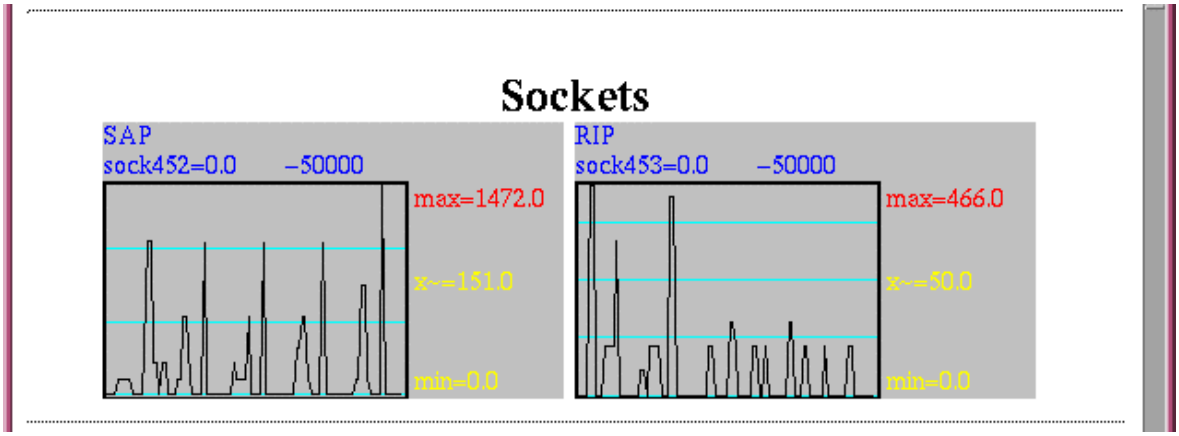


Figura Nº 32 Monitores de Sockets

Redes IP:

En este gráfico se indica la red y la máscara de la Subred IP que se está monitoreando, indicando el tráfico que existe entre la red local conectada a Netgraph y la red del gráfico en cuestión.

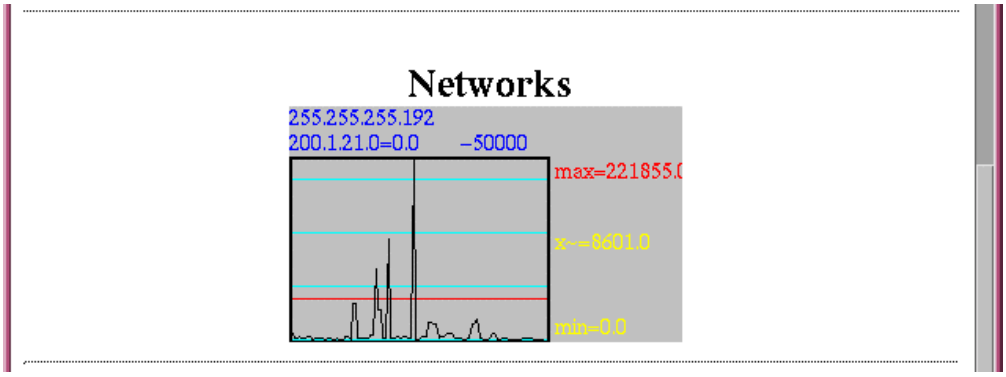


Figura Nº 33 Monitor de subred IP

Direcciones IP:

El gráfico muestra el tráfico de las Direcciones IP indicadas que se comunican o atraviesan la red que se está monitoreando. Este gráfico no debe confundirse con el tráfico TOTAL generado por esa IP en particular, sino que es su tráfico que es visto en esta red, vale decir, el aporte que hace estas IP a la red local, pudiendo pertenecer o no a ella.

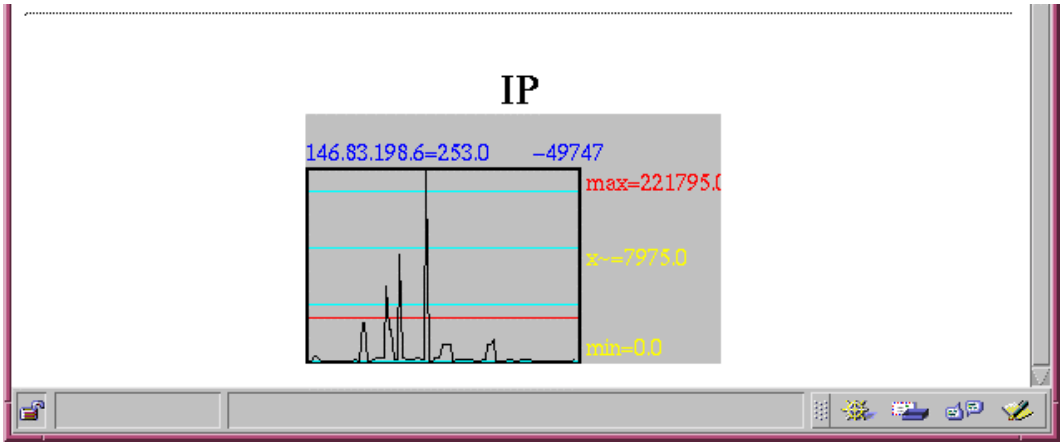


Figura Nº 34 Monitor de dirección IP

8.- Al finalizar el monitoreo, se DEBE presionar el botón de EXIT, con lo que se llega a la siguiente pantalla:

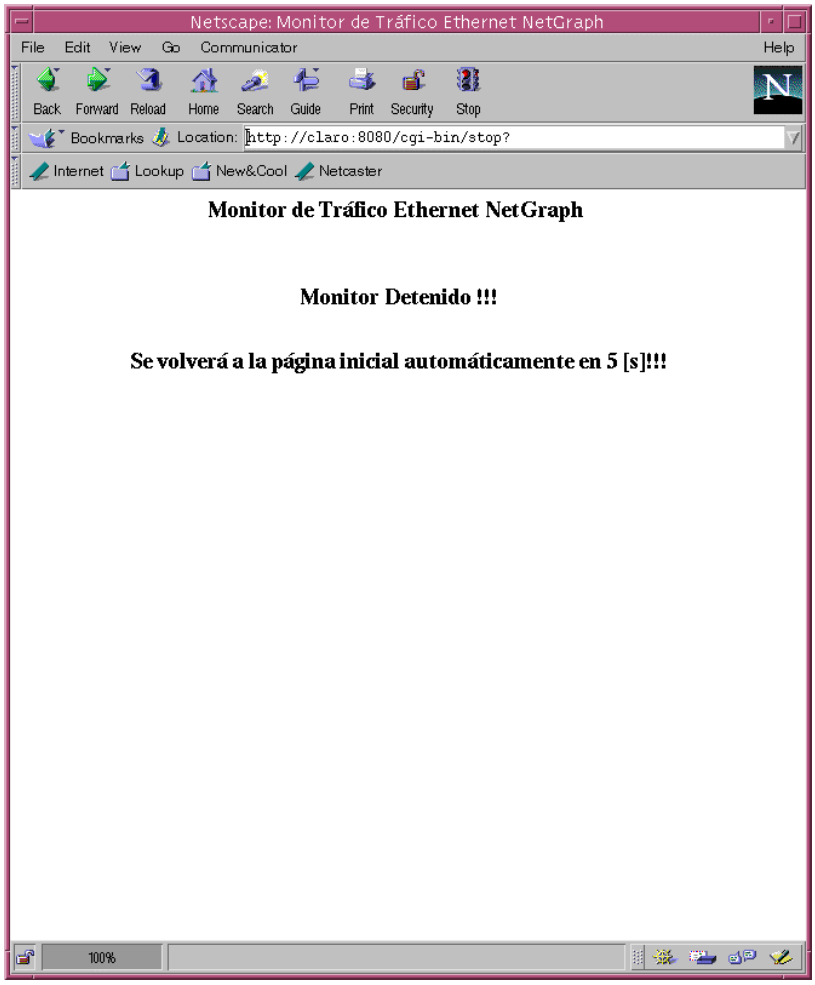


Figura Nº 35 Monitor Netgraph detenido

Sin intervención del usuario, se retornará a la página principal del Netgraph. Este tiempo de cinco segundos se debe a que antes de iniciar otra sesión de Netgraph se deben cerrar apropiadamente todos los capturadores, archivos FIFO, archivos normales para un apropiado trabajo en un nuevo monitoreo.

9.- Al finalizar la sesión de Netgraph, se debe detener además el servidor Web que fue ejecutado al inicio de la sesión con el comando www.start. El servidor Web de Netgraph se detiene con el comando www.stop como lo muestra la siguiente figura.

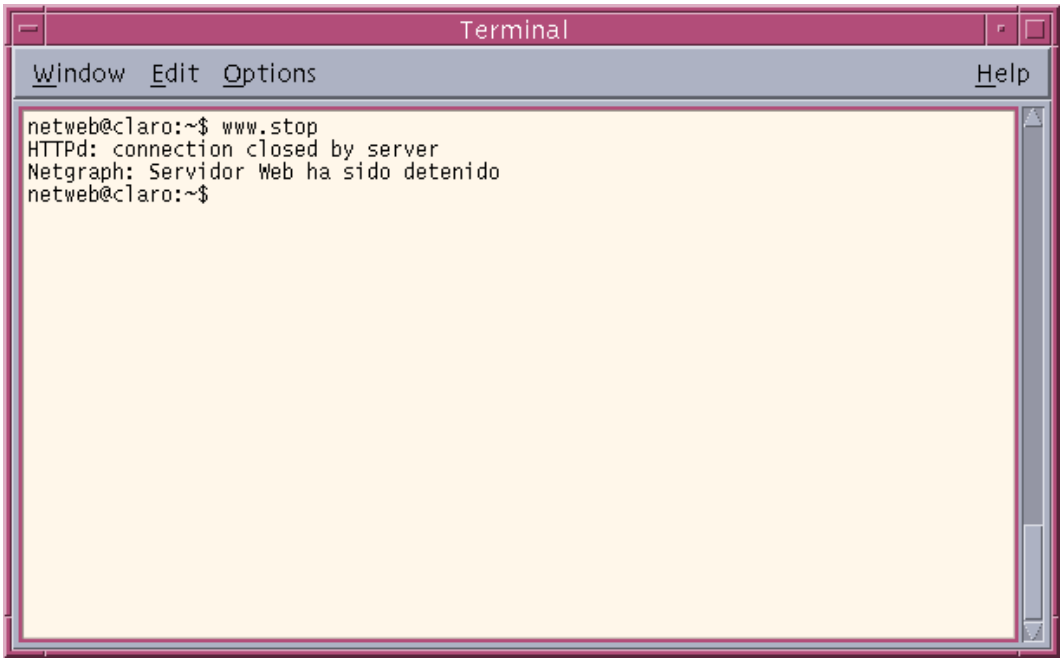


Figura Nº 36 Ejecución de www.stop

8.2.2.- Reporte Histórico

Una vez finalizada la sesión de Netgraph, toda la información (data a graficar) queda almacenada en archivos de tipo texto en el subdirectorio “data”. Los datos son directamente puestos en los archivos de la misma forma en que los programas en Java la reciben, esto es, “servicio xxxx” donde servicio indica el servicio que se estaba monitoreando y “xxxx” es la cantidad de bytes de esa muestra.

Por ejemplo: en el archivo “data/ip0” se muestran los datos del primer IP a monitorear, donde cada entrada del archivo corresponde a la cantidad de bytes que traficó esa dirección IP en el lapso de tiempo especificado por el “tiempo entre muestras” que por defecto es de cinco segundos.

Contenido del archivo “data/ip0”:

```
ip0 0
ip0 737
ip0 0
ip0 764
ip0 0
ip0 827
ip0 2224
ip0 2546
ip0 3846
ip0 4628
ip0 0
ip0 1762
ip0 2394
ip0 1754
ip0 1968
```

Esto se puede ingresar a cualquier programa del tipo estadístico para su análisis en mayor profundidad.

9.- Trabajo Futuro

El programa Netgraph ha finalizado su primera etapa, con la investigación, desarrollo e implementación de la versión inicial de Netgraph. Quedan todavía algunas características que se desea que Netgraph tenga para ser una herramienta aún más útil y poderosa.

A continuación se presentan algunas de estas características deseables. A medida que Netgraph empiece a ser ocupado en situaciones reales, se descubrirán nuevas características o modificaciones para aumentar su potencialidad.

- Versiones para otras plataformas: En el campo donde se aplicará Netgraph, es necesario contar con las versiones de Netgraph para varias plataformas de hardware como las estaciones de trabajo UNIX Hewlett-Packard (HPUX), Linux (UNIX para PC), Silicon Graphics (SGI), IBM RS/6000 (AIX) y Windows NT.

- Agregar monitoreo para Ethernet “Novell RAW” e “IEEE 802.3 SNAP”. Esta característica no es muy requerida, pero si es importante para poder completar todas las versiones de Ethernet para su posible monitoreo.

- Separar el tráfico de una IP por servicios: Esta característica es relativamente compleja pero muy útil cuando se desea saber exactamente qué servicios está ocupando una dirección IP en particular.

Estas características no son imposibles de agregar a Netgraph 1.0 para formar la versión de Netgraph 2.0. La primera de ellas es la más complicada ya que debe adaptar varias sentencias del lenguaje C a su equivalente en las otras máquinas. En el caso de las estaciones UNIX, esto es simple, pero para portar este programa a Windows NT se requiere de un estudio más acabado, ya que se ocuparon muchas de las cualidades de UNIX que Windows NT no tiene. Un ejemplo claro de esto es la ausencia de archivos tipo FIFO en Windows NT versión 4.0; si en futuras versiones esta opción se incluye, entonces será más fácil portar Netgraph.

Cabe destacar de que aún cuando falten estas tres características, Netgraph es único en su género y agregar estas características sólo aumenta su potencial para realizar un monitoreo más completo.

10.- Proyecciones Comerciales

Las proyecciones comerciales de Netgraph son buenas. En varias ocasiones se ha tenido la oportunidad de mostrar Netgraph a empresarios del área de Redes y sus opiniones han servido para obtener tres conclusiones fundamentales.

La primera de ellas es que no se ha visto un programa con características similares que pudiese entregar detalle en tiempo real de lo que sucede en una Red de Area Local Ethernet. Esto es importante, porque la mayor parte de la información que se obtiene es histórica y estadística.

La segunda observación es la necesidad de que Netgraph se porte a otras plataformas de hardware dado que no todas las empresas poseen una estación de trabajo SUN para poder realizar funciones de monitoreo de redes. En la pequeña y mediana empresa existe un dominio casi total de computadores tipo IBM PC compatibles con los más variados sistemas operativos: Novell, DOS, OS/2, SCO UNIX, Windows y Windows NT. En las empresas grandes existe un predominio de PC y servidores de tipo IBM (VM, MVS, AS/400, UNIX), SUN, HP y Windows NT.

Netgraph, por sus características, no está dirigido a la pequeña y mediana empresa. La razón fundamental es que no tienen la necesidad de monitorear su tráfico dado que no es muy grande en el presente momento. El área donde Netgraph tiene cabida es el la gran

empresa (control, monitoreo y estadística) y en las Universidades, donde el desarrollo de nuevas aplicaciones requiere de un monitor adaptable a las necesidades.

La última observación es que Netgraph debiese tener una versión para Linux y Windows NT para poder ser instalados en Notebooks. De esta manera Netgraph se puede "mover" entre las redes realizando monitoreo desde varias partes de la gran red de la empresa o de la Universidad. Cabe destacar de que existen pocas redes aisladas y por ende es necesario realizar el monitoreo de cada una de ellas. Esto es posible con Netgraph dado que se puede ejecutar en forma remota pudiendo así tener múltiples Netgraph en redes distintas y el Administrador puede conectarse a cada uno de ellos cuando desee.

11.- Conclusiones

El desarrollar un monitor de tráfico en tiempo real constituye un gran desafío personal y profesional dado que se debe empezar casi desde cero, teniendo, en ocasiones, muchas formas de lograr el objetivo debiendo escoger sin tener precedentes o modelos de otros monitores.

Este desafío significó investigar a fondo un protocolo (Ethernet) tan trastocado por la historia y estudiar los protocolos de capas superiores que ocupan Ethernet, que a su vez han ido creciendo y modificándose según las necesidades y no según un modelo estructurado como el modelo OSI de la ISO.

La idea nace fundamentalmente de la experiencia propia y de la de otros administradores de red, de la necesidad de contar con una herramienta de fácil manejo que pudiese mostrar los aspectos más exactos de la red local que se monitorea, de poder detectar focos de congestión, de mostrar múltiples servicios en tiempo real al mismo tiempo, pudiendo así comparar cualitativamente y cuantitativamente la ocupación de los servicios de una red.

Netgraph visualiza el tráfico de una red Ethernet permitiendo detectar y asistir en la solución de problemas práctico de la administración de redes, entre ellos: la congestión, la supervisión, monitoreo de servicios; todos ellos en tiempo real ayudando a que el

administrador pueda tomar decisiones en el momento que se produzca una situación anormal.

Una de las ventajas de Netgraph es que es flexible y configurable en forma fácil por el usuario, siendo muy útil tanto para el monitoreo constante, obtención de datos para luego analizarlos con programas estadísticos especializados (reporte histórico), como para monitorear nuevos servicios en desarrollo.

De la comparación hecha en el Capítulo 6, Netgraph se destaca por ser una herramienta complementaria a las ya existentes constituyendo un nuevo ángulo de acercamiento para combatir los problemas usuales de redes Ethernet y en particular redes Internet.

Una de las ventajas de Netgraph es que permite monitorear los servicios más estándares del momento y los servicios que aún no se fabrican, por lo que la obsolescencia de Netgraph no ocurre tan rápido como otros programas. Así, Netgraph se adapta a las necesidades del usuario de ese momento.

Es importante ir agregando nuevas características a Netgraph para mantenerla como una herramienta única y poderosa en el monitoreo de redes. Por ello, este software debe ser conocido ampliamente por la comunidad para recibir aportes y generar una segunda versión con las sugerencias de los usuarios.

11.- Bibliografía

[1] What is Ethernet?

http://seamonkey.ed.asu.edu/~alex/teaching/network/ethernet.html

[2] Ethernet Frequently Asked Questions (FAQ)

http://www.uga.edu/~ucns/lans/docs/ethernet.faq

[3] What is Ethernet?

http://cord.iupui.edu/~adspiece/ethernet.html

[4] Ethernet Frame Formats

http://www.optimized.com/tech_cmp/enformat.html

[5] Ethernet Frame Types: Don Provan's Definitive Answer

http://www.dataflux.com.mx/novell/faq/nvfaq-1.html

[6] Ethernet and IEEE 802.3 Frame Formats

http://web.syr.edu/~jmwobus/comfaqs/faq-ethernet-format

[7] Ethernet Type Codes

http://www.cisco.com/univercd/data/doc/software/9_1/rpc_r/19172.htm

RFC 1700, páginas 168 a 171

[8] SNMP

http://www.concentric.net/~tkvallil/snmp.html

[9] NOCOL

ftp://ftp.navya.com/pub/vikas/nocol.tar.Z

[10] Netman

ftp://ftp.cs.curtin.edu.au/pub/netman

[11] HTML

http://www.dcsc.utfsm.cl/~maraboli/html-manual/

[12] Netscape's Javascript Handbook

http:///www.netscape.com/eng/mozilla/3.0/handbook/javascript/index.html

[13] Java Home Page

http://java.sun.com

[14] Tcpdump

ftp://ftp.ee.lbl.gov/tcpdump.tar.Z

[15] Financial PortFolio (Stock Demo)

http://www.javasoft.com:80/applets/applets/StockDemo/details.html

[16] Netscape Web Browser

http://home.netscape.com

[17] HP Openview

http://www.hp.com

[18] Netguardian

http://master.di.fc.ul.pt/software/netguard.html

Anexo Nº 1

Formatos de Paquete de las cuatro versiones Ethernet

A continuación se muestra un esquema con los cuatro formatos de paquete existentes con los largos correspondientes [5][6]:

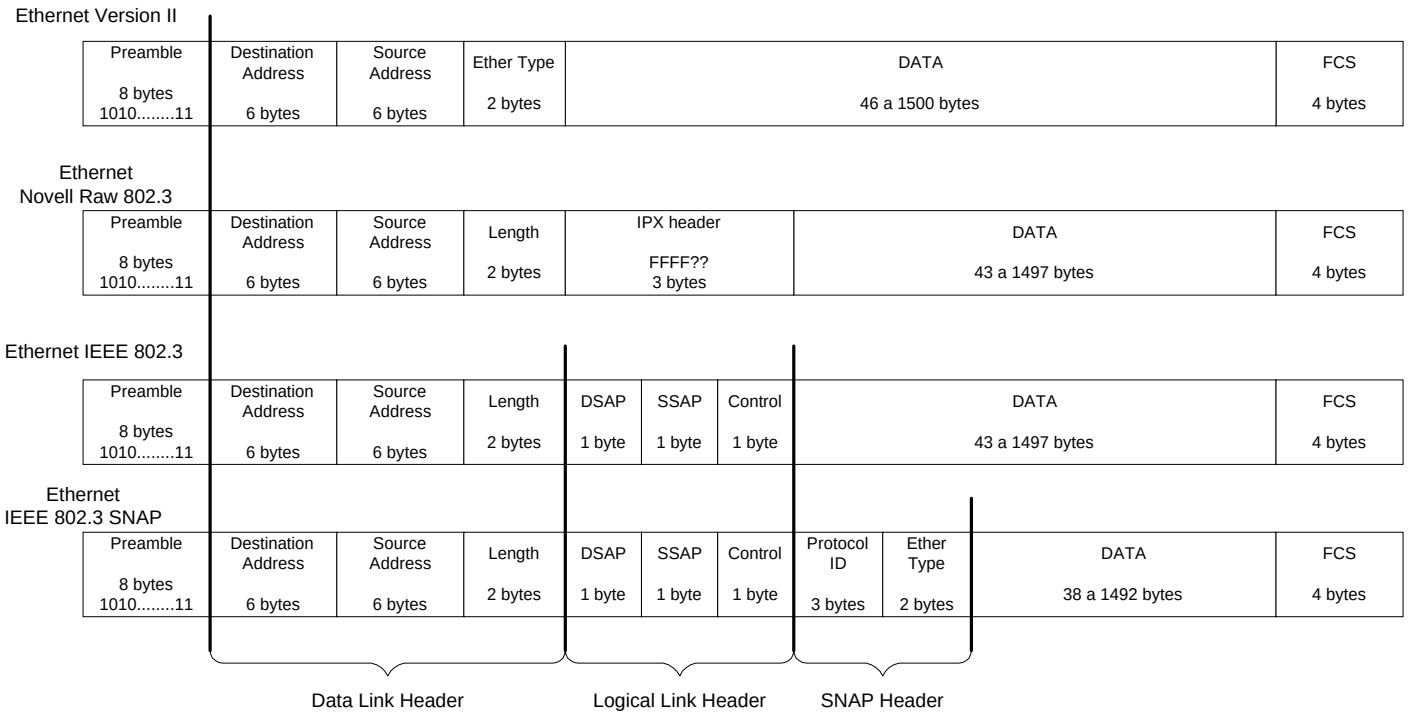


Figura Nº 37 Diagrama de las cuatro versiones de Ethernet

1.- Preámbulo (Preamble):

Independientemente del formato de paquete que sea, la sincronización digital entre las distintas tarjetas Ethernet debe ser la misma, por lo que este campo contiene 62 bits '1' y '0' en forma alternada finalizando con 2 bits '1' (en total 8 bytes), permitiendo ajustar los tiempos en ambas tarjetas (computadores o equipos de comunicaciones) para tener una transmisión digital sincronizada.

2.- Data Link Header:

2.1.- Dirección de Destino (Destination Address):

Corresponde a la dirección Ethernet (6 bytes) de la tarjeta Ethernet de destino del paquete a transmitir. Si esta dirección se compone enteramente de '1', entonces significa que es un mensaje de Broadcast, vale decir, un mensaje para todas las estaciones de la red local. Los primeros 3 bytes de esta dirección están normados por la IEEE y a cada fabricante de tarjetas Ethernet le corresponde un único trío.

2.2.- Dirección de Fuente (Source Address):

Corresponde a la dirección Ethernet (6 bytes) de la tarjeta Ethernet que envía el paquete a transmitir. Los primeros 3 bytes de esta dirección están normados por la IEEE y a cada fabricante de tarjetas Ethernet le corresponde un único trío, por lo que se puede saber "cual" es la maquina que envía el paquete.

Toda Dirección Ethernet (fuente o destino) es única en el mundo, excluyendo la posibilidad de suplantación o de errores en una red LAN.

2.3.- Ether Type (sólo Versión II):

Este campo indica el tipo de protocolo (superior) que está ocupando el formato de paquete Ethernet Versión II. En otras palabras, diferencia los distintos tipos de protocolos de capas superiores que puedan ocupar Ethernet. Por ejemplo: IP tiene un Ether-Type de valor 0x0800, ARP tiene valor 0x0806, IPX tiene 0x8137. Todos los valores son asignados por la IEEE en el RFC1700 y poseen valores mayores de 0x05DC (1500 decimal).

2.4.- Length (largo del paquete):

Este campo es ocupado por IEEE 802.3, IEEE 802.3 SNAP y Novell Raw e indica el largo del paquete, en cuanto a la sumatoria de los campos del Data Link Header y la Data. Por ende, este campo tiene los valores extremos de:

Mínimo: 6 + 6 + 2 + 46 = 60 bytes

Máximo: 6 + 6 + 2 + 1500 = 1514 bytes

NOTA:

Lo anterior muestra claramente que el Ether Type empieza desde el valor de 1500 (0x05DC) (oficialmente, pero en la práctica, empieza desde 0x0600 o 1536), y el Length termina en 1514 máximo, permitiendo que las versiones de Ethernet no se confundan y puedan ser utilizados al mismo tiempo en la misma red LAN.

3.- Logical Link Header (proviene de la IEEE 802.2)

La idea de este LLC es el de proveer de más información a las capas superiores indicando en qué buffer de memoria se coloca la información recibida por la tarjeta.

3.1.- DSAP (Destination Service Access point):

Este campo corresponde a un puntero en el buffer de memoria de la estación receptora del paquete, el cual, es utilizado por la "tarjeta receptora" para saber en cual buffer colocar esta información. Esto es particularmente útil en situaciones donde un usuario está usando múltiples protocolos. En el caso de ser un paquete del tipo IEEE 802.3 SNAP, este campo contiene el valor 0xAA.

3.2.- SSAP (Source Service Access point):

Este campo es análogo al DSAP, pero se refiere a la estación emisora. En el caso de ser un paquete del tipo IEEE 802.3 SNAP, este campo contiene el valor 0xAA.

3.3.- Control Byte:

Este byte indica el tipo de LLC (Logical Link Header).

4.- SNAP (SubNetwork Access Protocol) Header.

4.1.- Protocol ID (Vendor Code):

Este campo de 3 bytes indica el código del fabricante de la tarjeta de red (NIC), generalmente es igual a los 3 primeros bytes del Source Address y en otros casos es igual a cero.

4.2.- Ether Type (Local Code):

Este campo de 2 bytes corresponde al Ether-Type del paquete. Aquí es donde se aplica la "compatibilidad" entre el estándar IEEE 802.3 SNAP y Versión II.

5.- DATA:

Es en este campo donde reside la información transmitida y que generalmente consiste de headers de capas superiores (TCP/IP, IPX, etc.) y la data verdadera.

6.- FCS (Frame Check Sequence o CRC):

Este campo de 4 bytes contiene un checksum que permite revisar la integridad del paquete recibido para ser entregado a las capas superiores o descartado.

NOTA:

En el caso del protocolo IPX, éste puede ser transmitido por las 4 versiones de Ethernet, con los valores de campos indicados a continuación:

Versión II

Ether-Type = 0x8137

Novell Raw

Este protocolo sólo puede transmitir paquetes tipo IPX, pero para distinguirse de la norma IEEE 802.2 los primeros 2 bytes de la DATA deben ser 0xFFFF. Este es el protocolo por defecto de las redes Novell hasta antes de la Versión Novell 4.0.

IEEE 802.3

DSAP=0xE0, SSAP=0xE0, Control=0x03.

Este corresponde al tipo de paquetes por defecto para una Novell 4.0.

IEEE 802.3 SNAP

SNAP Protocol ID = 0x000000, SNAP Ether-Type=0x8137. Este Protocolo casi nunca es ocupado por IPX, es muy usando por Apple-Talk.

Por otro lado, en el caso del Protocolo TCP/IP, sólo es posible ocupar las versiones de Ethernet II y la IEEE 802.3 SNAP.

Anexo Nº 2

Códigos Ether Type [7]

Ethernet		Exp. Ethernet		Description	References
decimal	Hex	decimal	octal		
000	**0000-05DC**	**-**	**-**	**IEEE802.3 Length Field**	**[XEROX]**
257	0101-01FF	-	-	Experimental	[XEROX]
512	0200	512	1000	XEROX PUP (see 0A00)	[8,XEROX]
513	0201	-	-	PUP Addr Trans (see 0A01)	[XEROX]
	0400			Nixdorf	[XEROX]
1536	**0600**	**1536**	**3000**	**XEROX NS IDP**	**[133,XEROX]**
	0660			DLOG	[XEROX]
	0661			DLOG	[XEROX]
2048	**0800**	**513**	**1001**	**Internet IP (IPv4)**	**[105,JBP]**
2049	0801	-	-	X.75 Internet	[XEROX]
2050	0802	-	-	NBS Internet	[XEROX]
2051	0803	-	-	ECMA Internet	[XEROX]
2052	0804	-	-	Chaosnet	[XEROX]
2053	0805	-	-	X.25 Level 3	[XEROX]
2054	**0806**	**-**	**-**	**ARP**	**[88,JBP]**
2055	0807	-	-	XNS Compatability	[XEROX]
2076	081C	-	-	Symbolics Private	[DCP1]
2184	0888-088A	-	-	Xyplex	[XEROX]
2304	0900	-	-	Ungermann-Bass net debugr	[XEROX]
2560	0A00	-	-	Xerox IEEE802.3 PUP	[XEROX]
2561	0A01	-	-	PUP Addr Trans	[XEROX]
2989	0BAD	-	-	Banyan Systems	[XEROX]
4096	1000	-	-	Berkeley Trailer nego	[XEROX]
4097	1001-100F	-	-	Berkeley Trailer encap/IP	[XEROX]
5632	1600	-	-	Valid Systems	[XEROX]
16962	4242	-	-	PCS Basic Block Protocol	[XEROX]
21000	5208	-	-	BBN Simnet	[XEROX]
24576	6000	-	-	DEC Unassigned (Exp.)	[XEROX]
24577	6001	-	-	DEC MOP Dump/Load	[XEROX]
24578	6002	-	-	DEC MOP Remote Console	[XEROX]
24579	6003	-	-	DEC DECNET Phase IV Route	[XEROX]
24580	6004	-	-	DEC LAT	[XEROX]
24581	6005	-	-	DEC Diagnostic Protocol	[XEROX]
24582	6006	-	-	DEC Customer Protocol	[XEROX]
24583	6007	-	-	DEC LAVC, SCA	[XEROX]
24584	6008-6009	-	-	DEC Unassigned	[XEROX]
24586	6010-6014	-	-	3Com Corporation	[XEROX]
28672	7000	-	-	Ungermann-Bass download	[XEROX]
28674	7002	-	-	Ungermann-Bass dia/loop	[XEROX]

28704	7020-7029	-	-	LRT	[XEROX]
28720	7030	-	-	Proteon	[XEROX]
28724	7034	-	-	Cabletron	[XEROX]
32771	8003	-	-	Cronus VLN	[131,DT15]
32772	8004	-	-	Cronus Direct	[131,DT15]
32773	8005	-	-	HP Probe	[XEROX]
32774	8006	-	-	Nestar	[XEROX]
32776	8008	-	-	AT&T	[XEROX]
32784	8010	-	-	Excelan	[XEROX]
32787	8013	-	-	SGI diagnostics	[AXC]
32788	8014	-	-	SGI network games	[AXC]
32789	8015	-	-	SGI reserved	[AXC]
32790	8016	-	-	SGI bounce server	[AXC]
32793	8019	-	-	Apollo Computers	[XEROX]
32815	802E	-	-	Tymshare	[XEROX]
32816	802F	-	-	Tigan, Inc.	[XEROX]
32821	**8035**	**-**	**-**	**Reverse ARP**	**[48,JXM]**
32822	8036	-	-	Aeonic Systems	[XEROX]
32824	8038	-	-	DEC LANBridge	[XEROX]
32825	8039-803C	-	-	DEC Unassigned	[XEROX]
32829	803D	-	-	DEC Ethernet Encryption	[XEROX]
32830	803E	-	-	DEC Unassigned	[XEROX]
32831	803F	-	-	DEC LAN Traffic Monitor	[XEROX]
32832	8040-8042	-	-	DEC Unassigned	[XEROX]
32836	8044	-	-	Planning Research Corp.	[XEROX]
32838	8046	-	-	AT&T	[XEROX]
32839	8047	-	-	AT&T	[XEROX]
32841	8049	-	-	ExperData	[XEROX]
32859	805B	-	-	Stanford V Kernel exp.	[XEROX]
32860	805C	-	-	Stanford V Kernel prod.	[XEROX]
32861	805D	-	-	Evans & Sutherland	[XEROX]
32864	8060	-	-	Little Machines	[XEROX]
32866	8062	-	-	Counterpoint Computers	[XEROX]
32869	8065	-	-	Univ. of Mass. @ Amherst	[XEROX]
32870	8066	-	-	Univ. of Mass. @ Amherst	[XEROX]
32871	8067	-	-	Veeco Integrated Auto.	[XEROX]
32872	8068	-	-	General Dynamics	[XEROX]
32873	8069	-	-	AT&T	[XEROX]
32874	806A	-	-	Autophon	[XEROX]
32876	806C	-	-	ComDesign	[XEROX]
32877	806D	-	-	Computgraphic Corp.	[XEROX]
32878	806E-8077	-	-	Landmark Graphics Corp.	[XEROX]
32890	807A	-	-	Matra	[XEROX]
32891	807B	-	-	Dansk Data Elektronik	[XEROX]
32892	807C	-	-	Merit Internodal	[HWB]
32893	807D-807F	-	-	Vitalink Communications	[XEROX]
32896	8080	-	-	Vitalink TransLAN III	[XEROX]
32897	8081-8083	-	-	Counterpoint Computers	[XEROX]
32923	809B	-	-	Appletalk	[XEROX]

32924	809C-809E	-	-	Datability	[XEROX]
32927	809F	-	-	Spider Systems Ltd.	[XEROX]
32931	80A3	-	-	Nixdorf Computers	[XEROX]
32932	80A4-80B3	-	-	Siemens Gammasonics Inc.	[XEROX]
32960	80C0-80C3	-	-	DCA Data Exchange Cluster	[XEROX]
	80C4			Banyan Systems	[XEROX]
	80C5			Banyan Systems	[XEROX]
32966	80C6	-	-	Pacer Software	[XEROX]
32967	80C7	-	-	Applitek Corporation	[XEROX]
32968	80C8-80CC	-	-	Intergraph Corporation	[XEROX]
32973	80CD-80CE	-	-	Harris Corporation	[XEROX]
32975	80CF-80D2	-	-	Taylor Instrument	[XEROX]
32979	80D3-80D4	-	-	Rosemount Corporation	[XEROX]
32981	80D5	-	-	IBM SNA Service on Ether	[XEROX]
32989	80DD	-	-	Varian Associates	[XEROX]
32990	80DE-80DF	-	-	Integrated Solutions TRFS	[XEROX]
32992	80E0-80E3	-	-	Allen-Bradley	[XEROX]
32996	80E4-80F0	-	-	Datability	[XEROX]
33010	80F2	-	-	Retix	[XEROX]
33011	80F3	-	-	AppleTalk AARP (Kinetics)	[XEROX]
33012	80F4-80F5	-	-	Kinetics	[XEROX]
33015	80F7	-	-	Apollo Computer	[XEROX]
33023	80FF-8103	-	-	Wellfleet Communications	[XEROX]
33031	8107-8109	-	-	Symbolics Private	[XEROX]
33072	8130	-	-	Hayes Microcomputers	[XEROX]
33073	8131	-	-	VG Laboratory Systems	[XEROX]
	8132-8136			Bridge Communications	[XEROX]
33079	**8137-8138**	**-**	**-**	**Novell, Inc.**	**[XEROX]**
33081	8139-813D	-	-	KTI	[XEROX]
	8148			Logicraft	[XEROX]
	8149			Network Computing Devices	[XEROX]
	814A			Alpha Micro	[XEROX]
33100	**814C**	**-**	**-**	**SNMP**	**[JKR1]**
	814D			BIIN	[XEROX]
	814E			BIIN	[XEROX]
	814F			Technically Elite Concept	[XEROX]
	8150			Rational Corp	[XEROX]
	8151-8153			Qualcomm	[XEROX]
	815C-815E			Computer Protocol Pty Ltd	[XEROX]
	8164-8166			Charles River Data System	[XEROX]
	817D-818C			Protocol Engines	[XEROX]
	818D			Motorola Computer	[XEROX]
	819A-81A3			Qualcomm	[XEROX]
	81A4			ARAI Bunkichi	[XEROX]
	81A5-81AE			RAD Network Devices	[XEROX]
	81B7-81B9			Xyplex	[XEROX]
	81CC-81D5			Apricot Computers	[XEROX]
	81D6-81DD			Artisoft	[XEROX]
	81E6-81EF			Polygon	[XEROX]

	81F0-81F2			Comsat Labs	[XEROX]
	81F3-81F5			SAIC	[XEROX]
	81F6-81F8			VG Analytical	[XEROX]
	8203-8205			Quantum Software	[XEROX]
	8221-8222			Ascom Banking Systems	[XEROX]
	823E-8240			Advanced Encryption Syste	[XEROX]
	827F-8282			Athena Programming	[XEROX]
	8263-826A			Charles River Data System	[XEROX]
	829A-829B			Inst Ind Info Tech	[XEROX]
	829C-82AB			Taurus Controls	[XEROX]
	82AC-8693			Walker Richer & Quinn	[XEROX]
	8694-869D			Idea Courier	[XEROX]
	869E-86A1			Computer Network Tech	[XEROX]
	86A3-86AC			Gateway Communications	[XEROX]
	86DB			SECTRA	[XEROX]
	86DE			Delta Controls	[XEROX]
34543	86DF	-	-	ATOMIC	[JBP]
	86E0-86EF			Landis & Gyr Powers	[XEROX]
	8700-8710			Motorola	[XEROX]
	8A96-8A97			Invisible Software	[XEROX]
36864	**9000**	**-**	**-**	**Loopback**	**[XEROX]**
36865	9001	-	-	3Com(Bridge) XNS Sys Mgmt	[XEROX]
36866	9002	-	-	3Com(Bridge) TCP-IP Sys	[XEROX]
36867	9003	-	-	3Com(Bridge) loop detect	[XEROX]
65280	FF00	-	-	BBN VITAL-LanBridge cache	[XEROX]
	FF00-FF0F			ISC Bunker Ramo	[XEROX]

Anexo Nº 3

Breve Descripción de HP Openview y NetGuard

1.- HP Openview. (Windows 3.11, 95 y NT) [18]

Este monitor permite realizar las siguientes tareas:

- Integra 13 aplicaciones para el manejo de PC, Servidores, nodos de red, Impresoras, UPS y manejo de tráfico.

Figura Nº 38 Monitor de UPS e Impresora

- Descubre en 30 minutos la configuración de la red.

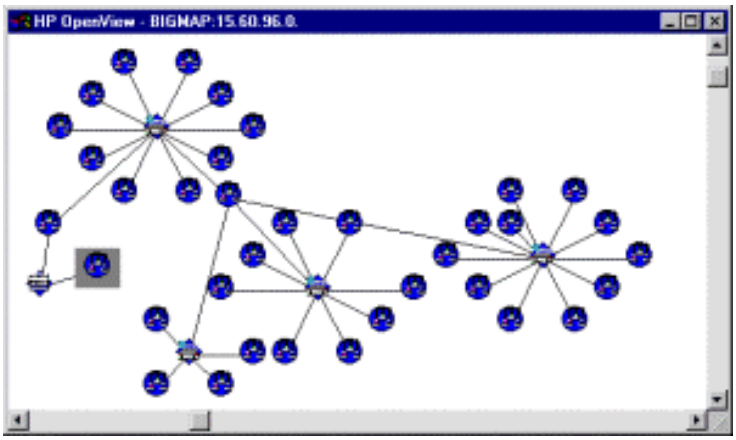


Figura Nº 39 Autodiscovery

- Trabaja con los protocolos IP e IPX

El manejo de tráfico se refiere a la activación de alarmas vía SNMP cuando el tráfico sobrepasa ciertos umbrales, etc. además de las funciones de estadística y monitoreo de ciertos paquetes para detección de problemas.

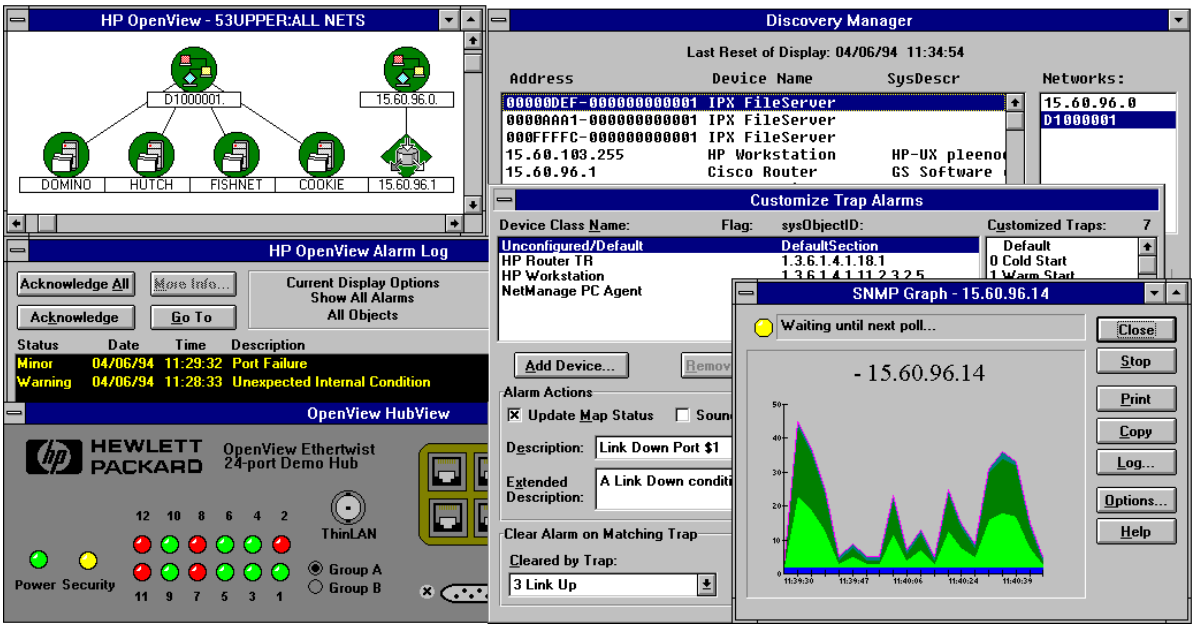


Figura Nº 40 Visión Global del SNMP

1.- NetGuardian. (Windows 3.11, 95 y NT)

Este programa incluye un Manager, un Agent y un graficador.

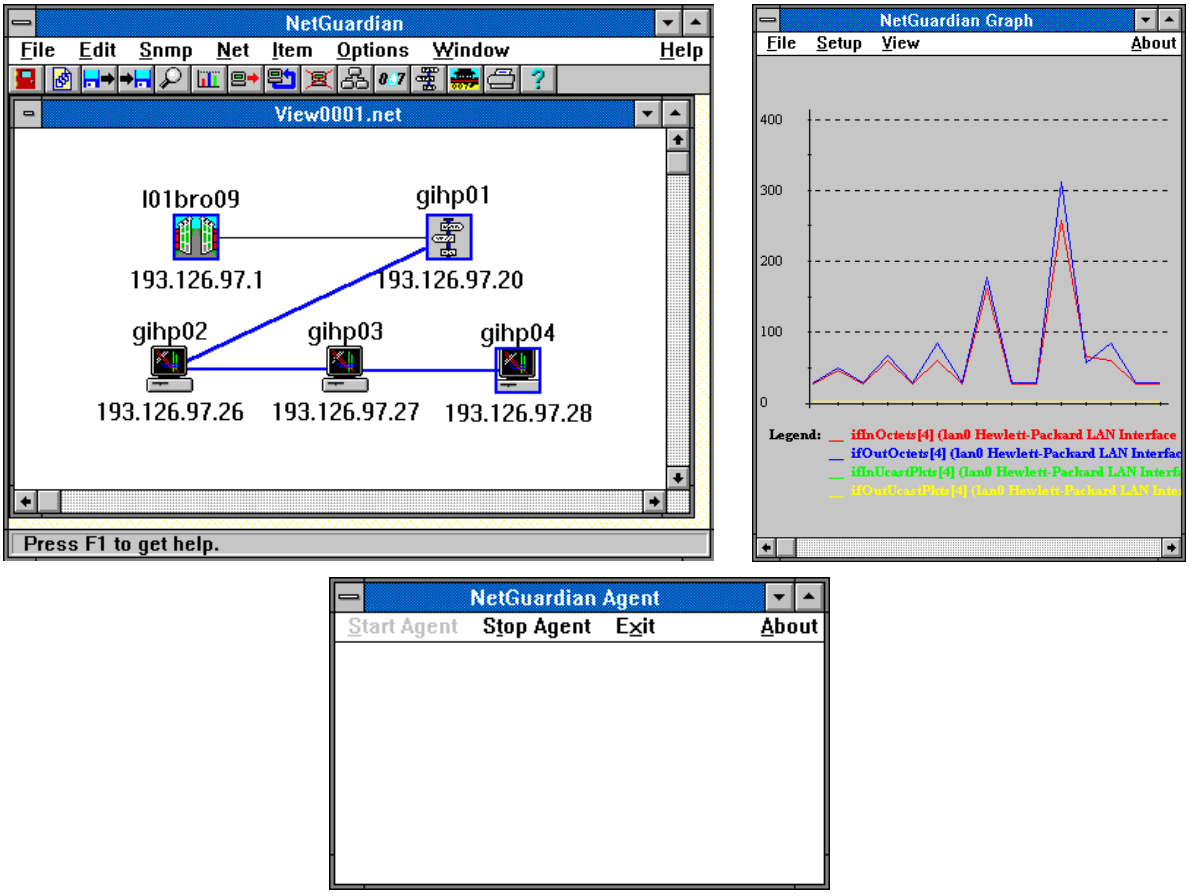


Figura Nº 41 Visión Global del NetGuardian

Fue desarrollado en la Facultad de Ciencias de la Universidad de Lisboa [19] y permite realizar funciones SNMP simples, ya que se utiliza para monitorear actividades de PC (el Agente es para Windows). Aunque se puede combinar con otros agentes y Managers, el programa todavía tiene algunas funciones que no están depuradas en su versión 2 beta 3.

Anexo Nº 4

Listado del programa Configure.c

```
#include "netgraph.h"

char date[50], pwd[50];

void get_date()
{
  FILE *in;

  system("/usr/bin/date > config/date");

  if ( (in = fopen( "config/date", "r") ) == NULL)
      {
       printf("No se puede abrir archivo config/date\n");
       exit(1);
      }

  fgets( date, 50, in );
}

void get_pwd()
{
  FILE *in;

  system("/usr/bin/pwd > config/pwd");

  if ( (in = fopen( "config/pwd", "r") ) == NULL)
      {
       printf("No se puede abrir archivo ./config/pwd\n");
       exit(1);
      }

  fgets( pwd, 50, in );

  pwd[ strlen(pwd) - 1] = 0;
}

void save_www_start_stop()
{

  FILE *out;

  if ( (out = fopen( "./www.start", "w+") ) == NULL)
      {
       printf("No se puede crear/abrir archivo ./www.start\n");
       exit(1);
      }

  get_date();

  fprintf( out, "#!/bin/sh -fh\n#Creado con fecha : %s#\n", date );
  fprintf( out, "%s/httpd/httpd -d %s/httpd -f %s/httpd/conf/httpd.conf\n", pwd, pwd, pwd);

  printf("\nEscribiendo ./www.start\n");
  system("chmod 744 www.start");

  if ( (out = fopen( "./www.stop", "w+") ) == NULL)
      {
       printf("No se puede crear/abrir archivo ./www.stop\n");
       exit(1);
      }

  fprintf( out, "#!/bin/sh -fh\n#Creado con fecha : %s#\n#SIGINT = 2 en sh\n\n", date );
  fprintf( out, "httppid=`/bin/cat %s/httpd/logs/httpd.pid`\n", pwd );
  fprintf( out, "kill -2 \"$httppid\"\n");

  printf("Escribiendo ./www.stop\n");
```

```
  system("chmod 744 www.stop");
}

void save_httpd_conf()
{
  char ss[40];

  sprintf(ss, "printf \"%s\n\" \"ServerRoot %s/httpd\" > config/server", "%s", pwd );
  system( ss );
  system( "cd config; cat httpd.conf1 server httpd.conf2 >httpd.conf ; cp httpd.conf
../httpd/conf/");
  printf("Escribiendo ./httpd/conf/httpd.conf\n");

  sprintf(ss, "printf \"%s\n\" \"<Directory %s/httpd/cgi-bin>\" > config/acc1", "%s", pwd );
  system( ss );
  sprintf(ss, "printf \"%s\n\" \"<Directory %s/httpd/htdocs>\" > config/acc2", "%s", pwd );
  system( ss );
  system( "cd config ; cat access.conf1 acc1 access.conf2 acc2 access.conf3 >access.conf; cp
access.conf ../httpd/conf/");
  printf("Escribiendo ./httpd/conf/access.conf\n");

  sprintf(ss, "printf \"%s\n\" \"DocumentRoot %s/httpd/htdocs\" > config/srm1", "%s", pwd );
  system( ss );
  sprintf(ss, "printf \"%s\n\" \"Alias /icons/ %s/httpd/icons/\" > config/srm2", "%s", pwd );
  system( ss );
  sprintf(ss, "printf \"%s\n\" \"ScriptAlias /cgi-bin/ %s/httpd/cgi-bin/\" > config/srm3",
"%s", pwd );
  system( ss );
  system( "cd config ; cat srm.conf1 srm1 srm.conf2 srm2 srm.conf3 srm3 srm.conf4 > srm.conf;
cp srm.conf ../httpd/conf");
  printf("Escribiendo ./httpd/conf/srm.conf\n");
}


void save_html()
{
  char ss[150], host[20];

  gethostname( host, 20 );

  sprintf(ss, "printf \"%s\" \"<FORM NAME=myform ACTION=http://%s:8080/cgi-bin/step1
METHOD=GET> \" > config/index1", "%s", host );
  system( ss );

  system( "cd config ; cat index.html1 index1 index.html2 > index.html; cp index.html
../httpd/htdocs/config");
  printf("Escribiendo ./httpd/htdocs/config/index.html\n");
}

void htpasswd()
{
  char ss[150];

  sprintf(ss, "printf \"%s\" \"AuthUserFile %s/httpd/htdocs/.htpasswd \" > config/ht1", "%s",
pwd );
  system( ss );

  system( "cd config ; cat ht1 htaccess1 > .htaccess; cp .htaccess ../httpd/htdocs");
  printf("Escribiendo ./httpd/htdocs/.htaccess\n");

  printf("\nIngrese passwd para las Paginas Web (usuario=netweb)\n");
  system("httpd/support/htpasswd -c httpd/htdocs/.htpasswd netweb");
}

void main()
{
  printf("\n Configuracion del Netgraph v1.0\n\n");

  get_pwd();

  printf("\n Configurando Netgraph con el path actual ---> %s\n", pwd);

  save_www_start_stop();
  save_httpd_conf();
```

```
  save_html();
  htpasswd();
  printf("\n Configuracion Lista...\n\n");
}
```

Anexo Nº 5

Listado del programa netgraph.c

Listado del netgraph.h

```
#include <sys/time.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
#include <string.h>

int net_belong (char *ip, char *net, char *netmask );
int ip_belong (char *ipa, char *ipb );
```

Listado del netgraph.c

```
#include "netgraph.h"

#define BUFSIZE 250          /* tamano de linea de output de tcpdump */
#define MAX_PORT 20            /* tamano maximo de ports a monitorear */
#define MAX_PROTO 10           /* tamano maximo de protocols a monitorear */
#define MAX_SOCK 10            /* tamano maximo de sockets a monitorear */
#define MAX_NET  10            /* tamano maximo de nets a monitorear */
#define MAX_IP  10             /* tamano maximo de ip a monitorear */

FILE *ptr, *out, *debug;
FILE *port_fifo[MAX_PORT], *proto_fifo[MAX_PROTO], *sock_fifo[MAX_SOCK], *ip_fifo[MAX_IP],
*net_fifo[MAX_NET];
FILE *port_data[MAX_PORT], *proto_data[MAX_PROTO], *sock_data[MAX_SOCK], *ip_data[MAX_IP],
*net_data[MAX_NET];
int dflg = 0;

int len=0, sumlen=0, proto, port1, port2, win;
unsigned long tpo;
char ipp1[20], ipp2[20], info[50];

int num_port=0, num_proto=0, num_sock=0, num_ip=0, num_net=0;

unsigned int port_array[ MAX_PORT ][ 2 ];
unsigned int proto_array[ MAX_PROTO ][ 2 ];
unsigned int sock_array[ MAX_SOCK ][ 2 ];

struct net_struct {
  char net[20];
  char netmask[20];
  unsigned int netbyte;
};

struct net_struct netw[ MAX_NET ];

struct ip_struct {
  char ip[20];
  unsigned int ipbyte;
};

struct ip_struct ipip[ MAX_IP ];

void read_config()
{
```

```
  FILE *cfg;
  char str[100], comm[5]="", arg2[20]="", arg3[10]="" ;

  if ( (cfg=fopen("./netgraph.cfg", "r" )) == NULL)
    {
      printf("\n\n\tNo se encontro el archivo ./netgraph.cfg\n\n");
      exit(1);
    }

  while ( fgets( str, 100, cfg ) != NULL )
    {

      if ( str[0] == '#' )                /* eliminar comentarios si es que hay */
       continue;

      sscanf( str, "%s %s %s", comm, arg2, arg3 );

      if ( strstr( comm, "port" ) != NULL )          /* si es "port" entonces */
       {
         if ( num_port >= MAX_PORT )
           {
             printf("netgraph.cfg : ports > MAX_PORT ... ignorando comando %s\n", str );
             continue;
           }
         port_array[num_port++][0] = atoi(arg2);
         /* printf("port_array[%d][0] = %d\n", num_port - 1,  port_array[num_port-1][0]);
          */
       }
      else
       {

          if ( strstr( comm, "net" ) != NULL )       /* si es "net" entonces */
            {
              if ( num_net >= MAX_NET )
               continue;

              strcpy( netw[num_net].net , arg2 );
              strcpy( netw[num_net++].netmask , arg3 );

              /*  printf("netw[%d].net = %s\t\t", num_net - 1,  netw[num_net-1].net );
              printf("netw[%d].netmask = %s\n", num_net - 1,  netw[num_net-1].netmask );
              */
            }
          else
             {
              if ( strstr( comm, "win" ) != NULL )        /* si es "window" entonces */
                {
                  win = atoi(arg2);
                }
               else
                 {
                   if ( strstr( comm, "proto" ) != NULL )          /* si es "proto" entonces
*/
                     {
                       if ( num_proto >= MAX_PROTO )
                         {
                           printf("netgraph.cfg : proto > MAX_PROTO ... ignorando comando
%s\n", str );
                           continue;
                         }
                       sscanf( str, "%s %x %s", comm, &proto_array[num_proto++][0], arg3 );

                       // proto_array[num_proto++][0] = atoi(arg2);
                       //printf("proto_array[%d][0] = %x dec=%d \n", num_proto -
1,proto_array[num_proto-1][0],proto_array[num_proto-1][0]);

                     }
                   else
                     if ( strstr( comm, "sock" ) != NULL )          /* si es "sock" entonces
*/
                       {
                         if ( num_sock >= MAX_SOCK )
                           {
```

```
                            printf("netgraph.cfg : sock > MAX_SOCK ... ignorando comando
%s\n", str );
                            continue;
                          }
                        sock_array[num_sock++][0] = atoi(arg2);
                        /*printf("sock_array[%d][0] = %d\n", num_sock - 1,
sock_array[num_sock-1][0]);
                         */
                      }
                    else
                      if ( strstr( comm, "ip" ) != NULL )          /* si es "ip" entonces */
                        {
                          if ( num_ip >= MAX_IP )
                            {
                             printf("netgraph.cfg : ip > MAX_IP=10 ... ignorando comando
%s\n", str );
                             continue;
                            }
                          strcpy( ipip[num_ip++].ip , arg2 );

                          /* printf("ipip[%d].ip = %s\t\t", num_ip - 1,  ipip[num_ip-1].ip
);
                           */
                        }
                      else
                        printf("netgraph.cfg : comando no reconocido (ignorado) : %s \n",
str );

                }

             }
         }
    }
}

void check_lock()
{
  FILE *lock;

  if ( (lock = fopen ("./netgraph.lock", "r" ) ) != NULL )
    {
      printf("ya existe un archivo lock\n" );
      exit(0);
    }
  fclose( lock );
}

void save_pid_and_lock()
{
  FILE *ppid;
  int i;

  i = (int )getpid();

  /*  printf("Mi PID es %d\n", i );
   */

  if ( (ppid = fopen( "./netgraph.pid", "w+") ) == NULL)
      {
       printf("No se puede abrir archivo ./netgraph.pid\n");
       exit(1);
      }

  fprintf( ppid, "%d\n", i );

  fclose( ppid );

  if ( (ppid = fopen( "netgraph.lock", "w+") ) == NULL)
      {
       printf("No se puede abrir archivo ./netgraph.lock\n");
       exit(1);
      }

  fclose( ppid );
```

```
}

void init_var()
{
  int i;

  for (i = 0 ; i < num_port; i++ )
      port_array[i][1] = 0 ;

  for (i = 0 ; i < num_proto; i++ )
      proto_array[i][1] = 0 ;

  for (i = 0 ; i < num_sock; i++ )
      sock_array[i][1] = 0 ;

  for (i = 0 ; i < num_net; i++ )
      netw[i].netbyte = 0 ;

  for (i = 0 ; i < num_ip; i++ )
      ipip[i].ipbyte = 0 ;

}


void cleanup()
{
  int i;
  char tmp[50];

  fprintf(stderr, "\n\nCerrando archivos de captura...\n" );

  for ( i = 0 ; i < num_port; i++ )
    {
      fclose( port_fifo[i] );
      sprintf( tmp, "rm -f ./httpd/cgi-bin/port%d.fifo", i );
      system( tmp );
      sprintf( tmp, "rm -f ./httpd/cgi-bin/port%d", i );
      system( tmp );
      fclose( port_data[i] );
    }

  for ( i = 0 ; i < num_proto; i++ )
    {
      fclose( proto_fifo[i] );
      sprintf( tmp, "rm -f ./httpd/cgi-bin/proto%d.fifo", i );
      system( tmp );
      sprintf( tmp, "rm -f ./httpd/cgi-bin/proto%d", i );
      system( tmp );
      fclose( proto_data[i] );
    }

  for ( i = 0 ; i < num_sock; i++ )
    {
      fclose( sock_fifo[i] );
      sprintf( tmp, "rm -f ./httpd/cgi-bin/sock%d.fifo", i );
      system( tmp );
      sprintf( tmp, "rm -f ./httpd/cgi-bin/sock%d", i );
      system( tmp );
    }

  for (i = 0 ; i < num_net; i++ )
    {
      fclose( net_fifo[i] );
      sprintf( tmp, "rm -f ./httpd/cgi-bin/net%d.fifo", i );
      system( tmp );
      sprintf( tmp, "rm -f ./httpd/cgi-bin/net%d", i );
      system( tmp );
      fclose( net_data[i] );
    }

  for (i = 0 ; i < num_ip; i++ )
    {
      fclose( ip_fifo[i] );
      sprintf( tmp, "rm -f ./httpd/cgi-bin/ip%d.fifo", i );
```

```
      system( tmp );
      sprintf( tmp, "rm -f ./httpd/cgi-bin/ip%d", i );
      system( tmp );
      fclose( ip_data[i] );
    }

  fclose( out );
  fclose( ptr );

  system("rm -f ./netgraph.lock");
  system("rm -f ./netgraph.pid");

  exit(0);
}

void mk_fifos_cgi_data()
{
  int i;
  char tmp[100];

  for ( i = 0; i < num_port; i++ )
    {

      sprintf( tmp, "./httpd/cgi-bin/port%d.fifo", i );

      if ( mkfifo( tmp, 644 ) == -1 )
       {
       printf("No se pudo CREAR archivo fifo %s\n", tmp);
       cleanup();
       exit( 1 );
       }

      sprintf( tmp, "chown netweb:monitor ./httpd/cgi-bin/port%d.fifo", i );
      system ( tmp );

      sprintf( tmp, "chmod 604 ./httpd/cgi-bin/port%d.fifo", i );
      system ( tmp );

      sprintf( tmp, "./httpd/cgi-bin/port%d.fifo", i );

      if ( (port_fifo[i] = fopen( tmp , "a+" ) ) == NULL )
       {
         printf("No se pudo ABRIR archivo fifo %s\n", tmp);
       cleanup();
         exit(2);
       }

      setbuf( port_fifo[i], NULL );

      sprintf( tmp, "cp ./httpd/cgi-bin/applet-cgi ./httpd/cgi-bin/port%d", i );
      system ( tmp );

      sprintf( tmp, "./data/port%d.data", i );
      if ( (port_data[i] = fopen( tmp , "w+" ) ) == NULL )
       {
         printf("No se pudo ABRIR archivo data %s\n", tmp);
         cleanup();
         exit(2);
       }

      setbuf( port_data[i], NULL );
    }

  for ( i = 0; i < num_proto; i++ )
    {

      sprintf( tmp, "./httpd/cgi-bin/proto%d.fifo", i );

      if ( mkfifo( tmp, 644) == -1 )
       {
       printf("No se pudo CREAR archivo fifo %s\n", tmp);
       cleanup();
       exit( 1 );
       }
```

```
    sprintf( tmp, "chown netweb:monitor ./httpd/cgi-bin/proto%d.fifo", i );
    system ( tmp );

    sprintf( tmp, "chmod 604 ./httpd/cgi-bin/proto%d.fifo", i );
    system ( tmp );

    sprintf( tmp, "./httpd/cgi-bin/proto%d.fifo", i );

    if ( (proto_fifo[i] = fopen( tmp , "a+" ) ) == NULL )
     printf("No se pudo ABRIR archivo fifo %s\n", tmp);

    setbuf( proto_fifo[i], NULL );

    sprintf( tmp, "cp ./httpd/cgi-bin/applet-cgi ./httpd/cgi-bin/proto%d", i );
    system ( tmp );

    sprintf( tmp, "./data/proto%d.data", i );
    if ( (proto_data[i] = fopen( tmp , "w+" ) ) == NULL )
     {
       printf("No se pudo ABRIR archivo data %s\n", tmp);
       cleanup();
       exit(2);
     }
    setbuf( proto_data[i], NULL );
  }

for ( i = 0; i < num_sock; i++ )
  {
    sprintf( tmp, "./httpd/cgi-bin/sock%d.fifo", i );

    if ( mkfifo( tmp, 644) == -1 )
     {
     printf("No se pudo CREAR archivo fifo %s\n", tmp);
     cleanup();
     exit( 1 );
     }

    sprintf( tmp, "chown netweb:monitor ./httpd/cgi-bin/sock%d.fifo", i );
    system ( tmp );

    sprintf( tmp, "chmod 604 ./httpd/cgi-bin/sock%d.fifo", i );
    system ( tmp );

    sprintf( tmp, "./httpd/cgi-bin/sock%d.fifo", i );

    if ( (sock_fifo[i] = fopen( tmp , "a+" ) ) == NULL )
     printf("No se pudo ABRIR archivo fifo %s\n", tmp);

    setbuf( sock_fifo[i], NULL );

    sprintf( tmp, "cp ./httpd/cgi-bin/applet-cgi ./httpd/cgi-bin/sock%d", i );
    system ( tmp );

    sprintf( tmp, "./data/sock%d.data", i );
    if ( (sock_data[i] = fopen( tmp , "w+" ) ) == NULL )
     {
       printf("No se pudo ABRIR archivo data %s\n", tmp);
       cleanup();
       exit(2);
     }
    setbuf( sock_data[i], NULL );
  }

for ( i = 0; i < num_net; i++ )
  {
    sprintf( tmp, "./httpd/cgi-bin/net%d.fifo", i );

    if ( mkfifo( tmp, 644) == -1 )
     {
     printf("No se pudo CREAR archivo fifo %s\n", tmp);
     cleanup();
     exit( 1 );
     }
```

```
      sprintf( tmp, "chown netweb:monitor ./httpd/cgi-bin/net%d.fifo", i );
      system ( tmp );

      sprintf( tmp, "chmod 604 ./httpd/cgi-bin/net%d.fifo", i );
      system ( tmp );

      sprintf( tmp, "./httpd/cgi-bin/net%d.fifo", i );

      if ( (net_fifo[i] = fopen( tmp , "a+" ) ) == NULL )
       printf("No se pudo ABRIR archivo fifo %s\n", tmp);

      setbuf( net_fifo[i], NULL );

      sprintf( tmp, "cp ./httpd/cgi-bin/applet-cgi ./httpd/cgi-bin/net%d", i );
      system ( tmp );

      sprintf( tmp, "./data/net%d.data", i );
      if ( (net_data[i] = fopen( tmp , "w+" ) ) == NULL )
       {
         printf("No se pudo ABRIR archivo data %s\n", tmp);
         cleanup();
         exit(2);
       }
      setbuf( net_data[i], NULL );
    }

  for ( i = 0; i < num_ip; i++ )
    {
      sprintf( tmp, "./httpd/cgi-bin/ip%d.fifo", i );

      if ( mkfifo( tmp, 644) == -1 )
       {
       printf("No se pudo CREAR archivo ip-fifo %s\n", tmp);
       exit( 1 );
       }

      sprintf( tmp, "chown netweb:monitor ./httpd/cgi-bin/ip%d.fifo", i );
      system ( tmp );

      sprintf( tmp, "chmod 604 ./httpd/cgi-bin/ip%d.fifo", i );
      system ( tmp );

      sprintf( tmp, "./httpd/cgi-bin/ip%d.fifo", i );

      if ( (ip_fifo[i] = fopen( tmp , "a+" ) ) == NULL )
       printf("No se pudo ABRIR archivo ip-fifo %s\n", tmp);

      setbuf( ip_fifo[i], NULL );

      sprintf( tmp, "cp ./httpd/cgi-bin/applet-cgi ./httpd/cgi-bin/ip%d", i );
      system ( tmp );

      sprintf( tmp, "./data/ip%d.data", i );
      if ( (ip_data[i] = fopen( tmp , "w+" ) ) == NULL )
       {
         printf("No se pudo ABRIR archivo data %s\n", tmp);
         cleanup();
         exit(2);
       }
      setbuf( ip_data[i], NULL );
    }

}


void port_stat()
{

  int i;

  for ( i = 0; i < num_port; i++ )
    {
      if ( port1 == port_array[i][0] )
```

```
         port_array[i][1] += len;

      if ( port2 == port_array[i][0] )
         port_array[i][1] += len;
    }
}

void proto_stat()
{

  int i;

  //if ( proto != 0x800)
    //printf( "\ntpo=%lu len=%d proto=%x proto(d)=%d port1=%d port2=%d\n", tpo, len, proto,
proto, port1, port2 );

  for ( i = 0; i < num_proto; i++ )                /* variable proto es HEXADECIMAL,
Protocolo 0800 es proto=2048 */
    {
        //if ( proto != 0x800)
          //printf("testing proto=%x proto(d)=%d con proto_array[%d][0]=%d
proto_array[%d][0]=%x hexa\n", proto, proto, i, proto_array[i][0], i, proto_array[i][0]);

      if ( (proto<1537 && proto_array[i][0]==0 ) || proto==proto_array[i][0] )    /* 0x600
=1536 */
       {
         proto_array[i][1] += len;
         //printf( "tpo=%lu len=%d proto=%x proto(d)=%d port1=%d port2=%d\n", tpo, len,
proto, proto, port1, port2 );
         //printf( "coincide proto=%x con proto_array[i][0]=%d\n", proto,
proto_array[i][0]);
       }
    }
}

void sock_stat()
{

  int i;

  if ( strstr( info, "ipx" ) == NULL )
    return;

  for ( i = 0; i < num_sock; i++ )
    {
      if ( port1 == sock_array[i][0] )
         sock_array[i][1] += len;

      if ( port2 == sock_array[i][0] )
         sock_array[i][1] += len;
    }
}

void net_stat()
{

  int i;

  for ( i = 0; i < num_net; i++ )
    {
      if ( net_belong( ipp1, netw[i].net, netw[i].netmask ) == 1 )
         netw[i].netbyte += len;

      if ( net_belong( ipp2, netw[i].net, netw[i].netmask ) == 1 )
         netw[i].netbyte += len;
    }

}

void ip_stat()
{

  int i;
```

```
  for ( i = 0; i < num_ip; i++ )
    {
      if ( ip_belong( ipp1, ipip[i].ip ) == 1 )
         ipip[i].ipbyte += len;

      if ( ip_belong( ipp2, ipip[i].ip ) == 1 )
         ipip[i].ipbyte += len;
    }

}

void statistica ()
{
  int i;

  for (i = 0 ; i < num_port; i++ )
    {
      fprintf(port_fifo[i], "port%d %d\n", i, port_array[i][1] );
      fprintf(port_data[i], "port%d %d\n", i, port_array[i][1]);
      port_array[i][1] = 0 ;
    }

  for (i = 0 ; i < num_proto; i++ )
    {
      fprintf(proto_fifo[i], "proto%d %d\n", i, proto_array[i][1] );
      fprintf(proto_data[i], "proto%d %d\n", i, proto_array[i][1]);
      proto_array[i][1] = 0 ;
    }
  for (i = 0 ; i < num_sock; i++ )
    {
      fprintf(sock_fifo[i], "sock%d %d\n", i, sock_array[i][1] );
      fprintf(sock_data[i], "sock%d %d\n", i, sock_array[i][1] );
      sock_array[i][1] = 0 ;
    }
  for (i = 0 ; i < num_net; i++ )
    {
      fprintf(net_fifo[i], "net%d %d\n", i, netw[i].netbyte );
      fprintf(net_data[i], "net%d %d\n", i, netw[i].netbyte );
      netw[i].netbyte = 0 ;
    }

  for (i = 0 ; i < num_ip; i++ )
    {
      fprintf(ip_fifo[i], "ip%d %d\n", i, ipip[i].ipbyte );
      fprintf(ip_data[i], "ip%d %d\n", i, ipip[i].ipbyte );
      ipip[i].ipbyte = 0 ;
    }

  alarm( win );

  if ( dflg > 0 )
    {
      fprintf(debug, "\nRecibi un SIGINT...Imprimiendo Stats..\n");

      /*************** imprime stats de ports **************/

      fprintf( debug, "\nport\t");
      for (i = 0 ; i < num_port; i++ )
       fprintf( debug, "%7u\t", port_array[i][0] );

      fprintf( debug, "\nbytes\t");
      for (i = 0 ; i < num_port; i++ )
       fprintf( debug, "%7u\t", port_array[i][1] );

      fprintf( debug,"\n\t");

      for (i = 0 ; i < num_port; i++ )
       fprintf(debug, "port%d %d\t", i, port_array[i][1] );

      fprintf(debug, "\n\n");

      /*************** imprime stats de protos **************/

      fprintf( debug, "\nproto\t");
```

```
      for (i = 0 ; i < num_proto; i++ )
       fprintf( debug, "%7u\t", proto_array[i][0] );

      fprintf( debug, "\nbytes\t");
      for (i = 0 ; i < num_proto; i++ )
       fprintf(debug, "%7u\t", proto_array[i][1] );

      fprintf(debug, "\n\t");

      for (i = 0 ; i < num_proto; i++ )
         fprintf( debug, "proto%d %d\t", i, proto_array[i][1] );

      fprintf(debug, "\n\n");

      /*************** imprime stats de sock ***************/

      fprintf(debug, "\nsock\t");
      for (i = 0 ; i < num_sock; i++ )
       fprintf(debug, "%7u\t", sock_array[i][0] );

      fprintf(debug, "\nbytes\t");
      for (i = 0 ; i < num_sock; i++ )
       fprintf(debug, "%7u\t", sock_array[i][1] );

      fprintf(debug, "\n\t");

      for (i = 0 ; i < num_sock; i++ )
         fprintf( debug, "sock%d %d\t", i, sock_array[i][1] );

      fprintf(debug, "\n\n");

      /*************** imprime stats de nets ***************/

      fprintf(debug, "\nred\t");
      for (i = 0 ; i < num_net; i++ )
       fprintf(debug, "%10s\t", netw[i].net );

      fprintf(debug, "\nmask\t");
      for (i = 0 ; i < num_net; i++ )
       fprintf(debug, "%10s\t", netw[i].netmask );

      fprintf(debug, "\nbytes\t");
      for (i = 0 ; i < num_net; i++ )
       fprintf(debug, "%10u\t", netw[i].netbyte );

      fprintf(debug, "\n\t");

      for (i = 0 ; i < num_net; i++ )
         fprintf(debug, "net%d %d\t", i, netw[i].netbyte );

      fprintf( debug, "\n\n");

      /*************** imprime stats de ips ***************/

      fprintf(debug, "\nip\t");
      for (i = 0 ; i < num_ip; i++ )
       fprintf(debug, "%10s\t", ipip[i].ip );

      fprintf(debug, "\nbytes\t");
      for (i = 0 ; i < num_ip; i++ )
       fprintf(debug, "%10u\t", ipip[i].ipbyte );

      fprintf(debug, "\n\t");

      for (i = 0 ; i < num_ip; i++ )
         fprintf(debug, "ip%d %d\t", i, ipip[i].ipbyte );

      fprintf( debug, "\n\n");

    }
}

void main(int argc, char **argv)
{
```

```
char aux[40], cmd[70], buf[BUFSIZ];

int c;
extern char *optarg;
extern int optind;
int errflg = 0;
char *ifile = "le0";

while ( (c = getopt(argc, argv, "di:h") ) != EOF)
  switch (c) {
     case 'd':
       dflg++;
       break;

     case 'i':
       ifile = optarg;
       break;

     case 'h':
       errflg++;
  }

if (errflg) {
  fprintf(stderr, "sintaxis: netgraph [-d] [-i <interface> ] [-h] \n");
  exit (2);
}

if ( dflg > 0 )
    {
     if ( ( debug = fopen( "./netgraph.debug", "w+") ) == NULL)
       {
         printf("No se puede abrir./netgraph.debug \n");
         exit(1);
       }
     setbuf( debug, NULL );
    }

if ( strstr(ifile, "le" ) == NULL || strstr(ifile, ":" ) != NULL )
  {
    /* interfase no es del tipo le0 le1...etc. */

    if ( strstr(ifile, "hme" ) == NULL || strstr(ifile, ":" ) != NULL )
     {
       /* interfase no es del tipo hme0 hme1...etc. */

       printf("Interface debe ser del tipo hme? o le? \n");
       exit(1);
     }
  }

sprintf(cmd, "/sbin/ifconfig %s 2>/tmp/ifcfg >/tmp/ifcfg2", ifile );
system( cmd );

ptr = fopen("/tmp/ifcfg", "r");

fgets( buf, BUFSIZ , ptr );

if ( strstr( buf, "no such interface" ) != NULL )
  {
    printf("no existe tal interfaz\n");
    exit(1);
  }
fclose(ptr);

sprintf( cmd, "./tcpdump -tt -e -q -n -l -i %s", ifile );
signal( SIGTERM, cleanup );

out = fopen ("out", "w+" );

init_var();
read_config();
check_lock();
save_pid_and_lock();
```

```
  signal( SIGINT, cleanup );
  signal( SIGALRM, statistica );
  alarm( win );

  mk_fifos_cgi_data();

  if ( (ptr = popen(cmd, "r") ) == NULL)
      {
       printf("No se puede ejecutar ./tcpdump\n");
      }

  do
    {
  /*
    ejemplo de linea capturada..

         tpo   %s   proto len  ip       port >   ip         port : info
    845579273.547304 0800 76 200.1.19.13. 23 > 200.1.21.84. 1024 : tcp 22 (DF)

         tpo   %s   proto len  ipx-id             sock >   ipx-id             sock : info
    849130103.891918 0038 70 50.00:00:c0:e3:6b:5d. 453 > 50.ff:ff:ff:ff:ff:ff. 453 :ipx-rip-
resp 16/1.2 32/1.2[|ipx 8]
    */

      tpo=1;

      while (tpo < 870542000 )
        {
          fgets( buf, BUFSIZ , ptr );
          sscanf( buf, "%lu %s %x %d %s %d %s %s %d %s", &tpo, aux, &proto, &len, ipp1,
&port1, aux, ipp2, &port2, info );
       }

      // sumlen += len;

      // fprintf( out, "\nlinea = %s", buf );
      // printf( "tpo=%lu len=%d proto=%x proto(d)=%d port1=%d port2=%d\n", tpo, len, proto,
proto, port1, port2 );

      if ( num_port >0 )
       port_stat();

      if ( num_proto >0 )
       proto_stat();

      if ( num_sock >0 )
       sock_stat();

      if ( num_net >0 )
       net_stat();

      if ( num_ip >0 )
       ip_stat();

      signal( SIGINT, cleanup );
      signal( SIGTERM, cleanup );
      signal( SIGALRM, statistica );

   }  while ( 1 );

  fclose (out);

}
```

Anexo Nº 6

Listado del programa step1.c

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#ifndef NO_STDLIB_H
#include <stdlib.h>
#else
char *getenv();
#endif
#include <string.h>

typedef struct {
    char name[128];
    char val[128];
} entry;

void getword(char *word, char *line, char stop);
char x2c(char *what);
void unescape_url(char *url);
void plustospace(char *str);

int n=0, i, num_net=0, port, upor, proto, sock, usoc;
int errorport=-1, errorproto=-1, errorsock=-1;
char net[20], netm[20], uproto[6], tmp[1000], tmp2[300], tmp3[300];

int final[20][3];
char final_net[10][3][20], final_uproto[20][10];


int esta( int val, int w )
{
  int i=0;

  while( final[i][w] != -1 )
    if ( final[i++][w] == val )
      return( 1 );

  return( 0 );
}

int esta_net( char *val, int w )
{
  int i=0;

  while( strcmp( final_net[i][w], "-1" ) != 0 )
    if ( strcmp( final_net[i++][w] , val ) == 0 )
      return( 1 );

  return( 0 );
}

void agrega_net( char *ip, int w )
{
  int i=0;

  if ( strcmp( final_net[0][w], "-1" ) != 0 )
    {
      while( strcmp( final_net[i++][w], "-1" ) != 0 );
      strcpy( final_net[i-1][w] , ip );
      strcpy( final_net[i][w], "-1" );
      //printf("agregue..%s..en [%d][%d]", final_net[i-1][w], i-1, w );
    }
  else
    {
```

```
      strcpy( final_net[0][w] , ip );
      strcpy( final_net[1][w], "-1" );
      //printf("agregue..%s..en [0][%d]", final_net[0][w], w );
    }

}

void agrega_uproto( char *p )
{
  int i=0;

  if ( strcmp( final_uproto[0], "-1" ) != 0 )
    {
      printf("yes...");
      while( strcmp( final_uproto[i++], "-1" ) != 0 );
      strcpy( final_uproto[i-1], p );
      strcpy( final_uproto[i], "-1" );
      printf("agregue..%s..en [%d]", final_uproto[i-1], i-1 );
    }
  else
    {
      strcpy( final_uproto[0], p );
      strcpy( final_uproto[1], "-1" );
      printf("agregue..%s..en final_uproto[0]", final_uproto[0] );
    }

}

void agrega( int val, int w )
{
  int i=0;

  if ( final [0][w] != -1 )
    {
      while( final[i++][w] != -1 );
      final[i-1][w] = val;
      final[i][w] = -1;
    }
  else
    {
      final[0][w] = val;
      final[1][w] = -1;
    }
}

int validate_ip( char *str )
{
  int pto=0, i=0;

  int n1, n2, n3, n4 ;

  while ( str[i] != 0 )
    if ( str[i++] == '.' )
      pto++;

  if ( pto != 3 )
    return( 1 );

  if ( inet_addr( str ) == -1 )
    return( 1 );

  sscanf( str, "%d.%d.%d.%d", &n1, &n2, &n3, &n4 );

  if ( (n1>255) || (n2>255) ||(n3>255) ||(n4>255) ||(n1<0) ||(n2<0) ||(n3<0) ||(n4<0) )
    return(1);

  return( 0 );
}

void main(int argc, char *argv[]) {
  entry entries[10000];
  register int x, m=0;
  int y, err=0;
  char *cl, hostname[20];
```

```
  printf("Content-type: text/html%c%c",10,10);

  if(strcmp(getenv("REQUEST_METHOD"),"GET")) {
    printf("This script should be referenced with a METHOD of GET.\n");
    printf("If you don't understand this, see this ");
    printf("<A HREF=\"http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/fill-out-
forms/overview.html\">forms overview</A>.%c",10);
    exit(1);
  }

  cl = getenv("QUERY_STRING");
  if(cl == NULL) {
    printf("No query information to decode.\n");
    exit(1);
  }

  for(x=0;cl[0] != '\0';x++) {
    m=x;
    getword(entries[x].val,cl,'&');
    plustospace(entries[x].val);
    unescape_url(entries[x].val);
    getword(entries[x].name,entries[x].val,'=');
  }

  /* printf("<H1>Query Results</H1>");
  printf("You submitted the following name/value pairs:<p>%c",10);
  printf("<ul>%c",10);

  for(x=0; x <= m; x++)
    printf("<li> <code>%s = %s</code>%c",entries[x].name, entries[x].val,10);
  printf("</ul>%c",10);
  */

  printf("<HTML>");
  printf("<HEAD>");
  printf("<TITLE>Configuraci&oacute;n de NetGraph TCP/IP</TITLE>");

  printf("<BODY BGCOLOR=\"#ffffff\" > ");

  printf("</HEAD>");

  final[0][0] = final[0][1] = final[0][2] = -1;
  strcpy( final_net[0][0], "-1" );
  strcpy( final_net[0][1], "-1" );
  strcpy( final_net[0][2], "-1" );
  strcpy( final_uproto[0], "-1" );

  for (x=4 ; x<=m ; x++ )
    {

      /* 0=port       1=proto 2=sock */

      // printf("entries[%d].name =%s--- entries[%d].val=%s---------- ",  x,entries[x].name,
x , entries[x].val);

      if (strstr( entries[x].name, "por" ) != NULL && ( strcmp(entries[x].val, "" ) != 0 ) )
       if ( esta( atoi( entries[x].val) , 0 ) == 0 )
         agrega( atoi( entries[x].val), 0 );

      if (strcmp( entries[x].name, "list_net" ) == 0 && ( strcmp(entries[x].val, "" ) != 0 )
)
         agrega_net( entries[x].val, 0 );

      if (strcmp( entries[x].name, "list_netm" ) == 0 && ( strcmp(entries[x].val, "" ) != 0 )
)
         agrega_net( entries[x].val, 1 );

      if (strcmp( entries[x].name, "list_ip" ) == 0 && ( strcmp(entries[x].val, "" ) != 0 ) )
         agrega_net( entries[x].val, 2 );

      if (strstr( entries[x].name, "upro" ) != NULL && ( strcmp(entries[x].val, "" ) != 0 ) )
           agrega_uproto( entries[x].val );
```

```
      if (strstr( entries[x].name, "proto" ) != NULL  && ( strcmp(entries[x].val, "" ) != 0 )
)
       if ( esta( atoi( entries[x].val) , 1 ) == 0 )
         agrega( atoi( entries[x].val), 1 );

      if (strstr( entries[x].name, "soc" ) != NULL  && ( strcmp(entries[x].val, "" ) != 0 ) )
       if ( esta( atoi( entries[x].val) , 2 ) == 0 )
         agrega( atoi( entries[x].val), 2 );
    }

  for ( y=0 ; y < 3; y++ )
    {
      x = 0;

      while( strcmp( final_net[x][y], "-1" ) != 0 )
       {
         if ( validate_ip(  final_net[x][y] ) == 1 )
           {
             printf("ERROR en IP %s%c", final_net[x][y], 10 );
             err = 1;
           }
         x++;
       }
    }

  if ( err == 1 )
    exit(1);

  /* printf("Ports%c", 10 );
  x=0;
  while( final[x][0] != -1 )
    printf("%d, ", final[x++][0] );

  printf("ip%c", 10 );
  x=0;
  while( strcmp( final_net[x][2], "-1") != 0 )
    printf( " %s, ", final_net[x++][2] );

  printf("net%c", 10 );
  x=0;
  while( strcmp( final_net[x][0], "-1") != 0 )
    printf("%s, ", final_net[x++][0] );

  printf("netm%c", 10 );
  x=0;
  while( strcmp( final_net[x][1], "-1") != 0 )
    printf( " %s, ", final_net[x++][1] );

  printf("uproto%c", 10 );
  x=0;
  while( strcmp( final_uproto[x], "-1") != 0 )
    printf( " %s, ", final_uproto[x++]);

  printf("Proto%c", 10 );
  x=0;
  while( final[x][1] != -1 )
    printf("%d, ", final[x++][1] );

  printf("Sock%c", 10 );
  x=0;
  while( final[x][2] != -1 )
    printf("%d, ", final[x++][2] );
    */

  gethostname( hostname, 20 );

  printf("<FORM NAME=\"myform\" ACTION=\"http://%s:8080/cgi-bin/step2\" METHOD=\"GET\">",
hostname );

  printf("<H1>Fijar Umbrales</H1>");
  printf("<center> <table border=2 >%c",10);
  printf("<td align=center colspan=6><b>Tama&ntilde;o de Ventana = %d min</b>",
atoi(entries[0].val) );
  printf("<input type=hidden name=win value=\"%d\">", atoi(entries[0].val) );
```

```
  printf("<tr><td>Monitor<td>MIN<td>MAX<td>Umbral<td>Ancho<td>Alto");

  x=0;
  while( final[x][0] != -1 )
    {
      printf("%c%c<tr>%c", 10, 10, 10);
      sprintf( tmp,  "<td>Port %d<td><INPUT TYPE=\"text\" NAME=\"minport%d\" MAXLENGTH=8
size=8 VALUE=\"%d\">", final[x][0],  final[x][0], atoi(entries[1].val) );
      sprintf( tmp2, "%c<td><INPUT TYPE=\"text\" NAME=\"maxport%d\" MAXLENGTH=8 size=8
VALUE=\"%d\">", 10,  final[x][0] , atoi(entries[2].val) );
      strcat( tmp, tmp2 );
      sprintf( tmp3, "%c<td><INPUT TYPE=\"text\" NAME=\"umbport%d\" MAXLENGTH=8 size=8
VALUE=\"%d\">", 10,  final[x][0], atoi(entries[3].val)  );
      strcat( tmp, tmp3 );
      sprintf( tmp3, "%c<td><INPUT TYPE=\"text\" NAME=\"widport%d\" MAXLENGTH=8 size=8
VALUE=250 >", 10,  final[x][0] );
      strcat( tmp, tmp3 );
      sprintf( tmp3, "%c<td><INPUT TYPE=\"text\" NAME=\"heiport%d\" MAXLENGTH=8 size=8
VALUE=150 >", 10,  final[x++][0] );
      strcat( tmp, tmp3 );
      printf( tmp );
    }

  x=0;
  while( final[x][1] != -1 )
    {
      printf("%c%c<tr>%c", 10, 10, 10);
      sprintf( tmp,  "<td>Protocol %d<td><INPUT TYPE=\"text\" NAME=\"minproto%d\" MAXLENGTH=8
size=8 VALUE=\"%d\">", final[x][1],  final[x][1], atoi(entries[1].val) );
      sprintf( tmp2, "%c<td><INPUT TYPE=\"text\" NAME=\"maxproto%d\" MAXLENGTH=8 size=8
VALUE=\"%d\">", 10,  final[x][1] , atoi(entries[2].val) );
      sprintf( tmp3, "%c<td><INPUT TYPE=\"text\" NAME=\"umbproto%d\" MAXLENGTH=8 size=8
VALUE=\"%d\">", 10,  final[x][1] , atoi(entries[3].val) );
      strcat( tmp, tmp2 );
      strcat( tmp, tmp3 );
      sprintf( tmp3, "%c<td><INPUT TYPE=\"text\" NAME=\"widproto%d\" MAXLENGTH=8 size=8
VALUE=250 >", 10,  final[x][1] );
      strcat( tmp, tmp3 );
      sprintf( tmp3, "%c<td><INPUT TYPE=\"text\" NAME=\"heiproto%d\" MAXLENGTH=8 size=8
VALUE=150 >", 10,  final[x++][1] );
      strcat( tmp, tmp3 );
      printf( tmp );
    }

  x=0;
  while( strcmp( final_uproto[x], "-1") != 0 )
    {
      //printf("final_uproto[%d]==%s--", xx, final_uproto[x] );

      printf("%c%c<tr>%c", 10, 10, 10);
      sprintf( tmp,  "<td>Protocol %s<td><INPUT TYPE=\"text\" NAME=\"minuproto%s\"
MAXLENGTH=8 size=8 VALUE=\"%s\">", final_uproto[x], final_uproto[x], entries[1].val  );
      sprintf( tmp2, "%c<td><INPUT TYPE=\"text\" NAME=\"maxuproto%s\" MAXLENGTH=8 size=8
VALUE=\"%d\">", 10,  final_uproto[x], atoi(entries[2].val)  );
      sprintf( tmp3, "%c<td><INPUT TYPE=\"text\" NAME=\"umbuproto%s\" MAXLENGTH=8 size=8
VALUE=\"%d\">", 10,  final_uproto[x], atoi(entries[3].val)  );
      strcat( tmp, tmp2 );
      strcat( tmp, tmp3 );

      sprintf( tmp3, "%c<td><INPUT TYPE=\"text\" NAME=\"widuproto%s\" MAXLENGTH=8 size=8
VALUE=250 >", 10, final_uproto[x] );
      strcat( tmp, tmp3 );
      sprintf( tmp3, "%c<td><INPUT TYPE=\"text\" NAME=\"heiuproto%s\" MAXLENGTH=8 size=8
VALUE=150 >", 10, final_uproto[x] );
      strcat( tmp, tmp3 );

      printf( tmp );
      x++;
    }

  x=0;
  while( final[x][2] != -1 )
    {
```

```
      printf("%c%c<tr>%c", 10, 10, 10);
      sprintf( tmp,  "<td>Socket %d<td><INPUT TYPE=\"text\" NAME=\"minsock%d\" MAXLENGTH=8
size=8 VALUE=\"%d\">", final[x][2],  final[x][2], atoi(entries[1].val) );
      sprintf( tmp2, "%c<td><INPUT TYPE=\"text\" NAME=\"maxsock%d\" MAXLENGTH=8 size=8
VALUE=\"%d\">", 10,  final[x][2], atoi(entries[2].val)  );
      sprintf( tmp3, "%c<td><INPUT TYPE=\"text\" NAME=\"umbsock%d\" MAXLENGTH=8 size=8
VALUE=\"%d\">", 10,  final[x][2], atoi(entries[3].val)  );
      strcat( tmp, tmp2 );
      strcat( tmp, tmp3 );
      sprintf( tmp3, "%c<td><INPUT TYPE=\"text\" NAME=\"widsock%d\" MAXLENGTH=8 size=8
VALUE=250 >", 10,  final[x][2] );
      strcat( tmp, tmp3 );
      sprintf( tmp3, "%c<td><INPUT TYPE=\"text\" NAME=\"heisock%d\" MAXLENGTH=8 size=8
VALUE=150 >", 10,  final[x++][2] );
      strcat( tmp, tmp3 );
      printf( tmp );
    }

  x=0;
  while( strcmp( final_net[x][0], "-1") != 0 )
    {
      printf("%c%c<tr>%c", 10, 10, 10);
      sprintf( tmp,  "<td>Network %s<br>Netmask %s<td><INPUT TYPE=\"text\" NAME=\"minnet%d\"
MAXLENGTH=8 size=8 VALUE=\"%d\">", final_net[x][0], final_net[x][1], x, atoi(entries[1].val)
);
      sprintf( tmp2, "%c<td><INPUT TYPE=\"text\" NAME=\"maxnet%d\" MAXLENGTH=8 size=8
VALUE=\"%d\">", 10,  x, atoi(entries[2].val)  );
      sprintf( tmp3, "%c<td><INPUT TYPE=\"text\" NAME=\"umbnet%d\" MAXLENGTH=8 size=8
VALUE=\"%d\">", 10,  x, atoi(entries[3].val)  );

      strcat( tmp, tmp2 );
      strcat( tmp, tmp3 );

      sprintf( tmp3, "%c<td><INPUT TYPE=\"text\" NAME=\"widnet%d\" MAXLENGTH=8 size=8
VALUE=250 >", 10, x );
      strcat( tmp, tmp3 );
      sprintf( tmp3, "%c<td><INPUT TYPE=\"text\" NAME=\"heinet%d\" MAXLENGTH=8 size=8
VALUE=150 >", 10, x );
      strcat( tmp, tmp3 );

      sprintf( tmp3, "<input type=hidden name=net%d value=\"%s %s\">", x, final_net[x][0],
final_net[x][1] );
      strcat( tmp, tmp3 );
      printf( tmp );
      x++;
    }

  x=0;
  while( strcmp( final_net[x][2], "-1") != 0 )
    {
      printf("%c%c<tr>%c", 10, 10, 10);
      sprintf( tmp,  "<td>IP %s<td><INPUT TYPE=\"text\" NAME=\"minip%d\" MAXLENGTH=8 size=8
VALUE=\"%d\">", final_net[x][2],  x, atoi(entries[1].val)  );
      sprintf( tmp2, "%c<td><INPUT TYPE=\"text\" NAME=\"maxip%d\" MAXLENGTH=8 size=8
VALUE=\"%d\">", 10,  x, atoi(entries[2].val)  );
      sprintf( tmp3, "%c<td><INPUT TYPE=\"text\" NAME=\"umbip%d\" MAXLENGTH=8 size=8
VALUE=\"%d\">", 10,  x, atoi(entries[3].val)  );
      strcat( tmp, tmp2 );
      strcat( tmp, tmp3 );

      sprintf( tmp3, "%c<td><INPUT TYPE=\"text\" NAME=\"widip%d\" MAXLENGTH=8 size=8
VALUE=250 >", 10, x );
      strcat( tmp, tmp3 );
      sprintf( tmp3, "%c<td><INPUT TYPE=\"text\" NAME=\"heiip%d\" MAXLENGTH=8 size=8
VALUE=150 >", 10, x );
      strcat( tmp, tmp3 );

      sprintf( tmp3, "<input type=hidden name=ip%d value=\"%s\">", x, final_net[x][2] );
      strcat( tmp, tmp3 );
      printf( tmp );
      x++;
    }

  printf("%c</table>%c",10, 10 );
```

```
  printf("<center>");
  printf("<table width=300>");
  printf("<tr>");
  printf("<td><INPUT TYPE=\"submit\" VALUE=\"Ingresar Valores\" >");
  printf("<td><INPUT TYPE=\"reset\" VALUE=\"Resetear Valores\">");
  printf("</table></center>");
}
```

Anexo Nº 7

Listado del programa step2.c

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

#ifndef NO_STDLIB_H
#include <stdlib.h>
#else
char *getenv();
#endif
#include <string.h>

typedef struct {
    char name[128];
    char val[128];
} entry;

void getword(char *word, char *line, char stop);
char x2c(char *what);
void unescape_url(char *url);
void plustospace(char *str);
char hostname[20];

int n=0, port, proto, sock, pid;
char port_nam[10], proto_nam[10], sock_nam[10], tmp[10];

entry entries[10000];

int port_num[]={ 20, 21, 23, 25, 53, 69, 79, 80, 109, 110, 119, 520, 2049, -1};
int proto_num[]={ 0, 8137, 800, 806, 9000, -1};
int sock_num[]={ 451, 452, 453, 455, 456, -1};

char *port_name[]={ "FTP-data", "FTP", "Telnet", "SMTP", "DNS", "TFTP", "Finger", "WWW",
"POP2", "POP3", "NNTP", "RIP", "NFS" };
char *proto_name[]={ "IEEE802.3", "IPX", "IP", "ARP", "Loopback"};
char *sock_name[]={ "NCP", "SAP", "RIP", "NetBIOS", "DIAG" };

char *num2name( int num, int index)
{
  int i=0;

  if ( index == 0 )
    {
      while ( port_num[i] != -1 )
          if ( port_num[i++] == num )
            return( port_name[i-1] );

      sprintf(tmp, "port%d", num );
      return ( tmp );
    }

  if ( index == 1 )
    {
      while ( proto_num[i] != -1 )
       if ( proto_num[i++] == num )
         return( proto_name[i-1] );

      sprintf(tmp, "proto%d", num );
      return ( tmp );
    }

  if ( index == 2 )
    {
      while ( sock_num[i] != -1 )
       if ( sock_num[i++] == num )
         return( sock_name[i-1] );

      sprintf(tmp, "sock%d", num );
      return ( tmp );
```

```
    }

return("NULL");

}

int crea_cfg()
{
  FILE *cfg;
  int x=1, n=0;
  char network[20], netmask[20], uproto[10];

  if ( (cfg = fopen( "../../netgraph.cfg", "w") ) == NULL)
    {
      printf("No se puede abrir archivo netgraph.cfg en /usr/local/netgraph\n");
      return(1);
    }

  fprintf( cfg, "window %d%c", atoi(entries[0].val), 10 );


    /* Ports */

    while( strstr( entries[x].name, "minport" ) != NULL )
      {
       sscanf( entries[x].name, "minport%d", &port );
       fprintf( cfg, "port%d %d %d %d %d %c", n++, port, atoi(entries[x].val),
atoi(entries[x+1].val), atoi(entries[x+2].val), 10 );
       x = x + 5;
      }

    /* Protocols */

    n=0;
    while( strstr( entries[x].name, "minproto" ) != NULL )
      {
       sscanf( entries[x].name, "minproto%d", &proto );
       fprintf( cfg, "proto%d %d %d %d %d %c", n++, proto, atoi(entries[x].val),
atoi(entries[x+1].val), atoi(entries[x+2].val), 10 );
       x = x + 5;
      }

    /* Protocols User defined */

    while( strstr( entries[x].name, "minuproto" ) != NULL )
      {
       sscanf( entries[x].name, "minuproto%s", uproto );
       fprintf( cfg, "proto%d %s %d %d %d %c", n++, uproto, atoi(entries[x].val),
atoi(entries[x+1].val), atoi(entries[x+2].val), 10 );
       x = x + 5;
      }

    /* Sockets */

    n=0;
    while( strstr( entries[x].name, "minsock" ) != NULL )
      {
       sscanf( entries[x].name, "minsock%d", &sock );
       fprintf( cfg, "sock%d %d %d %d %d %c", n++, sock, atoi(entries[x].val),
atoi(entries[x+1].val), atoi(entries[x+2].val), 10 );
       x = x + 5;
      }

    /* Networks */

    n=0;
    while( strstr( entries[x].name, "minnet" ) != NULL )
      {
       sscanf( entries[x+5].val, "%s %s", network, netmask );
       fprintf( cfg, "net%d %s %s %d %d %d %c", n++, network, netmask, atoi(entries[x].val),
atoi(entries[x+1].val), atoi(entries[x+2].val), 10 );
       x = x + 6;
      }
```

```
    /* IP */

    n=0;
    while( strstr( entries[x].name, "minip" ) != NULL )
      {
       sscanf( entries[x+5].val, "%s", network );
       fprintf( cfg, "ip%d %s %d %d %d %c", n++, network, atoi(entries[x].val),
atoi(entries[x+1].val), atoi(entries[x+2].val), 10 );
       x = x + 6;
      }

    fclose( cfg );
    return( 0 );
}

int crea_index()
{
  FILE *index;
  int x=1, n=0, win;
  char network[20], netmask[20], uproto[10];

  if ( ( index = fopen( "../htdocs/netgraph/index.html", "w") ) == NULL)
    {
      printf("No se puede abrir archivo index.html en ../htdocs/netgraph/index.html\n");
      return(1);
    }

  win = atoi(entries[0].val);
  gethostname( hostname, 20 );

  fprintf( index, "<html><head><title>Monitor de Tr&aacute;fico Ethernet
NETGRAPH</title></head>\n");
  fprintf( index, "<body BGCOLOR=\"#ffffff\"><CENTER><h2>Monitor de Tr&aacute;fico Ethernet
NETGRAPH</h2></CENTER>\n");

  fprintf( index, "<FORM NAME=\"myform\" ACTION=\"http://%s:8080/cgi-bin/stop\"
METHOD=\"GET\">\n", hostname);
  fprintf( index, "<center>\n");
  fprintf( index, "<INPUT TYPE=\"submit\" VALUE=\"EXIT\" >\n");
  fprintf( index, "</center></form>\n");

  fprintf( index, "<hr><br><CENTER><h1>Ports<br>\n\n");


  /* Ports */

  while( strstr( entries[x].name, "minport" ) != NULL )
    {
      sscanf( entries[x].name, "minport%d", &port );

      strcpy( port_nam, num2name( port, 0 ));

      fprintf( index, "\n<applet codebase=classes code=QuoteChart.class width=%d
height=%d>\n", atoi(entries[x+3].val), atoi(entries[x+4].val));
      fprintf( index, " <param name=livedata value=\"http://%s:8080/cgi-bin\">\n", hostname);
      fprintf( index, " <param name=stock value=port%d>\n", n);
      fprintf( index, " <param name=label value=port%d>\n", port);
      fprintf( index, " <param name=label2 value=%s>\n", port_nam);
      fprintf( index, " <param name=history value=\"%dm\">\n", win);
      fprintf( index, " <param name=fudge value=false>\n");
      fprintf( index, " </applet>\n");

      n++;
      x = x + 5;
    }

  /* Protocols */

  fprintf( index, "<hr><br><CENTER><h1>Protocols<br>\n\n");
  n=0;
  while( strstr( entries[x].name, "minproto" ) != NULL )
    {
      sscanf( entries[x].name, "minproto%d", &proto );
```

```
      strcpy( proto_nam, num2name( proto, 1 ));

      fprintf( index, "\n<applet codebase=classes code=QuoteChart.class width=%d
height=%d>\n", atoi(entries[x+3].val), atoi(entries[x+4].val));
      fprintf( index, " <param name=livedata value=\"http://%s:8080/cgi-bin\">\n", hostname);
      fprintf( index, " <param name=stock value=proto%d>\n", n);
      fprintf( index, " <param name=label value=proto%d>\n", proto);
      fprintf( index, " <param name=label2 value=%s>\n", proto_nam);
      fprintf( index, " <param name=history value=\"%dm\">\n", win);
      fprintf( index, " <param name=fudge value=false>\n");
      fprintf( index, " </applet>\n");

      n++;
      x = x + 5;
      }

  /* Protocols User Defined*/

  while( strstr( entries[x].name, "minuproto" ) != NULL )
    {
      sscanf( entries[x].name, "minuproto%s", uproto );

      strcpy( proto_nam, uproto);

      fprintf( index, "\n<applet codebase=classes code=QuoteChart.class width=%d
height=%d>\n", atoi(entries[x+3].val), atoi(entries[x+4].val));
      fprintf( index, " <param name=livedata value=\"http://%s:8080/cgi-bin\">\n", hostname);
      fprintf( index, " <param name=stock value=proto%d>\n", n);
      fprintf( index, " <param name=label value=proto%s>\n", uproto);
      fprintf( index, " <param name=label2 value=%s>\n", proto_nam);
      fprintf( index, " <param name=history value=\"%dm\">\n", win);
      fprintf( index, " <param name=fudge value=false>\n");
      fprintf( index, " </applet>\n");

      n++;
      x = x + 5;
      }

  /* Sockets */

  fprintf( index, "<hr><br><CENTER><h1>Sockets<br>\n\n");
  n=0;
  while( strstr( entries[x].name, "minsock" ) != NULL )
    {
      sscanf( entries[x].name, "minsock%d", &sock );

      strcpy( sock_nam, num2name( sock, 2 ));

      fprintf( index, "\n<applet codebase=classes code=QuoteChart.class width=%d
height=%d>\n", atoi(entries[x+3].val), atoi(entries[x+4].val));
      fprintf( index, " <param name=livedata value=\"http://%s:8080/cgi-bin\">\n", hostname);
      fprintf( index, " <param name=stock value=sock%d>\n", n);
      fprintf( index, " <param name=label value=sock%d>\n", sock);
      fprintf( index, " <param name=label2 value=%s>\n", sock_nam);
      fprintf( index, " <param name=history value=\"%dm\">\n", win);
      fprintf( index, " <param name=fudge value=false>\n");
      fprintf( index, " </applet>\n");

      n++;
      x = x + 5;
      }

  /* Networks */

  fprintf( index, "<hr><br><CENTER><h1>Networks<br>\n\n");
  n=0;
  while( strstr( entries[x].name, "minnet" ) != NULL )
    {
      sscanf( entries[x+5].val, "%s %s", network, netmask );
      fprintf( index, "\n<applet codebase=classes code=QuoteChart.class width=%d
height=%d>\n", atoi(entries[x+3].val), atoi(entries[x+4].val));
      fprintf( index, " <param name=livedata value=\"http://%s:8080/cgi-bin\">\n", hostname);
      fprintf( index, " <param name=stock value=net%d>\n", n);
      fprintf( index, " <param name=label value=%s>\n", network);
```

```
      fprintf( index, " <param name=label2 value=%s>\n", netmask);
      fprintf( index, " <param name=history value=\"%dm\">\n", win);
      fprintf( index, " <param name=fudge value=false>\n");
      fprintf( index, " </applet>\n");

      n++;
      x = x + 6;
      }

    /* IP */

  fprintf( index, "<hr><br><CENTER><h1>IP<br>\n\n");
    n=0;
    while( strstr( entries[x].name, "minip" ) != NULL )
      {
       sscanf( entries[x+5].val, "%s", network );
      fprintf( index, "\n<applet codebase=classes code=QuoteChart.class width=%d
height=%d>\n", atoi(entries[x+3].val), atoi(entries[x+4].val));
      fprintf( index, " <param name=livedata value=\"http://%s:8080/cgi-bin\">\n", hostname);
      fprintf( index, " <param name=stock value=ip%d>\n", n);
      fprintf( index, " <param name=label value=%s>\n", network);
      fprintf( index, " <param name=history value=\"%dm\">\n", win);
      fprintf( index, " <param name=fudge value=false>\n");
      fprintf( index, " </applet>\n");

      n++;
      x = x + 6;
      }

    fclose( index );
    return( 0 );
}

int check_lock()
{
  FILE *lock;
  FILE *ppid;

  if ( (lock = fopen ("../../netgraph.lock", "r" ) ) != NULL )
    {
      if ( ( ppid = fopen ("../../netgraph.pid", "r" ) ) != NULL )
        {
          fscanf( ppid, "%d", &pid );
          fclose( ppid );
        }
      else
        return(3);

      fclose( lock );
      return(1);
    }

  fclose( lock );
  return(0);
}

void run_netgraph()
{
  chdir("/usr/local/netgraph");
  system("./netgraph &");

}

void main(int argc, char *argv[]) {
  register int x,m=0;
  char *cl;

  printf("Content-type: text/html%c%c",10,10);

  if(strcmp(getenv("REQUEST_METHOD"),"GET")) {
        printf("This script should be referenced with a METHOD of GET.\n");
        printf("If you don't understand this, see this ");
        printf("<A HREF=\"http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/fill-out-
forms/overview.html\">forms overview</A>.%c",10);
```

```
        exit(1);
    }

    cl = getenv("QUERY_STRING");
    if(cl == NULL) {
        printf("No query information to decode.\n");
        exit(1);
    }
    for(x=0;cl[0] != '\0';x++) {
        m=x;
        getword(entries[x].val,cl,'&');
        plustospace(entries[x].val);
        unescape_url(entries[x].val);
        getword(entries[x].name,entries[x].val,'=');
    }

    /* printf("<H1>Query Results</H1>");
    printf("You submitted the following name/value pairs:<p>%c",10);
    printf("<ul>%c",10);

    for(x=0; x <= m; x++)
      printf("<li> <code>%s = %s</code>%c",entries[x].name, entries[x].val,10 );
    printf("</ul>%c",10);
    */

    if ( check_lock() == 1 )
      {
       printf("ya esta corriendo el netgraph en pid= %d", pid );
       exit(1);
      }

    crea_cfg();

    crea_index();

    printf("<HTML>");
    printf("<HEAD>");
    printf("<TITLE>Monitor de Tr&aacute;fico NetGraph </TITLE>");
    printf("</HEAD>");

    printf(" <BODY BGCOLOR=\"#ffffff\" > %c", 10);
    printf("<center><h2>Monitor de Tr&aacute;fico NetGraph </h2><br><br>");

    printf("<h2>Configuracion LISTA !!!</h2>");
    printf("<br><h2>Ejecute el comando ./capture.start !!!</h2>");
    printf("<br><h2>y siga el Link !!!</h2>");
    printf("<center><h3><A HREF=\"/netgraph/\"> Net Graph</A></P>");

    printf("</BODY>");
    printf("</HTML>");

    chdir("/usr/local/netgraph");

    /*system("sh netgraph.start");
     */

    exit(0);
}
```

Anexo Nº 8

Listado del programa stop.c

```
#include <stdio.h>
#include <stdlib.h>

#ifndef NO_STDLIB_H
#include <stdlib.h>
#else
char *getenv();
#endif

#include <string.h>
#include <signal.h>
#include <unistd.h>

typedef struct {
    char name[128];
    char val[128];
} entry;

void getword(char *word, char *line, char stop);
char x2c(char *what);
void unescape_url(char *url);
void plustospace(char *str);

char hostname[20];

int check_lock()
{
  FILE *lock;

  if ( (lock = fopen ("../../netgraph.lock", "r" ) ) != NULL )
    {
      fclose( lock );
      return(1);
    }

  fclose( lock );
  return(0);
}

void main(int argc, char *argv[]) {
    entry entries[10000];
    register int x,m=0;
    char *cl;

    printf("Content-type: text/html%c%c",10,10);

    if(strcmp(getenv("REQUEST_METHOD"),"GET")) {
        printf("This script should be referenced with a METHOD of GET.\n");
        printf("If you don't understand this, see this ");
        printf("<A HREF=\"http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/fill-out-
forms/overview.html\">forms overview</A>.%c",10);
        exit(1);
    }

    cl = getenv("QUERY_STRING");
    if(cl == NULL) {
        printf("No query information to decode.\n");
        exit(1);
    }
    for(x=0;cl[0] != '\0';x++) {
        m=x;
        getword(entries[x].val,cl,'&');
        plustospace(entries[x].val);
        unescape_url(entries[x].val);
        getword(entries[x].name,entries[x].val,'=');
    }

    /*    printf("<H1>Query Results</H1>");
```

```
    printf("You submitted the following name/value pairs:<p>%c",10);
    printf("<ul>%c",10);

    for(x=0; x <= m; x++)
      printf("<li> <code>%s = %s</code>%c",entries[x].name, entries[x].val,10);
    printf("</ul>%c",10);
    */

    /* STOP the NETGRAPH */

    if( check_lock() == 0 )
      {
       printf("no esta corriendo el netgraph%c", 10 );
       exit(1);
      }

    chdir("../..");
    system("sh capture.stop" );

    gethostname( hostname, 20 );

    printf("<HTML>");
    printf("<HEAD>");
    printf("<TITLE>Monitor de Tr&aacute;fico Ethernet NetGraph </TITLE>");

    printf(" <BODY BGCOLOR=\"#ffffff\" > %c", 10);

    printf(" <SCRIPT language=\"javascript\"> %c", 10);
    printf(" <!-- hide script from non-JS browsers%c", 10);

    printf(" setTimeout (\"location='http://%s:8080/'\", 5000); // time in ms %c", hostname,
10);

    printf(" // end script --> %c", 10);
    printf(" </SCRIPT> %c", 10);

    printf("</HEAD>");

    printf("<BODY>");
    printf("<center><h2>Monitor de Tr&aacute;fico Ethernet NetGraph </h2><br><br>");

    printf("<h2>Monitor Detenido !!!</h2>");
    printf("<br><h2>Se volver&aacute; a la p&aacute;gina inicial autom&aacute;ticamente en 5
[s]!!!</h2>");

    printf("</BODY>");
    printf("</HTML>");

}
```

Anexo Nº 9

Listado del programa config.js

```
function Add(form) {

        var ll = form.list_net.length++;
        var llm = form.list_netm.length++;

        if ( ll > 9 )
                {
                alert("Solo puede ingresar 10 Networks" );
                return;
                }

        if ( form.net.value == "" )
                {
                alert( "No ha ingresado valor en Network" );
                return;
                }

        if ( form.netm.value == "" )
                {
                alert( "No ha ingresado valor en Netmask" );
                return;
                }

        form.list_net.options[ll].text = form.net.value;
        form.list_netm.options[ll].text = form.netm.value;

        form.net.value = "";
        form.netm.value = "";

}

function Kill(form) {
        var i = form.list_net.selectedIndex;
        var im = form.list_netm.selectedIndex;

        if ( i == -1 && im == -1 )
                return;

        if ( i == -1 )
                {
                item = im;
                form.list_net.options[item].selected = "true";
                }

        if ( im == -1 )
                {
                item = i;
                form.list_netm.options[item].selected = "true";
                }

        form.net.value = form.list_net.options[item].text;
        form.netm.value = form.list_netm.options[item].text;

        for ( var ii = item; ii < form.list_net.length; ii++ )
                {
                if ( ii == form.list_net.length - 1)
                        form.list_net.options[ii].text = "";
                else
                        form.list_net.options[ii].text = form.list_net.options[ii+1].text ;
                }

        form.list_net.options[item].selected = "";
        form.list_net.length--;

        for ( var ii = item; ii < form.list_netm.length; ii++ )
                {
                if ( ii == form.list_netm.length - 1)
```

```
                        form.list_netm.options[ii].text = "";
                else
                        form.list_netm.options[ii].text = form.list_netm.options[ii+1].text ;
                }

        form.list_netm.options[item].selected = "";
        form.list_netm.length--;
}

function Addip(form) {

        var ll = form.list_ip.length++;

        if ( ll > 9 )
                {
                alert("Solo puede ingresar 10 IP" );
                return;
                }

        if ( form.ip.value == "" )
                {
                alert( "No ha ingresado valor en IP Address" );
                return;
                }

        form.list_ip.options[ll].text = form.ip.value;

        form.ip.value = "";
}

function Killip(form) {

        var item = form.list_ip.selectedIndex;

        if ( item == -1 )
                return;

        form.ip.value = form.list_ip.options[item].text;

        for ( var ii = item; ii < form.list_ip.length; ii++ )
                {
                if ( ii == form.list_ip.length - 1)
                        form.list_ip.options[ii].text = "";
                else
                        form.list_ip.options[ii].text = form.list_ip.options[ii+1].text ;
                }

        form.list_ip.options[item].selected = "";
        form.list_ip.length--;
}

function notempty( input ) {

        if ( input.value == "" )
                {
                alert( "No debe dejar el campo \"Tamano de la ventana\" sin numero" );
                input.focus();
                return( false);
                }
        else
          return( true);
}

function checknum( input, msg ) {

        var str = input.value

        if ( str == "" )
                return( true );

        msg = msg + " no es un numero: " + input.value ;

        for ( var i = 0 ; i < str.length; i++ ) {
                var ch = str.substring( i , i+1 );
```

```
                if (( ch < "0" || "9" < ch )) {
                        alert( msg );
                        return (false );
                }
        }
        return( true );
}

function Run(form) {

        for ( ii = 0 ; ii < form.list_net.length; ii++ )
                form.list_net.options[ii].selected = "true";

        for ( ii = 0 ; ii < form.list_netm.length; ii++ )
                form.list_netm.options[ii].selected = "true";

        for ( ii = 0 ; ii < form.list_ip.length; ii++ )
                form.list_ip.options[ii].selected = "true";

        var ret0 = notempty( form.win );
        var ret1 = checknum( form.win, "Tamano de la ventana" );
        var ret2 = checknum( form.upor1, "Port A" );
        var ret3 = checknum( form.upor2, "Port B" );
        var ret4 = checknum( form.upor3, "Port C" );
        var ret5 = checknum( form.upor4, "Port D" );
        var ret6 = checknum( form.upor5, "Port E" );
        var ret7 = checknum( form.upro1, "Proto A" );
        var ret8 = checknum( form.upro2, "Proto B" );
        var ret9 = checknum( form.upro3, "Proto C" );
        var ret10 = checknum( form.upro4, "Proto D" );
        var ret11 = checknum( form.upro5, "Proto E" );
        var ret12 = checknum( form.usoc1, "Socket A" );
        var ret13 = checknum( form.usoc2, "Socket B" );
        var ret14 = checknum( form.usoc3, "Socket C" );
        var ret15 = checknum( form.usoc4, "Socket D" );
        var ret16 = checknum( form.usoc5, "Socket E" );

        var ret = (ret0 && ret1) && (ret2  && ret3)  && (ret4 && ret5) && (ret6 && ret7) &&
(ret8 && ret9) && (ret10 && ret11) && (ret12 && ret13) && (ret14 && ret15) && ret16;

        if ( ret==true )
                document.myform.submit();
}
```

Anexo Nº 10

Listado del programa applet-cgi.c

```
#include <stdio.h>
#include <sys/time.h>

#ifndef NO_STDLIB_H
#include <stdlib.h>
#else
char *getenv();
#endif

#include <string.h>
#include <signal.h>
#include <unistd.h>

typedef struct {
    char name[128];
    char val[128];
} entry;

char  tmp[20], line[20], line2[20], aux[20], ip[20], net[20], netm[20];

FILE *out, *fifo_in, *ptr, *ptr2;
int win, data, pid, k;

int port, proto, max, min, thres;

void getword(char *word, char *line, char stop);
char x2c(char *what);
void unescape_url(char *url);
void plustospace(char *str);

void read_cfg( char *who )
{
  FILE *cfg;
  char str[200];


  if ( ( cfg = fopen( "../../netgraph.cfg", "r") ) == NULL)
    {
      printf("no-puedo-abrir-cfg-file\n");
      exit(0);
    }

  fscanf( cfg, "window %d", &win );

  do {

    if ( strstr( who, "port" ) != NULL )
       fscanf( cfg, "%s %d %d %d %d", str, &port, &min, &max, &thres );

    if ( strstr( who, "proto" ) != NULL )
       fscanf( cfg, "%s %d %d %d %d", str, &proto, &min, &max, &thres );

    if ( strstr( who, "sock" ) != NULL )
       fscanf( cfg, "%s %d %d %d %d", str, &port, &min, &max, &thres );

    if ( strstr( who, "net" ) != NULL )
       fscanf( cfg, "%s %s %s %d %d %d", str, net, netm, &min, &max, &thres );

    if ( strstr( who, "ip" ) != NULL )
       fscanf( cfg, "%s %s %d %d %d", str, ip, &min, &max, &thres );

    } while ( strcmp(who, str) != 0 );

  fclose( cfg );
}

int kill_before_pid( char *who )
{
```

```
  FILE *ppid;
  int i;
  char tmp[100];

  sprintf( tmp, "/tmp/%s.pid", who );

  if ( (ppid = fopen( tmp, "r") ) == NULL)
    {
      return(0);
    }

  fscanf( ppid, "%d", &i );

  printf("old-pid-%d\n", i);

  sprintf(tmp, "kill -KILL %d\n", i);

  system( tmp );

  fclose(ppid);

  return( 1 );
}

void save_pid( char *who )
{
  FILE *ppid;
  int i;
  char tmp[30], tmp2[30];

  i = (int )getpid();

  sprintf( tmp, "rm /tmp/%s.pid", who );
  sprintf( tmp2, "/tmp/%s.pid", who );

  system( tmp );

  if ( (ppid = fopen( tmp2, "w+") ) == NULL)
      {
       printf("No se puede abrir archivo SAVE\n");
      }

  printf("new-pid-%d\n", i);

  fprintf( ppid, "%d\n", i );
  fclose( ppid );
}

void cleanup()
{

  fclose( fifo_in );

  exit(1);

}


void main(int argc, char *argv[]) {
    entry entries[10000];
    register int x,m=0;
    char *cl;
    int i;
    struct timeval init, now;

    setbuf(stdout, NULL);

    printf("Content-type: text/html%c%c",10,10);

    if(strcmp(getenv("REQUEST_METHOD"),"GET")) {
        printf("This script should be referenced with a METHOD of GET.\n");
        printf("If you don't understand this, see this ");
        printf("<A HREF=\"http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/fill-out-
forms/overview.html\">forms overview</A>.%c",10);
```

```
    exit(1);
}

cl = getenv("QUERY_STRING");
if(cl == NULL) {
    printf("No query information to decode.\n");
    exit(1);
}
for(x=0;cl[0] != '\0';x++) {
    m=x;
    getword(entries[x].val,cl,'&');
    plustospace(entries[x].val);
    unescape_url(entries[x].val);
    getword(entries[x].name,entries[x].val,'=');
}

/*    out = fopen( "/tmp/out.html", "w+" );

fprintf(out, "<H1>Query Results</H1>");
fprintf(out, "You submitted the following name/value pairs:<p>%c",10);
fprintf(out, "<ul>%c",10);

for(x=0; x <= m; x++)
  fprintf(out, "<li> <code>%s = %s</code>%c",entries[x].name, entries[x].val,10);
fprintf(out, "</ul>%c",10);
fprintf(out, "argv[0] = %s\n", argv[0] );

fclose(out);
*/

signal( SIGINT, cleanup );
signal( SIGTERM, cleanup );

sscanf(entries[0].name, "dhle,60,%ds", &k );

printf("MISS-%d--pid-%d\n", k, (int)getpid() );

read_cfg( argv[0] );
kill_before_pid( argv[0] );
save_pid( argv[0] );

sprintf( tmp, "./%s.fifo", argv[0] );

if ( (fifo_in = fopen( tmp , "r" ) ) == NULL )
  {
   printf("No se puede abrir fifo file en r\n");
   exit(1);
  }

if (k != 60*win )
  {

   printf("SON-MENOS-DE-%d-SEGUNDOS\n", 60*win);

   for ( i = k/5 ; i > 0 ; i-- )
     {
       fgets( line, 50, fifo_in );

       gettimeofday( &now, NULL );

       sscanf( line, "%s%d", aux, &data );
       printf("old-%d-data-%d\n", i, data);
       printf("@%s -%d %d\n", aux, i*5 ,data );

       if ( i == k/5 )
         init.tv_sec = now.tv_sec;

       if ( ( now.tv_sec - init.tv_sec ) > win/2 )
         {
          printf("RESCALING-%s\n", argv[0] );
          break;
         }
       else
         init.tv_sec = now.tv_sec;
```

```
      }
  }
else
  {
   printf("SON-%d-SEGUNDOS\n", 60*win );

   fgets( line, 50, fifo_in );
   gettimeofday( &init, NULL );
   sscanf( line, "%s%d", aux, &data );

   printf("time1-%lu\n", init.tv_sec );

   printf("COMI-1-DATO\n" );

   fgets( line2, 50, fifo_in );
   gettimeofday( &now, NULL );

   printf("time2-%lu\n", now.tv_sec );

   printf("COMI-2do-DATO\n" );

   if ( ( (now.tv_sec) - (init.tv_sec) ) > win/2 )
     {
       /* es 1a vez y hay 0 datos */
       printf("PERO-ES-1a-VEZ\n");

       printf("data1-%d\n", data);
       printf("@%s -%d %d\n", aux, win, data );

       sscanf( line2, "%s%d", aux, &data );
       printf("data2-%d\n", data);
       printf("%s\n", line2 );
     }
   else
     {
       /* son 6*win datos en el fifo y ya comi 2 de ellos */
       printf("PERO-ESTAN-EN-COLA\n");

       printf("old-%d-data-%d\n", 60 , data);
       printf("@%s -%d %d\n", aux , 60*win ,data );

       sscanf( line2, "%s%d", aux, &data );
       printf("old-%d-data-%d\n", 59, data);
       printf("@%s -%d %d\n", aux, 59*win ,data );

       /* hay (2*6*win - 2) datos mas para comer */

       for ( i = (2*6*win - 2) ; i > 0 ; i-- )
         {
           signal( SIGINT, cleanup );
           signal( SIGTERM, cleanup );

           fgets( line, 50, fifo_in );

           gettimeofday( &now, NULL );

           sscanf( line, "%s%d", aux, &data );
           printf("old-%d-data-%d\n", i, data);
           printf("@%s -%d %d\n", aux, i*5 ,data );

           if ( i == (2*6*win - 2) )
             init.tv_sec = now.tv_sec;

           if ( ( now.tv_sec - init.tv_sec ) > win/2 )
             {
               printf("RESCALING-%s\n", argv[0] );
               break;
             }
           else
             init.tv_sec = now.tv_sec;
         }
     }
  }
```

```
    printf("^%s %d\n", argv[0], max );
    printf("_%s %d\n", argv[0], min );
    printf("<%s %d\n", argv[0], thres );

    do {

      signal( SIGINT, cleanup );
      signal( SIGTERM, cleanup );

      fgets( line, 50, fifo_in );

      sscanf( line, "%s%d", aux, &data );

      printf("data-%d\n", data);
      printf("%s\n", line );

    } while ( 1 );

    fclose( fifo_in );

    printf("EXIT-TING\n");
}
```

Anexo Nº 11

Listado del programa Quotechart.java

```
/*
 * @(#)QuoteChart.java1.15 95/12/09 James Gosling and Jim Graham
 *
 * Copyright (c) 1994-1995 Sun Microsystems, Inc. All Rights Reserved.
 *
 * Permission to use, copy, modify, and distribute this software
 * and its documentation for NON-COMMERCIAL or COMMERCIAL purposes and
 * without fee is hereby granted.
 * Please refer to the file http://java.sun.com/copy_trademarks.html
 * for further important copyright and trademark information and to
 * http://java.sun.com/licensing.html for further important licensing
 * information for the Java (tm) Technology.
 *
 * SUN MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF
 * THE SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED
 * TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE, OR NON-INFRINGEMENT. SUN SHALL NOT BE LIABLE FOR
 * ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR
 * DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.
 *
 * THIS SOFTWARE IS NOT DESIGNED OR INTENDED FOR USE OR RESALE AS ON-LINE
 * CONTROL EQUIPMENT IN HAZARDOUS ENVIRONMENTS REQUIRING FAIL-SAFE
 * PERFORMANCE, SUCH AS IN THE OPERATION OF NUCLEAR FACILITIES, AIRCRAFT
 * NAVIGATION OR COMMUNICATION SYSTEMS, AIR TRAFFIC CONTROL, DIRECT LIFE
 * SUPPORT MACHINES, OR WEAPONS SYSTEMS, IN WHICH THE FAILURE OF THE
 * SOFTWARE COULD LEAD DIRECTLY TO DEATH, PERSONAL INJURY, OR SEVERE
 * PHYSICAL OR ENVIRONMENTAL DAMAGE ("HIGH RISK ACTIVITIES").  SUN
 * SPECIFICALLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS FOR
 * HIGH RISK ACTIVITIES.
 */

import java.lang.Math;
import java.util.Hashtable;
import java.awt.*;
import java.applet.Applet;
import java.util.Vector;
import java.util.StringTokenizer;

/**
 * A simple class to read a stock quote data stream.
 *
 * @author     James Gosling
 * @author     Jim Graham
 * @version    1.15, 09 Dec 1995
 */
public
class QuoteChart extends Applet implements StockWatcher {
  float samples[] = new float[100];
  int sampletimes[] = new int[100];
  boolean started = true;
  public String stockSymbol = "SUNW";

  ChartScrollDaemon daemon = null;
  boolean fudge = false;
  StockStreamParser ssp;

  private float ymin = 1e20f, ymax = -1e20f;
  private float yprom = 0;
  private float prevymin, prevymax;
  private float high, low, quote, yesterday;
  private int quoteTime = 0;
  private int yesterdayTime = currentTime();
  private Color graphbg;
  private Color graphfg;
  private Color quotefg;
  private Color zerofg;
  private Color tickfg;
```

```
private Color legendfg;
private Color errorfg;
private Color yellowfg;
private int graphBorder = 2;
private String graphLabel = "Latest quote";
private String graphLabel2 = " ";
private static Hashtable colorCache = new Hashtable();
private Font fractfont;
private FontMetrics fractfm;
private boolean feedError = false;

private boolean changeHist = false;
private int histIndex = 0;
private int numHist = 0;
private int histLengths[];
private int histTotals[];
private int histUnits[];

private boolean pressed = false;
private boolean tracking = false;

private String quoteWhole = "momento...";
private String quoteFract = "";
private String diffWhole = "";
private String diffFract = "";
private String maxWhole = "";
private String maxFract = "";
private String minWhole = "";
private String minFract = "";

private static final int SECONDS = 0;
private static final int MINUTES = 1;
private static final int HOURS = 2;
private static final int DAYS = 3;

private static final String unitStrs[] = {
  " second", " minute", " hour", " day"
};
private static final int unitMults[] = {
  1, 60, 60 * 60, 24 * 60 * 60
};

static {
  colorCache.put("red", Color.red);
  colorCache.put("green", Color.green);
  colorCache.put("blue", Color.blue);
  colorCache.put("cyan", Color.cyan);
  colorCache.put("magenta", Color.magenta);
  colorCache.put("yellow", Color.yellow);
  colorCache.put("orange", Color.orange);
  colorCache.put("pink", Color.pink);
  colorCache.put("white", Color.white);
  colorCache.put("lightgray", Color.lightGray);
  colorCache.put("gray", Color.gray);
  colorCache.put("darkgray", Color.darkGray);
  colorCache.put("black", Color.black);
}

public static int currentTime() {
  return (int) (System.currentTimeMillis() / 1000L);
}

/**
 * Initialize the applet. Get attributes.
 */
public void init() {
  String s;

  s = getParameter("fudge");
  if (s != null) fudge = s.equals("true");
  graphbg = getColorAttribute("graphbg", Color.lightGray);
  graphfg = getColorAttribute("graphfg", Color.black);
  quotefg = getColorAttribute("quotefg", Color.blue);
  zerofg = getColorAttribute("zerofg", Color.red);
```

```
  tickfg = getColorAttribute("tickfg", Color.cyan);
  legendfg = getColorAttribute("legendfg", new Color(0, 128, 64));
  errorfg = getColorAttribute("errorfg", Color.red);
  yellowfg = getColorAttribute("yellowfg", Color.yellow );
  s = getParameter("graphborder");
  if (s != null) graphBorder = Integer.valueOf(s).intValue();
  s = getParameter("stock");
  if (s != null) stockSymbol = graphLabel = s;
  convertTimeScales(getParameter("history"));
  s = getParameter("label");
  if (s != null) graphLabel = s;

  s = getParameter("label2");
  if (s != null) graphLabel2 = s;

  Font font = new Font("TimesRoman", Font.PLAIN, 14);
  setFont(font);
  fractfont = new Font("TimesRoman", Font.PLAIN, 10);
  fractfm = getFontMetrics(fractfont);

}

void convertTimeScales(String s) {
  histIndex = 0;
  Vector histSpecs = new Vector();
  if (s != null) {
    StringTokenizer st = new StringTokenizer(s, ", ");
    while (st.hasMoreTokens()) {
      histSpecs.addElement(st.nextToken());
    }
  }
  if (histSpecs.size() == 0) {
    if (stockSymbol.equals("NEATO")) {
      histSpecs.addElement("20m");
    } else {
      histSpecs.addElement("1d");
    }
  }
  numHist = histSpecs.size();
  histLengths = new int[numHist];
  histUnits = new int[numHist];
  histTotals = new int[numHist];
  for (int i = 0; i < numHist; i++) {
    String histSpec = (String) histSpecs.elementAt(i);
    int max = histSpec.length();
    int timeUnits = SECONDS;
    switch (histSpec.charAt(max-1)) {
    case 'd':
      timeUnits = DAYS;
      max--;
      break;
    case 'h':
      timeUnits = HOURS;
      max--;
      break;
    case 'm':
      timeUnits = MINUTES;
      max--;
      break;
    case 's':
      max--;
      break;
    }
    histLengths[i] = Integer.parseInt(histSpec.substring(0, max));
    histUnits[i] = timeUnits;
    histTotals[i] = histLengths[i] * unitMults[timeUnits];
  }
}

  private Color getColorAttribute(String name, Color defcolor) {
     String s = getParameter(name);
     if (s == null)
         return defcolor;
     s = s.toLowerCase();
```

```
    defcolor = (Color) colorCache.get(s);
    if (defcolor != null)
        return defcolor;
    int colorval = java.lang.Integer.parseInt(s, 16);
    int r = (colorval >> 16) & 0xff;
    int g = (colorval >> 8) & 0xff;
    int b = colorval & 0xff;
    defcolor = new Color(r, g, b);
    colorCache.put(s, defcolor);
    return defcolor;
}

/**
 * Run the chart.
 * This method loops through the available URLs attempting
 * connections until one succeeds.  It then calls the readStream
 * method to handle the actual reading of data.
 * @see java.lang.Thread
 */
public synchronized void startSSP() {
    int now = currentTime();
    int histstart = now - histTotals[histIndex];
    adjustQuotes(now);
    for (int i = 0; i < sampletimes.length; i++) {
        if (sampletimes[i] > histstart) {
           histstart = sampletimes[i];
        }
    }
    ssp = new StockStreamParser(this, getDocumentBase(),
                                fudge, 60, now - histstart);
    ssp.setStock(stockSymbol);
    ssp.start();
}

public synchronized void feedEOF(StockStreamParser ssp) {
    if (this.ssp == ssp) {
        this.ssp = null;
    }
}

boolean verifySymbol(String symbol) {
    boolean matches = stockSymbol.equals(symbol);
    if (!matches) {
        System.out.println(stockSymbol + ": " + symbol );
    }
    return matches;
}

public void newHigh(String symbol, double value) {
    if (verifySymbol(symbol)) {
        high = (float) value;
        ymax = Math.max(high, ymax);
        ymin = Math.min(high, ymin);
    }
}

public void newLow(String symbol, double value) {
    if (verifySymbol(symbol)) {
        low = (float) value;
        ymax = Math.max(low, ymax);
        ymin = Math.min(low, ymin);
    }
}

public void newClose(String symbol, double value) {
    if (verifySymbol(symbol)) {
        yesterday = (float) value;
        yesterdayTime = 0;
        ymax = Math.max(yesterday, ymax);
        ymin = Math.min(yesterday, ymin);
        diffWhole = formatdiff(quote, yesterday);
        diffFract = formatdifffract(quote, yesterday);
    }
}
```

```
public void newQuote(String symbol, double value) {
    if (verifySymbol(symbol)) {
        insertQuote((float) value);
    }
}

public void newQuote(String symbol, double value, int time) {
    if (verifySymbol(symbol)) {
        insertQuote((float) value, time);
    }
}

public void setFeedError(String symbol, boolean error) {
    if (verifySymbol(symbol)) {
        feedError = error;
    }
}

public void flushQuotes() {
    repaint();
}

void insertQuote(float value) {
    insertQuote(value, currentTime());
}

void insertQuote(float value, int time) {
    int now = currentTime();

    int i;
    float maxtmp=0, mintmp=10000000;

    if (time <= 0) {
        // Fake data files contain relative history.
        time += now;
    }
    if (time < yesterdayTime) {
        yesterdayTime = time;
        yesterday = value;
    } else if (yesterday == 0) {
        yesterday = value;
    }
    if (time > now) {
        return;
    }
    int totalhist = histTotals[histIndex];
    int posdiff = (now - time) * samples.length / totalhist;
    int histpos = samples.length - posdiff - 1;
    if (histpos < 0) {
        return;
    }
    if (posdiff == 0 && sampletimes[histpos] != 0) {
        adjustQuotes(now);
    }
    if (time <= sampletimes[histpos]) {
        return;
    }
    //      ymax = Math.max(value, ymax);
    //      ymin = Math.min(value, ymin);

    for ( i=0; i < samples.length; i++)
      {
        yprom += samples[i];

        if ( samples[i] > maxtmp )
          maxtmp = samples[i];

        if ( samples[i] < mintmp )
          mintmp = samples[i];
      }

    ymax = maxtmp;
    ymin = mintmp;
```

```
    yprom = yprom / samples.length;

    if ( ymax == 0 )
      {
        ymax = 0;
        ymin = -1;
      }

    // System.out.println("ymin=" +ymin+ " ymax=" +ymax+ " yprom=" +yprom );

    samples[histpos] = value;
    sampletimes[histpos] = time;
    if (posdiff == 0) {
        quote = value;
        quoteWhole = formatwhole(quote);
        quoteFract = formatfract(quote);
        diffWhole = formatdiff(quote, yesterday);
        diffFract = formatdifffract(quote, yesterday);
    }
}

public void adjustQuotes(int now) {
    int totalhist = histTotals[histIndex];
    for (int i = 0; i < samples.length; i++) {
        int time = sampletimes[i];
        if (time != 0) {
            // For each valid sample, check its new index
            // and move it down if necessary.
            sampletimes[i] = 0;
            int posdiff = (now - time) * samples.length / totalhist;
            int histpos = samples.length - posdiff - 1;
            if (histpos < 0) {
                if (time < sampletimes[0]) {
                    continue;
                } else {
                    histpos = 0;
                }
            }
            samples[histpos] = samples[i];
            sampletimes[histpos] = time;
        }
    }
}

public void scroll() {
    int now = currentTime();
    if (started) {
        if (fudge && ssp == null) {
            insertQuote(quote + (float) (Math.floor(Math.random()*9)-4)/16,
                        now);
        } else {
            adjustQuotes(now);
        }
        repaint();
    }
}

private String formatwhole(float q) {

  if ( q == 0 )
    return Float.toString( 0 );

  return Float.toString((float) Math.floor(q));
}

private String formatfract(float q) {
    int numerator = (int) ((q - Math.floor(q)) * 16);
    int denominator = 16;
    while (numerator != 0 && (numerator & 1) == 0) {
        numerator >>= 1;
        denominator >>= 1;
    }
    return (numerator == 0) ? "" : (numerator + "/" + denominator);
}
```

```
private String formatdiff(float now, float before) {
    float diff = now - before;
    if (diff == 0)
        return "";
    int wholediff = (int) Math.floor(Math.abs(diff));
    String s = (diff < 0) ? "-" : "+";
    if (wholediff != 0)
        s = s + wholediff;
    return s;
}

private String formatdifffract(float now, float before) {
    float diff = now - before;
    return (diff == 0) ? "" : formatfract(Math.abs(diff));
}

private int drawStringWidth(Graphics g, FontMetrics fm,
                        String s, int x, int y) {
    g.drawString(s, x, y);
    return fm.stringWidth(s);
}

/**
 * Paint the current frame.
 */
public void paint(Graphics g) {
    Font font = getFont();
    FontMetrics fm = getFontMetrics(font);
    int sl = samples.length;
    int legendWidth = fm.stringWidth("999999999999"); /* + fractfm.stringWidth("9/16");*/
    int legendLeft = size().width - legendWidth;
    int fontheight = fm.getHeight();
    int fontascent = fm.getAscent();
    int fontdescent = fm.getDescent();
    int graphTop = 2*fontheight + graphBorder;
    int graphBottom = size().height - graphBorder;
    if (changeHist)
        graphBottom -= fontheight + 2;
    int graphLeft = graphBorder;
    int graphRight = legendLeft - graphBorder - 1;
    int graphWidth = graphRight - graphLeft + 1;
    int graphHeight = graphBottom - graphTop + 1;

    try {
        int sw;

        for (int i = 1; i <= graphBorder; i++) {
            g.draw3DRect(graphLeft - i, graphTop - i,
                         graphWidth-1 + i * 2, graphHeight-1 + i * 2,
                         false);
        }
        g.setColor(graphbg);
        g.fillRect(graphLeft, graphTop, graphWidth, graphHeight);

        g.setColor(quotefg);
        g.setFont(font);

        drawStringWidth(g, fm, graphLabel2, 0, fontascent);
        sw = drawStringWidth(g, fm, graphLabel + "=", 0, fontascent + fontheight);

        if (ymin > ymax) {
            g.drawString("momento...", sw, fontascent);
        } else {
            if (feedError) {
                g.setColor(errorfg);
            }

            sw += drawStringWidth(g, fm, quoteWhole, sw, fontascent + fontheight);

            //      g.setFont(fractfont);
            //      sw += drawStringWidth(g, fractfm, quoteFract, sw, fontascent);

            sw += 25;
```

```
                g.setFont(font);
                sw += drawStringWidth(g, fm, diffWhole, sw, fontascent+ fontheight );

                //      g.setFont(fractfont);
                //      g.drawString(diffFract, sw, fontascent);

                // g.setFont(font);
                // sw += drawStringWidth(g, fm, " p=" + formatwhole(yprom), sw, fontascent);

                // g.setFont(font);
                // g.drawString("dodil2...", 0, 2*fontascent);
            }

            if (ymin <= ymax) {
                float xf;
                float yo;
                float yf;
                int x, y;

                xf = (graphWidth - 1)/(float)sl;
                if (ymin < ymax) {
                    yo = ymax;
                    yf = (float) (graphHeight - 1) / (ymin - ymax);
                } else {
                    yo = ymax + 1;
                    yf = (float) (graphHeight - 1) / -2;
                }

                if (prevymax != ymax) {
                    maxWhole = formatwhole(ymax);
                    maxFract = formatfract(ymax);
                    prevymax = ymax;
                }
                if (prevymin != ymin) {
                    minWhole = formatwhole(ymin);
                    minFract = formatfract(ymin);
                    prevymin = ymin;
                }

                // System.out.println("-------ymin=" +ymin+ " ymax=" +ymax+ " yprom=" +yprom
);

                g.setColor(zerofg);
                g.setFont(font);
                sw = drawStringWidth(g, fm,  " max=" + maxWhole, legendLeft, graphTop +
fontascent);
                /*             g.setFont(fractfont);
                g.drawString(maxFract, legendLeft + sw, graphTop + fontascent);
                */

                g.setColor(yellowfg);
                g.setFont(font);
                sw = drawStringWidth(g, fm,  " x~=" + formatwhole(yprom), legendLeft,
(graphBottom - graphTop)/2 + graphTop );


                g.setFont(font);
                sw = drawStringWidth(g, fm,  " min=" + formatwhole( ymin ), legendLeft,
graphBottom - fontdescent);

                /*             g.setFont(fractfont);
                g.drawString(minFract, legendLeft + sw, graphBottom - fontdescent);
                */

                if (changeHist) {
                    g.setFont(font);
                    sw = drawStringWidth(g, fm,
                                         ("Last "
                                          +histLengths[histIndex]
                                          +unitStrs[histUnits[histIndex]]
                                          +((histLengths[histIndex] != 1)
                                            ? "s"
                                            : "")
```

```
                                +" shown"),
                               1, graphBottom + fontascent + 2);
            if (numHist > 1) {
                g.draw3DRect(0, graphBottom + 2,
                             sw + 2, fontheight + 2,
                             !pressed);
            }
        }

        if (ymin < ymax) {
            float tickScale;
            tickScale = (float) Math.floor(Math.log(ymax - ymin)
                                           / Math.log(2.0)) - 1.0f;
            float tickFreq = (float) Math.pow(2.0, tickScale);
            g.setColor(tickfg);
            for (float ytick = (float) (Math.floor(ymin / tickFreq)
                                        * tickFreq);
                 ytick <= ymax; ytick += tickFreq) {
                if (ytick < ymin)
                    continue;
                y = (int)((ytick-yo)*yf) + graphTop;
                g.drawLine(graphLeft, y, graphRight, y);
            }
        }
        y = (int)((yesterday-yo)*yf) + graphTop;
        if ( y > graphTop )
          {
            g.setColor(zerofg);
            g.drawLine(graphLeft, y, graphRight, y);
          }

        g.setColor(graphfg);
        int px = graphLeft;
        int py = y = (int)((yesterday-yo)*yf) + graphTop;

        if ( y < graphTop )
          y = graphTop;

        if ( py < graphTop )
          py = graphTop;

        for (int i = 0; i < sl; i++) {
            if (sampletimes[i] != 0) {
                y = (int)((samples[i]-yo)*yf) + graphTop;

                if ( y < graphTop )
                  y = graphTop;

                if (i == 0) {
                    py = y;
                }
            }
            x = (int) (i * xf) + graphLeft;
            g.drawLine(px, py, x, y);
            px = x;
            py = y;
        }
    }
} catch(Exception e) {
    e.printStackTrace();
}
}

public void mouseDown(int x, int y) {
    if (numHist > 1 && y > (size().height - 2
                            - getFontMetrics(getFont()).getHeight())) {
        pressed = tracking = true;
        repaint();
    }
}

public void mouseDrag(int x, int y) {
    if (tracking) {
        if (y <= (size().height - 2
```

```
                    - getFontMetrics(getFont()).getHeight())) {
                if (pressed) {
                    pressed = false;
                    repaint();
                }
            } else {
                if (!pressed) {
                    pressed = true;
                    repaint();
                }
            }
        }
    }

    public void mouseUp(int x, int y) {
        tracking = false;
        if (pressed) {
            stop();
            histIndex++;
            if (histIndex >= histLengths.length) {
                histIndex = 0;
            }
            for (int j = 0; j < samples.length; j++) {
                sampletimes[j] = 0;
            }
            ymin = Math.min(Math.min(quote, yesterday), low);
            ymax = Math.max(Math.max(quote, yesterday), high);
            start();
            pressed = false;
            repaint();
        }
    }

    /**
     * Start the applet by forking an animation thread.
     */
    public synchronized void start() {
        started = true;
        if (ssp == null) {
            startSSP();
        }
        if (daemon == null) {
            int deltat = histTotals[histIndex] / samples.length;
            daemon = new ChartScrollDaemon(this, deltat);
            daemon.start();
        }
    }

    /**
     * Stop the applet.
     */
    public synchronized void stop() {
        started = false;
        if (ssp != null) {
            ssp.stop();
            ssp = null;
        }
        if (daemon != null) {
            daemon.stopchart();
            daemon = null;
        }
    }
}

class ChartScrollDaemon extends Thread {
    QuoteChart chart;
    int delay;

    public ChartScrollDaemon(QuoteChart chart, int delay) {
        this.chart = chart;
        this.delay = delay;
    }

    public synchronized void run() {
```

```
        while (chart != null && chart.daemon == this) {
            chart.scroll();
            try {
                wait(delay * 1000);
            } catch (InterruptedException e) {
                break;
            }
        }
    }

    public synchronized void stopchart() {
        chart = null;
        notify();
    }
}
```