HERRAMIENTA PARA MONITORIZACIÓN DE DIRECCIONES IP Y SERVICIOS EN LA RED DE DATOS DE LA UNIVERSIDAD INDUSTRIAL DE SANTANDER

JOHN FREDY RICO PRADILLA 1942248
MARYORIS RICO PRADILLA 1951950

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍA ELÉCTRICA ELECTRÓNICA Y TELECOMUNICACIONES Y ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMATICA
2004

HERRAMIENTA PARA MONITORIZACIÓN DE DIRECCIONES IP Y SERVICIOS EN LA RED DE DATOS DE LA UNIVERSIDAD INDUSTRIAL DE SANTANDER

JOHN FREDY RICO PRADILLA 1942248
MARYORIS RICO PRADILLA 1951950

Trabajo presentado como requisito parcial para optar al título de Ingeniero Electrónico e Ingeniero de Sistemas

Director
PHD. OSCAR GUALDRON GONZALEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍA ELÉCTRICA ELECTRÓNICA Y TELECOMUNICACIONES Y ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMATICA
2004

AGRADECIMIENTOS

Los autores expresan sus agradecimientos a:

Nuestro director de Proyecto, Phd. OSCAR GUALDRON GONZALEZ por su orientación, recomendaciones, exigencias, empeño e interés en el desarrollo de este trabajo.

La UNIVERSIDAD INDUSTRIAL DE SANTANDER, especialmente al laboratorio de electrónica por permitirnos realizar las prácticas correspondientes al desarrollo de nuestro trabajo de grado.

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

TITULO
HERRAMIENTA PARA MONITORIZACIÓN DE DIRECCIONES IP Y SERVICIOS EN LA RED DE DATOS DE LA UNIVERSIDAD INDUSTRIAL DE SANTANDER "ESCANEADOR DE RED[*]

AUTORES
JOHN FREDY RICO PRADILLA
MARYORIS RICO PRADILLA[**]

PALABRAS CLAVES
Monitorización, Direcciones IP, Escáner de Red, Software, Sockets, Puertos TCP, Puertos UDP

DESCRIPCION

El escáner de red es un software que permite monitorizar los servicios, direcciones IP en uso y la red de servidores que ofrece la universidad, consultando una base de datos elaborada previamente donde se encuentran almacenadas las direcciones IP y los servicios autorizados por el administrador de la red de datos de la UIS.

El software realiza cuatro procesos de escaneo de direcciones IP y puertos; el primer proceso permite monitorizar una dirección IP para saber si esta disponible en ese momento y si está registrada, además analiza los puertos TCP y UDP más conocidos contrastando esta información con la autorizada en la base de datos. El segundo proceso escanea una subred, tomando como entrada una dirección IP y una máscara de Subred para monitorear qué direcciones IP se encuentran en uso y corroborar si están registradas. Con tercer proceso se revisa un rango de direcciones IP que verifica si las direcciones que están en uso se encuentran autorizadas por el administrador de la red. La activación del último proceso empieza un chequeo automático cada cierto intervalo de tiempo a las direcciones IP de los servidores registrados en la base de datos con el fin de saber si ha ocurrido una falla y de esta manera tomar los correctivos necesarios.

Este software también cuenta con la opción de revisar en la base de datos los registros de las eventualidades almacenados tales como fecha, hora, direcciones IP y servicios no autorizados durante los procesos de escaneo.

Con este sistema se contribuye a la monitorización de los dispositivos de la red administrada por la División de Servicios de Información y al mejoramiento de la gestión de la red en la UIS.

---

[*] Proyecto de Grado

[**] Facultad de Ciencias Fisicomecánicas, Ingeniería Eléctrica Electrónica y Telecomunicaciones e Ingeniería de Sistemas e Informática. Director: Phd Oscar Gualdrón González

TITLE
TOOL FOR MONITORIZATION OF IP ADDRESS AND SERVICES IN THE NET OF DATA FROM INDUSTRIAL UNIVERSITY OF SANTANDER "NETWORK SCANER" [*]

AUTHORS:
JOHN FREDY RICO PRADILLA
MARYORIS RICO PRADILLA[**]

KEY WORDS:
Monitorization, Ip address, Network scan, Software, TCP Ports, UDP Ports.

DESCRIPTION

Network Scanner is a software that allow monitoring the service, IP address in use and Net of services that offer the University, consulting a base of data previously elaborated, where IP address and services authorized by the administrator of net data from UIS stored.

Software realices four processes of scanning Ip address and ports; First process allow to monitoring an Ip address to know if it`s available in that moment and Ip it´s registed, Besides, it analyzes TCP and UDP port the most well known ports comparing this information with the authorized one in base of data. Second process scanning a Subnet, taking as entry an IP direction and a mask of subnet monitoring what IP address are in use and corroborate if they are registed. Third process checking a rank of IP address that verify if the administrator of net. The activation from the last of time, to the Ip address from servants registed in base of data, with the purpose to know if have been a flaw and this way to take the necessary correctives.

This software has the option to inspect in base of data the registers of the eventualities stored such as, date, hour, Ip address and services no authorized during the processes of scanning.

With this system contribute to monitorization of the devices from net administed by the Division of Service of Information and to improvement of management from the net at UIS.

---

[*] Project of Grade

[**] Faculty of Physicalmechanics Sciences, Electrical Electronis Engeneering and Telecommunication and Engineering of System and Computer Science.
Director: Phd Oscar Gualdrón González

1. DESCRIPCIÓN DEL PROYECTO

1.1 Planteamiento del problema

La UNIVERSIDAD INDUSTRIAL DE SANTANDER “UIS” ente educativo por excelencia del nororiente colombiano, se esfuerza cada día por brindar un mejor servicio a la comunidad que la integra. Esta misión se logra gracias al compromiso de todos los miembros de la comunidad académica.

Los servicios de transmisión de datos se han convertido en un elemento fundamental para toda la organización; la universidad no es la excepción y para ello la División de Servicios de Información incluye entre sus responsabilidades garantizar un buen desempeño de los dispositivos que conforman esta red.

Para abordar algunos aspectos de esta problemática se hace necesario un escáner de red que verifique las direcciones IP en uso, los servicios disponibles en uso y los servidores de la universidad.

Por otra parte es importante conocer con gran certeza los servicios que están autorizados por el administrador de la red, para cada estación, con el fin de compararlos con aquellos que se encuentran disponibles en cada una de las estaciones.

También es necesario tener un reporte de anomalías en una base de datos, donde se registren eventos detectados incluyendo particularidades del mismo como la dirección IP de la estación, los servicios no autorizados, la fecha y la

hora para posteriormente tomar acciones correctivas que permitan brindar soluciones rápidas al problema encontrado.

El objetivo de este trabajo es implementar una herramienta de software que satisfaga todas las necesidades anteriormente descritas y que represente una contribución para el mejoramiento de la gestión de red en la UIS.

1.2 OBJETIVOS

1.2.1 Objetivo General

Diseñar e implementar una herramienta de software que permita la monitorización de direcciones IP y servicios instalados en la red de datos que ofrece la Universidad Industrial de Santander.

1.2.2 Objetivos Específicos

- Identificar el problema referente al control de la asignación de direcciones IP y servicios no autorizados en la red de datos de la UIS.
- Seleccionar la plataforma y el lenguaje de programación para la implementación de la herramienta y la interfaz del usuario, que permita un buen desempeño, fácil manejo y desarrollo de las tareas necesarias.
- Diseñar una herramienta software que permita realizar un escaneado de red y proporcione información sobre los servicios y direcciones IP que no se encuentran autorizadas por el administrador de la red.
- Implementar la herramienta de software que brinde control y soporte para la verificación de las direcciones IP en uso y los servicios instalados.

- Validar el correcto funcionamiento de la herramienta contrastando los datos obtenidos con los establecidos en una base de datos.
- Diseño de la interfaz de usuario para una buena presentación del sistema y fácil entendimiento de los resultados que procesa el software.

1.3 JUSTIFICACIÓN

Dada la importancia de controlar la asignación de direcciones IP en uso y la red de servidores que ofrece la universidad, se hace necesario diseñar una herramienta que realice las tareas de control y verificación, que detecte el uso de servicios y direcciones IP no autorizadas, y suministre información oportuna de las fallas de servidores.

La red de datos institucional presenta un permanente crecimiento y a su vez el número de direcciones IP asignadas, lo que requiere un mayor control para evitar conflictos que puedan generar retrasos en el funcionamiento normal de las diferentes subredes instaladas en la universidad; estos inconvenientes se pueden solucionar de una manera más rápida utilizando una aplicación que realice la tarea de escáner de red.

El escáner de red permite realizar trabajos de monitorización de servicios y direcciones IP ayudando a identificar problemas que frecuentemente se presentan en el manejo de red de la universidad, como son el uso de servicios no autorizados y las fallas de servidores.

Esta aplicación además de facilitar la solución, puede ser modificada y adaptada de acuerdo a las necesidades que se requieran por haber sido desarrollada en la institución.

Existen en el mercado un gran número de aplicaciones que monitorizan y realizan escaneo de una red de datos, pero ninguna que se ajuste

plenamente a las necesidades de la institución, porque no está personalizada.

Por otra parte la adquisición de un software que no haya sido desarrollada en la universidad impediría su mejoramiento e integración con proyectos futuros ya que no se contaría con su código fuente para poder hacer las modificaciones y personalización de acuerdo a los requerimientos solicitados.

Contar con el código fuente de una herramienta software hace que pueda adaptarse y se potencie el desarrollo y la investigación de nuevas herramientas para gestión de red.

Es importante resaltar que la universidad requiere trabajos de investigación que ayuden al mejoramiento de los servicios que se están prestando a la comunidad, algunas ventajas que presenta la implementación de esta herramienta de software son las siguientes:

- Eliminar la necesidad de adquirir un software que no se pueda manipular, por no tener el código fuente.
- Permitir la integración de nuevos servicios para el mejoramiento y la actualización del software.
- Reportar las fallas y servicios no autorizados en una base de datos oportunamente, permitiendo mayor control de la red llevando dicho registro.

2. PROTOCOLOS TCP/IP Y PRINCIPIOS DE GESTION DE REDES

El inicio de los protocolos que fueron usados como base para lo que hoy conocemos como Internet o familia de protocolos TCP/IP se remonta a finales de los años 60 y fue iniciado por el departamento de defensa de los Estados Unidos con la creación de una red llamada ARPANET.

Lo que se busca con TCP/IP es una manera de interconectar hosts de forma general que sea válido para cualquier plataforma, sistema operativo y tipo de red. La familia de protocolos que se eligieron para permitir que Internet sea una *Red de redes* es TCP/IP. Se habla de una familia de protocolos ya que son varios los protocolos que la constituyen, aunque se conoce de manera más abreviada como protocolo TCP/IP.

El modelo TCP/IP está formado por las siguientes capas en donde cada capa se encarga de un determinado proceso:

Capa de aplicación
Capa de transporte
Capa de inter-red
Capa de acceso a la red
Capa física

Figura 2.1 Modelo en capas de TCP/IP

El nivel más bajo es la *capa física*. Se refiere al medio físico por el cual se transmite la información, hace referencia a los conectores y las señales que representan ceros y unos por ejemplo las tarjetas de interfaz de red de Ethernet.

La *capa de acceso a la red.* Una vez que tenemos un enlace físico, se establece la forma de cómo se transmiten datos al medio. En esta capa no se especifica ningún protocolo, los datos se organizan en unidades llamadas tramas donde cada trama tiene una cabecera que contiene una dirección e información de control y una cola que se usa para detectar errores.

La capa de Acceso a la Red y la capa Física no hacen parte del protocolo TCP/IP.

La capa de inter-red. El protocolo principal en esta capa es IP. No es un protocolo fiable ni es orientado a conexión.

La *capa de transporte* En esta capa se encuentran dos protocolos (TCP y UDP) que nos ayudan a establecer comunicación de extremo a extremo. Uno de ellos es orientado a conexión (TCP) y el otro no (UDP).

En la capa de aplicación se encuentran los protocolos de más alto nivel y son con los cuales estamos más familiarizados y los que nos prestan servicios como: correo electrónico, páginas Web, FTP, TELNET.

La interacción entre hosts de grandes y pequeñas redes se hace posible gracias a que cada computador que la compone se identifica plenamente con una única dirección (dirección IP) y de esta manera pueden comunicarse también con redes exteriores interconectadas por enrutadores.

2.1 Direcciones IP

El mecanismo mediante el cual dos o más host se pueden comunicar y compartir información sin tener que preocuparse de que la información llegue al host destino y no a otros es porque cada host se identifica de manera única mediante una dirección IP. En el caso de los computadores que están

conectados a internet no pueden existir de manera simultanea dos direcciones idénticas. La única forma de que se estén usando dos direcciones IP iguales simultáneamente es que sean direcciones IP privadas.

Las direcciones IP se clasifican en:

- **Direcciones IP públicas**. Es necesario tener una IP pública para usar internet y son accesibles desde cualquier host conectado a la red de redes.
- **Direcciones IP privadas (reservadas)**. Este rango de direcciones se usan para conectar redes o subredes privadas ya que estas direcciones no son accesibles desde Internet, pero si pueden acceder a Internet por medio de un router con una dirección IP pública.

A su vez, las direcciones IP pueden ser:

- **Direcciones IP estáticas (fijas)**. Son las direcciones que se asignan de manera permanente a un host, normalmente estas direcciones las usan los servidores o host que deben estar siempre accesibles. Para poder usar una dirección IP fija hay que contratarla con un proveedor de servicios de Internet.
- **Direcciones IP dinámicas**. Son las direcciones que se asignan de manera momentánea cada vez que un usuario se conecta a internet, normalmente esto ocurre cuando se usa el servicio de internet por medio de línea telefónica, partiendo del hecho de que no todos los usuarios se conecten al mismo tiempo, un proveedor de internet podrá tener más usuarios que direcciones IP.

Las direcciones IPv4 están formadas por 4 octetos que normalmente se representan de forma decimal 123.123.13.133 donde cada paquete separado por un punto puede variar de 0 a 255

Las direcciones IP también se pueden representar de manera hexadecimal, desde la 00.00.00.00 hasta la FF.FF.FF.FF o en binario, desde la 00000000.00000000.00000000.00000000 hasta la 11111111.11111111.11111111.11111111.

Dependiendo del número de direcciones IP que se necesiten para cada red, se han clasificado de la siguiente manera A, B, C, D y E. La clase D se usa para multidifusión y las direcciones de clase E están reservadas.

	0 1 2 3	4	8	16	24	31
Clase A	0	Red	Host			
Clase B	1 0	Red		host		
Clase C	1 1 0	Red			host	
Clase D	1 1 1 0	Grupo de multicast (multidifusión)				
Clase E	1 1 1 1	(direcciones reservadas: no se pueden utilizar)				

Tabla 2.1 Clasificación de las redes IP

Nota: *Las direcciones usadas en Internet están definidas en el RFC 1166.*

2.2 Direcciones IP especiales y reservadas

Existen algunos rangos de direcciones IP que se encuentran comprendidos en las clases A, B y C que se asignan solo para propósitos especiales, como por ejemplo indicar que se trata de mi propio Host ó direccionar una subred privada que no esta conectada directamente a internet y si lo hace es por medio de un servidor Proxy con una IP pública.

Bits de red	**Bits de host**	**Significado**	**Ejemplo**
todos 0		Mi propio host	0.0.0.0
todos 0	Host	Host indicado dentro de mi red	0.0.0.10
Red	todos 0	Red indicada	192.168.1.0
todos 1		Difusión a toda la red	255.255.255.255
Red	todos 1	Difusión a la red indicada	192.168.1.255
127	Cualquier valor válido de host	Loopback (mi propio host)	127.0.0.1

Tabla 2.2 Direcciones IP especiales y reservadas

En la tabla 2.3 se muestra rangos de direcciones que se usan para direccional redes privadas.

Clase	**Rango de direcciones reservadas de redes**
A	10.0.0.0
B	172.16.0.0 – 172.31.0.0
C	192.168.0.0 - 192.168.255.0

Tabla 2.3 Rango de Direcciones IP Reservadas

2.3 Máscara de subred

Es aquella por la cual nos valemos para poder hacer subredes del tamaño más apropiado y que se ajuste de acuerdo a nuestras necesidades. También nos puede indicar si otra dirección IP pertenece a nuestra subred.

La siguiente tabla muestra las máscaras de subred correspondientes a cada clase:

Clase	Máscara de subred
A	255.0.0.0
B	255.255.0.0
C	255.255.255.0

Tabla 2.4 máscaras de subred correspondientes a cada clase.

Si expresamos la máscara de subred de clase A en notación binaria, tenemos:

11111111.00000000.00000000.00000000

En el caso anterior los unos nos indican la dirección de la subred y los ceros la dirección que puede tomar cada host.

2.4 Protocolo IP

Es el protocolo de la capa de interred o de internet, su principal objetivo es el de interconectar redes llevando la información en datagramas con las siguientes características:

- Es no orientado a conexión debido a que cada uno de los paquetes puede seguir rutas distintas entre el origen y el destino. Entonces pueden llegar duplicados o desordenados.
- Es no fiable porque los paquetes pueden perderse, dañarse o llegar retrasados.

Nota: El protocolo IP está definido en el RFC 791.

2.5 Protocolo ICMP

El principal objetivo del protocolo ICMP (Protocolo de mensajes de control de Internet) es el de informar acerca de errores y control en lo nodos. Una de las herramientas más conocidas que usa el protocolo ICMP es el comando PING que sirve para saber si una dirección IP esta siendo utilizada en ese momento.

E protocolo ICMP únicamente informa de incidencias en la red pero no toma ninguna decisión. Esto será responsabilidad de las capas superiores. Los mensajes ICMP viajan en el campo de datos de un datagrama IP.

Nota: El formato y significado de cada mensaje ICMP está documentado en la RFC 792.

A continuación se muestra una tabla que contiene el significado de los mensajes y campos más comunes usados por ICMP.

Campo de tipo	Tipo de mensaje ICMP
0	Respuesta de eco (*Echo Reply*)
3	Destino inaccesible (*Destination Unreachable*)
4	Disminución del tráfico desde el origen (*Source Quench*)

5	Redireccionar (cambio de ruta) (*Redirect*)
8	Solicitud de eco (*Echo*)
11	Tiempo excedido para un datagrama (*Time Exceeded*)
12	Problema de Parámetros (*Parameter Problem*)
13	Solicitud de marca de tiempo (*Timestamp*)
14	Respuesta de marca de tiempo (*Timestamp Reply*)
15	Solicitud de información (obsoleto) (*Information Request*)
16	Respuesta de información (obsoleto) (*Information Reply*)
17	Solicitud de máscara (*Addressmask*)
18	Respuesta de máscara (*Addressmask Reply*)

Tabla 2.5 Campos y Tipos de mensajes ICMP

2.6 Solicitud y respuesta de eco

Como sabemos que una de las utilidades más comunes para saber si un host está conectado a la red es usando el comando PING, a continuación explicaremos brevemente su funcionamiento.

Los mensajes de solicitud y respuesta de eco, tipos 8 y 0 respectivamente, se utilizan para comprobar si existe comunicación entre 2 hosts a nivel de la capa de red. Estos mensajes comprueban que las capas física, acceso al medio y interred estén correctas. Sin embargo, no dicen nada de las capas de transporte y de aplicación las cuales podrían estar mal configuradas.

1. A envía un mensaje ICMP de tipo 8 (*Echo*) a B
2. B recibe el mensaje y devuelve un mensaje ICMP de tipo 0 (*Echo Reply*) a A
3. A recibe el mensaje ICMP de B y muestra el resultado en pantalla

Figura 2.2 Ilustración del comando ping

Nota: El comando ping 127.0.0.1 informa de si están correctamente implementados los protocolos TCP/IP en nuestro host. No informa de si la tarjeta de red de nuestro host está correcta.

Algunos hosts en Internet tienen deshabilitadas las respuestas de eco como medida de seguridad. En estos casos hay que utilizar otros mecanismos para detectar si responde.

2.7 Puertos

Para lograr que dos host se comuniquen es necesario que cada uno posea además de una dirección IP un puerto disponible, para cada dirección IP están asociados 65536 puertos esto significa que un host puede establecer comunicación con muchos host al mismo tiempo usando para cada comunicación la misma dirección IP y un puerto diferente. Las aplicaciones utilizan estos puertos para recibir y transmitir mensajes.

Cuando se establece una comunicación se asigna un número de puerto disponible de manera dinámica que normalmente es superior al 1024 ya que los puertos con números inferiores están asignados para tareas específicas y son los llamados puertos “bien conocidos” que son usados por aplicaciones

servidoras. Estos puertos están definidos en la RFC 1700. A continuación se enumeran los puertos "*well-known"* más usuales:

Palabra clave	Puerto	Descripción
	0/tcp	Reserved
	0/udp	Reserved
tcpmux	1/tcp	TCP Port Service Multiplexer
Rje	5/tcp	Remote Job Entry
Echo	**7/tcp/udp**	**Echo**
Discard	9/tcp/udp	Discard
systat	11/tcp/udp	Active Users
daytime	13/tcp/udp	Daytime
Qotd	17/tcp/udp	Quote of the Day
chargen	19/tcp/udp	Character Generador
ftp-data	20/tcp	File Transfer [Default Data]
ftp	**21/tcp**	**File Transfer [Control]**
telnet	**23/tcp**	**Telnet**
Smtp	**25/tcp**	**Simple Mail Transfer**
time	37/tcp/udp	Time
nameserver	42/tcp/udp	Host Name Server
nicname	43/tcp/udp	Who Is
domain	**53/tcp/udp**	**Domain Name Server**
Bootas	67/udp/udp	Bootstrap Protocol Server
tftp	69/udp	Trivial File Transfer
gopher	70/tcp	Gopher
Finger	79/tcp	Finger
www-http	**80/tcp**	**World Wide Web HTTP**
dcp	93/tcp	Device Control Protocol
supdup	95/tcp	SUPDUP
hostname	101/tcp	NIC Host Name Server
iso-tsap	102/tcp	ISO-TSAP
Gppitnp	103/tcp	Genesis Point-to-Point Trans Net
rtelnet	107/tcp/udp	Remote Telnet Service
pop2	109/tcp	Post Office Protocol - Version 2
pop3	**110/tcp**	**Post Office Protocol - Version 3**
Sunrpc	111/tcp/udp	SUN Remote Procedure Call
auth	113/tcp	Authentication Service
sftp	115/tcp/udp	Simple File Transfer Protocol
nntp	**119/tcp**	**Network News Transfer Protocol**

Ntp	123/udp	Network Time Protocol
pwdgen	129/tcp	Password Generator Protocol
netbios-ns	137/tcp/udp	NETBIOS Name Service
netbios-dgm	138/tcp/udp	NETBIOS Datagram Service
netbios-ssn	**139/tcp/udp**	**NETBIOS Session Service**
snmp	161/udp	SNMP
Snmptrap	162/udp	SNMPTRAP
Irc	194/tcp	Internet Relay Chat Protocol

Tabla 2.6 Nombres de los puertos más conocidos

Los puertos tienen una memoria intermedia *(buffer)* situada entre los programas de aplicación y la red. De tal forma que las aplicaciones transmiten la información a los puertos. Aquí se va almacenando hasta que pueda enviarse por la red. Una vez transmitida la información se recibe en el puerto destino donde se irá almacenando hasta que la aplicación esté preparada para recibirla.

2.8 Protocolo UDP

El protocolo UDP es un protocolo del nivel de transporte que usa datagramas para su comunicación, debido a que este protocolo no es orientado a conexión éste incorpora información suficiente en el datagrama para que pueda llegar a su destino, pero la aplicación destino no devuelve un acuse de recibido. El protocolo UDP también al igual que TCP usa puertos para su comunicación, se vale de IP para su transporte y posee las siguientes características:

- *No orientado a conexión.* No se establece una conexión previa con el otro extremo para transmitir un mensaje UDP. Los mensajes se envían sin más y éstos pueden duplicarse o llegar desordenados al destino.
- *No fiable*. Los mensajes UDP se pueden perder o llegar dañados.

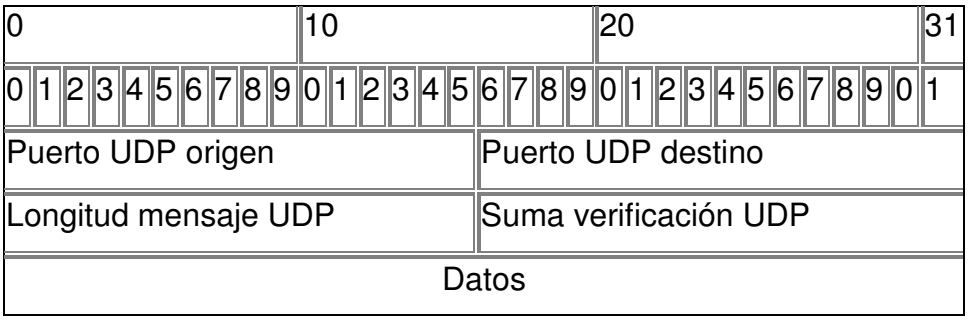


Figura 2.5 *Formato del mensaje UDP*

- **Puerto UDP de origen** (16 bits, opcional). Número de puerto de la máquina origen.
- **Puerto UDP de destino** (16 bits). Número de puerto de la estación destino.
- **Longitud del mensaje UDP** (16 bits). Especifica la longitud medida en bytes del mensaje UDP incluyendo la cabecera. La longitud mínima es de 8 bytes.
- **Suma de verificación UDP** (16 bits, opcional). Suma de comprobación de errores del mensaje. Para su cálculo se utiliza una *pseudo-cabecera* que también incluye las direcciones IP origen y destino. Para conocer estos datos, el protocolo UDP debe interactuar con el protocolo IP.
- **Datos**. Aquí viajan los datos que se envían a las aplicaciones.

2.9 Protocolo TCP

Es un protocolo del nivel de transporte que se usa para establecer comunicaciones orientadas a conexión asegurando que la transmisión sea

fiable y exitosa, este protocolo al igual de los demás protocolos se vale de IP para trasportar sus segmentos. TCP cuenta principalmente con las siguientes características:

- *Orientado a conexión*. Es necesario establecer una conexión previa entre las dos estaciones antes de poder transmitir algún dato. A través de esta conexión los datos llegarán siempre a la aplicación destino de forma ordenada y sin duplicados. Finalmente, es necesario cerrar la conexión.
- *Fiable.* La información que envía el emisor llega de forma correcta al destino.

El protocolo TCP también usa el concepto de puertos para su comunicación, de esta manera es posible que se puedan establecer varias conexiones bidireccionales al mismo tiempo.

2.9.1 Conexiones

Para establecer una conexión son necesarios dos pares *dirección IP:puerto*. No puede haber dos conexiones iguales en un mismo instante en toda la Red. Aunque bien es posible que un mismo host tenga dos conexiones distintas y simultáneas utilizando puertos diferentes.

2.9.2 Formato del segmento TCP

Para poder transmitir una información el protocolo TCP divide en uno o más segmentos para su transmisión. Cada uno de estos segmentos viaja en el campo de datos de un datagrama IP. Para facilitar el control de flujo de la

información los bytes de la aplicación se numeran. De esta manera, cada segmento indica en su cabecera el primer byte que transporta. A continuación veremos como está conformado un segmento TCP y una breve descripción de lo que significa cada uno de los campos que lo compone.

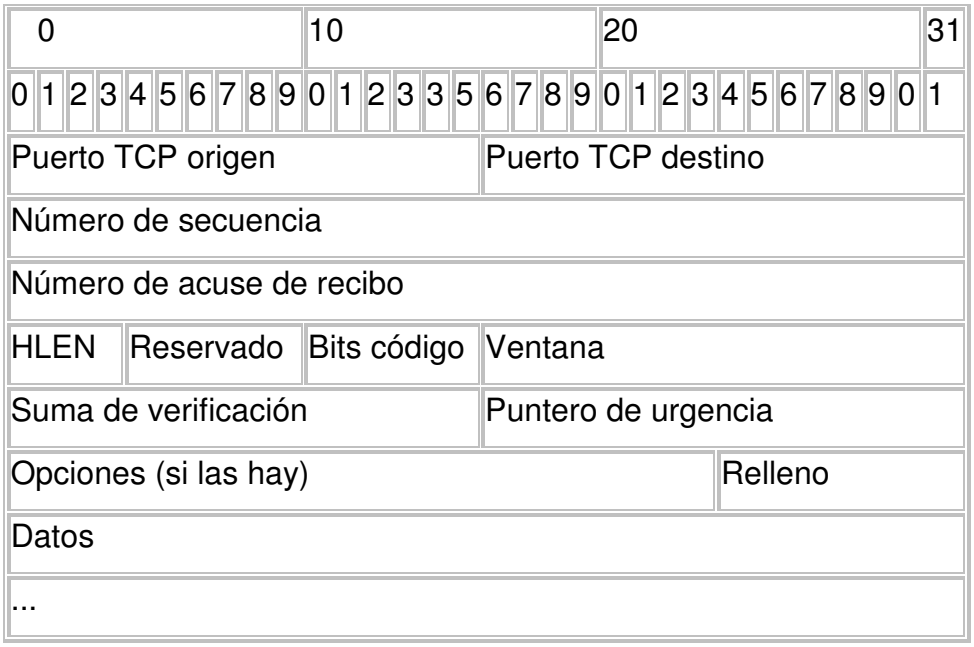


Figura 2.6 Formato del datagrama TCP

- **Puerto fuente** (16 bits). Puerto de la estación origen. Al igual que el puerto destino es necesario para identificar la conexión actual.
- **Puerto destino** (16 bits). Puerto de la estación destino.
- **Número de secuencia** (32 bits). Indica el número de secuencia del primer byte que trasporta el segmento.
- **Número de acuse de recibo** (32 bits). Indica el número de secuencia del siguiente byte que se espera recibir. Con este campo se indica al otro extremo de la conexión que los bytes anteriores se han recibido correctamente.

- **HLEN** (4 bits). Longitud de la cabecera medida en múltiplos de 32 bits (4 bytes). El valor mínimo de este campo es 5, que corresponde a un segmento sin datos (20 bytes).
- **Reservado** (6 bits). Bits reservados para un posible uso futuro.
- **Bits de código** o indicadores (6 bits). Los bits de código determinan el propósito y contenido del segmento. A continuación se explica el significado de cada uno de estos bits (mostrados de izquierda a derecha) si está a 1.
- **URG.** El campo *Puntero de urgencia* contiene información válida.
- **ACK.** El campo *Número de acuse de recibo* contiene información válida, es decir, el segmento actual lleva un ACK. Observemos que un mismo segmento puede transportar los datos de un sentido y las confirmaciones del otro sentido de la comunicación.
- **PSH.** La aplicación ha solicitado una operación *push* (enviar los datos existentes en la memoria temporal sin esperar a completar el segmento).
- **RST.** Interrupción de la conexión actual.
- **SYN.** Sincronización de los números de secuencia. Se utiliza al crear una conexión para indicar al otro extremo cual va a ser el primer número de secuencia con el que va a comenzar a transmitir (veremos que no tiene porqué ser el cero).
- **FIN.** Indica al otro extremo que la aplicación ya no tiene más datos para enviar. Se utiliza para solicitar el cierre de la conexión actual.

- **Ventana** (16 bits). Número de bytes que el emisor del segmento está dispuesto a aceptar por parte del destino.
- **Suma de verificación** (24 bits). Suma de comprobación de errores del segmento actual. Para su cálculo se utiliza una *pseudo-cabecera* que también incluye las direcciones IP origen y destino.
- **Puntero de urgencia** (8 bits). Se utiliza cuando se están enviando datos urgentes que tienen preferencia sobre todos los demás e indica el siguiente byte del campo *Datos* que sigue a los datos urgentes. Esto le permite al destino identificar donde terminan los datos urgentes. Nótese que un mismo segmento puede contener tanto datos urgentes (al principio) como normales (después de los urgentes).
- **Opciones** (variable). Si está presente únicamente se define una opción: el tamaño máximo de segmento que será aceptado.
- **Relleno.** Se utiliza para que la longitud de la cabecera sea múltiplo de 32 bits.
- **Datos.** Información que envía la aplicación.

2.10 Nombres de dominio

Como no es muy sencillo acordarse de direcciones IP para visitar páginas, en internet se usan nombres de dominio del estilo de www.uis.edu.co. Para que esto sea posible es necesario un proceso previo de conversión de nombres de dominio a direcciones IP, ya que el protocolo IP requiere direcciones IP al enviar sus datagramas. Este proceso se conoce como resolución de

nombres. Existen protocolos encargados de hacer este tipo de conversiones, cuyo funcionamiento es totalmente transparente para los usuarios.

2.11 LOS SOCKETS

Los *sockets* son canales o mecanismos de comunicación entre dos host que permiten que un proceso emita o reciba información con otro proceso.

También pudiera verse como un punto de comunicación entre dos agentes por el cual se puede emitir o recibir información.

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

Los pasos a seguir para establecer una comunicación usando los sockets son:

- El proceso servidor crea un socket con nombre y espera la conexión.
- El proceso cliente crea un socket sin nombre.
- El proceso cliente realiza una petición de conexión al socket servidor.
- El cliente realiza la conexión a través de su socket con el servidor, de igual manera el servidor puede aceptar más peticiones por otros clientes.

Es muy común que un socket servidor lance un proceso hijo para que una

vez realizada la conexión se ocupe del intercambio de información con el proceso cliente mientras el proceso padre servidor sigue aceptando conexiones. Se puede eliminar esta característica haciendo que el servidor solo acepte una conexión.

Todo socket viene definido por dos características fundamentales:

- El tipo del socket, que indica la naturaleza del mismo, el tipo de comunicación que puede generarse entre los sockets.
- El dominio del socket especifica el conjunto de sockets que pueden establecer una comunicación con el mismo.

2.12 TIPOS DE SOCKETS

En la comunicación entre socket se pueden presentar las siguientes características de acuerdo con el tipo de socket usado:

- Fiabilidad de transmisión.
- Mantenimiento del orden de los datos.
- No duplicación de los datos.
- El modo orientado a conexión en la comunicación.
- Envío de mensajes urgentes.

Los tipos disponibles son los siguientes:

* Tipo **SOCK_DGRAM**: sockets para comunicaciones no orientadas a conexión, usan datagramas de tamaño limitado. Este tipo de socket usa el protocolo UDP.

* Tipo **SOCK_STREAM**: para comunicaciones fiables orientadas a conexión, de dos vías y con tamaño variable de los mensajes de datos. Este socket usa el protocolo TCP.

* Tipo **SOCK_RAW**: permite el acceso a protocolos de más bajo nivel como el IP que se encuentra en el nivel de red.

* Tipo **SOCK_SEQPACKET**: tiene las características del SOCK_STREAM pero además el tamaño de los mensajes es fijo.

2.13 Seguridad en la redes

2.13.1 Políticas de seguridad

Las redes no pueden calificarse como seguras o inseguras puesto que el término no es absoluto. Cada grupo define el nivel de acceso permitido o negado. Por ejemplo algunas organizaciones almacenan datos valiosos, tales organizaciones definen como red segura un sistema que evita que gente ajena acceda a sus computadores.

Otras organizaciones colocan información a disposición de usuarios externos pero prohíben que cambien los datos. Tales organizaciones podrían definir un sistema seguro como el que permite acceso arbitrario a los datos pero que incluyen mecanismos para evitar cambios no autorizados; en cambio otros grupos se enfocan en tener privada la comunicación y en consecuencia definen un red segura como aquella en la que nadie excepto el destinatario, puede interceptar y leer los mensajes.

Por último muchas organizaciones grandes necesitan una definición complicada de seguridad que permita acceso a datos y servicios seleccionados como públicos y evite el acceso y modificación de datos y servicios privados.

Los costos y beneficios de varias políticas de seguridad también añaden complejidad. En particular, no puede definirse una política de seguridad a menos que la organización entienda el valor de su información, que en muchos casos es difícil de estimar.

Otro aspecto es la responsabilidad en la que puede incurrir una organización si su información es incorrecta; por ejemplo si una persona no autorizada aumenta la tarifa de pagos de una base de datos de nómina, tal compañía, podría incurrir en costos arbitrarios porque se pagaría de más a los empleados.

Por tanto idear una política de seguridad de red resulta complicado ya que requiere que la organización estime el valor de su información y la política de seguridad debe aplicarse a la información almacenada en los computadores, así como a la información que atraviesa una red.

2.13.2 Aspectos de la seguridad

La organización debe decidir los aspectos de protección más importantes y buscar una media entre seguridad y facilidad de uso. Entre los aspectos a tener en cuenta tenemos los siguientes:

- *Integridad de datos:* la integridad se refiere a la protección contra los cambios ¿son iguales los datos que llegan al receptor que los transmitidos?.
- *Disponibilidad de datos:* la disponibilidad significa la protección contra interrupciones del servicio ¿permanecen accesibles los datos para usos adecuados?.
- *Confidencialidad e intimidad de datos:* la confidencialidad y la intimidad se refieren a protección contra espías e intervenciones: ¿están protegidos los datos contra el acceso no autorizado?.
- *Autenticidad:* se refiere a la verdad y legitimidad de los datos.

2.13.3 Responsabilidad y control

La mayoría de las organizaciones descubre que no puede diseñar una política de seguridad adecuada porque no se ha especificado la manera de asignar y controlar la responsabilidad de la información.
Algunos aspectos de la responsabilidad y control son:

- *Contabilidad:* la contabilidad se refiere de mantener una pista de auditoria: que asigne responsabilidad a algunas personas de los datos, así como mantener para el grupo registros de acceso y cambios.
- *Autorización:* la autorización significa la responsabilidad de cada elemento de información y la delegación de tal responsabilidad a otros, haciendo referencia a quién es el responsable de donde reside la información y cómo aprueba el acceso y los cambios.

El punto crítico tanto de la contabilidad como de la autorización es el control.

2.13.4 Mecanismos de seguridad

Un mecanismo de seguridad muy usado es el de cifra de comprobación y comprobación de redundancia cíclica (CRC). Para emplear tales técnicas el transmisor calcula una cifra entera pequeña en función de los datos de un paquete. El receptor recalcula la función de los datos que llegan y compara el resultado con la suma calculada por el transmisor.

Este mecanismo no puede garantizar completamente la integridad de los datos por dos razones:

- Si un hardware con fallas cambia la suma de comprobación al igual que a la de los datos es posible que la suma de comprobación alterada sea válida para los datos alterados, aunque la probabilidad de que cambios aleatorios generen una suma de comprobación correcta es extremadamente baja pero no nula.
- Si el cambio de los datos es resultado de un ataque planeado, el atacante puede crear una suma de comprobación válida para los datos alterados.

2.13.5 Filtrado de paquetes

El filtrado de paquetes se realiza mediante un programa o software que opera en un enrutador. El filtro consiste en evitar que los paquetes no autorizados pasen por un enrutador que une dos subredes. El administrador debe configurar el filtro para especificar los paquetes que pueden pasar por el enrutador y los que se bloquean.

Además de emplear la dirección fuente y destino un filtro de paquetes puede examinar el protocolo del paquete o el servicio de alto nivel al que corresponde al paquete. El administrador puede indicar combinaciones boolenas de fuente y destino y de tipo de servicio al software que realiza el filtrado de paquetes.

2.13.6 Concepto de firewall

Un filtro de paquetes configurado para proteger una organización contra el tráfico de internet se llama Internet firewall (o barrera de seguridad). Tal barrera se diseña para evitar que los problemas de Internet se extiendan a los computadores de una organización, como por ejemplo ataques por virus o accesos no autorizados.

Los firewalls son la herramienta de seguridad más importante para manejar conexiones de red entre dos organizaciones que no se tienen confianza. En particular, al limitar el acceso a un grupo pequeño de computadores, que evite que gente ajena acceda a los computadores de una organización o que inunde sus redes con tráfico indeseado.

Con un firewall el administrador puede restringir los paquetes de entrada a un grupo pequeño de computadores, aunque los computadores del grupo deben ser seguros, otros computadores de la organización no necesiten serlo.

3. TRABAJO EN RED EN JAVA

La posibilidad de escribir aplicaciones distribuidas de forma rápida y sencilla es uno de los principales atractivos de Java. Java es uno de las pocas herramientas que permiten a cualquier programador (sin necesidad de conocimientos avanzados de comunicaciones) escribir programas que se integren fácilmente en una red IP para acceder a bases de datos remotas, interactuar con otros programas (escritos o no en Java) y distribuir datos o aplicaciones a través de Internet.

3.1 Comunicación mediante el protocolo UDP usando java

3.1.1 La clase DatagramSocket

Un objeto java.net.DatagramSocket es un "conector" a través del cual enviamos y recibimos paquetes UDP. En la literatura técnica estos paquetes se denominan Datagramas.

La forma usual de crear un DatagramSocket para recibir paquetes es especificando un número de puerto en el constructor. De esta forma, este DatagramSocket estará "escuchando" en el puerto especificado, preparado para recibir cualquier paquete entrante.

Si queremos construir un DatagramSocket para enviar paquetes, no es necesario especificar el número de puerto, ya que nos es indiferente. No ocurre lo mismo en el caso anterior, ya que siempre querremos usar un puerto fijo conocido por el resto de aplicaciones.

```
DatagramSocket ds1 = new DatagramSocket(123);
/* Aquí usamos este DatagramSocket para recibir datos... */
```

```
/* ...              */
/* Hemos terminado, cerramos el socket      */
ds1.close();
DatagramSocket ds2 = new Datagramsocket();
/* Aquí lo usamos para transmitir datos...  */
/* ...              */
/* Hemos terminado, cerramos el socket      */
ds2.close();
```

3.1.2 La clase DatagramPacket

Esta clase representa a los paquetes de datos que vamos a recibir o transmitir a través de los objetos DatagramSocket. Estos paquetes constan de una cabecera (que incluye la dirección de origen y destino del paquete, el puerto, la longitud del paquete, un checksum, etc.) y un cuerpo (donde se encuentra el contenido real del paquete).

En Java accedemos a las distintas partes de un datagrama mediante los métodos de la clase java.net.DatagramPacket.

Veamos un ejemplo, donde enviamos un datagrama a una determinada dirección, suponiendo que tenemos un objeto InetAddress correctamente creado. Nótese el uso del método send() de la clase DatagramSocket para enviar el datagrama:

```
int tam = 1024;
InetAddress direcc = ...;
 byte[] datos = new byte[tam];
 int puerto = 543;
 for (int n=0;n<tam;n++){
        /* Generamos los datos que vamos a enviar */
        datos[n] = ...;
        }
```

```
DatagramSocket ds = new DatagramSocket();
 DatagramPacket dp = new DatagramPacket(datos, tam, direcc,
puerto);
 ds.send(dp);          /* Aquí enviamos el paquete     */
```

Ejemplo de cómo recibir datos

```
int tam = 1024;
 byte[] buffer = new byte[tam];

 int puerto = 987;
 DatagramSocket ds = new Datagramsocket(puerto);
 DatagramPacket dp = new DatagramPacket(buffer,tam);

 ds.receive(dp);
 // Ahora tenemos en buffer la información que nos interesa
```

3.1.3 La clase InetAddress

La forma de crear un objeto InetAddress es mediante el método estático InetAddress.getByName(String), que recibe un nombre de host en notación alfanumérica (por ejemplo " www.uis.edu.co " o "192.168.19.2" y devuelve un objeto InetAddress con esa dirección. Si la dirección no existe o no puede ser encontrada, este método lanza una UnknownHostException.

Por ejemplo, si queremos mandar una matriz de bytes al puerto 90 de la dirección "www.uis.edu.co", tenemos que escribir:

```
int tam = ...;
 int puerto = 90;
 String maquina = " www.uis.edu.co ";
 byte[] buffer = new byte[tam];
 // ...
 // Generamos el contenido del buffer
 // ...
 InetAddress direcc = InetAddress.getByName(maquina);
 DatagramSocket ds = new DatagramSocket();
 DatagramPacket dp = new DatagramPacket(buffer, tam, direcc,
puerto);
```

```
ds.send(dp);          /* Aquí enviamos el paquete     */
```

Si queremos enviar paquetes a nuestra propia máquina hay que usar como nombre de host la dirección "localhost" o "127.0.0.1". También podemos usar el método InetAddress.getLocalHost(), que devuelve un objeto InetAddress que "apunta" a la máquina local.

3.2 Comunicación mediante el protocolo TCP en Java

3.2.1 La clase Socket

Un objeto java.net.Socket es un "conector" a través del cual enviamos y recibimos datos mediante el protocolo TCP. A diferencia de los "conectores" java.net.DatagramSocket, que eran usados para enviar paquetes sueltos, estos "conectores" TCP sirven para enviar o recibir datos de forma continua, como si trabajáramos con un flujo InputStream o OutputStream.

El hecho de que el protocolo subyacente sea TCP nos permite olvidarnos de detalles relacionados con la pérdida de datos, ya que es el propio protocolo el encargado de hacerlo por nosotros. A todos los efectos, podemos tratar un Socket TCP como un canal carente de errores.

3.2.2 Utilización de un objeto Socket

La inicialización de estos objetos es más compleja que en el caso de DatagramSocket, ya que es necesario que previamente haya alguien "escuchando" en el extremo receptor.

Suponiendo que de alguna forma, hay un programa "escuchando" en el puerto 1234 de la máquina con dirección IP 192.168.19.2 la inicialización de nuestro Socket sería:

```
InetAddress d = InetAddress.getByName("192.168.19.2");
Socket s = new Socket(d,1234);
/* Utilizacion del socket */
...
/* Cerramos el socket */
s.close();
```

Una vez tenemos un Socket abierto con otra máquina, podemos obtener un flujo de entrada o de salida para poder recibir o transmitir datos. Esto se hace con los métodos Socket.getInputStream() y Socket.getOutputStream():

Veamos un ejemplo donde abrimos un socket, leemos los bytes que nos transmitan desde el otro extremo y los imprimimos en pantalla:

```
InetAddress d = InetAddress.getByName("192.168.19.2");
Socket s = new Socket(d, 1234);
Inputstream is = s.getInputStream();
while((int dato=is.read())!=-1){
System.out.println("Recibido " + dato);
}
is.close();
s.close();
```

3.3 Gestión de excepciones en Java

Como en todos los casos en que tengamos que trabajar con protocolos de red o sistemas de entrada/salida, tenemos que encargarnos de gestionar las posibles excepciones. Las más comunes son IOException (para los casos en los que haya problemas con la conexión) y UnknownHostException (cuando especificamos una dirección IP desconocida o incorrecta).

El manejo de excepciones permite implementar funciones de gestión de errores en los programas de Java.

Existen dos tipos de excepciones:

1) ***De tiempo de ejecución (runtime)*****:** son objetos de la clase java.lang.Throwable o de sus subclases Exception y Error. Se producen cuando se viola una regla fundamental de Java.

2) De programa: son creadas y lanzadas directamente por el programa cuando se detecta una condición que podría producir un error.

En un método se puede decidir:

a) si se **captura** y se **maneja** la excepción dentro del método, o

b) si se **declara** y se **lanza** al método donde está la sentencia que lo llamó, para que se capture allí o se declare.

Un método procesa una excepción cuando la captura (catch) y la maneja (handle).

Las excepciones deben ser procesadas (capturadas y manejadas) en algún punto (método) del programa, si no se procesa una excepción lanzada finalizaría la ejecución y se visualizaría un mensaje de error.

3.4 Manejo de hilos en Java

Una aplicación o applet que inicia su ejecución es un hilo sencillo o proceso.

Un programa multihilo contiene 2 o mas partes que pueden ejecutarse de manera concurrente. Cada parte del programa se llama hilo (thread) y cada hilo es un camino de ejecución diferente.

3.4.1 Ejemplo de un hilo en Java

```
public class HiloPrueba {
        public static void main(String args[]) {
                Xyz r = new Xyz();
```

```
                Thread t = new Thread(r);
                t.start();
                }
        }

class Xyz implements Runnable {
        int i;
        public void run() {
                i=0;
                while(true) {
                        System.out.println("Hello" + i++);
                        if(i==50) break;
                }
        }
}
```

El sistema de Multihilos de Java está compuesto por la clase Thread y la interface Runnable. La clase Thread encapsula un hilo de ejecución.

Para crear un nuevo Hilo, debe hacer una de las siguientes opciones: Extender la clase Thread o implementar la interface Runnable.

La clase Thread define diferentes métodos que ayudan a manejar los hilos, los más usados son:

start: Comenzar el hilo llamando su método run.

run: Punto de entrada para el Hilo.

getName: Obtener el nombre del hilo

getPriority: Obtener la prioridad del hilo

isAlive: Determinar si el hilo aun esta corriendo.

join: Esperar a que un hilo termine.

sleep: Suspender un hilo por un periodo de tiempo.

3.4.2 El hilo principal

El hilo principal es el que se inicia con la ejecución de la aplicación o el applet. Es el hilo desde el cual se crean el resto de hilos en el programa; debe ser el último hilo que termine su ejecución.

Se puede obtener una referencia a el usando el método

```
Thread hip = Thread.currentThread();
```

3.5 Manejo de base de datos en java

JDBC es la API estándar de acceso a Bases de Datos con Java, y se incluye con el Kit de Desarrollo de Java (JDK) a partir de la versión 1.1

Para trabajar con JDBC es necesario tener controladores (drivers) que permitan acceder a las distintas Bases de Datos: cada vez hay más controladores nativos JDBC. Sin embargo, ODBC es hoy en día la API más popular para acceso a Bases de Datos. Para el desarrollo de la aplicación se utilizará este puente JDBC/ODBC para acceder a una Base de Datos.

3.5.1 Preparación del entorno de desarrollo

Antes de comenzar a revisar las distintas clases proporcionadas por JDBC, vamos a abordar la creación de una fuente de datos ODBC, a través de la cuál accederemos a una Base de Datos Interbase. Hemos escogido este modo de acceso en lugar de utilizar un driver JDBC nativo para Interbase porque ello nos permitirá mostrar el uso de drivers ODBC: de este modo, aquél que no disponga de Interbase, podrá crear una Base de Datos Access,

o de cualquier otro tipo para el que sí tenga un driver ODBC instalado en su sistema.

El hecho de que para acceder una Base de Datos mediante ODBC se utilice el nombre de la fuente de datos, en lugar del nombre de la Base de Datos, nos permite cambiar la ubicación e incluso el nombre de la misma sin tener que modificar el código fuente de nuestros programas, ya que estos utilizarán el nombre de la fuente de datos para identificarla.

Como último paso, no debemos olvidar arrancar el servidor de Base de Datos.

Como es lógico, la API JDBC incluye varias clases que se deben utilizar para conseguir acceso a una Base de Datos. La Tabla 3.1. muestra la lista de clases e interfaces más importantes que JDBC ofrece, junto con una breve descripción. Estas clases se encuentran en el paquete *java.sql*.

Clase/Interface	**Descripción**
Driver	Permite conectarse a una Base de Datos: cada gestor de Base de Datos requiere un Driver distinto.
DriverManager	Permite gestionar todos los Drivers instalados en el sistema.
DriverPropertyInfo	Proporciona diversa información acerca de un Driver.
Connection	Representa una conexión con una Base de Datos. Una aplicación puede tener más de una conexión a más de una

	Base de Datos.
DatabaseMetadata	Proporciona información acerca de una Base de Datos, como las tablas que contiene, etc.
Statement	Permite ejecutar sentencias SQL sin parámetros.
PreparedStatement	Permite ejecutar sentencias SQL con parámetros de entrada.
CallableStatement	Permite ejecutar sentencias SQL con parámetros de entrada y salida, típicamente procedimientos almacenados.
ResultSet	Contiene las filas o registros obtenidos al ejecutar un SELECT.
ResultSetMetadata	Permite obtener información sobre un ResultSet, como el número de columnas, sus nombres, etc.

Tabla 3.1 Clases e interfaces definidos por JDBC.

Es conveniente cerrar las conexiones a Bases de Datos tan pronto como dejen de utilizarse, para liberar recursos rápidamente. Sin embargo, ha de tenerse en cuenta que establecer una conexión es una operación lenta, por lo que tampoco se debe estar abriendo y cerrando conexiones con frecuencia.

3.5.2 Consultas en JDBC

Un programa que realice una consulta y quiera mostrar el resultado de la misma requerirá del uso de varias clases: la primera es *DriverManager*, que permitirá llevar a cabo una conexión con una Base de Datos, conexión que se representa mediante un objeto que soporta el interface *Connection*. También será necesario además ejecutar una sentencia SELECT para llevar a cabo la consulta, que se representará por un objeto que soporte el interface *Statement* (o *PreparedStatement*, o *CallableStatement*. Una instrucción SELECT puede devolver diversos registros o filas: esta información es accesible mediante un objeto que soporte el interface *ResultSet*.

Es posible limitar el número máximo de registros devuelto al hacer un *executeQuery* mediante *setMaxRows*, y averiguar dicho número mediante *getMaxRows*. Es posible también obtener el último aviso generado al ejecutar una sentencia, mediante *getWarning*, así como limitar el tiempo en segundos que el controlador esperará hasta que el SGBD devuelva un resultado, mediante *setQueryTimeout*. Por último, el método *close* libera los recursos asociados a la sentencia.

El interface *ResultSet* es el que encapsula el resultado de una sentencia SELECT. Para recuperar la información es necesario acceder a las distintas columnas (o campos), y recuperarla mediante una serie de métodos *getString*, *getFloat*, *getInt*, etc. Al utilizar estos métodos debe indicarse el número correspondiente a la columna que estamos accediendo: si lo desconocemos, podemos averiguar el número correspondiente a una columna, dado el nombre, mediante *findColumn*. Por último, dado que un *ResultSet* puede contener más de un registro, para ir avanzando por la lista de registros que contiene debemos utilizar el método *next*, que devuelve un valor booleano indicando si existe otro registro delante del actual. Además de

estos métodos, *ResultSet* también cuenta con *getWarnings*, que devuelve el primer aviso obtenido al manipular los registros, así como *wasNull*, que indica si el contenido de la última columna accedida es un NULL SQL. Por último, el método *close* libera los recursos asociados al *ResultSet*.

Excepción	Descripción
SQLException	Error SQL.
SQLWarning	Advertencia SQL.
DataTruncation	Producida cuando se truncan datos inesperadamente, por ejemplo al intentar almacenar un texto demasiado largo en un campo.

Tabla 3.2 Excepciones que puede generar la API de JDBC

Al utilizar la API JDBC es posible obtener diversos errores debido a que se ha escrito incorrectamente una instrucción SQL, a que no se puede establecer una conexión con la Base de Datos por cualquier problema, etc.

4. CICLO DE VIDA DE LA APLICACION

El ciclo de vida del desarrollo de un sistema es un enfoque por fases de planeación y diseño que sostiene que los sistemas son desarrollados de mejor manera mediante el uso de un ciclo específico de actividades.

Estas etapas son presentadas en forma discreta, en la realidad no se llevan a cabo cada una como un paso separado, pues pueden suceder simultáneamente y algunos actividades pueden repetirse.

4.1 Identificación de las necesidades del sistema

La primera etapa a desarrollar es la identificación del problema en cuanto al control de asignación de direcciones IP y los servicios no autorizados, lo cual proporciona un enfoque sistemático para el diseño y la construcción del sistema planteado. En esta etapa es importante proceder minuciosamente para evitar confusiones u omitir requerimientos o funciones, ya que incurriríamos en gastos y pérdida de tiempo para el sistema; además se requiere más esfuerzo para corregir en etapas posteriores dichos problemas.

4.2 Análisis y diseño del sistema

Otra etapa es el análisis y diseño del software que permite realizar el escáner de red proporcionando información sobre los servicios y direcciones IP que no se encuentran autorizadas; se busca analizar cuidadosamente la entrada y el flujo de los datos, el proceso de los datos, el almacenamiento y la salida de la información que da el sistema.

Cuando se instala un sistema sin la planeación adecuada, lleva a grandes fracasos y frecuentemente deja de ser útil muy rápidamente, de ahí la necesidad de un buen análisis y diseño del sistema que implica un gran esfuerzo, pero con buenos resultados. Esta etapa tiene una serie de procesos que se llevan a cabo para mejorar el desempeño del software planteado; gran parte de este análisis y diseño involucra tanto a los usuarios como a los programadores.

4.3 Implementación del sistema

Otra etapa es la implementación de la herramienta que debe brindar control y soporte para verificar las direcciones IP en uso y los servicios instalados. Lo que se requiere en este proceso es asegurarse de que el sistema sea operacional y permitir que el usuario tome el control de la operación para su uso y evaluación. Es importante evaluar el desempeño del software con los usuarios que manipulan el sistema para modificar o corregir los problemas presentados. Además ofrecer una capacitación adecuada a los usuarios para que entiendan lo que el sistema hace y se pueda tomar decisiones e interpretar los resultados que entrega la herramienta.

4.4 Validación y control del sistema

La siguiente etapa del software es la validación de la herramienta, teniendo en cuenta la evolución del sistema para retroalimentar y mejorar así su desempeño y de esta manera dar continuidad a la implementación. Es importante observar la utilidad que el sistema ofrece a la universidad no sólo para este proyecto sino para sus posibles progresos futuros. Esta utilidad se

ve reflejada en las decisiones que se toman para la gestión y el control de asignaciones de los servicios de la red, porque la información se presenta en forma organizada; además existen reportes que son guardados para facilitar consultas futuras; se ahorra tiempo porque la información del escáner de red llega inmediatamente; el sistema puede ser actualizado más adelante de acuerdo a las necesidades que se presenten, esto permite que se conserve su valor y dé resultados satisfactorios, duraderos y permita tener mayor control de los recursos asignados a los servicios de la red de la universidad.

4.5 Diseño de interfaz del sistema

La última etapa es desarrollar o diseñar la interfaz que facilite a los usuarios acceder al sistema de tal forma que puedan manipular la información que necesiten, reduciendo los errores en la entrada de datos, además de incluir información para recordar a los usuarios la decisión que deben tomar ante determinada consulta.
En esta fase se desarrolla una interfaz de menús que proporciona al usuario una lista en pantalla de las selecciones disponibles, aquí el usuario no necesita conocer el sistema, pero sí que tarea realiza cada ítem del menú.

El diseño de las opciones de los menús, debe permitir a los usuarios extraer información significativa para el buen desempeño y control de las tareas.

Es importante tener en cuenta que una interfaz efectiva ayuda mucho a la satisfacción del usuario y que éste utilice el sistema para lo que fue diseñado.

5. METODOLOGIA DE DESAROLLO

Para la consecución de cualquier proyecto se hace necesario seguir una metodología que ayude a la realización de los objetivos y en consecuencia el proyecto; existen varios métodos y enfoques para la realización de un proyecto, uno de ellos es el ciclo de vida clásico, que es el que vamos a desarrollar.

Esta metodología está basada en fases, que a continuación se describirá cada una de ellas, enfocadas específicamente a la realización del sistema propuesto.

5.1 Identificación de las necesidades del sistema

Es la primera fase del ciclo de vida del desarrollo de sistemas, esta etapa es crítica para el éxito del resto del proyecto, debido a que nadie quiere desperdiciar el tiempo subsecuente resolviendo el problema equivocado.

En esta fase se estudia cual es la necesidad o el problema que se quiere resolver, se debe observar lo que honestamente está sucediendo, en nuestro caso específico poder controlar los servicios de las direcciones IP que ofrece la universidad y reportar las salidas de servicio de los servidores oportunamente.

Después de identificado el problema, consideramos que se puede resolver por medio del uso de un sistema de escaneo de red. El realizar este sistema de escáner permite que la universidad gane un avance en control y eficiencia de los servicios que presta.

El resultado de esta fase es saber si es factible solucionar el problema planteado y tomar una decisión para continuar el proyecto propuesto.

5.2 Análisis y diseño del sistema

En esta etapa se realiza el análisis de la información recopilada en la fase anterior, por medio de algunas estrategias como son, los diagramas de flujo de datos observando las entradas, el proceso y salidas de la información. A partir de los diagramas de flujo de datos se desarrolla un diccionario de datos que lista los conceptos de datos usados en el sistema.

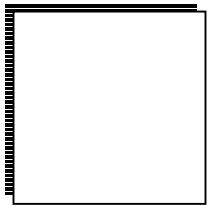
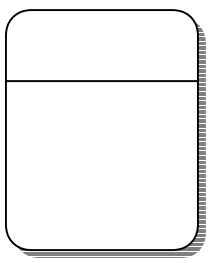
Además, se utiliza la información recolectada para realizar el diseño lógico del sistema propuesto. Se hacen procedimientos precisos para la captura de datos, con el fin de que los datos que se van a suministrar al sistema sean los correctos. Además se diseñan cómo se van a guardar los datos que más tarde se utilizarán para la toma de decisiones.

Con el fin de realizar un análisis adecuado para este proyecto se estudió la falta de control que tiene la universidad para los servicios de direcciones IP en uso y el reporte oportuno para las anomalías de los servidores. Finalmente se detectaron los principales problemas como son el control de asignaciones de direcciones IP y servicios que presenta la universidad.

Una de las ventajas de utilizar diagrama de flujo de datos es que permite describir cada elemento que es usado en el diagrama sin comprometerse demasiado pronto con la realización técnica. Este análisis se realiza para asegurarnos que todas las salidas necesarias puedan ser obtenidas a partir de los datos de entrada y la lógica de procesamiento reflejada en el diagrama.

La detección y corrección de errores y fallas de diseño de esta naturaleza en las primeras etapas del ciclo de vida del desarrollo de sistemas es mucho menos costosa que en las fases posteriores de programación, pruebas e implementación.

Ahora describiremos los símbolos utilizado para realizar el diagrama de flujo de datos. Se usan cuatro símbolos básicos para diagramar el movimiento de datos en los diagramas.

SIMBOLO	DESCRIPCION
	**ENTIDAD.** El cuadrado doble es usado para representar una actividad externa (otro departamento, una persona, un negocio o una máquina) que pueden enviar datos o recibirlos del sistema. Esta entidad también es llamada una fuente destino de datos y es considerada externa al estudio. Cada entidad externa es etiqueta con un nombre adecuado.
	FLUJO DE DATOS. Una flecha representa el movimiento de datos de un punto a otro acerca de una persona, lugar o cosa, también debe ser descrita con un nombre.
	**PROCESO.** Este rectángulo es usado para mostrar la aparición de un proceso de transformación. Los procesos siempre denotan un cambio o transformación de los datos y, por lo tanto, el flujo de datos que sale de un proceso siempre es etiquetado en forma diferente al que entra a él. Los procesos representan trabajo que

	está siendo desarrollado dentro del sistema y deben ser nombrados.
	**ALMACEN DE DATOS.** Es un rectángulo. Este símbolo simplemente muestra un recipiente para los datos que permita adición y recuperación de datos. Este almacenamiento puede representar un almacenamiento manual o un archivo de base datos computarizado

Tabla 5.1 Símbolos usados en un diagrama de flujo de datos.

5.2.1 Proceso de escaner de red

Este proceso permite verificar que host o servidores están disponibles y que servicios de puertos utilizan, este proceso permite a la universidad tener un mejor control y por lo tanto brindar un buen servicio para beneficio a la comunidad.

Figura 5.1 Proceso de escáner de red

5.2.2 Proceso de escaneo de una dirección IP

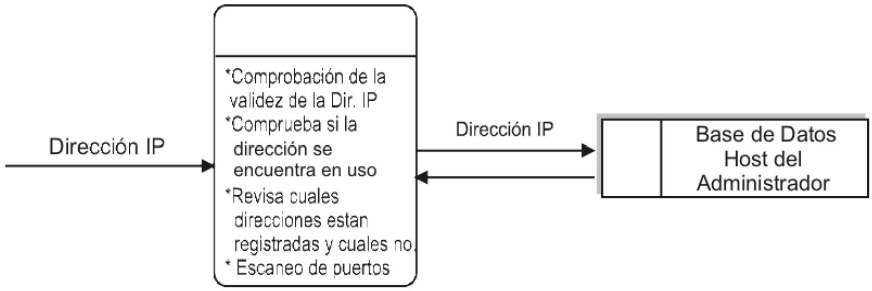


Figura 5.2 Proceso de escáner de una direcciones IP

5.2.3 Proceso de escaneo del rango de direcciones IP

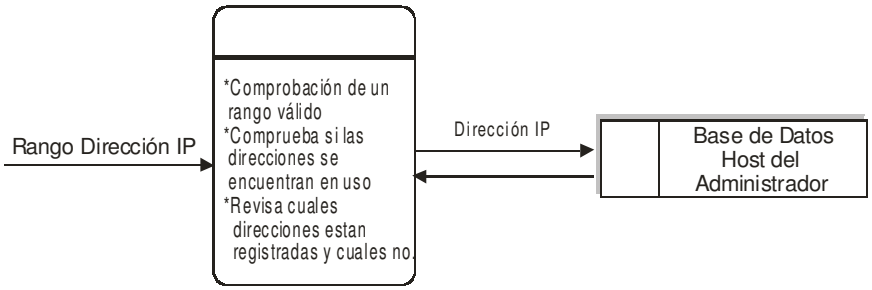


Figura 5.3 Proceso de escáner del rango de direcciones IP

5.2.4 Proceso de escaneo de una subred

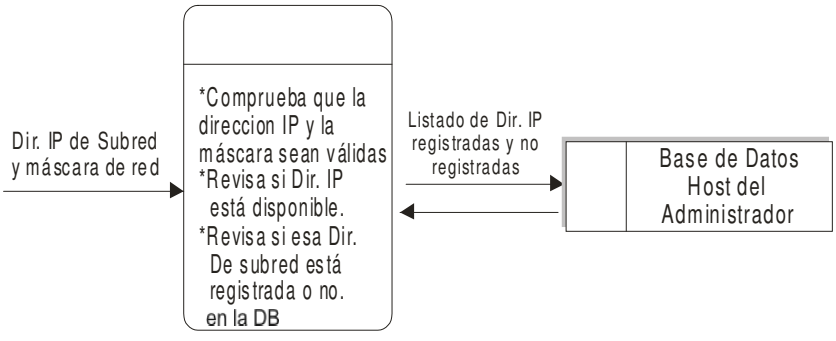


Figura 5.4 Proceso de escáner de una Subred

5.2.5 Escaneo automático de las direcciones IP de los servidores

Esta verificación empezará con un chequeo automático cada cierto intervalo de tiempo a las direcciones IP registradas en la base de Datos en la tabla de Servidores con el fin de saber si ha ocurrido alguna falla o cambio de estado los servidores que se están monitorizando.

5.3 Implementación del sistema

Se optó por utilizar el lenguaje de programación Java, porque tenía características especiales que permitía que la herramienta a desarrollar cumpliera con los requisitos de multiplataforma y facilidad de desarrollo de aplicaciones en entorno de red.

En esta fase se desarrolla y codifica el software, además de instalarlo en los equipos donde irá a funcionar.

5.3.1 Proceso de desarrollo de la Herramienta

Primera Etapa: En esta primera etapa se profundizó sobre el manejo de la estructura básica del lenguaje y métodos relacionados con el manejo de redes.

La implementación del Software se realizó por módulos para facilitar la programación y la corrección de errores.

La implementación del Software comenzó con la definición de un vector con los nombres de los puertos más conocidos (del 1 al 255). Seguidamente se desarrolló el método que escanea una dirección IP revisando si dicha

dirección se encontraba en servicio y qué puertos de los más conocidos se encontraban en uso.

Una vez realizado el método que escanea una dirección IP, se desarrolló el método que escanea un rango de direcciones IP, el cual revisa si cada dirección IP se encuentra en uso, a partir de este método se construyo el módulo que escanea una Subred.
Para la construcción de los tres métodos anteriores se hizo necesario implementar en primera instancia tres clases fundamentales:

Primero, verificar que los datos de entrada fueran válidos para lo cual se creó una clase que comprobara direcciones Ip y nombre de Host.
Segundo, se creó una clase que permitiera conocer si una dirección Ip se encuentra en uso, esta clase realiza una función parecida a la de un PING. Para lograr esta función se investigó a fondo conceptos de programación y se realizaron varias pruebas para lograr un resultado satisfactorio.
Tercero, se creó una clase que verificara si el servicio de un puerto en uso, era TCP o UDP.

En esta primera etapa se realizó una depuración de errores que causaban salidas erróneas de los tres módulos principales.

Segunda etapa: Como el proceso del método de verificar dirección IP o nombre de Host, era muy lento, surgió la necesidad de acelerar dicho proceso, para lo cual se implementó una clase adicional que pudiera revisar varios puertos de manera concurrente y la manera más apropiada fue recurrir al uso de los hilos, que es una de las características más importantes que ofrece Java para acelerar procesos que son independientes y que demandan tiempo.

Se desarrollaron dos clases, una que agilizara el proceso para el escaneo de puertos de una dirección IP y otra que agilizara el proceso de verificar las direcciones en uso de un Rango IP y de una Subred.

En este proceso se presentaron inconvenientes, uno de los cuales era que en la revisión de los últimos 10 procesos efectuados por los hilos, no se entregaba en su totalidad el resultado del escaneo de los puertos en el módulo de direcciones IP, puesto que el hilo principal tomaba el control antes que el proceso de los hilos terminara. Este inconveniente se presentó también en los métodos de Rango IP y Subred.

La solución fue recurrir a un método que garantizara la espera del último hilo y que este terminara completamente su proceso.

En el método de Dirección IP y Rango IP se verificaban de manera concurrente 10 procesos, después de muchas pruebas se estableció esta cifra, puesto que al analizar más se consumían demasiados recursos en el equipo el cual pudiera provocar un desbordamiento de memoria.

Tercera Etapa: Después de haber terminado dichos métodos con las funciones esenciales, era favorable facilitar el trabajo del administrador de la red, para lo cual se elaboró un esquema de una base de datos donde se encuentre almacenada la información correspondiente a las direcciones IP asignadas, el nombre de usuario y los servicios autorizados, como ftp, dns, telnet, smtp, http,… etc.

Con la ayuda del escáner de red se realiza la operación de escaneo de una dirección IP y los servicios de los puertos en uso, los resultados obtenidos se

contrastan con los almacenados en la base de datos y se registran las eventualidades encontradas, de esta manera el administrador de red tiene la información de una manera oportuna, tomando acciones pertinentes.

De la misma manera el módulo de Rango Ip y Subred también acceden a la base de datos para contrastar información de las direcciones autorizadas con las que se encuentran en uso realmente, quedando registrado en la base de datos aquellas direcciones IP que no se encuentran autorizadas por el administrador

Se implementó el servicio de escaneo de direcciones IP más importantes como son las de los servidores, con el fin de detectar si la dirección no se encuentra en servicio. De ser así, esta acción se reporta a la base de datos, y el administrador tomará las medidas correspondientes para dar solución oportuna.

Una operación que facilita aún más el trabajo del administrador fue la implementación en el escáner de red la opción que permite visualizar las direcciones IP registradas y los puertos autorizados de dicha dirección, las direcciones IP de los servidores, las direcciones IP en uso sin autorizar que fueron encontradas después de haber realizado un escaneo de una dirección IP o un Rango de direcciones y las fallas de los servidores.

5.3.2 Manejo de Variables y Tablas

5.3.2.1. Variables

Tipo de Variables	Estándar
Bolean	Las variables tipo booleano contienen valores true o false. Las variables tipo booleano sin inicializar se inician como false.

String	Es una secuencia de caracteres. Pueden ser tan largos como se desee en Java.
Integer	Es un entero con signo de 32 bits. (-2147483648 a 2147483647).
Date/time	Hora y fecha local
Sql andel	Setencias sql para invocar base de datos.

Tabla 5.2 Tipo de variables en java

5.3.2.2 Tablas

Tipo de Variables	Estándar
Dir_no_autorizadas	Se almacena las direcciones IP, el número del puerto, la clase de servicio TCP o UDP y la fecha y hora de las direcciones IP no registradas y que se encuentran en uso.
Direcciones	Se almacena la dirección IP, el nombre del host y el administrador a cargo.
Fallas_servidores	Se almacena la dirección IP, y la fecha y hora de salida de servicio.
Puertos	Se almacena el nombre del puerto y el tipo de servicio (TCP o UDP) asociado a cada dirección IP registrada de la tabla direcciones.
Servidores	Se encuentra almacenada la dirección Ip, el nombre del servidor y el nombre del administrador.

Tabla 5.3 Tipo de variables en la base de datos

5.3.3 Descripción del programa

5.3.3.1 Diseño de pantallas y menús

A continuación se muestra la pantalla principal del sistema escaneador de Red donde se encuentran todas las opciones del menú.

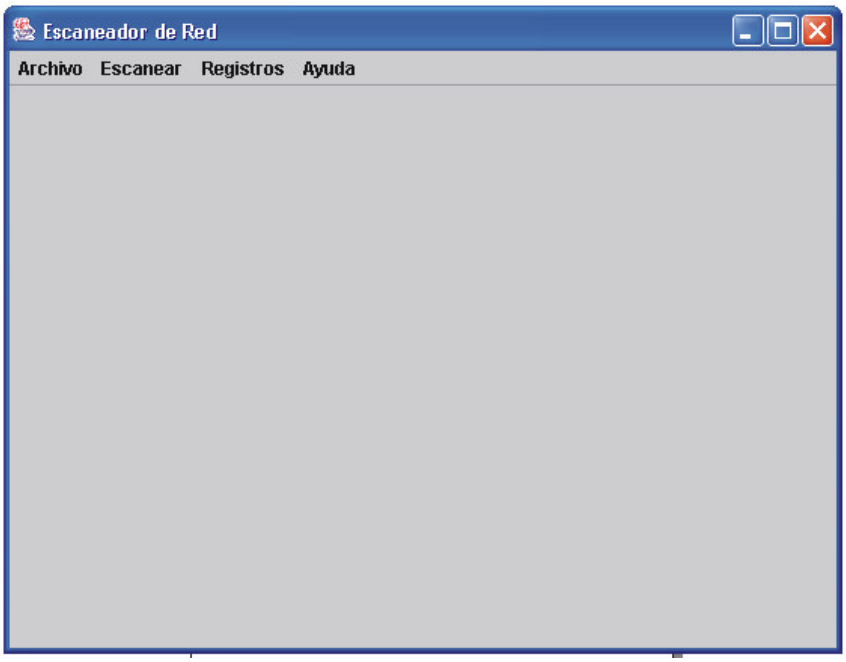


Figura 5.5 Pantalla principal del escáner de red

*** ARCHIVO**

En este menú sólo se muestra la opción para salir de la aplicación, con respuesta de SI ó NO.

Figura 5.6 Opción del menú archivo

- **ESCANEAR.** En este menú se encuentran los procesos básicos necesarios para realizar la tarea de escaneo de red y consta de las siguientes opciones:

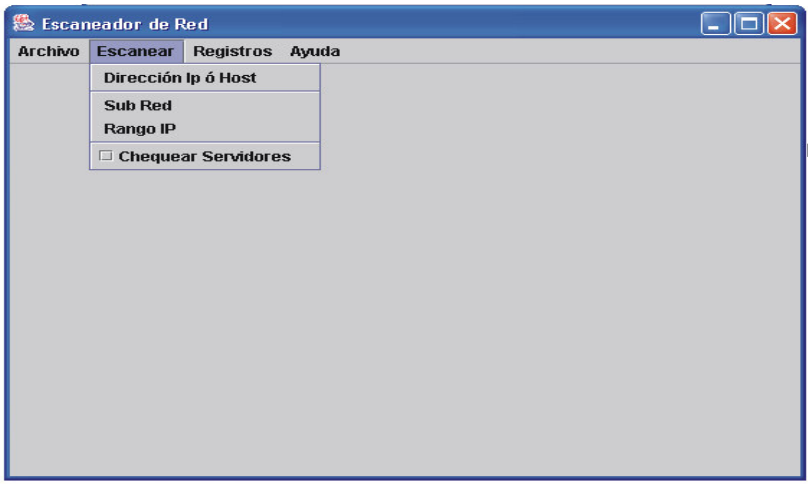


Figura 5.7 Opción del menú escanear

- *Dirección Ip o Nombre del Host.* Esta opción permite mediante la dirección IP, revisar si está disponible en ese momento y si está registrada, además analiza los puertos TCP y UDP más conocidos, registrando en una base de datos las eventualidades encontradas.

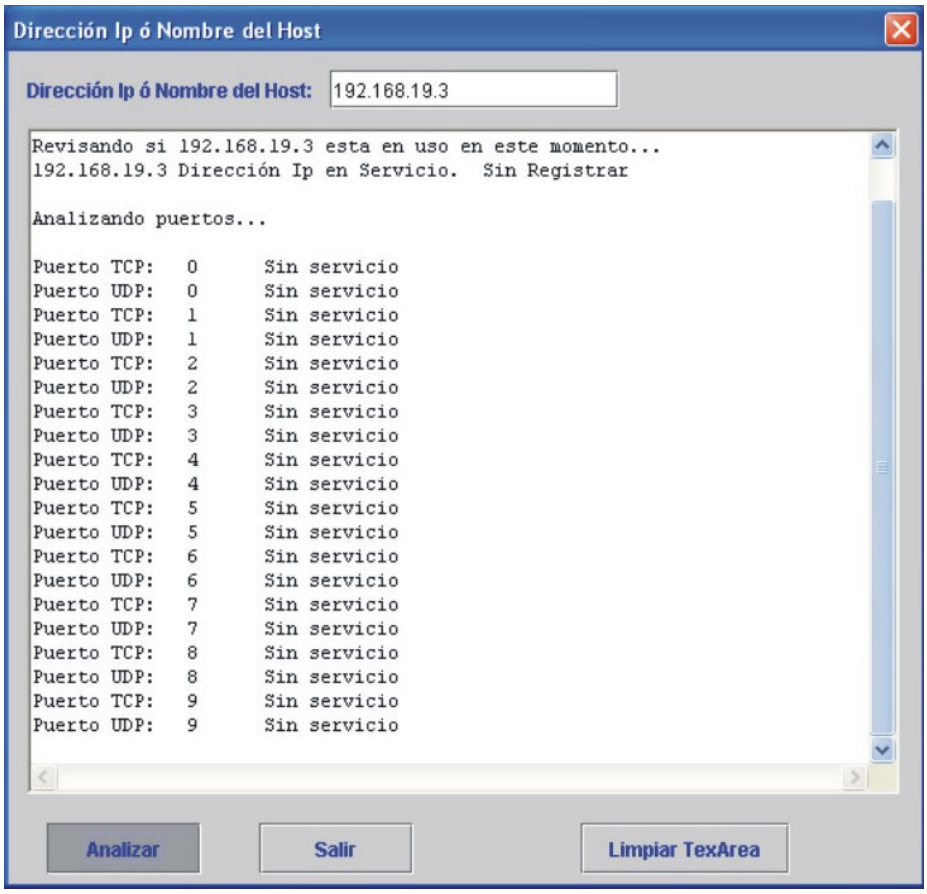


Figura 5.8 Resultados arrojados al realizar el escáner de una dirección IP

- *Sub red.* Con esta opción se analiza una sub-red, usando como datos de entrada una dirección IP y la máscara de subred correspondiente. Se revisará si que direcciones IP están siendo utilizadas y en caso de estarlo comprobará si están autorizadas y registrará en la base de datos dicho evento.

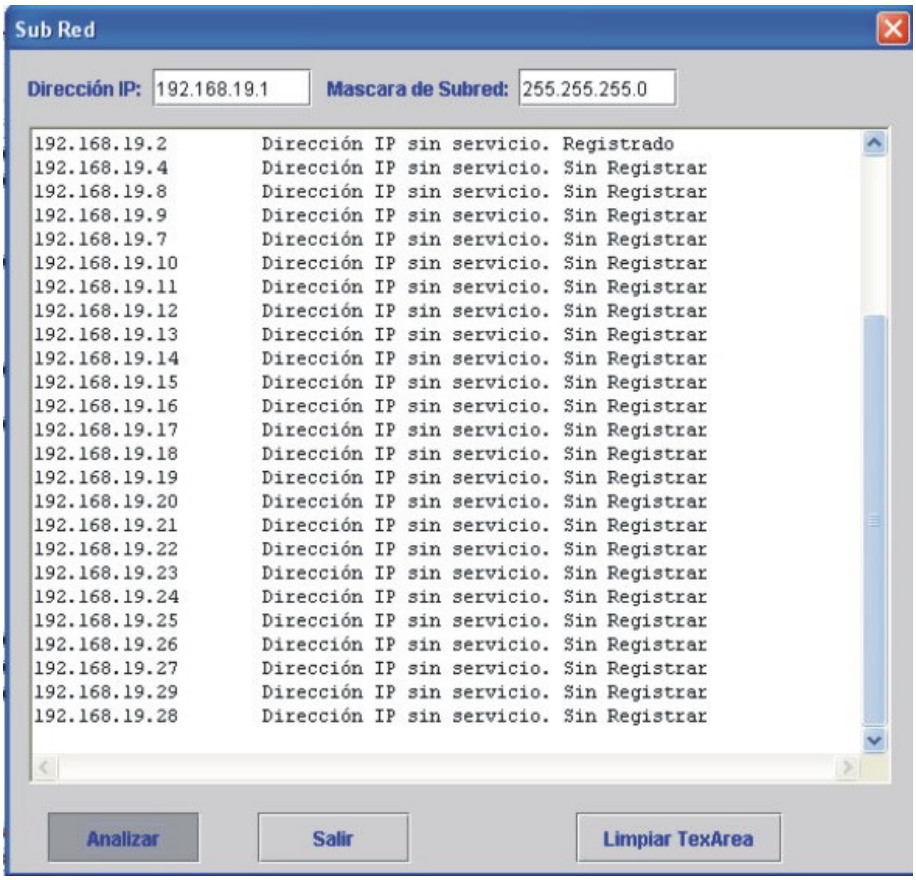


Figura 5.9 Resultados arrojados al realizar el escáner de una Subred

- *Rango IP.* Con esta opción se revisará el rango de direcciones IP tomando siempre como dirección IP inicial la de menor valor, se revisarán que direcciones IP están en uso y se verificará si estas tienen o no autorización comparando con la base de datos y registrando cualquier eventualidad.

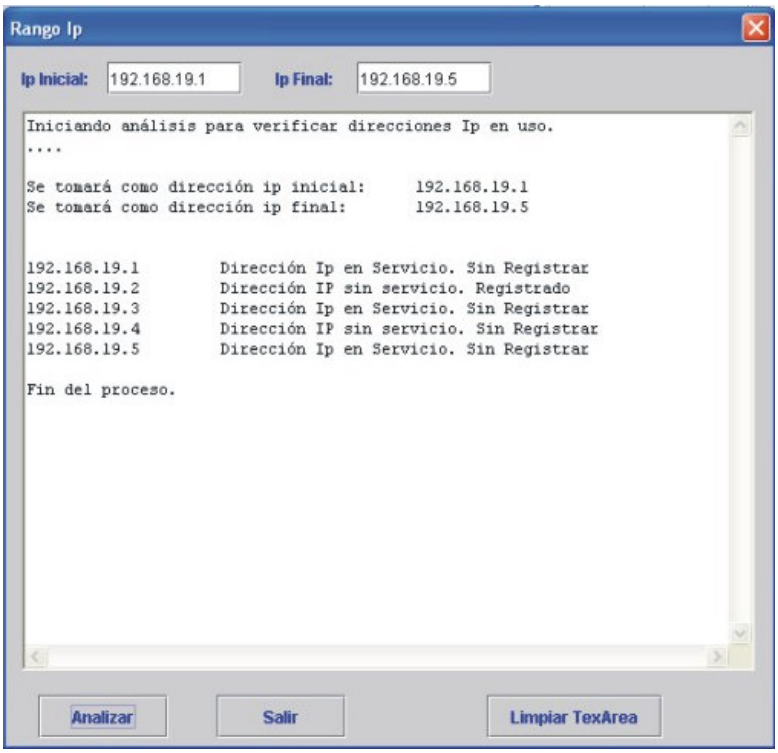


Figura 5.10 Resultados arrojados al realizar el escáner de un rango IP

- *Verificar servidores.* Al activar esta casilla de verificación se empezará una verificación automática cada cierto intervalo de tiempo a las direcciones IP registradas en la base de Datos en la tabla de Servidores con el fin de saber si ha ocurrido alguna falla o cambio de estado de los servidores que se están monitorizando.

*** REGISTROS**

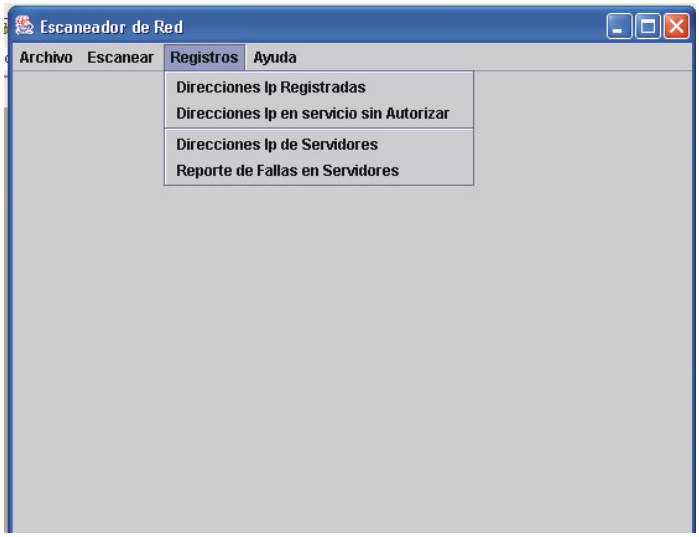


Figura 5.11 Opciones del menú Registros

- *Direcciones IP registradas:* Con esta opción tendremos acceso a la base de datos para consultar que direcciones IP se encuentran registradas por el administrador de la red.

- *Direcciones Ip en servicio sin autorizar:* Con esta opción tendremos acceso a la base de datos para consultar que direcciones IP se encuentran en uso y no están autorizadas por el administrador de la red.

- Direcciones IP de servidores: Con la cual podemos consultar que direcciones IP se encuentran registradas con permiso para prestar el servicio de Servidor.

- Reporte de salida de servicio en los servidores.

*** AYUDA**

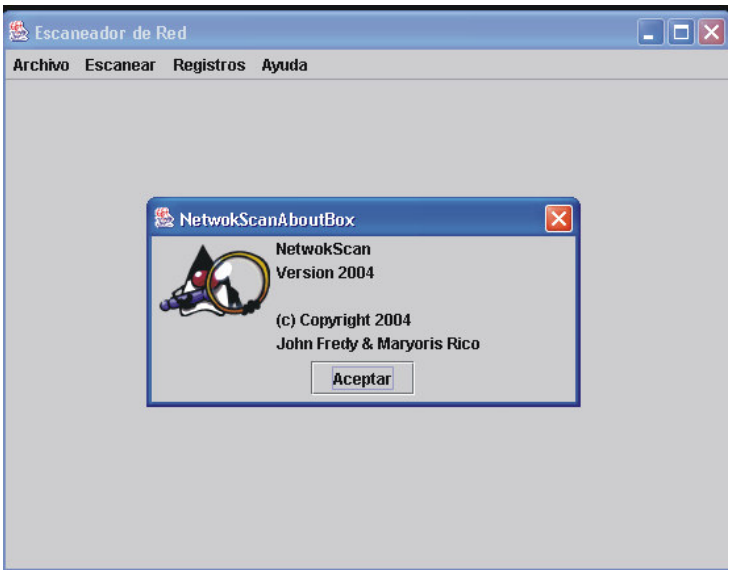


Figura 5.12 Visualización del menú ayuda

Después de la implementación debe validarse el nuevo sistema.

5.4 Validación y control del sistema

Antes de ser usado, el sistema debe ser probado. Es menos costoso encontrar problemas antes de que el sistema sea entregado a los usuarios o en este caso al administrador de la red. La herramienta es probada en sus laboratorios respectivos antes de entregar el resultado final. Primero se realiza una serie de pruebas para que se observen los problemas con datos de ejemplo y luego con datos reales del sistema actual para observar el desempeño del sistema.

Las pruebas se realizan a lo largo del desarrollo del sistema y no simplemente al final; esto significa sacar a la luz problemas no conocidos y demostrar el buen funcionamiento del programa.

Aunque las pruebas son tediosas constituyen un proceso que ayuda a asegurar la calidad del sistema. La prueba se realiza en subsistemas o módulos de programa conforme el trabajo avanza.

La prueba se hace en muchos niveles diferentes y a diversos intervalos. Antes de que el sistema sea puesto en producción, todos los programas deben ser probados en el escritorio, validadas con datos de prueba y revisados para ver si los módulos interactúan entre ellos, tal como se planeó.

También debe ser probado el sistema trabajando como un todo. Esto incluye probar también las interfaces.

Para el buen funcionamiento del sistema implementado se efectuaron las siguientes pruebas:

5.4.1 Pruebas de programas con datos de prueba

En esta etapa se probaron los programas en el escritorio para verificar la forma en que el sistema trabajaría.
Aquí se revisa si la rutina diseñada de cada módulo trabaja como fue escrita.
Las pruebas se hicieron con datos válidos e inválidos y luego esos datos fueron ejecutados para verificar las rutinas y corregir los errores encontrados y de esta manera con todos los módulos. Es importante revisar los datos de salida de las pruebas, para verificar que son los datos correctos.

Si un archivo fue creado y accesado revisar que tenga los datos reales, como la base de datos que lleva el registro de las direcciones IP.

5.4.2 Pruebas de enlace con datos de prueba

Después de probar cada módulo se pasa a realizar la prueba de enlace. Esta prueba revisa si los programas que son interdependientes trabajan como se planeó. Esta prueba se realiza con datos normales y luego con datos inválidos para asegurar que el sistema pueda detectar errores adecuadamente.

5.4.3 Prueba completa del sistema con datos reales

Después de realizar las pruebas de enlace, debe ser probado el sistema como una entidad completa con datos reales, como son: una dirección IP, una dirección de subred, y los servidores de la universidad.

El período de prueba es importante para valorar cómo interactúa el usuario con el sistema. Los conceptos a observar en este período son la facilidad de aprendizaje del sistema y la retroalimentación del sistema incluyendo lo que sucede cuando se recibe un mensaje de error, cómo reacciona el usuario al tiempo de respuesta del sistema dependiendo de la cantidad de información que procese.

Cuando el sistema tiene un buen diseño es más fácil mantenerlo y el gasto de dinero será menor. El mantenimiento se realiza para mejorar y actualizar el software de acuerdo a las necesidades que se presenten y evitar que el sistema deje de ser usado

5.5 Diseño de Interfaz del sistema

La estética y utilidad son importantes para crear una buena interfaz de usuario; además se debe entrenar a los tomadores de decisiones sobre la forma de Interpretar los datos que aparecen en pantalla para que les sean útiles.

Debido a que la salida útil es esencial para asegurar el uso y aceptación del sistema, se debe tener en cuenta varios objetivos como son:

- Diseñar la salida para que sirva al propósito deseado.
- Diseñar la salida para que el usuario entienda los datos.
- Mostrar los datos por módulos para que no haya sobrecarga de información que no se necesita.

Al igual que la salida es importante la calidad de la entrada de datos en un sistema. Es vital que las interfaces de entrada sean bien diseñadas para que los datos de salida sean los correctos.

Los diseños de las interfaces deben satisfacer objetivos de efectividad, precisión, facilidad de uso, consistencia, simplicidad.

La efectividad significa que las interfaces de entrada sirven a propósitos específicos del sistema, la precisión se refiere al diseño que asegura el llenado adecuado, la facilidad de uso significa que las interfaces son directas y no requieren tiempo adicional para descifrarlas; la consistencia significa, que los datos no van a cambiar de una aplicación a otra; y la simplicidad se refiere a mantener las interfaces sin amontonamientos de tal forma que el usuario enfoque su atención en ella.

6. CONCLUSIONES

Con este sistema se contribuye a la monitorización de los dispositivos de la red administrada por la División de Servicios de Información y al mejoramiento de la gestión de la red en la UIS.

Con el escáner de red se puede verificar y obtener información fundamental de las direcciones IP, los servicios disponibles en uso y los servidores de la universidad, para posteriormente tomar acciones correctivas que permitan brindar soluciones rápidas al problema encontrado.

Gracias a la implementación de este software de escáner de red se logra:

- Solucionar de una manera más rápida un inconveniente en la red de datos de la UIS.

- Controlar las asignaciones no autorizadas de direcciones IP y los servicios instalados, lo cual congestiona la red.

- La herramienta diseñada brinda la posibilidad de disminuir el tiempo de búsqueda de fallas que se presentan en los Servidores cuando salen de servicio.

- Proyectos como este, permite vincular al estudiante y darse cuenta de las falencias de la universidad, pero de igual manera nos brinda la oportunidad de dar soluciones y aplicar los conocimientos de ingeniería en la solución de dichos problemas, ayudando a la formación integral del ingeniero.

7. RECOMENDACIONES PARA LA CONITINUIDAD DE ESTE PROYECTO

- Implementar servicios que permitan mejorar el desempeño del software y satisfacer las nuevas necesidades del administrador de la red y de la universidad como ente educativo.

- Seguir con el control de la herramienta, buscando siempre mejorar la gestión de la red de datos de la universidad.

- Como las necesidades no surgen todas a la vez, cuando ocurra alguna, desarrollarla y hacer las pruebas necesarias para confirmar que el servicio implementado es funcional para el administrador de la red.

- Es importante establecer comunicación con el administrador, quien está encargado de controlar las asignaciones y servicios en la red de la universidad, esto ayuda a personalizar la herramienta, desarrollando módulos que realmente se van utilizar y no perder tiempo en cosas que no son funcionales.

8. BIBLIOGRAFIA

DOUGLAS E. COMER. - Redes de Computadoras, Internet e Interredes. - Editorial Prentice Hall. 1995

DEITEL Y DEITEL. Cómo programar en Java - Editorial Prentice Hall. México 1998.

SCHILDT Herbert, Java 2 Manual de Referencia - Editorial McGrawHill. Madrid 2001.

KENDALL & KENDALL, Análisis y diseño de sistemas. Editorial Prentice Hall. Mexico 1997.

9. REQUERIMIENTOS MINIMOS NECESARIOS PARA LA EJECUCION DEL SISTEMA ESCANER DE RED

REQUERIMIENTOS DE SOFTWARE.

- Sistema operativo Windows 98 o versión posterior, aunque puede ser cualquier sistema operativo haciendo adecuaciones mínimas puesto que la aplicación se encuentra hecha 100% en Java.
- Java™ 2 Runtime Environment, Standard Edition 1.4.2
- Debe estar creada la base de datos donde se encuentran registrados las direccines IP y servicios autorizados por el administrador de la red.
- El archivo ejecutable se llama *NetScan.bat*.

REQUERIMIENTOS DE HARDWARE

- Tarjeta de red.
- Estar conectado a una red.
- Espacio requerido en el disco 1 MB.
- Memoria de 64 MB (recomendable 128 Mb)
- Procesador MMX 200 Mhz. o Superior