UNIVERSIDAD AUTÓNOMA METROPOLITANA
UNIDAD AZCAPOTZALCO

UNA HERRAMIENTA DE GESTIÓN DE REDES VIRTUALES

TESIS

QUE PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

ABRAHAM JORGE JIMÉNEZ ALFARO.

DIRECTOR DE TESIS: DR. ROSSEN PETROV POPNIKOLOV

MÉXICO, D.F. **JULIO 2005**

INDICE

AGRADECIMIENTOS

A DIOS:

GRACIAS SEÑOR POR PERMITIRME CUMPLIR UNA META MÁS, POR ESTAR SIEMPRE CONMIGO Y CONCEDERME FORTALEZA EN LOS MOMENTOS DIFÍCILES.

POR HABERME DADO UNOS PADRES CARIÑOSOS A QUIENES LES DEBO EL ESTAR AQUÍ.

A MIS PADRES:

AGRADEZCO DE CORAZÓN TODO SU AMOR QUE ME HAN BRINDADO, EL APOYO INCONDICIONAL Y EL EJEMPLO MÁS MARAVILLOSO DE PERSEVERANCIA Y LUCHA ANTE LA VIDA.

AGRADEZCO A LA UNIVERSIDAD AUTÓNOMA METROPOLITANA, UNIDAD AZCAPOTZALCO Y EN GENERAL A TODOS LOS PROFESORES QUE COLABORARON EN MI DESARROLLO PROFESIONAL, ESPECIALMENTE AL DIRECTOR DE TESIS DR. ROSSEN PETROV POPNIKOLOV.

A MIS HERMANOS:

POR SU CONFIANZA Y APOYO QUE ME BRINDARON. GRACIAS POR LOS MOMENTOS COMPARTIDOS

A MI FAMILIA Y AMIGOS:

GRACIAS POR TODO SU AMOR, CARIÑO, RESPETO Y TERNURA; Y QUE JAMÁS LOS VOY A OLVIDAR.

A LA H. COMISIÓN REVISORA DE TESIS:

POR LOS CONSEJOS Y CORRECCIONES EN EL DESARROLLO DE ESTE TRABAJO

GRACIAS.....

RESUMEN

Un Futuro prometedor de la Conmutación Ethernet es la red virtual de área local, VLAN (*Virtual Local Area Network*). Una VLAN es una agrupación lógica de dispositivos o usuarios que se pueden agrupar por función, departamento o aplicación, sin importar la ubicación física del segmento. Los dispositivos en la VLAN están restringidos a comunicarse entre si están en la misma VLAN, de lo contrario será necesario establecer algún mecanismo de ruteo para el intercambio de información entre las VLAN´s.

En el trabajo de tesis se desarrolla una Herramienta de Gestión de Redes Virtuales con el fin de monitorear, configurar/reconfigurar VLAN's, consta de varios módulos tales como: SNMP GET, SNMP SET, SNMP GETNEXT, MONITOREO, SNMP VISUALIZADOR Y ESTADÍSTICAS DE RED. La herramienta es gratuita y de libre uso por parte de las comunidad de redes versus las herramientas propias de cada proveedor de VLAN, que valen de cientos a miles de pesos.

Los módulos de la herramienta de Gestión de Redes Virtuales pueden trabajar de forma local o remota, a través de la pila de protocolos TCP/IP (*Transfer Control Protocol/Internet Protocol*). El protocolo que se emplea para acceder a las bases de datos de configuración de los equipos o nodos, es el protocolo de administración simple de red, basado en UDP (*User Datagram Protocol*), SNMP (*Simple Network Management Protocol*), los puertos en uso son el 160 y 161 que son los que permiten el envío y recepción de información entre la estación de administración y la estación administrada.

Para analizar la red: a) se capturan los paquetes de datos; b) se desencapsula el paquete en los elementos de interés para el análisis de red y finalmente se presenta la información en pantalla, por lo que es posible capturar:

- **Paquetes, cuyo origen o destino sea un host determinado.**
- **Paquetes, cuya dirección origen o destino corresponda a una red determinada.**
- **Paquetes, cuyo puerto de origen sea algún número de puerto.**
- **Paquetes, TCP, UDP y los puertos correspondientes.**
- **Paquetes, cuya dirección destino corresponda a una red determinada.**
- **Cualquier combinación de los anteriores.**

El desarrollo modular de la herramienta permite que la integración de nuevos módulos, que brinda nuevas funcionalidades, sea flexible sin afectar a los que ya están en función, por mencionar un ejemplo: El módulo de seguridad SSH (*Security Shell*) y el módulo de Cortafuegos, para la seguridad de la información que se extrae de los equipos y del análisis de red.

Información adicional acerca del detalle de la herramienta de gestión de redes virtuales se encuentra en el desarrollo capitular de la tesis.

ABSTRACT

An important feature of Ethernet switching is the virtual local-area network (VLAN). A VLAN is a logical grouping of devices or users. These devices or users can be grouped by function, department, or application despite the physical LAN segment location. Devices on a VLAN are restricted to only communicating with devices that are on their own VLAN. Just as routers provide connectivity between different LAN segments, routers provide connectivity between different VLAN segments.

The thesis to provide a tool for monitoring, configuration or reconfiguration of VLANs through software. The tool has several modules, such as: SNMP GET, SNMP SET, SNMP GETNEXT, MONITOR, SNMP VIEWER and NETSTATISTICS. The porpuse was made a tool free, because each vendor has developed its own proprietary VLAN product.

The parts of the tool, modules, work in local and remote environments. Tha tool use the TCP/IP and UDP protocols for analizing traffic statistics of the network. There is a protocol called SNMP(Simple Network Management Protocol), the protocol is used to manage, monitor and configurate nodes in local or remote form. The protocol use the ports 160 and 161 to communicate information about nodes confgurations.

For analizing the network the tool capture frames and dispaly information about network performance, the tool provide filtering, security and traffic flow management, for example:

a) Capture frames about a particular node.

b) Capture frames about a particular network.

c) Capture frames with a particular evaluation criteria.

d) Whatever criteria.

The tool can enhance modular scalability, security, and network management without problem. The modular architecture permit to increase modules for security, for example a firewall or SSH protocol, so, the information from nodes will be protected. Therefore, the tool is flexible due to simplify task when additions, moves or changes to the architecture are necessary.

Additional Information is given in the structure of the thesis which is divided in several chapters.

ACRÓNIMOS

3Pty	Third-party reference point
3GPP	Third Generation Partnership Project
4GL	4th (Fourth) Generation Language

A

AAL	ATM Adaption Layer
AB	Architecture Board (Grupo del OMG)
ABR	Available Bit Rate
ABT	ATM Block Transfer
AC	Access Connector
AcC	Account Collation
ACA	Alarm Correlation Agent
ACI	Administrative and Control InterFace
ACID	Atomicity, Consistency, Isolation and Durability (Propiedades de las Transacciones)
ACSE	Association Control Service Element
ACSI	American Communications Services, Inc.
ACTS	Advanced Communications Technologies & Services
AD	Area Directors (del IETF)
ADM	Add/Drop Multiplexer
ADPCM	Adaptive Differential Pulse Code Modulation
ADSL	Asymmetric Digital Subscriber Line
AF	Auxiliary carry Flag
AI	Artificial Intelligence
AIN	Advanced Intelligent Network
AIP	Applicaation infraestructure Provider
AIS	Alarm Indication Signal
ALM	NetWare application loadable module
AMA	1.Accounting Management Application, 2.Automatic Message Accounting
AMATS	Automatic Message Accounting Teleprocessing System
AMO	Affected Managed Object
AMPS	Advanced Mobile Phone Service
AMS	Applications Management System
AMX	Administrator and Maintenance interface
AN	1.Alarm Notifier (Spectrum), 2. Active Network
ANEP	Active Network Encapsulation Protocol
ANSA	Advanced Network Systems Architecture project (da orígen al concepto DPE de TINA-C)
ANSI	American National Standards Institute
ANT	Access Network Transport
AP	1.Auxiliary projects (Proyecto de miembro de TINA-C), 2. Application Process
API	Application Programming Interface
ARA	Apple Remote Access
ARM	Asynchronous Response Mode
ARP	Address Resolution Protocol
AS	Access session (arquitectura de servicios)
ASCII	American Standard Code for Information Interchange
ASE	Application Seervice Elements
ASM	Adaptive System Management
ASN.1	Abstract Syntax Notation.l
ASP	Application Service Providers
ASPIC	Application Service Provider Industry Conservatium

ASR	1.Automated Speech Recognition, 2.Address Space Register, 3.Automated Send/Receive, 4.Answer/Seize Ractio
ATC	ATM transfer capability
ATIS	Alliance for Telecommunications Industry Solutions (umbrella organisation in US)
ATM	Asynchronous Transfer Mode (Redes de Banda Ancha)
ATMF	ATM Forum
ATMR	ATM Ring
AuC	Authentication Center

B

B-ISDN	Broadband Integrated services Digita Network
B-NT2	Network Termination 2
B-NTI	Network Termination 1
BBE	Blackground Block Error
BBER	Blackground Block Error Ratio
BCSM	Basic call state model (Red Inteligente)
BECN	Backward Explicit Congestion Notification
BER	Bit Error Rate BH Busy Hour
BIB	Binding Information Base
BICC	Bearer – Independent Call Control
Bkr	Broker reference point (TINA-C)
BLSR	Bidirectional Line Switch Ring
BM	Business Management
BML	Business Management Level
BMO	Base Managed Object
BODTF	Business Object Domain Task Force (OMG)
BOSS	Broadband Operations Support System
BP	Business Process
BPMS	Business Process Management System
BPR	Business Process Reengineering
BRI	Basic Rate Interface
BRLC	Broadband Remote Line Concentrator
BSC	Base Station Controller
BSMS	Broadband Service Management System
BSS	Base Station System
BTS	Base Transceiver Station

C

C	Lenguaje de Programación C
C++	Lenguaje de Programación C Orientado a Objeto
CAD	Computer Aided Design
CAD/CAM	Computer Aided Design/Computer Aided Manufacturing
CAMEL	Customer Advanced Mobility Enhanced Logic (Red Móvil)
CAN	Customer Access Network
CAP	1.Competitive Access Carrier, 2.Computer Aided Publishing, 3.Carrierless Amplitude Phase
CARS	Command and Response System
CASE	Computer Aided Software Engineering
CATV	1.Community Antenna TeleVision 2.Cable Television (Community Antenna Television)
CBR	1.Constant Bit Rate, 2.Case-Based Reasoning
CC	Connection co-ordinator Network resource architecture
CCF	Connection co-ordinator factory
CCITT	Consultative Committee on International Telegraphy and Telephony (ahora ITU)
CCM	CORBA Component Model (Grupo OMG)
CCR	Call Completion Record

CD	Cell Delay
CDDI	Cable Distributed Data Interface/Copper Data Distribution Interface
CDMA	Code (or Call) Division Multiple Access
CDPD	Cellular Digital Packet Data
CDR	Call Detail Records
CDR/AMA	Call Detail Records/Automatic Message Accounting
CDV	Cell Delay Variation
CDVT	Cell Delay Variation Tolerance
CE	Cell Error
CEC	Chargeable Event Collection
CEPT	European Conference of Posts and Telecommunications
CER	Call Event Record
CES	Circuit Emulation Switching
CFP	CNM Feature Packages
CI	Critical action Indicator
CIF	Central Information Facilities (TMforum)
CIM	Common Information Model
CIR	Committed Information Rate
CL	Cell Loss
CLEC	Competitive Local Exchange Carrier (Acceso)
CLI	1.Command Line Interface 2.Calling Line Identifier
CLLM	Consolidated Link Layer Management
CLNAP	Connectionless Network Access Protocol
CLNIP	Connectionless Network Interface Protocol
CLNS	Connectionless Network Service
CLP	Cell Loss Priority
CLR	1.Circuit Layout Records 2.Cell Loss Ratio
CM	Cell Minisertion
CMA	Connection Management Architecture
CMIP	Common Management Information Protocol
CMIP	Common Management Interface Protocol (ITU-T TMN)
CMIS	Common Management Information Services (ITU-T TMN)
CMISE	Common Management Information Service Element (ITU-T TMN)
CMOL	CMIP Over Logical link control
CMOT	CMIP Over TCP/IP
CMR	Cell Misinsertion Ratio
CN	Core Network
CNM	Customer Network Management
CO	1.CentralOffice, 2. CO Computational object Elements of the computational model
COBOL	Common Business Oriented Languaje (lenguaje de Programación)
COM	Component Object Model (producto de Microsoft)
ConS	Connectivity-service reference point (TINA-C)
COPS	Common Open Policy Service
CORBA	Common Object Request Broker Architecture (Arquitectura de OMG)
COSS	Common Object Service Specification (Arquitectura de OMG)
COTS	Comercial Off The Shelf (componentes de software prontos para usar)
CPE	Customer Premises Equipment
CpoF	Common point of Failure
CPU	Central Processing Unit
CRC	1.Cyclic Redundance Code, 2.Class-Responsibility-Collaboration
CRMA	Cyclic Reservation Multiple Access
CS	Computer System
CSA	Computer System Agent
CSB	Consortium Steering Board Management of TINA-C
CSCW	Computer-supported co-operative work (Servicios de teletrabajo de GroupWare)

CSLIP	Compressed Serial Line Interface Protocol
CSLN	Client-Server Layer Network Business Relationship (TINA.C)
CSM	Communication session manager (parte de la arquitectura de recursos de red)
CSMA/CD	Carrier Sense Multiple Access/with Collision Detection
CSMF	Communication session manager factory
CSPP	Computer Systems Policy Project (GII)
CSTA	Computer-supported telecom applications (Interfase para PABX's)
CSU	Channel Service Unit
CT	Core Team (Grupo residente permanente de ingenieros de TINA-C)
CT2	Cordless Telephony Generation 2
CTC	Consortium Technical (Comité de gestión técnica de TINA-C en su primer fase)
CTD	Cell Transfer delay
CTI	1.Computer Telephone Integration, 2.Computer Telephony Interface
CTM	Cordless telephone mobility
CTP	Connection Termination Point
CU	Control Unit
CWM	Common Warehouse Metamodel (término del OMG)

D

DACS	Digital Access Cross-connect System
DAI	Distributed Artificial Intelligence
DAVIC	Digital Audio-Vlsual Council (Foro de servicios de multimedia)
DB	Database
DBMS	Database Management System
DBR	Deterministic Bit Rate
DCE	1.Data Communication Equipment, 2.Distributed Computing Environment
DCF	Data Communication Function (bloque de función lógica ITU-T M3010)
DCN	Data Communication Network
DCOM	Distributed Component Object Model
DCS	1.Digital Cross-Connect, 2.Distributed Communication System
DCSC	Data Customer Support Center
DE	Discrete Element
DiffServ	Differentiated Services
DLC	1.Data Link Connection, 2.Digital Loop Carrier
DLCI	Data Link Connection Identifier
DLS	Data Link Switching
DME	Distributed Management Environment
DMH	Data Message Handler
DMI	Desktop Management Interface
DMT	Discrete Multitone
DMTF	1.Desktop Management Task Force (Foro de debate de la gestión de elementos de oficina) 2.Distributed Management Task Force
DN	Distinguished Name
DNS	1.Domain Name System /Service 2.Domain Naming Service
DOLMEN	Development for an Open Long-term Mobile neetwork ENvironment (proyecto ACTS)
DOT	Distributed Object Technology
DPE	Distributed Processing Environment (definido por TINA-C)
DPL	Degraded Performance Limit
DQDB	Dual Queue Dual Bus
DS	1.Directory Services 2.Differentiated Services
DSl	T1 1.544 Mbps (digital signal)
DS2	T2 6.312 Mbps (digital signal)
DS3	T3 44.736 Mbps (digital signal)
DS4	T4 274.176 Mbps (digital signal)
DSCP	DiffServ Code Point

DSI Dynamic Skeleton Interface
DSF Directory System Function
DSn Digital Speed Number Elements
DSL Digital Subscriber Line
DSS Digital signaling system
DSU Data Service Unit
DTC Domain Technology Committee (Grupo del OMG)
DTE Data Terminal Equipment
DTP Distributed Transaction Processing
DVB Digital Video Broadcasting
DWDM Dense WDM

E

EA 1.Extension Address, 2.Emergency Action
EAICD Emergency Action/Control Display Channel
EB Electronic Bonding
EC Electronic Commerce
ECA Event Correlation Agent
ECBP End-to-end Connection Blocking Probability
ECDTF Electronic Commerce Domain Task Force
ECIC Electronic Communications Implementation Committee (comite de trabajo de TCIF)
ECO Engineering Computational Object
EDC Error detection Code
EDGE Enhanced Data rates for GMS Evolution
EDI 1.Electronic Data Interface, 2. Electronic Data Interchange
EFD Event Forwarding Discriminators
EFS Error Free Seconds
EGP Exterior Gateway Protocol
EII European information Infrastructure
EIR 1.Equipment Identity Register, 2.Equipment Identity Requester
EL Element Layer
ELAN Extended LAN
EM Element Manager
E-MAIL Electronic Mail
EML Element Management Layer
EML-CP Element management layer connection performer
EML-TC Element management layer topology configurator
EMS 1.Element Management System, 2.Enterprise Management System
EQOS EURESCOM Quality of Service
ERP Enterprise Resource Planning
ES Errored Second
ESMR Enhanced Specialized Mobile Radio
ESN Emergency Service Number
ESR Errored Second Ratio
ETSI European Telecommunications Standards Institute
EURESCOM European Institute for Research and Strategic Studies in Telecommunications
Euro-ISDN European-Integrated Services Digital Networks

F

F Interfaz F (TMN communication protocol-ITU-T)
FAB Fulfillment, Assurance and Billing
FC Fault coordinator
FCAPS Fault, Configuration, Accounting, Performance, Security (ITU-T TMN)
FCC 1.Federal Communications Commission, 2. Flow connection controller (arquitec. de recurso de red)

FDD	Frequency Division Duplex
FDDI	Fiber Distributed Data Interface (datos)
FDDI II	Fiber Distributed Data Interface (voz y datos)
FDM	Frequency Division Multiplexing
FEBE	Far End Block Error
FECN	Forward Explicit Congestion Notification
FERF	Far End Reporting Failure
FHR	Fixed Hierarchical Routing
FIFO	First In, First Out
FITL	Fiber In The Loop
FLND	Foreign layer network domain
FOTS	Fiber Optic Terminating Systems
FR	Frame Relay
FRAD	Frame Relay Access Device
FRADs	Frame Relay Access Devices
FRS	Frame Relay Switch
FSA	1.Finite State Automata 2.Framework Study Area
FSN	Full-Service Network
FTAM	File Transfer Access and Management
FTD	Frame Transfer Delay
FTMP	File Transfer Management Protocol
FTP	File Transfer Protocol
FTTC	Fiber To The Curb

G

G- CDR	Gateway GPRS Support Node-Call Detail Record
GDMO	Guidelines for the Definition of Managed Objects (OSI)
GERAM	GSH/EDGE Ratio Access Network
GFR	Guaranted Frame Rate
GGS	GPRS Gateway Support Node
GII	Global Information Infrastructure
GIOP	Generic Inter-ORB Protocol (CORBA)
GIR	Graphic Information Requirements
GIS	Graphical Information System
GMM	Global multimedia mobility
GNE	Gateway Network Element
GNM	Generic Network Model
GOM	Generic Object Model
GOS	Grade Of Service
GPRS	General Racket Ratio Service
GRM	General Relationship Model
GSM	Global System for Mobile communication
GSM	Groupe Speciale Mobile (Red Móvil)
GSMP	General Switch Management Protocol
GTP	Group Termination Point
GUI	Graphical User Interface

H

HAN	Home Access Network
HCPN	Hybrid Circuit-switehed/Packet-based Network
HDSL	High-bit-rate Digital Subscriber Line
HDT	Host Digital Terminals
HDTV	High Definition Television
HEC	Header End Control
HFC	Hybrid Fiber/Coax
HFW	Hybrid Fiber/Wireless

HLR	Home location register
HLN	Home Lan Network
HMMO	Hypermedia Managed Object
HMMP	HyperMedia Management Protocol
HMMS	HyperMedia Management Schema
HMOM	HyperMedia Object Manager
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HU	High Usage
HVSP	High Value Service Provider

I

I-ETS	Interim European Telecommunication Standards
IA	Infrastructure Agent
IAB	Internet Architecture Board (ISOC)
IANA	Internet Asignation Number Authority (ISOC)
IAP	Internet access provider
IC	Interexchange Carrier
ICI	Interexchange Carrier Interface
ICF	Information Conversion Function
ICMP	Intemet Control MessageProtocol
IDEAS	Intelligent Dynamic Event Analysis Subsystem
IDL	Interface Definition Language (Estándar OMG)
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical & Electronics Engineers
IESG	Internet Engineering Steering Group (ISOC)
IETF	Internet Engineering Task Force (ISOC)
IEX	InterEXchange carrier
IH	Intermediate High usage
IIOP	Internet Inter-ORB Protocol (CORBA)
ILEC	Incumbent Local Exchange Carrier
ILMI	Interim Local Management Interface
ILMI MIB	Interim Local Management Interface Management Information Base
ILP	Inductive Logic Programming
IMEI	International Mobile Equipment Identity
IMT	International Mobile Telecommunications
IMT2000	Especificaciones de radio móvil de tercera Generación de ITU-T
IN	1.Intelligent Network, 2.Intelligent Node
INAP	Intelligent Network Application Protocol (Red Inteligente)
INMD	In-Service Non-intrusive Measuring Device
IntServ	Integrated Services
IOP	Inter-Orb Protocol (CORBA)
IOPS.ORG	Internet Operators Group
IOR	Interoperation Object Reference (CORBA)
IOS	Internetwork Operating System
IP	Internet Protocol
IPDV	IP Packet delay Variation
IPER	IP Packet Error Ratio
IPLR	IP Packet Loss Ratio
IPMH	Interprocess Message Handler
IPMHD	Interprocess Messaging Handler Daemon
IPPM	IP Peerformance Metrics
IPTD	IP Packet Transfer Delay
IPv6	Internet Protocol versión 6
IPX	Internet Package Exchange
IPX/SPX	Internet Package Exchange/ Sequenced Package Exchange

IRTF	Internet Research Task Force
iSAC	integrated Service Activation Control ler
iSAC/CA	integrated Service Activation Controller client application
iSAC/GUI	integrated Service Activation Controller Graphical User Interface
iSAC-NSAM	integrated Service Activation Controller Network Service Activation Manager
iSAC/PDT	integrated Service Activation Controller process definition tool
iSAC/TDT	integrated Service Activation Controller/Task Definition Tool
iSAC/WE	integrated Service Activation Controller/Workflow Engine
ISAM	Indexed Sequential Access Method
ISCP	Intelligent Service Control Point
ISDN	Integrated Services Digital Network
ISDN PRI	Integrated Services Digital Network Primary rate
ISM	In-Service Monitoring
ISNM	International Symposium on Integrated Network Management
ISO	1.International Standards Organization 2.International Organization for Standarization
ISOC	Internet SOCiety
ISP	Internet Service Provider
ISSI	Inter Switching Interface
IST	Information Society Tecnologies
ISUP	ISDN Signalling User Part
ISV	Independent Software Vendor
IT	Information Technology
ITAA	Information Tecnology Association of America
ITU	International Telecommunications Union
ITU-R	Iternational Telecommunication Union Radiocommunication sector
ITU-T	1.International Telecommunications Union- Telecommunications 2.International Telecommunications Union-Telecommunications Standarization Sector
IU	International Union
IVR	Interactive Voice Response
IXC	Interexchange Carrier

J

JaMAPI	Java Management Application Programming Interface (API)
JAVA	Lenguaje de programación (Marca registrada de SUN)
JIDM	Joint Inter Domain Management (X/Open y TMForum)
JIT	Just In Time
JMAPI	Ver JaMAPI
JPEG	Joint Photografic Experts Group
JTAPI	Java Telephony Application Programming Interface

K

Kbps	Kilo bits per second (mil bits/second)
kTN	Kernel transport network (enlace lógico de los nodos en ambiente DPE)

L

LAN	Local Area Network
LAPD	Link Access Procedure D
LAT	Local Area Transport
LATA	Local Access and Transit/Transport Area (las LEC fueron divididas en de Local Access y en de Transit Areas)
LDAP	Lightweight Directory Access Protocol
LDP	Label Distribution Protocol
LCN	Logical Channel Number

LEC Local Exchange Carrier (les fue requerido que presenten el tráfico en un punto denominado POP)
LEX Local Exchange
LHC Long-Haul Carriers (empresa de transporte inter-LATA, tambien denominada "Carrrier")
LLA Logical Layered Architecture (ITU-T layered architecture) (Bellcore)
LLC Logical Link Control
LLCI Logical Link Control 1
LLND Local layer network domain
LMDS Local Multipoint Distribution Service
LN Layer network (arquitectura de recurso de red)
LNB Layer network binding
LNBM Layer network binding manager
LNC Layer network co-ordinator
LND Layer network domain
LNFed Layer Network Federation Business relationship (TINA-C)
LNTC Layer network topology configurator
LOA Library Object Adapter
LOS Loss of Service
LSR Label Switching Router
LT Logical Terminal
LTE Line Terminating Equipment
LTP Link termination point

M

M2M Manager-to-Manager
MA Management Application
MAC Media Access Control
MAF Management Application Function (reeefiere a las funciones FCAPS de TMN-ITU-T)
MAN Metropolitan Area Network
MAP Mobile Access Protocol Mobile networking
MBONE Multicast Backbone
Mbps Megabits per second (Millon de bits/second)
MBR Model-Based Reasoning
MBS Maximun Burst Size
MCF Message Communication Function
MCR Mean Cell Rate
MCU Multipoint Control Unit
MD Mediation Device
MF 1.Mediation Function (bloque de función lógica ITU-T M3010) , 2.Management Function
MI Management Information
MIB Management Information Base
MIF Management Information Format
MIM Management Information Model
MISA Management of Integrated SDH and ATM Networks (proyecto ACTS)
MIT Management Information Tree
MMDS Microwave Multipoint Distribution Service
MO Managed Object
MOCS Managed Objects Configuration System
MOF 1.Management Object Format. 2.Meta-Object Facility (término del OMG)
MOM Manager Of Managers
MOP Maintenance Operations Protocol
MOS Mean Opinion Score
MPEG-2 Motion Picture Experts Group (norma 2)

MPLS Multiprotocol Label Switching
MR Modified Read
MS Mobile Station
MSA Multiple Software Agent
MSC Mobile Switching Center
MSM Multimedia Service Management
MTBF Mean Time Between Failure
MTBO Mean Time Between Outage
MTIE Maximun Time Interval Error
MTPS Mean Time to Provide Service
MTRS Mean Time to Provide Restore Service
MTTP Mean Time to Provide Provision
MTTR Mean Time to Provide Repair
MTSO Mobile Telephone Switching Office
MUX Multiplexor

N

N-ISDN Narrow band Integrated Service Digital Network
NAU Network Addressable Unit
NC Network Connector
NCA Network Computing Architecture (marca de fábrica de Oracle Corporation)
NCCE Native Computing and Communication Environment (TINA)
NCIH North Carolina Information Highway
NE Network Element
NEF 1.Network Element Function (bloque de función lógica ITU-T M3010) 2.Network Element Facility
NEM Network element management
NER Network effectiveness Ratio
NFC Network flow connection (arquitectura de recurso de red)
NFEP Network flow end-point (elemento de la arquitectura de recurso de red)
NGN Next Generation Network
NGOSS New Generation of Operations Systems and Software (marca registrada de TMForum)
NII National Information Infrastructure
NIU Network Interface Units
NLM 1.Network Layer/Level Managemen, 2.NetWare Loadable Module.
NM Network Management
NMA Network Monitoring and Analysis
NMDG Network Meassurement Development Group
NMF Network Management Forum (ahora TeleManagement Forum)
NML Network Management Layer
NML-CP Network management layer connection performer
NML-TC Network management layer topology configurator
NMO Network Management Objects
NMS Network Management System
NNI 1.Network Node Interface, 2.Network to Network Interface
NNM Network Node Manager
NO Network Operator
NOC Network Operations Center
NP Network Performance
NP&D Network Planning and Development
NPC Network Parameter Control
NRA Network resource architecture
NRIM Network resource information model
NSAP Network Service Access Point
NSDB Network and Service DataBase

NSM	Network Security Manager
NSP	1.Network Server/Service Provider 2.Network Servic Provider
NT	Network Termination
NTl	Network Termination type 1
NT2	Network Termination type 2
NTCM	Network topology configuration management
NTE	Network Terminating Equipment
NVRAM	Non-Volatile Rapid Access Memory
NWCTP	Network connection termination point
NWTTP	Network trail termination point (elemento de la arquitectura de recurso de red)
NX	NetExpert
NXCMIP	NetExpert Common Management Information Protocol
NXV	NetExpert Vectors

O

OAM	Operation, Administration, and Maintenance
OAM&P	Operation, Administration, Maintenance, and Provisioning
OCE	Operation Creation Environment
OCL	Object Constraint Language
ODA	Object Database Adapter
ODBC	Open Database Connectivity
ODBMS	Object Database Management System
ODL	Object Definition Language (extensión IDL para describir objetos TINA)
ODMA	Open Distributed Management Architecture
ODMG	Object Data Management Group
ODP	Open Distributed Processing (ISO standard)
ODTA	Open Distributed Telecommunication Architecture
OI	Outage Intensity
OID	Object Interface Definition
OIF	Optical Interworking Forum
OLAP	On-Line Analytical Processing
OLE	Object Linking and Embedding
OLTP	On-Line Transaction Processing
OMA	I.ObjectManagementArchitecture, 2.Open Management Architecture
OMG	Object Management Group (CORBA , UML)
OMNIPoint	Open Management Interoperability Point
OMS	Off-line Management System
OMT	Object modeling technique
ONP	Open Network Provision
OO	1.Object Oriented, 2. Object Orientation
OOA	Object Oriented Analysis
OODBMS	Object Oriented Database Management System
OODBS	Object Oriented Database System
OOSE	Object Oriented Software Engineering
OPX	Operator interface
OQL	Object Query Language
ORB	Object request broker (objeto en el Kernel de la arquitectura CORBA)
ORDBS	Object Relational Database System
OS	1.Operation System (ITU-T TMN), 2.Operational Support
OSF	1.Operations Systems Functions (bloque de función lógica ITU-T M3010), 2.Open Software Foundation
OSI	Open System Interconnection (de ISO)
OSM	Open Service Model (for Global Information Brokerage and Distribution)
OSS	1. Operations Support System, 2. Operations Systems and Software
OT	Object Technology

OTN Optical Transport Network
OTS 1.Object Transaction Service , 2.Off The Shelf

P

PA Provider agent
PAD Packet Assembler/Disassembler
PBX Private Branch Exchange
PC Personal Computer
PCM Pulse Code Modulation
PCMCIA Personal Computer Memory Card International Association
PCR Peak Cell Rate
PCS Personal Communication Service
PDH Plesiochronous digital hierarchy
PDN Public Data Network
PDP Pocket Data Protocol
PDU Protocol Data Unit
PEI Peak Emission Interval
PHB Per Hop Behavior
PHY Physical Layer
PIB Policy Information Base
PICS Plug-in Inventory Control System
PIN Personal Identification Number
PIR Peak Information Rate
PLAN Programming Language for Active Networks
PLM Product Line Management
PLMN Public Land Mobile Network
PNO Public Network Operator
PMD Physical Medium Dependent
PNO Public Network Operator
PO Post Office (nombre también de la oficina telefónica)
POA Portable Object Adapter
POH Path Over Head
POP Point Of Presence
POS Persistent Object Service
POSC Petrotechnical Open Systems Corporation
POTS Plain Old Telephone Service
PPP Point to Point Protocol
PRI Primary Rate Interface
PRM Performance Report Message
PROSPECT Prospect of Multidomain Management in the Expected Open Services Market (proyecto ACTS)
PS Packet Switching
PSDN Public Switched Data Network
PSS 1.Provider service session, 2. Persistent State Service
PSTN Public Switched Telephone Network
PTC Platform Technology Committee (Grupo del OMG)
PTT Postal Telephone and Telegraph
PVC 1.Permanent Virtual Circuit, 2. Private Virtual Circuit

Q

Q3 Interfaz Q3 (TMN communication protocol-ITU-T)
q3 TMN reference point
QA 1.Q Adapter, 2.Quality Assurance
QAF 1.Quality Assurance Function, 2.Q Adapter Function (bloque de función lógica ITU-T M3010)

QoS Quality of Service
QOSF Quality of Service Forum
QSDG Quality of Service Development Group
Qx Interfaz Qx (TMN communication protocol-ITU-T)

R

RADSL Rate-Adaptive Digital Subscriber Line
RAM 1.Rapid Access Memory, 2.Random Access Memory
RAN Radio Access Network
RBOC Regional Bell Operating Company (7 empresas regionales en que se dividió AT&T en 1984, para actuar como LEC's)
RBR Rule-Based Reasoning
RDBMS Relational Database Management Systems
RDC Remote Diagnostics Center
RDN Relative Distinguished Name
RDT Recall Dial Tone
REFORM Resource Failure and restORation Management in ATM-based (proyecto ACTS)
Ret Retailer reference point (TINA-C)
ReTINA Real-time TINA compliant DPE (proyecto ACTS)
RF Radio Frequency
RFC Request for Comments
RFI Request for Information
RFP Request for Proposal
RFR/S Requests for refinement and solutions (Proceso de adopción de especificaciones en TINA)
RFSD Ready for Service Date
RLA Resource Level Accounting
RMI Remote Method Invocation
RM-ODP Reference model - open distributed programming (estandard ISO)
RMON Remote Monitoring Specification
ROI Return On Investment
ROSE Remote Operations Service Element
RP Reference Point
RPC Remote Procedure Call
RPO Reference Performance Objective
RSVP Resource reSerVation Protocol
RT Response Time
RTA Response Time Agent
RtR Retailer-to-retailer reference point (TINA-C)

S

S-CDR Serving GPRS Support Node-Call Detail Record
S&OP Sales and Order Processing
SA 1.Service Adapter, 2. Security Agent, 3. Service architecture, 4. Software Agent, 5.Service Availability
SAP Service Access Point
SBR Statistical Bit Rate
SCE Service creation environment (Red Inteligente)
SCF Service control function
SCMP Simple Connection Management Protocol
SCP 1.System Control Program, 2. Service Control Point (Red Inteligente)
SCR Sustainable Cell Rate
SCREEN Service CReation Engineering Environment (proyecto ACTS)
SD Service Delivery
SDA Software Distribution Agent
SDF Service data function

SDH	Synchronous Digital Hierarchy
SDL	Specification Description Language
SDSL	Single-line Digital Subscriber Line
SE	1.Software Engineering, 2.Service Element
SECB	Severely Errored Cell Block
SECBR	Severely Errored Cell Block Ratio
SEFS	Severely Erred Framing Seconds
SEP	Severely Errored Period
SEPI	Severely Errored Period Intensity
SES	Severely Errored Second
SESR	Severely Errored Second Ratio
SF	Service factory (Instancia la sesión de servicio de los CO's)
SFC	Stream flow connection
SFEP	Stream flow end-point
SG	Study Group
SGML	Standardized Generalized Markup Language
SGSN	Serving GPRS Support Node
SIB	Service-independent building block (Red Inteligente)
SIM	SIMulator
SIP	1.SMDS Interface Protocol, 2.Services Instance Provision
SIP/SAP	Session Identification Protocol/Session Announcement Protocol
SIR	Sustained Information Rate
SLA	Service Level Agreement
SLIP	Serial Line Internet Protocol
SLM	Service Level Management
SLO	Service Level Objective
SLR	Service Level Report
SLS	Service Level Specification
SLSU	Service Level Specification and Usage
SM	System/Service Management
SMAE	System Management Application Entities
SMASE	System Management Application Service Element
SMC	System Management Communication
SMDS	1.Switched Megabit Data Service, 2.Switched Multimegabit Data Service
SME	Small & Medium Business
SMF	1.Systems/Service Management Function, 2.System/Station Management Function
SMI	1.Structure of Management Information, 2.System Management Information
SMK	Shared Management Knowledge
SML	Service Management Layer
SMO	Service Management Object
SMP	Service management point (Red Inteligente)
SMS	Service Management System
SMTP	Simple Mail Transfer Protocol
SNA	Systems Network Architecture
SNI	System network Interface
SNMP	Simple Nctwork Management Protocol (de IETF)
SO	Switching Office
SOCS/SOAC	Service Order Control System and Service Order Analysis and Control
SOHO	Small Office Home Office
SONET	Synchronous Optical Network
SP	1. service Provider, 2.Service Parameter
SP&D	Service Planning and Development
SPINA	Subscriber PersonalIdentification Number Access
SPINI	Subscriber Personal Identification Number Intercept

SPIRIT	Service Providers Integrated Requirements for Information Technology (inicio de TMforum)
SPT	Spare Part Tracking
SPVC	Semi-Permanent Virtual Circuit
SPX	Sequenced Packet Exchange
SQL	Structured Query Language
SQM	Service Quality Management
SR	Service Resourcing
SRES	Signed Response
SRM	Service Resource Management
SS	Service session (arquitectura de servicio)
SS7	Signaling Systcm number 7
SSF	Service switching function
SSM	Service session manager (arquitectura de servicio)
SSP	Service Switching Point (Red Inteligente)
STASE	Secure Transformation Application Service Element (de ROSE)
STDL	State Transition Diagram Language
STDM	Statistical Time Division Multiplexing
STG	State Transition Graph
STM-N	Synchronous Transfer Module level N
STP	Signaling Transfer Point
STP	Signaling Transfer Point
STU	Set Top Unit
SVC	1.Switched Virtual Call, 2.Switched Virtual Circuit, 3.Switched Virtual Connection
SW	Switch
SXF	AccessCNM Core

T

TI(T1M1)	Committee T1 (patrocinado por ATIS)(reporta desarrollos de interfaces de red)
TI/E1	High capacity networks designed for the digital transmission of voice, data, and video.
TA	Traffic Agent
TAB	TINA Architecture Board
TAPI	Telephony Application Programming Interface (Marca de fábrica de Microsoft)
TARP	Target Address Resolution Protocol
TC	1.Technology Committe, 2. Tandem connection (arquitectura de recurso de red)
TCA	1.Threshold Crossing Alarms 2.Traffic Conditioning Agreement
TCM	Tandem connection manager
TCIF	Telecommunications Industry Forum (patrocinado por ATIS)
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TCO	Total Cost of Ownership
TCon	Terminal connectivity reference point (TINA)
TCSM	Terminal communication session manager (arquitectura de recurso de red)
TDD	Time Division Duplex
TDM	Time Division Multiplexing
TDMA	Time Division Multiple Access
TDS	Testing and diagnostic server
TEl	Terminal Type 1
TE2	Terminal Type 2
TELCO	Telephone Companie
TEWG	Traffic Engineering Working Group
TEX	Transfer Exchange
TFC	Terminal flow connection
TFT	Traffic Flow Template
TFTP	Telnet File Transfer Protocol, (a veces referenciado como trivial)

TIE	Time Interval Error
TIM	Technology Integration Map (TM Forum)
TINA-C	Telecommunication Information Network Architecture-Consortium
TIPHON	Telecommunication and Internet Protocol Harmonization over Networks
TIRKS	Trunk Integrated Record Keeping System
TLI	Transaction Language-1 (de Bellcore)
TLA	Terminal layer adaptor
TLTP	Topological link termination point
TM	Trail manager
TMF	TeleManagement Forum (formalmente Network Management Forum)
TMN	Telecommunications Management Network
TMS	Traffic Management System
TM	Terminal Multiplexer
TOG	The Open Group (grupo formado con X/Open y OSF)
TOM	Telecom Operations Map (TM Forum)
TONICS	TelOps Network Integrated Control System
TP	Transaction Processing
TOSCA	TINA Open Service Creation Architecture (proyecto ACTS)
TSAG	Telecommunication Standarization Advisory Group
TSAPI	Telephony Service Application Programming Interface (Marca de fábrica de Novell)
TSP	Telecom Service Provider
TTA	Trouble Tickering Agent
TTP	Trail Termination Point
TT	Trouble Ticket
TTT	The TINA Trial (demostraciones en gran escala en 1998 de TINA-C)
TV	Television

U

UA	User agent
UAP	User application
UAS	Unavailable seconds
UBR	Unspecified Bit Rate
UDP	User Datagram Protocol
UDP/IP	User Datagram Protocol/Intemet Protocol
UME	UNI Management Entry
UML	Unified Modeling Language (Métodos orientados al Objeto)
UMTS	Universal Mobile Telecom System (Estándard de la próxima generación de móviles)
UNI	User Network Intertace
UPC	User Parameter Control
UPL	Unacceptable Performance Limit
UPSR	Unidirectional Path Switch Ring
URC	1.Uniform Resource Citations, 2.Uniform Resource Characteristics
URL	Uniform Resource Locator
URN	Universal Resource Names/Numbers
USM	User session manager (arquitectura de servicio)
USS	1.User-to-SCP signaling , 2.User service session (arquitectura de servicio)
UTC	Universal Time Coordinated
UTRA	UMTS Terrestrial Radio Access

V

VA	1.Visual Agent, 2.Virtual Address
VAR	Value Added Reseller
VBR	Variable Bit Rate
VC	1.Virtual Circuit, 2.Virtual Connection, 3. Virtual channel

VCC	Virtual Channel Connection
VCI	Virtual Circuit Identifier
VCL	Virtual Channel Link
VCN	Virtual Circuit Number
VDSL	1.Very high-rate Digital Subscriber Line, 2.Virtual Digital Subscriber Line
VITAL	Validation of Integrated Telecommunications Architectures for the Long-term (proyecto ACTS)
VLR	1.Visitor Location Register, 2.Visible Location Requester
VoD	Video On Demand
VoIP	Voice over IP
VP	Virtual Path
VPC	Virtual Path Connection
VPI	Virtual Path Identifier
VPN	Virtual Private Network
VT	Vertical Tab

W

WAN	Wide Area Network
WAP	Wireless Access Protocol
WBEM	Web-Based Enterprise Management
WDM	Wave Division Multiplexing
WFA	Work Force Administration
WM	Workin Memory
WS	Workstation
WSF	Work Station Function (bloque de función lógica ITU-T M3010)
WTSA	World Telecommunications Standarization Access
WWD	World Wide Demo (primera demostración de TINA-C en Telecom'95)
WWW	World Wide Web

X

X	Interfase entre dos TMN's separados según ITU-T
xDSL	X Digital Subscriber Line (donde x es genérico)
XIWT	Cross-Industry Working Team
XMI	XML Interchange
XML	EXtensible Markup Language
XMP	X/Open standards for Management Protocols (API)
XOM	X/Open standards for OSI abstract data Manipulation (API)
X/Open	Asociación X/Open
XOT	X.25 Over TCP/IP

LISTADO DE TABLAS Y FIGURAS

ANTECEDENTES

CAPITULO I.- GESTIÓN DE REDES.

CAPITULO II. PROTOCOLO DE ADMINISTRACIÓN DE RED SIMPLE, SNMP (*SIMPLE NETWORK MANAGEMENT PROTOCOL*)

CAPITULO III. HERRAMIENTA DE GESTIÓN DE REDES VIRTUALES.

CAPITULO IV. PRUEBAS Y RESULTADOS DE LA HERRAMIENTA DE GESTIÓN DE REDES VIRTUALES.

APÉNDICE A. CODIGO FUENTE DE LA HERRAMIENTA DE GESTION DE REDES VIRTUALES

INTRODUCCION

En el campo de las tecnologías de información la tendencia más importante en este momento la constituyen los sistemas distribuidos y las redes de computadoras. Siendo así, la mayoría de las computadoras trabajan conectadas a una red a través de la cual los usuarios pueden acceder a recursos remotos, comunicarse, trabajar en grupo, entre otros. La mayoría de las organizaciones tienden a distribuir sus sistemas informáticos o aplicaciones; a medida que aumenta la criticidad y la importancia de estos sistemas, la complejidad y la inversión realizada en las redes de computadoras crece de forma paralela. Llega entonces el momento en que los sistemas se hacen demasiado complejos para ser administrados y se hacen imprescindibles técnicas y herramientas que permitan llevar a cabo dicha gestión de manera distribuida, esto implica que las redes deben garantizar que los sistemas funcionen y, cuándo no es así, minimizar el tiempo en el que el sistema está parado, esto es, optimizar la confiabilidad y la disponibilidad. Sin embargo, es posible que con la infraestructura de red con la que se cuente no sea suficiente.

En la actualidad, las redes tradicionales están cada vez más congestionadas y sobrecargadas. Para una población de usuarios de red en constante crecimiento, algunos factores adicionales se han combinado para hacer necesaria la expansión de las capacidades de las redes tradicionales:

- El entorno multiusuario-multitarea, presente en los sistemas operativos actuales (Windows, Unix, MacOS y otros) permite transacciones de red simultáneas. Esta capacidad ha dado como resultado una enorme demanda de recursos de red.
- Los sistemas operativos actuales son más rápidos. Los usuarios pueden iniciar varias transacciones de red simultáneas y pueden aumentar sus demandas de recursos de red.

Recientemente, como una solución a esta problemática, se encuentran las **redes virtuales de área local,** VLAN's (Virtual Local Area Network). Una VLAN es una agrupación lógica de dispositivos que no se limita a un segmento físico. Los dispositivos, o los usuarios de la VLAN se pueden agrupar por funciones, departamentos, aplicaciones, protocolos, entre otros, independientemente de la ubicación física de su segmento.

La gestión de red[1] es el conjunto de tareas de monitorización, configuración, información y control, necesarias para operar de manera efectiva una red. Estas tareas pueden estar distribuidas sobre diferentes nodos de la red, lo cual puede requerir repetidas acciones de recogida de datos y su análisis, cada vez que sucede un nuevo evento en la red. Atendiendo a esta definición, la herramienta de gestión de redes virtuales, se ocupa de las tareas de configuración y la monitorización de la VLAN.

La herramienta de gestión de redes virtuales permite a través de diferentes módulos: la monitorización, configuración e informe de la actividad de la red. Estos datos pueden usarse para la detección de problemas (p. ej. los cuellos de botella), determinar los umbrales de operatividad (necesarios para la determinación de los acuerdos de nivel de servicio), y la planificación de la capacidad de la VLAN. Además, se tiene la capacidad de poder configurar o reconfigurar la red virtual con el fin de proporcionar los servicios para la adaptación de los recursos de la red. Por lo que, la herramienta de gestión de redes virtuales consta de los siguientes módulos, cada uno con los detalles y submódulos respectivos:

- Módulo de configuración/Reconfiguración de la red virtual
- Módulo de monitoreo e informe de la disponibilidad y la utilización de la red

Con este propósito la tesis se estructura en los siguientes capítulos:

CAPITULO I. GESTIÓN DE RED.

El capítulo presenta la gestión de redes y las redes de área local Virtuales VLAN (*Virtual Local Area Network*). Con respecto a la gestión de redes se analizan a detalle las dos grandes categorías de división: La monitorización y El control. Así también, las principales funciones de la gestión de red. Con respecto a las VLAN's [23] se incursiona en la definición, configuraciones, ventajas y desventajas. Este capitulo trasciende en los fundamentos y da la pauta para encaminarse al desarrollo de de la Herramienta de Gestión de Redes Virtuales.

CAPITULO II. PROTOCOLO DE ADMINISTRACIÓN DE RED SIMPLE, SNMP (*SIMPLE NETWORK MANAGEMENT PROTOCOL*)

Dentro de un entorno de red gestionado con el protocolo de administración de red simple, SNMP (S*imple Network Management Protocol*) habrá un conjunto de nodos de la red que se encarguen de la gestión y un conjunto de componentes de la red (hosts, concentradores, ruteadores, modems, etc.) que podrán ser gestionados por estas estaciones. El capitulo describe de manera detallada el uso de SNMP y su aplicación e implantación en la herramienta de gestión.

CAPITULO III. HERRAMIENTA DE GESTIÓN DE REDES VIRTUALES.

En el capitulo se proporciona la descripción técnica de la herramienta de gestión de redes virtuales.

CAPITULO IV. PRUEBAS Y RESULTADOS DE LA HERRAMIENTA DE GESTIÓN DE REDES VIRTUALES.

El capitulo presenta las pruebas y los resultados del empleo de la herramienta de gestión de redes virtuales.

Finalmente las conclusiones, la bibliografía y los apéndices referentes a los códigos de los programas.

ANTECEDENTES

Existen herramientas actuales que ofrecen funcionalidades relacionadas a la gestión de red, así como a la posible configuración de las mismas, algunas de éstas son eminentemente propietarias y requieren del uso de licencias para poder emplearlas en los entornos de red y en la práctica profesional, así también, tiene protocolos propietarios que deben de emplearse en sus equipos. A continuación se especifican algunas de las herramientas más representativas y la funcionalidad asociada a las mismas.

MRTG: MULTI ROUTER TRAFFIC GRAPHER

MRTG es una avanzada utilidad gráfica escrita por Tobias Oetiker y Dave Rand para representar gráficamente los datos que los gestores SNMP leen de los agentes SNMP. Produce unas vistosas páginas HTML con gráficos GIF sobre el tráfico entrante y saliente en los interfaces de red prácticamente tiempo real. Con esta herramienta se evita el tener que trabajar directamente con las utilidades CMU-SNMP mediante línea de comandos. Ésta es la herramienta más potente y fácil de utilizar que he encontrado en la Internet.

MRTG utiliza una implementación de SNMP escrita completamente en Perl, por tanto, no es necesario instalar otros paquetes. El programa principal está escrito en "C" para acelerar el proceso de toma de muestras y la generación de imágenes GIF. Los gráficos son generados con la ayuda de la biblioteca GD escrita por Thomas Boutell, autor de la FAQ WWW.

El paquete contiene algunas utilidades para analizar los interfaces de enlace, extraer sus características y generar los ficheros de configuración base, que luego se pueden modificar para adaptarlos a las necesidades concretas.

Otra característica interesante del MRTG es la cantidad de información que produce. Permite cuatro niveles de detalle para cada interface: tráfico en las últimas 24 horas, la última semana, el último mes y un gráfico anual. Esto permite recoger información para realizar estadísticas. Guarda toda esta información en una base de datos utilizando un algoritmo de consolidación que impide que los ficheros crezcan de forma desmesurada.

También genera una página principal que contiene las imágenes GIF de los detalles diarios de cada interface del ruteador, lo que permite hacerse una idea general de qué es lo que está pasando en el ruteador con un sólo vistazo (véase figura 1).

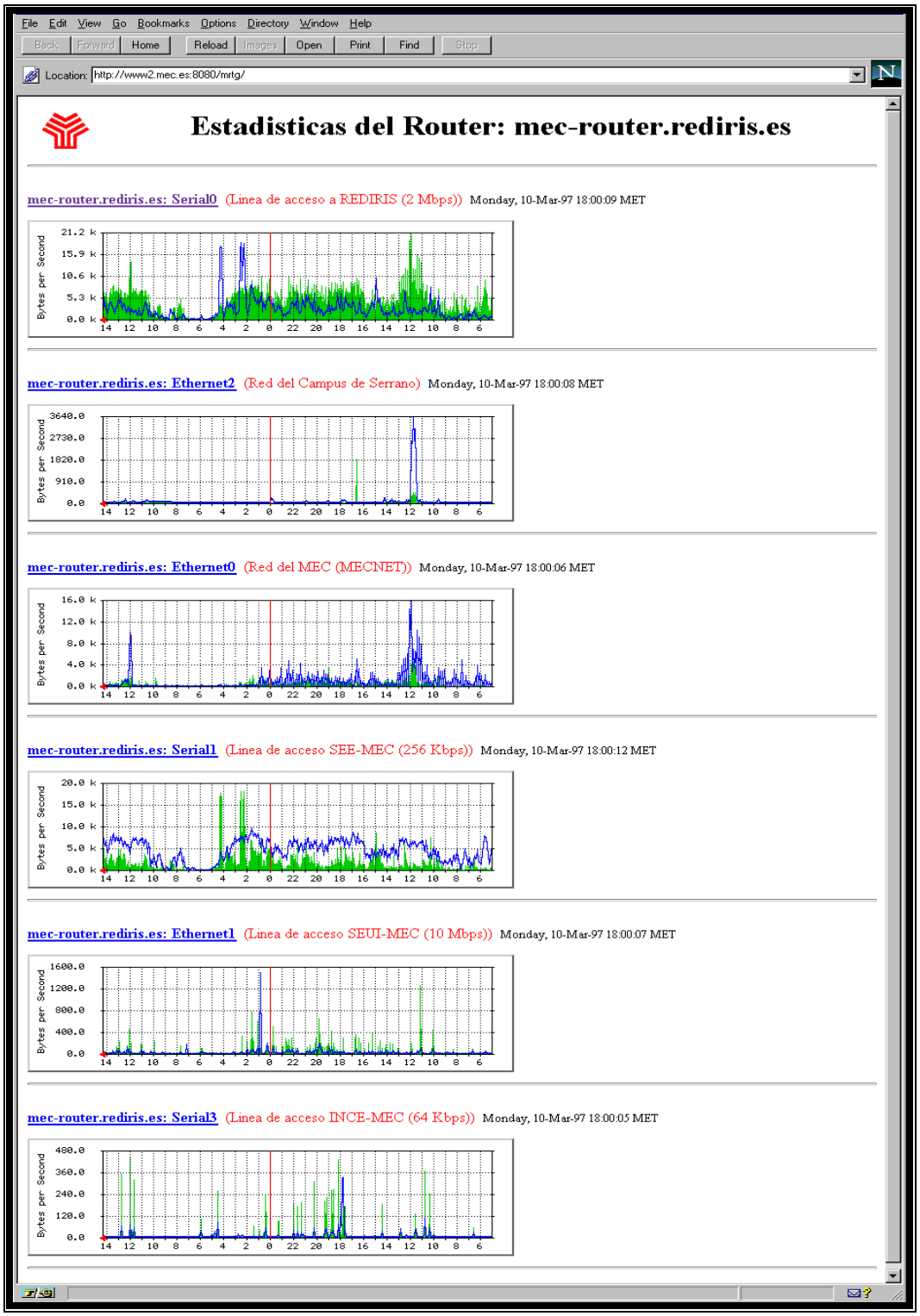


Figura 1.- Entorno de trabajo de MRTG: MULTI ROUTER TRAFFIC GRAPHER

MIB-BROWSER

El programa **MIB-Browser** permite explorar la estructura de una MIB de forma amigable. Además, ofrece la posibilidad de enviar peticiones de lectura o escritura a un agente SNMP(Véase figura 2).

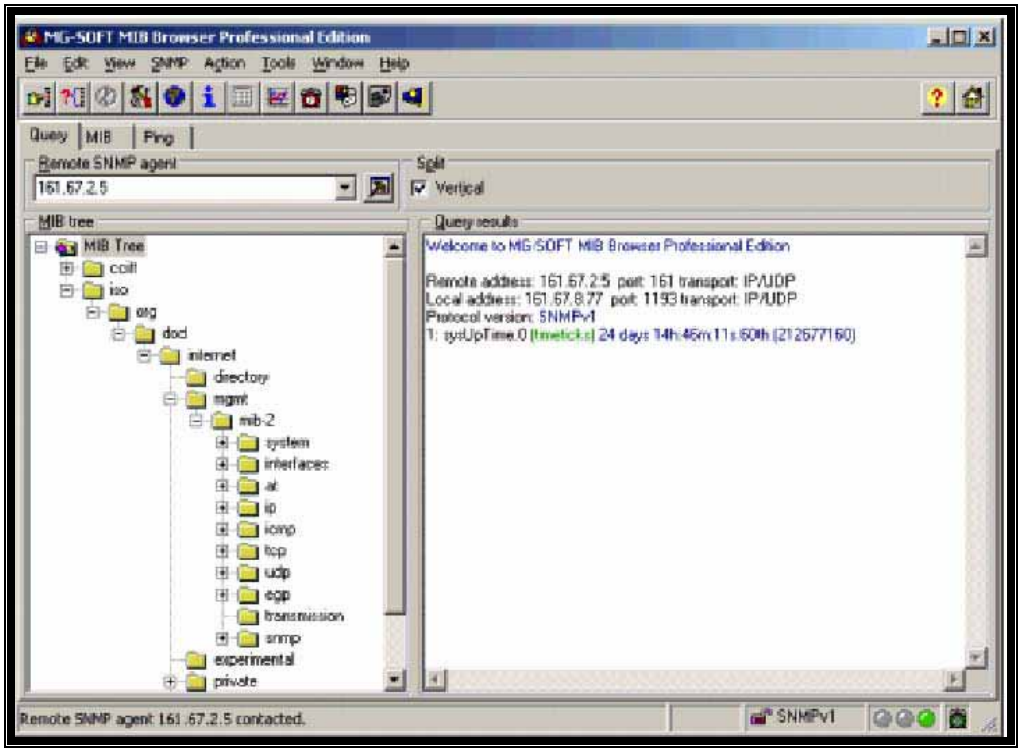


Figura 2.- Explorar la estructura de la MIB con MIB BROWSER

Se pueden apreciar en el área de trabajo, tres carpetas etiquetadas como Query, MIB (*Management Information Base*) y Ping. En la carpeta "Query" se puede recorrer la MIB del agente cuya dirección IP se especifica. En la carpeta "MIB" se pueden seleccionar las MIBs que se van a considerar de entre el conjunto de diversas MIBs que trae el programa. Por último, en la carpeta "Ping", se ofrece la posibilidad de ejecutar un "ping" hacia una cierta máquina, para comprobar si es alcanzable, también, es posible lanzar un "traceroute", para obtener la secuencia de ruteadores en la ruta hacia el destino especificado.

Este es el aspecto de la aplicación cuando se selecciona la carpeta MIB:

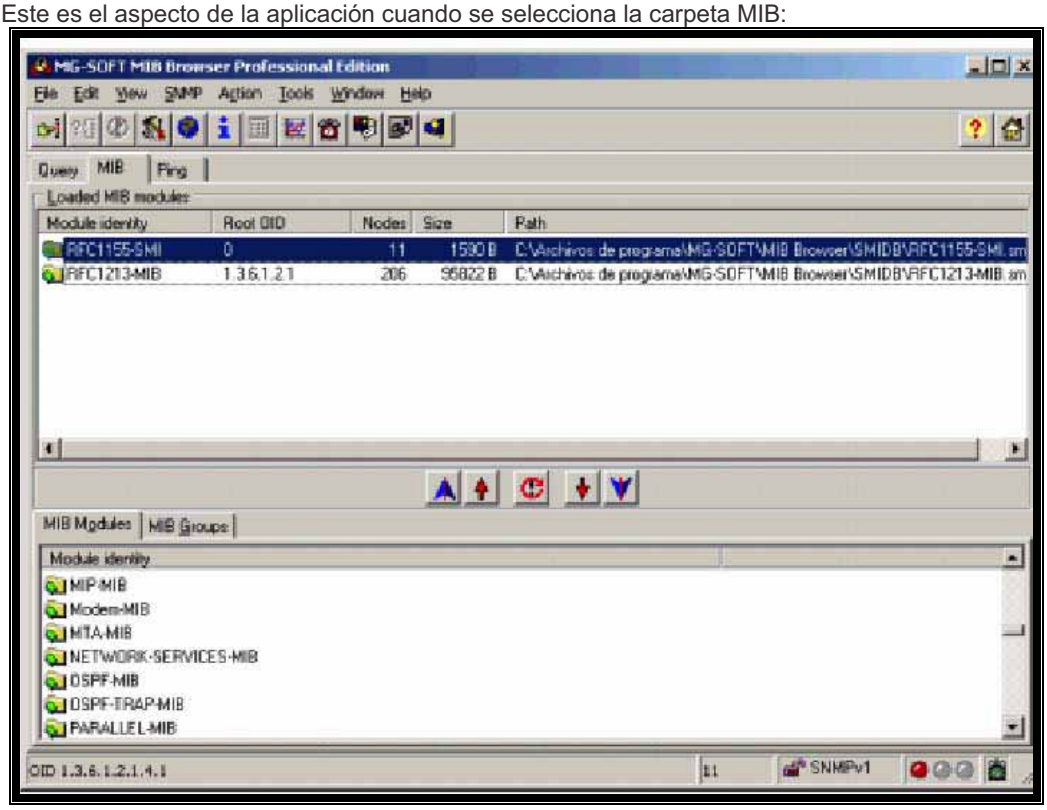


Figura 3.- Posibles MIB's a consultar con MIB BROWSER.

Es posible observar como en la parte inferior aparece un listado de las MIBs disponibles (MIB Modules). También hay una carpeta adicional etiquetada como MIB Groups. En ella, hay grupos de MIBs relacionadas entre sí, por ejemplo, el conjunto de MIBs relacionadas con SNMPv1, con SNMPv2, entre otros. Mediante los botones centrales, se pueden cargar y eliminar módulos de MIB de la aplicación, de manera que cuando se realicen consultas a la MIB de un agente, el programa sólo 'conocerá' los objetos de la MIB pertenecientes a aquellos módulos que tenga cargados en un momento dado. En la carpeta de Query (Consulta) se dispone de varios controles. Sirven para indicar la dirección IP o el nombre del nodo que contiene el agente SNMP remoto, para configurar diversos parámetros relacionados con el protocolo, y para mostrar la información necesaria. En un panel se muestra toda la estructura de la MIB, representada en forma de árbol, donde cada grupo de puede desplegar o contraer, para permitir una navegación rápida y sencilla por todos los objetos de las MIBs cargadas, y en otro panel se muestran los mensajes que se obtienen como resultado de las consultas.

La siguiente figura(véase figura 4), muestra todos estos controles. Se puede observar como la dirección IP del agente es 161.67.17.17, y se haya desplegado el contenido del grupo system de la MIB-2. El botón situado en la parte superior izquierda, que representa una mano con el dedo extendido, sirve para contactar con el agente, pidiéndole un valor del grupo system. En concreto, solicita al agente especificado (en este caso, 161.67.17.17) el valor del objeto SysUpTime. De esta manera, se sabe si el agente SNMP del nodo indicado está activo. En la figura se puede ver la respuesta obtenida al emplear esta función sobre el nodo indicado.

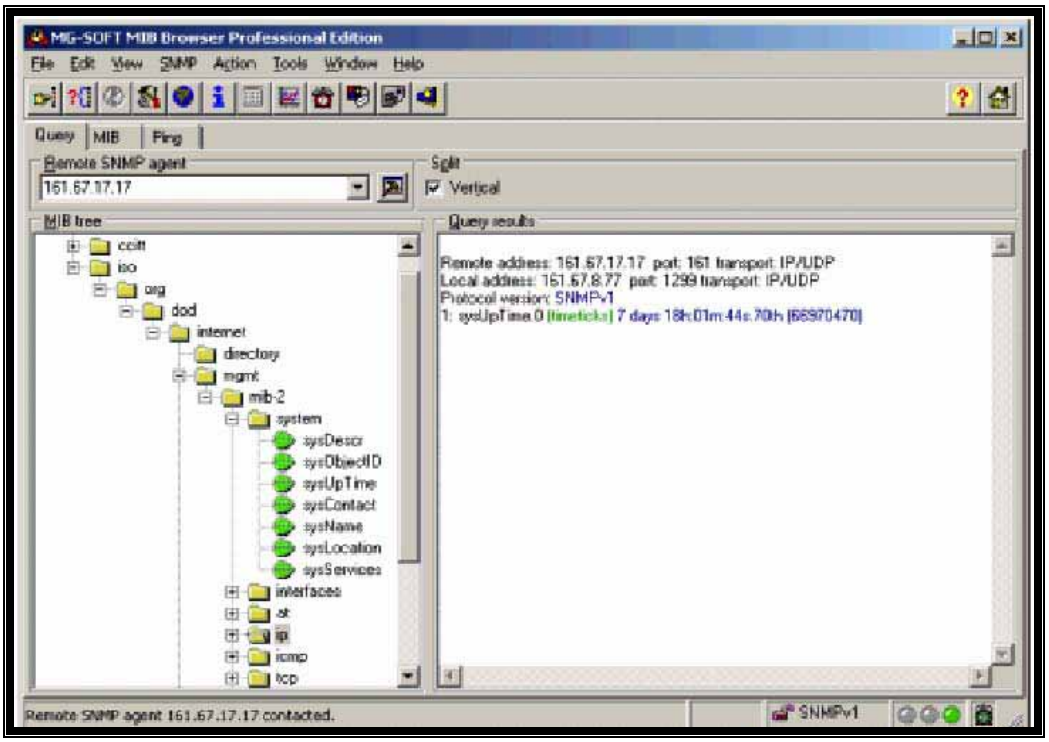


Figura 4.- Controles Asociados a MIB BROWSER

A la derecha de la caja donde se especifica la dirección IP, hay un botón con un martillo ( ) que sirve para configurar los parámetros relacionados con las consultas SNMP. Al pulsarlo se abre la ventana que se muestra en la siguiente figura (véase figura 5).

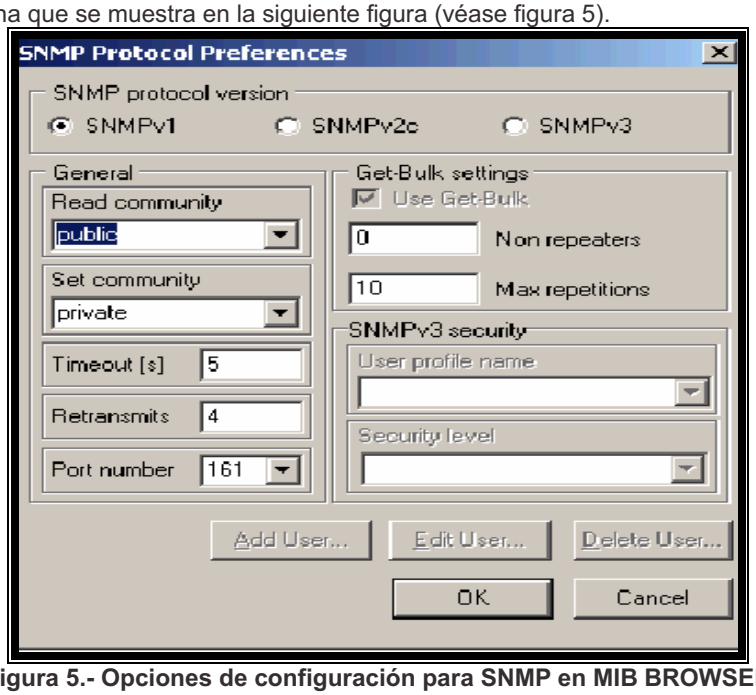


Figura 5.- Opciones de configuración para SNMP en MIB BROWSER

Los parámetros a configurar son:

1.- Versión de SNMP a emplear (al tratarse de una versión de evaluación, sólo admite SNMPv1)

2.- Parámetros generales:

- Nombre de comunidad de lectura: este nombre permite obtener los valores a los objetos de la MIB.
- Nombre de comunidad de modificación: este nombre permite modificar los valores de los objetos de la MIB.
- Temporizador y número de retransmisiones: regulan el comportamiento del protocolo ante la pérdida de paquetes de Get/GetNext/Set.
- Número de puerto: el puerto donde está el agente SNMP (por defecto, es el puerto bien-conocido 161, el agente SNMP implementa el *servidor* del protocolo)

3.- Parámetros relacionados con SNMPv2 (Get-Bulk settings) y SNMPv3. Una vez configurados los datos necesarios para realizar la consulta, navegando en el árbol de la MIB se puede seleccionar un nodo (hojas) o grupo de interés (carpetas). Al pulsar el botón derecho del ratón se abre un menú contextual. La opción Walk permite obtener el valor del objeto o del conjunto de objetos que haya en el grupo seleccionado. La información obtenida se muestra en el panel Query Results. Otra opción interesante es Table View, que muestra el contenido de las tablas (carpetas azules) de una manera más conveniente. Y la opción Properties muestra las propiedades del objeto seleccionado, tal y como aparecen en la descripción textual de la MIB.

OTROS PROGRAMAS

Existe un programa similar llamado Router-Stats, escrito por Lain Lea, el autor del famoso programa lector de correo "tin". Router-Stats actualiza los gráficos una vez al día y muestra información estadística muy interesante sobre la utilización por horas y otros aspectos. El único problema es que se apoya en muchos programas externos. (SMU-SNMP para las tareas con SNMP, GNUPLOT para trazar gráficos, NetPBM y GIFTOOL para trabajar con gráficos).

Hay otra categoría de software que da un paso más allá en la tarea de gestión de redes, ofreciendo una solución completa tanto para monitorizar como para configurar toda la red. Este tipo de solución permite obtener una compleja representación gráfica de la red y ojear fácilmente los nodos que la componen, verificando detalles de configuración específicos y otras cuestiones de interés.

A este nivel podemos hablar de dos soluciones comerciales ampliamente utilizadas: "HP-OpenView" de Hewlett-Packard y "SunNet Manager" de Sun. Estas herramientas ofrecen una plataforma integrada para la gestión de los recursos de red, a través de impresionantes interfaces gráficas. Entre otras utilidades, disponen de herramientas para localizar los nodos de la red en los que se están ejecutándo agentes SNMP. Otra característica importante es la capacidad de integrar productos de otros fabricantes, como el CiscoWork de Cisco (véase figura 6), que permite al administrador mantener una base de datos con todas las configuraciones de los ruteadores e incluso monitorizar gráficamente los paneles traseros de los ruteadores con todas sus conexiones.

Los dos inconvenientes fundamentales de estos productos es que son comerciales y no están disponibles de forma libre, tanto el código fuente como las bibliotecas relacionadas, además de que hay que pagar cierta cantidad de dinero si se requieren funciones adicionales agregándose las licencias implicadas por distribución.

Existen otras soluciones disponibles con una funcionalidad más o menos similar. Uno de los paquetes es el Scotty. Scotty es un paquete basado en TCL que permite crear programas específicos a las necesidades de la red propia, empleando un API de alto nivel de cadenas de caracteres. Un paquete similar es el Tkined. Es un editor de red que ofrece extensiones para crear un entorno de trabajo completo, integrando algunas herramientas para localizar redes IP, soporte para el proceso de instalación de la red o resolución de problemas en redes IP utilizando SNMP en combinación con otras utilidades estándar (por ejemplo traceroute). Scotty también incluye un visor gráfico MIB que permite explorar fácilmente información MIB.

EL SOFTWARE LLAMADO CLUSTER MANAGEMENT SUITE.

Este producto permite configurar y mantener redes virtuales de área local(VLAN's) localmente(véase figura 6); muestra una pantalla en dónde es posible realizar tareas tales como: configurar puertos de acceso estático y Multi-VLAN y configurar dominios de acceso con el VIRTUAL TRUNK PROTOCOL(VTP).

Sin embargo, para funcionar este software necesita dominios de trabajo basados en el protocolo VTP (VIRTUAL TRUNK PROTOCOL), por lo que su inconveniente principal está en la necesidad de crear obligatoriamente CLUSTERS (CÚMULOS), además de no proporciona posibilidades de monitoreo de la VLAN.

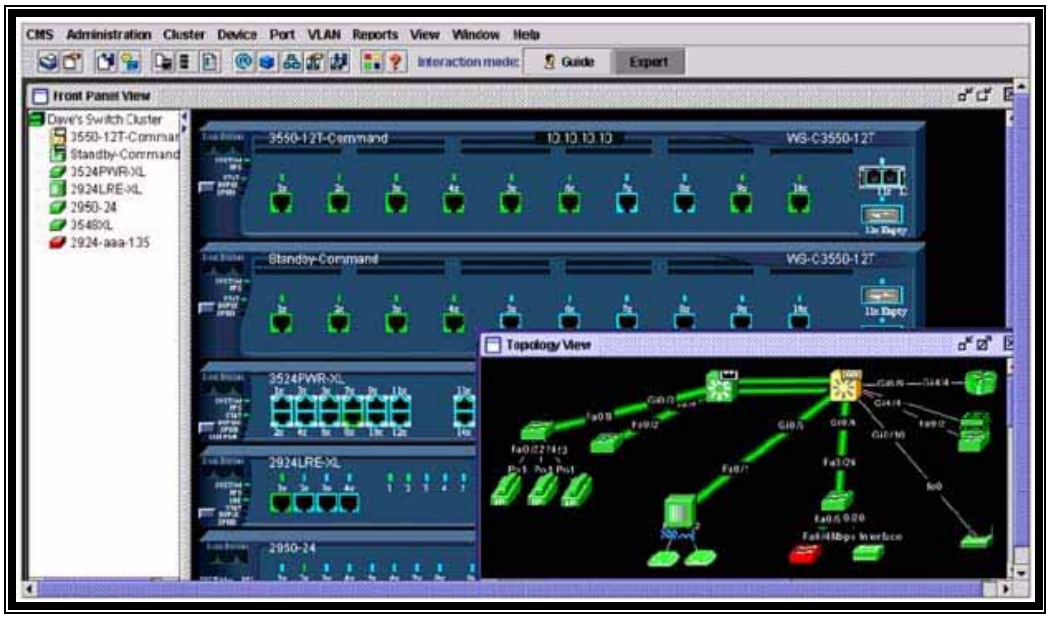


Figura 6.- El software CLUSTER MANAGEMENT SUITE.

LA CLI(COMMAND LINE INTERFACE).

Cada proveedor de VLAN's proporciona algún método para su configuración, basado en el IOS (Sistema Operativo) que se accesa a través de la línea de comandos. Con éste método es posible realizar tareas locales, tales como: configuración de puertos de acceso estático y Multi-VLAN; configurar dominios de acceso con VIRTUAL TRUNK PROTOCOL(VTP); añadir y modificar VLAN's..etc.

ETHEREAL

Ethereal es una aplicación que ofrece una interfaz sencilla de utilizar y permite visualizar los contenidos de las cabeceras de los protocolos involucrados en una comunicación de una forma muy cómoda.

INTERFAZ Y MENÚS

Ethereal funciona en modo gráfico y está programado con la librería de controles GTK. La ventana principal de la aplicación se divide en tres partes de visualización y una zona inferior de trabajo con filtros (véase figura 7).

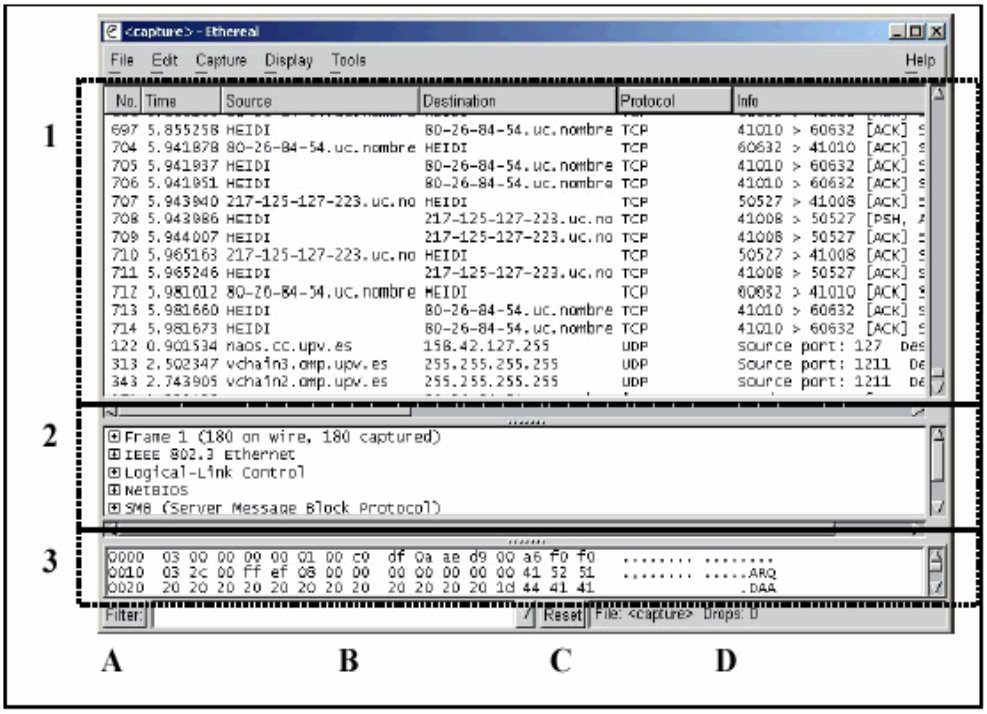


Figura 7.- Interfaz Gráfica de Ethereal
Corregir grafica

En la primera parte (1) se muestra la información más relevante de los paquetes capturados, como, por ejemplo, las direcciones IP y puertos involucrados en la comunicación. Seleccionando un paquete en esta sección es posible obtener información detallada sobre él, en las otras dos secciones de la pantalla que se comentan a continuación.

En la parte central de la ventana (2) se muestra, utilizando los controles tree-view, cada uno de los campos de cada una de las cabeceras de los protocolos que ha utilizado el paquete para moverse de una máquina a la otra. Así, si se han capturado una serie de paquetes de, por ejemplo, una conexión telnet, se pueden ver las cabeceras del protocolo TCP, del IP y de los que tengamos debajo de ellos (trama Ethernet, por ejemplo, en una red Ethernet).

La tercera parte de la ventana (3) se muestra un volcado hexadecimal del contenido del paquete. Seleccionando cualquier campo en la parte central de la ventana se muestran en negrita los datos correspondientes del volcado hexadecimal y los datos reales que están viajando por la red.

En la barra inferior, aparecen cuatro componentes muy interesantes a la hora de hacer análisis de capturas: Creación de filtros (A), filtro actual (B), borrar filtro (C) y mensajes adicionales (D). Para obtener un mayor detalle de estos menús, se debe consultar la guía del usuario en la página Web de la aplicación.

Todas las opciones que pueden ser empleadas, son accesibles por medio de los menús, la aplicación contiene los siguientes menús(véase figura 8).

Figura 8.- Menús de Ethereal.

• File: Menú con los ítems para abrir, guardar archivos de captura. Permite imprimir y salir de la aplicación.

• Edit: Menú para encontrar tramas concretas, ir a una trama y marcar tramas. También tiene las opciones de preferencias, de captura y visualización de filtros y protocolos.

• Capture: Inicia o detiene la captura de paquetes.

• Display: Permite cambiar las opciones de visualización, correspondencia y coloreado de marcos.

• Tools: Este menú contiene los pluggins instalados, seguimiento de un paquete TCP (interesante función), sumario de los paquetes capturados y la visualización de las estadísticas de protocolos empelados.

• Ayuda. Ayuda de la aplicación y “acerca de”.

CAPTURA DE PAQUETES

La captura de paquetes se realiza por medio de la opción de menú capture → start. Al seleccionar la opción aparecerá la caja de diálogo con las preferencias para la captura de los paquetes (véase figura 9).

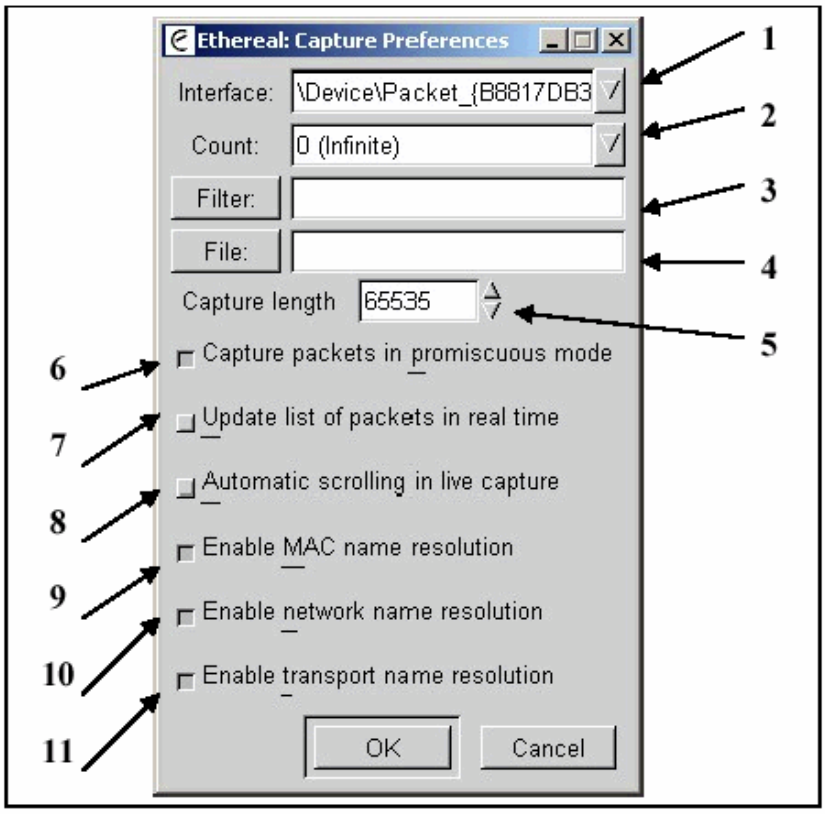


Figura 9.- Ventana de preferencias de Captura.

Las opciones que permite esta ventana son las siguientes:

1. **Interface**. Es el interface de captura que se va a emplear para la sesión, en algunos sistemas es equivalente a las conexiones de red de las que se dispone (PPP, Red, etc…). Si al seleccionar una opción no se capturan datos, se debe a que no se ha seleccionado la interfaz correcta.

2. **Cuenta**. Es el número de paquetes que se desean capturar. En el caso de dejarse a 0, se capturarán todos los paquetes hasta que se detenga manualmente la sesión (o se sature el sistema).

3. **Filtro**. Determina el filtro que se aplica a la captura.

4. **Archivo**. En caso de que se desee que la captura sea volcada hacia un archivo, se puede seleccionar por medio de este cuadro de texto.

5. **Longitud**. Marca la cantidad máxima de bytes de cada trama para que sean capturados.

6. **Captura de datos en modo promiscuo**. En caso de seleccionarse esta opción, se capturarán todos los datos posibles que sean alcanzables por el host monitor. Si no se selecciona, sólo se capturarán los paquetes que entren o salgan al host monitor.

7. **Actualización de la lista en tiempo rea**l. Actualiza la ventana de paquetes capturados a la vez que éstos son capturados. Esta opción tiene el problema de cargar en exceso el sistema.

8. **Desplazamiento de la lista de paquetes capturado**s. Permite que la ventana con el contenido de los paquetes capturados, se actualice en tiempo real.

9. **Permitir resolución de nombres MAC (*Media Access Control*)**. Este botón permite controlar si Ethereal traduce o no los primeros tres bytes de las direcciones MAC al nombre del fabricante a quien ese prefijo ha sido asignado por el IETF.

10. **Permite resolución del nombre de la red**. Este botón permite que controlar si Ethereal traduce o la direcciones IP a nombres del dominio del DNS. Haciendo click en este botón, la lista de paquetes capturados tendrá información más útil, pero provocará las correspondientes peticiones de operaciones de búsqueda, que pueden influir en la captura.

11. **Permitir la resolución de nombres de transporte**. Este botón permite controlar si Ethereal traduce o no los números de puerto del protocolo. Una vez configurada la ventana con las opciones que se desean, al presionar el botón de OK, comenzará la captura de paquetes, si se presiona CANCEL, no se capturará ningún paquete y se regresará al estado anterior. Durante la captura, aparecerá la caja de diálogo que muestra la evolución (véase figura 10).

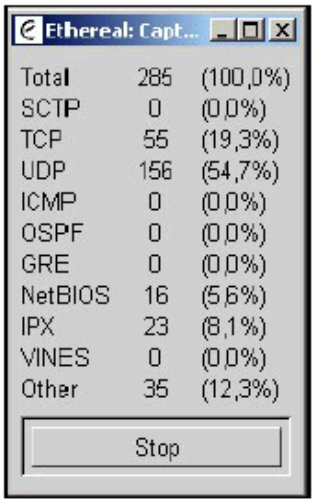


Figura 10.- Evolución de la captura.

JUSTIFICACION

En el mundo actual, en el que la informática gira en torno al concepto de red, el trabajo de los administradores de sistemas es muy complejo. Su misión consiste en mantener en funcionamiento recursos tales como ruteadores, concentradores, servidores, así como, cada dispositivo crítico que conforma la red.

Hay gran cantidad de motivos por los cuales un administrador necesita monitorizar entre otros: la utilización del ancho de banda, el estado de funcionamiento de los enlaces, la detección de cuellos de botella, detectar y solventar problemas con el cableado, administrar la información de encaminamiento entre máquinas, etc. La monitorización de la red es también un buen punto desde el que comenzar el estudio de los problemas de seguridad.

En muchos casos, la red de una organización está enlazada mediante costosos enlaces a redes de área extensa (WAN) o con la Internet, y cuyos costes dependen del volumen de tráfico. Es muy importante mantener un registro estadístico del tráfico que circula por estos enlaces. La tarificación de este tipo de líneas se realiza en función del número de paquetes enviados y recibidos. A este respecto se emite la pregunta: ¿ **QUÉ ES SNMP** ?.

La respuesta a todas las necesidades antes expuestas, es el protocolo llamado protocolo de administración de red simple, SNMP (*Simple Network Management Protocol*), diseñado en los años 80, su principal objetivo fue el integrar la gestión de diferentes tipos de redes mediante un diseño sencillo y que produjera poca sobrecarga en la red.

SNMP opera en el nivel de aplicación, utilizando el protocolo de transporte UDP, por lo que ignora los aspectos específicos del hardware sobre el que funciona. La gestión se lleva a cabo al nivel de IP, por lo que se pueden controlar dispositivos que estén conectados en cualquier red accesible desde la Internet, y no únicamente aquellos localizados en la propia red local. Evidentemente, si alguno de los dispositivos de encaminamiento con el dispositivo remoto a controlar no funciona correctamente, no será posible su monitorización ni reconfiguración.

El protocolo SNMP está compuesto por dos elementos: el agente (agent), y el gestor (manager). Es una arquitectura cliente-servidor, en la cual el agente desempeña el papel de servidor y el gestor hace el de cliente.

El agente es un programa que ha de ejecutase en cada nodo de red que se desea gestionar o monitorizar. Ofrece un interfaz de todos los elementos que se pueden configurar. La Base de información de administración, MIB (*Management Information Base*), representa la parte del servidor, en la medida que tiene la información que se desea gestionar y espera comandos por parte del cliente.

El gestor es el software que se ejecuta en la estación encargada de monitorizar la red, y su tarea consiste en consultar los diferentes agentes que se encuentran en los nodos de la red los datos que estos han ido obteniendo.

Hay un comando especial en SNMP, llamado trap, que permite a un agente enviar datos que no han sido solicitados de forma explícita al gestor, para informar de eventos tales como: errores, fallos en la alimentación eléctrica, etc.

En esencia, el SNMP es un protocolo muy sencillo puesto que todas las operaciones se realizan bajo el paradigma de carga-y-almacenamiento (load-and-store), lo que permite un juego de comandos reducido. Un gestor puede realizar sólo dos tipos diferentes de operaciones sobre un agente: leer o escribir un valor de una variable en el MIB del agente. Estas dos operaciones se conocen como petición-de-lectura (get-request) y petición-de-escritura (set-request).

Hay un comando para responder a una petición-de-lectura llamado respuesta-de-lectura (get-response), que es utilizado únicamente por el agente.

La posibilidad de ampliación del protocolo está directamente relacionado con la capacidad del MIB de almacenar nuevos elementos. Si un fabricante quiere añadir un nuevo comando a un dispositivo, como puede ser un ruteador, tan sólo tiene que añadir las variables correspondientes a su base de datos (MIB).

Casi todos los fabricantes implementan versiones agente de SNMP en sus dispositivos: ruteadores, concentradores, sistemas operativos etc.

Existen herramientas de uso comercial que empiezan a profundizar en este nuevo tópico, La VLAN, sin embargo es necesario pagar en primera instancia el costo del software, y en segunda instancia la distribuciones ya que por lo regular es para un sólo equipo. Así también, el código no es libre de distribución, por lo que para cada nueva funcionalidad hay que pagar tanto el soporte como la actualización de la herramienta.

Si consideramos ambos elementos como determinantes en los nuevos escenarios de gestión de redes (SNMP y VLAN's), aunado a la compra de los productos propietarios de los diversos proveedores y a la dependencia que esto conlleva, se justifica:

1.-El Desarrollo de una Herramienta de Gestión de Redes Virtuales independiente de los protocolos propietarios, mecanismos de etiquetado, mecanismos de comunicación, soporte y pago de licencias a los proveedores de servicios de VLAN.

2.- El proporcionar una herramienta de código libre y compilable, capaz de integrar nuevos componentes y funcionalidades, sin afectar a los ya existentes.

3.- Incursionar en la gestión de redes y sus funciones, así como en el análisis de protocolos a detalle; con el fin de formar e integrar una plataforma de conocimiento sólida y sustentable para el desarrollo computacional en este campo.

OBJETIVOS

Los objetivos son los elementos a cumplir durante el desarrollo del trabajo de tesis, para el caso que nos competen son:

OBJETIVO GENERAL.

Desarrollar una herramienta de gestión que permita la configuración y monitorización de una red virtual de área local.

Para lo anterior se hace necesario considerar los siguientes objetivos específicos

OBJETIVOS PARTICULARES.

1.-Crear una VLAN experimental, de tamaño reducido, para su gestión vía la herramienta de gestión que se pretende desarrollar.

2.- Diseñar y desarrollar una herramienta para gestionar redes virtuales

2.1.- Crear un programa de monitoreo que permita observar las disponibilidades y el uso de la red virtual.

2.2.- Crear un programa que permita la configuración-reconfiguración, el control y la consulta de información de la red virtual.

3.- Implantar la herramienta de gestión en la red virtual (VLAN) experimental para demostrar su funcionamiento.

CAPITULO I
LA GESTIÓN DE REDES

INTRODUCCIÓN

El capítulo presenta la gestión de redes y las redes de área local Virtuales VLAN (*Virtual Local Area Network*). Con respecto a la gestión de redes se analizan a detalle las dos grandes categorías de división: La monitorización y el control. Así también, las principales funciones de la gestión de red. Con respecto a las VLAN's se incursiona en la definición, configuraciones, ventajas y desventajas. Este capitulo trasciende en los fundamentos y da la pauta para encaminarse al desarrollo de de la Herramienta de Gestión de Redes Virtuales.

1.- LA GESTIÓN DE REDES.

La gestión de red [1, 4] es el conjunto de tareas de monitorización, configuración, información y control, necesarias para operar de manera efectiva una red. Estas tareas pueden estar distribuidas sobre diferentes nodos de la red, lo cual puede requerir repetidas acciones de recogida de datos y su análisis, cada vez que sucede un nuevo evento en la red.

Las redes son cada vez más importantes en empresas y organizaciones tomando en consideración:

- La Tendencia a redes más grandes, más complejas, más heterogéneas,...
- La red y las aplicaciones distribuidas se hacen imprescindibles.
- Los costos de gestión de la red aumentan.
- La gestión de la red no se puede hacer a mano:
- Se requieren herramientas de gestión de red automatizadas.
- Herramientas estándar que funcionen sobre equipos de distintos tipos y distintos fabricantes: Estaciones de usuario, conmutadores, ruteadores, equipos de telecomunicación.

Las funciones de gestión de red se pueden agrupar en dos grandes categorías [1, 3, 4]:

- **Monitorización:** Funciones de "lectura".
    - Observar y analizar el estado y comportamiento de la
    - configuración de red y sus componentes.
    - Abarca: prestaciones, fallos y costes.
- **Control:** Funciones de "escritura".
    - Alterar parámetros de los componentes de la red
    - Abarca: configuración y seguridad.

Todas las áreas funcionales abarcan monitorización y control, y cumplen tres objetivos fundamentales:

- **Identificación de la información**: identificar la información a monitorizar.
- **Diseño de mecanismos de monitorizació**n: como obtener esa información.
- **Utilización de la información**: para qué utilizar la información obtenida dentro de las distintas áreas funcionales de gestión de red.

La información a monitorizar puede encontrase en una de las siguientes categorías:

1.- Estática: Caracteriza la configuración actual de la red y de sus elementos (p.e. El número de puertos de un router). Esta información cambiará con muy poca frecuencia.

2.- Dinámica: Información relacionada con eventos en la red (p.e. La transmisión de un paquete por la red)

3.- Estadística: Información que puede ser derivada de la información dinámica (p.e. El número medio de paquetes transmitidos por unidad de tiempo por un sistema final).

La información Estática: Es generada y almacenada por el propio elemento de red (p.e. un ruteador almacena su propia configuración).

La información Dinámica: Puede almacenarla el propio elemento, u otro encargado de ello (p.e. En una LAN cada elemento puede almacenar el número total de paquetes que envía, o un elemento de la LAN puede estar escuchando y recoger esa información (se denomina **Monitor remoto).** Un monitor remoto no puede recoger cierta información propia de un elemento (p.e. El número de sesiones abiertas).

La información Estadística: Puede ser generada por cualquier elemento que tenga acceso a la información dinámica en base a dos opciones básicas:

- Puede enviarse toda la información dinámica al gestor de red para que realice las estadísticas.
- Si el gestor no necesita toda la información, ésta puede ser resumida por el propio elemento antes de enviarla al gestor, ahorrando procesamiento en el gestor y generando menos tráfico en la red.

1.1. AREAS DE LA GESTIÓN DE REDES.

La gestión de redes incursiona en distintos campos cada uno con sus propias peculiaridades, estos campos son [1,4]:

- Gestión de fallos
- Gestión de contabilidad
- Gestión de la configuración
- Gestión de prestaciones
- Gestión de seguridad

1.1.1.- GESTIÓN DE FALLOS

La gestión de fallos tiene una gran componente “manual”. Una vez localizado el fallo, los procedimientos para solucionarlo suelen realizarse por el administrador (desplazarse hasta el lugar donde está el equipo, sustituir todo o parte, etc.)

La gestión de fallos tiene que ver con situaciones referentes a:

- Detección, aislamiento y corrección de fallos
- Cuando ocurre un fallo, tan pronto como sea posible:
  - Determinar exactamente **dónde** está el fallo.
  - **Aislar** el resto de la red, para que pueda seguir operando sin interferencia.
- **Reconfigurar** la red para minimizar el impacto de operar sin el componente averiado.
- **Reparar o reemplazar** el componente averiado para devolver la red al estado inicial

- Distinción entre fallo y error:
  - **Error**: Un solo evento (ejemplo, se pierde un paquete)
  - **Fallo**: Un funcionamiento anormal que requiere una intervención para ser subsanada
  - El fallo se manifiesta por un funcionamiento incorrecto (se corta la línea física) o por exceso de errores (en una línea se pierden muchos paquetes)

En un entorno complejo puede ser difícil diagnosticar los fallos. Principalmente debido a:

1.- Múltiples causas potenciales: Cuando hay múltiples tecnologías, los posibles puntos de fallo y tipos de fallo aumenta (p.e. En la siguiente figura si falla o tiene muchos errores la comunicación entre el cliente y el servidor puede ser debido a muchas causas).

2.- Demasiadas observaciones relacionadas: Un solo fallo puede afectar a muchos elementos, generando mucha información de fallos que obscurecerá la causa real (p.e. En la figura 1.1 el fallo de la línea de 2 Mb. Afectará a todas las comunicaciones entre los equipos Token-ring y Ethernet, así como a las comunicaciones telefónicas, o un fallo en un nivel de las comunicaciones puede causar degradación o fallos en todos los niveles superiores: un fallo en la línea será detectado como un fallo de enlace por el router y de transporte y aplicación en los equipos).

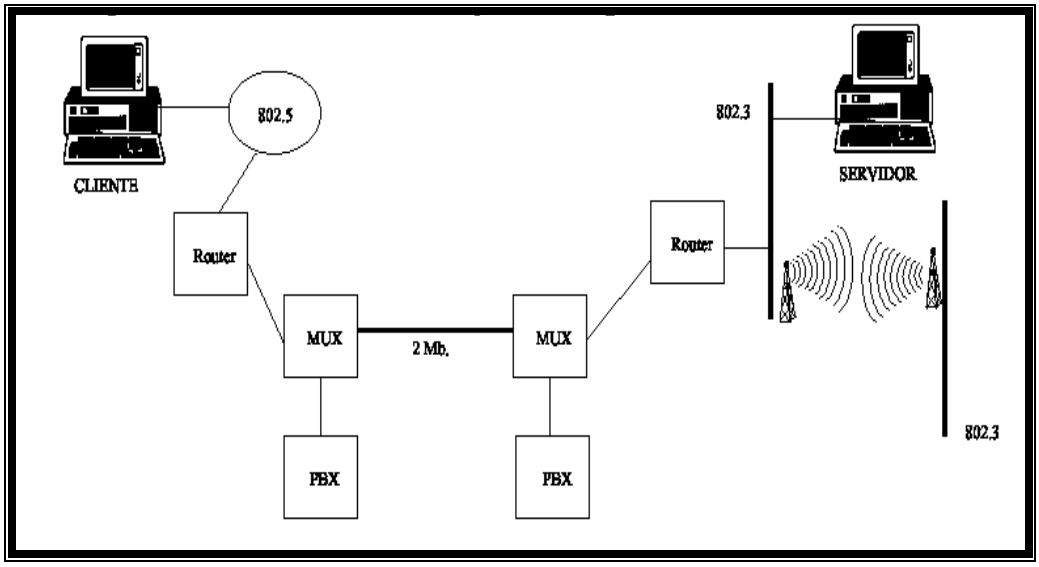


Figura 1.1.- Gestión de Fallos en una red.

- **Interferencia entre procedimientos de diagnóstico y de recuperación local**: Procedimientos de recuperación local pueden destruir evidencias importantes de la naturaleza del fallo, no permitiendo el diagnóstico.

- **Ausencia de herramientas de prueba automática**s: Las pruebas para aislar fallos son difíciles y costosas para el administrador.

1.1.2.- GESTIÓN DE CONTABILIDAD

Consiste en recoger información sobre el uso de los recursos por los usuarios. La información que se recoge depende mucho de las necesidades de la corporación:

- En una red interna puede ser bastante general para saber cuanto usan un recurso cada uno de los departamentos, y por tanto que porcentaje del costo que les corresponde.

- En otros casos (p.e. Sistemas que ofrecen servicios públicos) el uso debe detallarse por cuenta, proyecto, o individualmente con el objeto de cobrar por el uso de los recursos. En este caso la información debe ser más detallada y exacta que en el ejemplo anterior.

Los recursos que son objeto de gestión de costos suelen ser:

- **Recursos de comunicación**: LANs, WANs, líneas alquiladas, líneas conmutadas alquiladas, conmutadores PBXs, etc.

Hardware de computación: Servidores, estaciones de trabajo,...

Sistemas y software: Utilidades software y aplicaciones en servidores, un centro de proceso de datos.

Servicios: Todos los servicios de información y servicios de comunicaciones comerciales disponibles.

La información recogida y almacenada depende de las necesidades de la organización. Ejemplos típicos de información por usuario puede ser:

- **Identificación de usuario**: proporcionada por el generador de una transacción o petición de servicio.
- **Receptor**: identifica el recurso de red utilizado.
- **Número de paquetes**: cantidad de datos transmitidos.
- **Nivel de seguridad**: Identificación de las prioridades de la transmisión y el procesamiento.
- **Sellos temporales**: asociados con cada evento de transmisión o procesamiento (p.e. Comienzo y final de una transacción).
- **Códigos de estado de la red**: Indican la naturaleza de errores o mala utilización detectados.
- **Recursos utilizados**: Recursos involucrados en una transacción o evento de servicio.

1.1.3.- GESTIÓN DE CONFIGURACIÓN.

Se ocupa de la inicialización, mantenimiento y apagado de componentes individuales y subsistemas lógicos de la red. Los componentes son tanto recursos físicos (servidores, ruteadores, etc.) como objetos lógicos de bajo nivel (p.e. Un temporizador de retransmisión de un protocolo de transporte).

La gestión de configuración puede indicar el proceso de inicialización identificando y especificando las características del los componentes y recursos que constituyen “la red”. También se especifican valores iniciales o por defecto para los diferentes atributos, de forma que los recursos gestionados comiencen a operar en los estados deseados, tengan los valores de atributos deseados y las relaciones adecuadas con otros componentes de la red.

Durante el funcionamiento de la red, la gestión de configuración es responsable de realizar cambios en respuesta a comandos del usuario o en respuesta a otras funciones de gestión de red (p.e., si una función de gestión de prestaciones detecta que el tiempo de respuesta se está degradando dado a un mal balance de carga, la gestión de configuración puede hacer ajustes para conseguir el nivel de carga adecuado, o si la gestión de fallos detecta y aísla un fallo, la

gestión de configuración puede alterar la configuración para recuperar el tráfico por un camino alternativo).

Las funciones asociadas a la configuración se relacionan a:

Definición de la información de configuración.
Establecimiento y modificación de valores de atributos.
Establecimiento y modificación de relaciones.
Operaciones de inicialización y apagado de la red.

- Distribución de software.

Examen de valores y relaciones.
Informes sobre el estado de la configuración.

Las dos últimas son funciones de monitorización. Normalmente se examinan mediante sondeo y los cambios se notifican mediante informe de eventos.

1.1.3.1.- DEFINICIÓN DE LA INFORMACIÓN DE CONFIGURACIÓN.

La información de configuración describe la naturaleza y estado de los recursos que forman la red (tanto físicos como lógicos). Esta información incluye una especificación del recurso y de los atributos de ese recurso (p.e., nombre, dirección, número de identificación, estados, características operacionales, versión del software). Esta información (en realidad, toda la información de gestión) se puede estructurar de diversas formas:

- **Una sencilla lista estructurada de campos de datos**: cada uno almacenando un valor simple. Un ejemplo es SNMP.
- **Una base de datos orientada a objetos**: cada elemento de la red es representado por uno o varios objetos. Cada objeto contiene los atributos cuyos valores reflejan las características del elemento. Un objeto puede tener *comportamientos* asociados, tales como notificaciones ante la ocurrencia de ciertos eventos. También se puede definir herencia de atributos. Un ejemplo es el modelo OSI.
- **Una base de datos relacional**: cada entrada en la base de datos contiene valores que reflejan las características de un elemento de la red. La estructura de la base de datos refleja las relaciones entre los elementos de la red. Un ejemplo es Sun Net Manager (utiliza Sybase).

Aunque la información es accesible por el **gestor**, normalmente se almacena próxima al recurso en cuestión (en su **agente** o en un **agente proxy** si el recurso no soporta directamente un agente). Esta función de control de red debe permitir al administrador:

- Especificar el rango y el tipo de valores que puede tomar un determinado atributo de un agente.
- El rango puede ser una lista de todos los valores posibles o los valores superior e inferior permitidos.
- Definir nuevos tipos de objetos o tipos de datos, dependiendo del tipo de la base de datos. Lo ideal sería poder definir estos objetos *on-line, en línea,* y que se crearan en los correspondientes **agentes** y **proxies**. Sin embargo, en todos los sistemas existentes hoy en día esta función se realiza *off-line, fuera de línea,* como parte del proceso de configuración del elemento de la red, en vez ser posible hacerlo dinámicamente.

1.1.3.2.- ESTABLECIMIENTO Y MODIFICACIÓN DE VALORES.

Esta función debe permitir al **gestor** establecer y modificar de manera remota atributos de los **agentes** y **proxies**, con dos limitaciones:

- Un administrador debe ser autorizado a realizar la modificación de un atributo en particular en un **agente** concreto en un momento concreto (esta es una función de gestión de seguridad).
- Algunos atributos reflejan la "realidad" de un recurso y no tiene sentido que se pueda modificar (p.e., el número de puertos de un ruteador).

La modificación de un atributo, obviamente, modifica la información de configuración de un **agente**. Estas modificaciones, en general, se pueden clasificar en tres categorías:

1.- Actualización de la base de datos: Se cambia un valor en la base de datos del **agente** sin que este cambio afecte en nada más a dicho **agente** (p.e., cambio de la información de contacto: nombre y dirección de la persona responsable de ese recurso). El **agente** responde a un comando de este tipo actualizando la base de datos y enviando una confirmación al **gesto**r.

2.- Actualización de la base de datos que implica modificación del recurso: Además de cambiar un valor en la base de datos del **agente** la modificación tiene un efecto sobre el recurso (p.e., si el atributo de estado de un puerto físico se modifica a disabled, el agente además de modificar la base de datos, deshabilita el puerto, de tal forma que éste no seguirá activo).

3.- Actualización de la base de datos que implica una acción: En algunos sistemas de gestión el **gestor** no dispone de "comandos de acción" directos. En vez de estos comandos, disponen de determinados atributos en la base de datos de los **agentes** que cuando se establecen, hacen que el **agente** inicie cierta acción (p.e., un *ruteador*, puede tener un parámetro de reinicialización en su base de datos. Cuando un **gestor** autorizado pone a ***TRUE*** ese parámetro el equipo se reinicializará y pondrá el valor a ***FALS*E**).

1.1.3.3.- ESTABLECIMIENTO Y MODIFICACIÓN DE RELACIONES.

Una relación describe una asociación, conexión o condición que existe entre recurso o componentes de la red: topologías, jerarquías, conexiones físicas o lógicas, dominios de gestión (conjunto de recursos que comparten un conjunto común de atributos o un conjunto de recursos que comparten la misma autoridad de gestión). La gestión de configuración debe permitir la modificación, adicción o eliminación de relaciones on-line sin deshabilitar toda o parte de la red. Ejemplos de este tipo de operaciones pueden ser el permitir al administrador "cortar" una conexión entre dos nodos, o designar una dirección alternativa o de *backup* para utilizar en caso de que el destino primario de una petición de conexión no responda.

1.1.3.4.- INICIALIZACIÓN Y APAGADO DE LA RED.

La gestión de configuración debe incluir mecanismos que permitan al administrador inicializar o apagar la red o una subred. La inicialización incluye verificar que todos los atributos y relaciones se han establecido adecuadamente, notificar al administrador que algún recurso, atributo o relación necesita ser establecido y validar los comandos de inicialización del administrador. El apagado de la red, requiere disponer de mecanismos para solicitar determinadas estadísticas o información de estado antes de que finalice la operación de apagado.

1.1.3.5.- DISTRIBUCIÓN DE SOFTWARE.

La gestión de configuración debe permitir distribuir software a los sistemas finales (servidores, terminales,...) y sistemas intermedios (puentes, ruteadores,...). Para ello se debe disponer de facilidades que permitan: solicitudes de carga de software, transmisión de las versiones de software especificadas y actualización de la configuración de los sistemas.

Además de ejecutables, también se pueden distribuir tablas y otros datos que controlen el comportamiento de un nodo (p.e., tablas de encaminamiento que requieran intervención del administrador por motivos de costos, prestaciones o seguridad y que por tanto no puede calcularlas directamente el propio nodo mediante un algoritmo matemático). Se necesita un mecanismo de control de versiones de software que permita cargar diferentes versiones de software o tablas de encaminamiento en base a determinadas condiciones, tal como tasas de error.

1.1.4.- GESTIÓN DE SEGURIDAD.

La introducción de los ordenadores en las organizaciones, supuso la necesidad de preservar la seguridad de la información en dos aspectos:

- La utilización de sistemas de tiempo compartido obligó a buscar mecanismos de protección de información dentro del sistema ***(computer securit*y**).
- La utilización de redes de ordenadores obligó a buscar mecanismos de protección de información en su transmisión ***(network security***). Esto es especialmente importante si los datos atraviesan o están disponibles a través de redes publicas.

La gestión de seguridad en redes se ocupa de estas dos áreas incluyendo, obviamente, la seguridad del propio sistema de gestión de red:

- **Privacidad**: La información debe estar accesible para lectura sólo a personas autorizadas. Los accesos incluyen: leer, visualizar, imprimir, o incluso revelar la propia existencia del objeto.
- **Integridad**: Los recursos deben ser modificables sólo por usuarios autorizados. Las modificaciones incluyen: escribir, cambiar, cambiar estado, eliminar y crear.
- **Disponibilidad**: Los recursos deben estar disponibles a los usuarios autorizados.

Las funciones de gestión de seguridad son:

- Mantenimiento de la información de seguridad.
- Control de acceso a los recursos.
- Control del proceso de cifrado.

1.1.4.1.- MANTENIMIENTO DE LA INFORMACIÓN DE SEGURIDAD.

La gestión de seguridad debe seguir la actividad, o intentos de actividad, sobre estos objetos para detectar y resolver intentos de ataques, o ataques conseguidos. Esto incluye las siguientes funciones:

- *Log* de eventos.
- Monitorización de registros de seguridad.
- Monitorización de utilización y usuarios de recursos de seguridad.
- Avisos de violaciones de seguridad.

- Recepción de avisos de violaciones de seguridad.
- Mantener y examinar *logs* de seguridad.
- Mantener *backups, respaldos,* de los datos referentes a seguridad
- Mantener perfiles de usuarios y utilizaciones para recursos específicos para permitir la verificación del cumplimiento de las políticas de seguridad definidas.

Los objetos que intervienen: claves, información de autentificación, derechos de acceso y parámetros operacionales de los mecanismos y servicios de seguridad.

1.1.4.2.- CONTROL DE ACCESO A LOS RECURSOS.

Consiste en la autentificación y autorización de accesos a recursos. Para ello se mantienen perfiles de usuarios y perfiles de utilización para recursos específicos y se establecen prioridades de acceso.

1.1.4.3.- CONTROL DEL PROCESO DE CIFRADO.

Cifrado del intercambio de información entre agentes y gestores cuando sea necesario. Incluye la elección de algoritmos de cifrado y mecanismos de distribución de claves.

1.1.5.- GESTIÓN DE PRESTACIONES.

La gestión de prestaciones tiene dos partes: **monitorización** y **control. Monitorizar** consiste en observar y analizar la actividad de la red. **Controlar** es realizar ajustes para mejorar las prestaciones.

Algunas cuestiones que atañen al gestor de la red son:

- ¿Cuál es la utilización de la red?
- ¿Hay un tráfico excesivo?
- ¿Se ha reducido la productividad a niveles inaceptables?
- ¿Existen cuellos de botella?
- ¿Está aumentando el tiempo de respuesta?

Para ello hay que monitorizar algunos recursos de la red: Recoger información, analizarla y usar los resultados del análisis para ajustar los parámetros de la red

Se deben de elegir los indicadores adecuados para monitorizar adecuadamente las prestaciones de la red.

Los indicadores se pueden dividir en dos categorías:

- **Medidas orientadas a servicios:** Medidas que permiten mantener los niveles de determinados servicios a satisfacción de los usuarios. Son los más importantes:
  - Disponibilidad.
  - Tiempo de respuesta.
  - Fiabilidad.
- **Medidas orientadas a eficiencia:** Medidas que permiten mantener los niveles de satisfacción anteriores al mínimo coste posible.
  - Prestaciones *(througtput)*.
  - Utilización.

1.1.5.1.- MEDIDAS ORIENTADAS A SERVICIOS.

- **Disponibilidad:** Porcentaje de tiempo que una red, un dispositivo o una aplicación está disponible para el usuario.
- **Tiempo de respuesta:** Cuanto tarda en aparecer la respuesta en el terminal del usuario cuando éste realiza una acción.
- **Fiabilidad:** Porcentaje de tiempo en el que no ocurren errores en la transmisión y entrega de información.

1.1.5.2.- MEDIDAS ORIENTADAS A EFICIENCIA.

- **Prestaciones *(throughput*):** La tasa a la que ocurren eventos a nivel de aplicación (p.e. Transacciones, mensajes, transferencia de archivos).
- **Utilización:** Porcentaje de la capacidad teórica de un recurso (p.e. Un concentrador, una línea de transmisión, un conmutador) que se está utilizando.

1.2.- ARQUITECTURA FUNCIONAL DE MONITORIZACIÓN.

Los cuatro elementos principales que intervienen en la monitorización son:

1. Aplicación de monitorización: gestiona las funciones de monitorización que son visibles al usuario, tal como gestión de prestaciones, fallos y costos.

2. Gestor: El módulo de la red que recoge la información del resto de los elementos de la red.

3. Agente: Recoge y almacena información de uno o varios elementos de la red y los envía al **gestor**.

4. Objetos gestionados: información de gestión que representa recursos y su actividad.

5. Agente de monitorización: módulo que genera sumarios y análisis estadísticos de la información de gestión. Si este módulo es remoto al **gestor**, actúa como un agente y comunica la información estadística que genera a dicho gestor (véase figura 1.2 y 1.3).

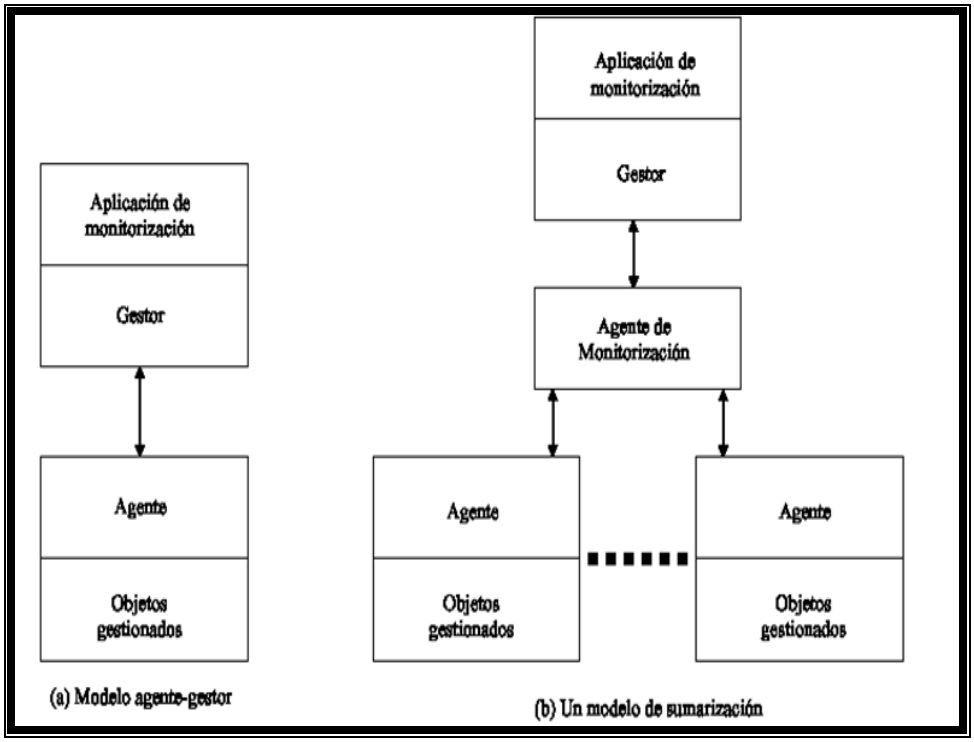


Figura 1.2.- Arquitectura Funcional de Monitorización.

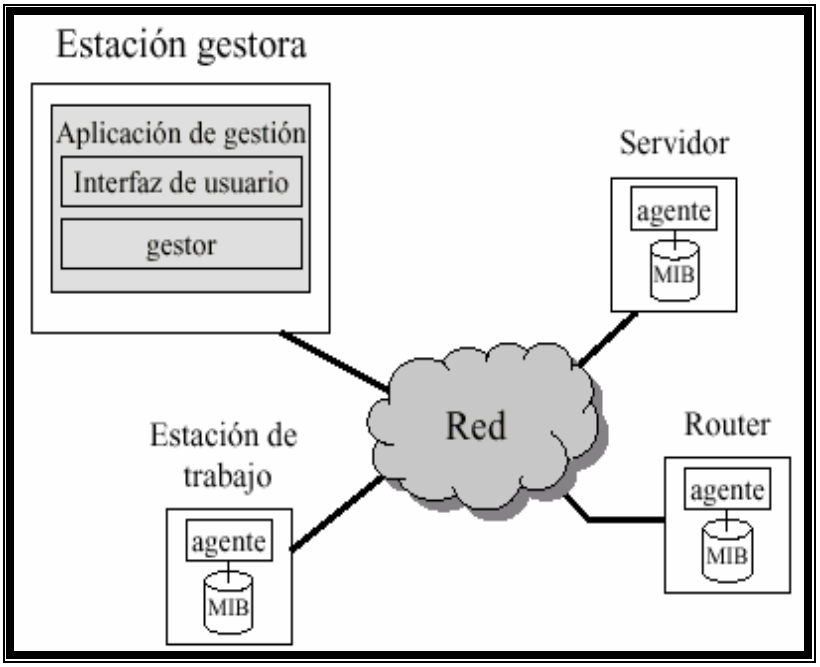


Figura 1.3.- Arquitectura Funcional de Monitorización (cont.).

Los módulos funcionales anteriores pueden dar lugar a diversas configuraciones:

(a) La estación que ejecuta la **aplicación de monitorización** es también un elemento de la red, y debe gestionarse. Por ello normalmente incluye un **Agente** y unos **Objetos gestionados** (véase figura 1.4).

Figura 1.4.- Estación como elemento de red.

(b) Configuración normal de monitorización de otros elementos de red. El **gestor** y los **agentes** deben compartir el mismo protocolo de gestión y sintaxis y semántica de la MIB (véase figura 1.5).

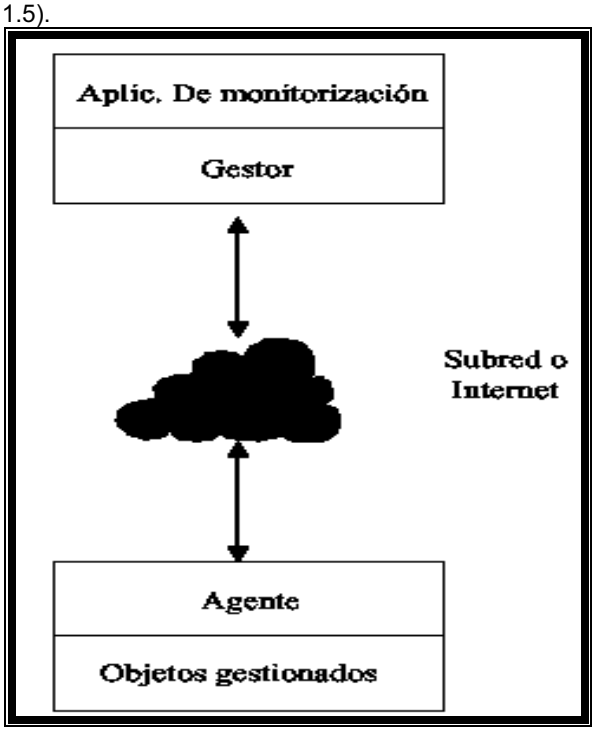


Figura 1.5.- Configuración de otros elementos de la red.

(c) Configuración con **agentes** que monitorizan el tráfico de una red, **monitores remotos** o **monitores externos** (véase figura 1.6)**.**

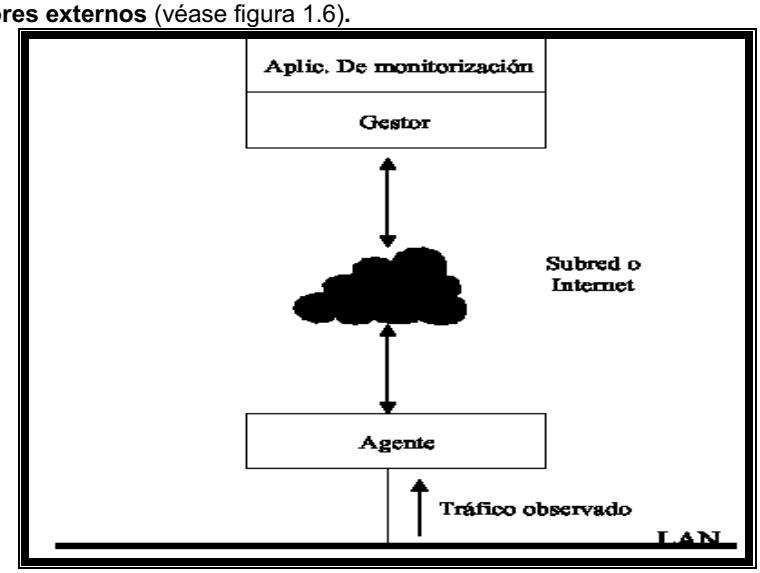


Figura 1.6.- Monitores remotos.

(d) Cuando existen elementos que no utilizan el mismo protocolo de gestión que el **gestor**, se utilizan **proxie**s (véase figura 1.7).

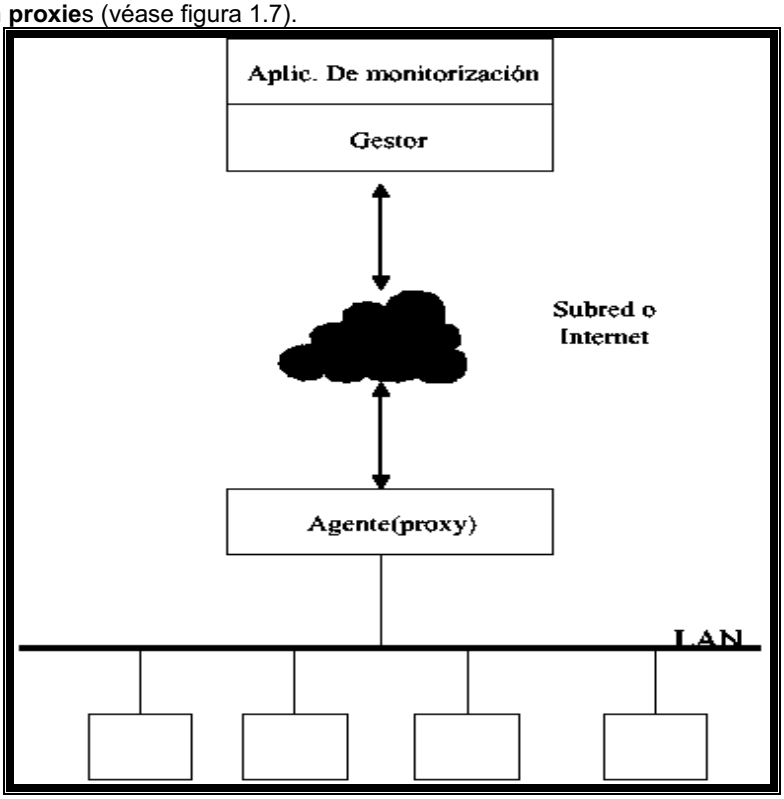


Figura 1.7.- Proxies.

1.2.1.- FORMAS DE COMUNICACIÓN DE INFORMACIÓN DE MONITORIZACIÓN.

La información de monitorización es recopilada y almacenada por los **agentes** y enviada a uno o más **gestore**s. Para el envío de dicha información se usan dos técnicas:

- **Sondeo *(polling*):** El **gestor** solicita información al **agente**, que responderá a la petición.
- **Informe de eventos *(event reporting*):** La iniciativa de la comunicación es tomada por el **agente**, teniendo que estar por tanto el **gestor** a la espera de información de este tipo.

Se suelen usar conjuntamente en la monitorización de una red. El énfasis en el uso de uno u otro dependerá de la propia naturaleza de la red y de los protocolos de gestión utilizados.

1.2.1.1.- SONDEO.

Interacción petición/respuesta iniciada por el **gesto**r. El **gestor** puede pedir información sobre:

El valor de uno o varios elementos de información. El **agente** devolverá valores almacenados en su MIB local. Este tipo de petición puede ser dos tipos:

Específica: Pidiendo el valor de una o varias variables concretas.
Genérica o de búsqueda: Solicitando información que cumpla ciertos criterios de búsqueda.

Información sobre la estructura de la MIB del agente. Se utiliza para:

Mantener actualizada la información que el **gestor** tiene de los elementos la red. Para ello se hace un sondeo periódico.
Conocer la configuración de la red (y sus elementos) que está gestionando
Investigar un área en detalle ante un problema.
Soporte a usuarios: generar informes solicitados por los usuarios o respondes sus preguntas.

1.2.1.2.- INFORME DE EVENTOS.

El **agente** toma la iniciativa de enviar información al **gestor**. Se utiliza para:

- Comunicar la ocurrencia de eventos relevantes (p.e. Un cambio de estado) o inusuales (p.e. Un fallo).
- Generar un informe periódico al **gestor** del estado actual de los elementos gestionados por el **agente**.

Este sistema de comunicación es útil, sobre todo, para:

- Detectar problemas tan pronto como se producen.
- Monitorizar objetos que cambian con poca frecuencia de estado (en este caso es más eficiente que el sondeo).

1.3.- ARQUITECTURA FUNCIONAL DE CONTROL.

Los cuatro elementos principales que intervienen en el control son:

1. Aplicación de control: gestiona las funciones de control que son visibles al usuario, tal como gestión de configuración y seguridad.

2. Gestor: El módulo de la envía peticiones de operación al resto de los elementos de la red.

3. Agente: Recibe los comandos de control y actúa convenientemente sobre los elementos de la red que depende de él.

4. Objetos gestionados: información de gestión que representa recursos y su actividad.

De la información de gestión. Si este módulo es remoto al **gestor**, actúa como un agente y comunica la información estadística que genera a dicho gestor (véase figura 1.8).

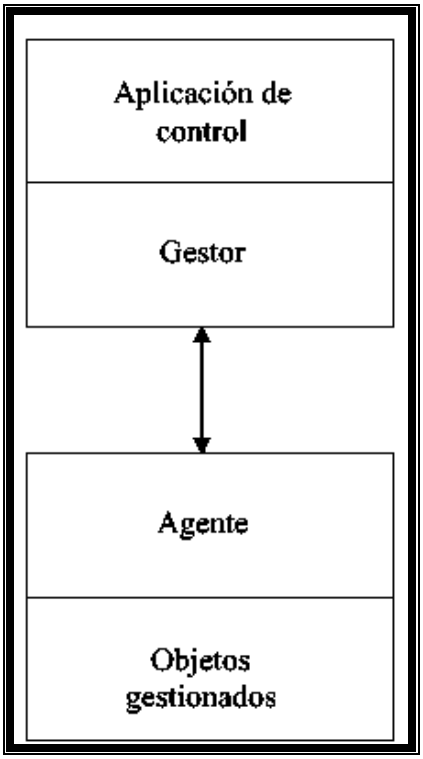


Figura 1.8.- Modelo Agente-Gestor.

Los módulos funcionales anteriores pueden dar lugar a diversas configuraciones:

(a) La estación que ejecuta la **aplicación de control** es también un elemento de la red, y debe gestionarse. Por ello normalmente incluye un **Agente** y unos **Objetos gestionados** véase figura 1.9).

Figura 1.9.- Agente-Modelos Gestionados.

(b) Configuración normal de control de otros elementos de red. El **gestor** y los **agentes** deben compartir el mismo protocolo de gestión y sintaxis y semántica de la MIB.

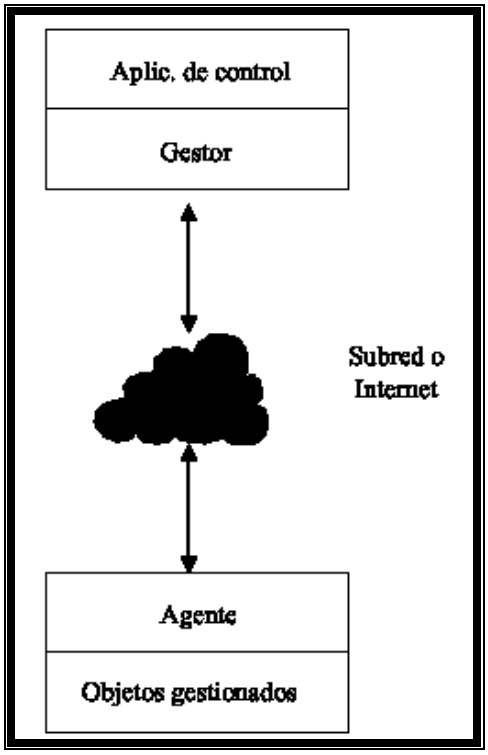


Figura 1.10.- Agente en red.

(c) Cuando existen elementos que no utilizan el mismo protocolo de gestión que el gestor, se utilizan proxies (véase figura 1.11).

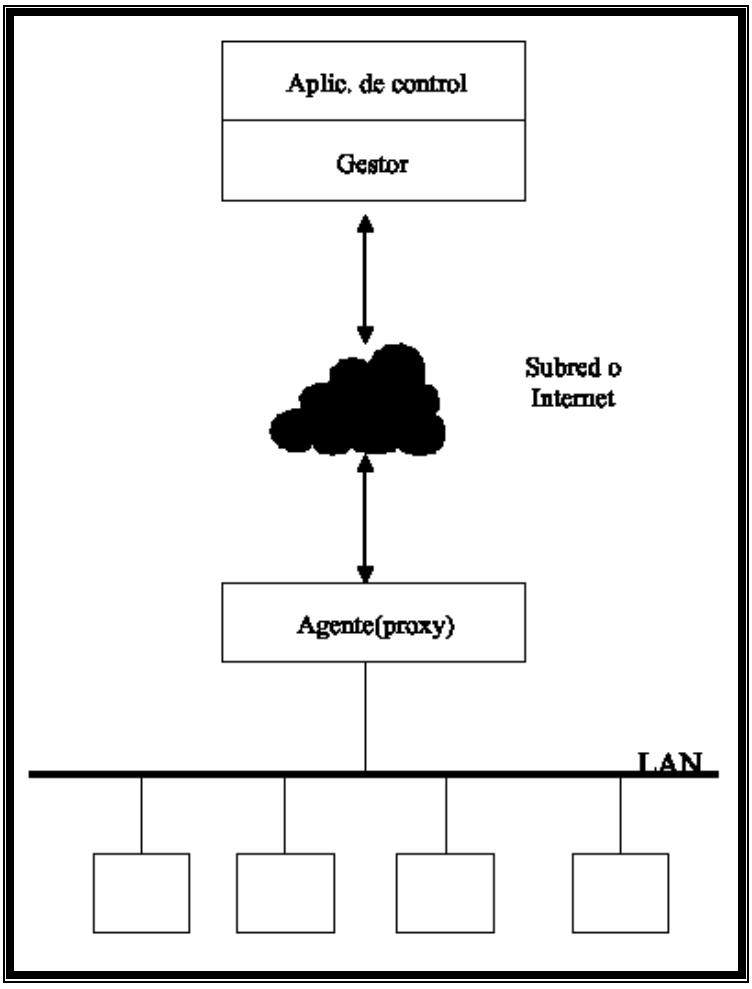


Figura 1.11.- Agente Proxy.

1.4.- FORMAS DE COMUNICACIÓN DE INFORMACIÓN DE CONTROL.

El gestor comienza siempre la comunicación enviando comandos a los agentes. En función del tipo de comando, puede haber una respuesta del agente o no. Los comandos son generados por diversos motivos (véase figura 1.12):

- Iniciativa del administrador de la red.
- Comandos preprogramados:
  - Periódicos.
  - Respuesta a eventos o sucesos.

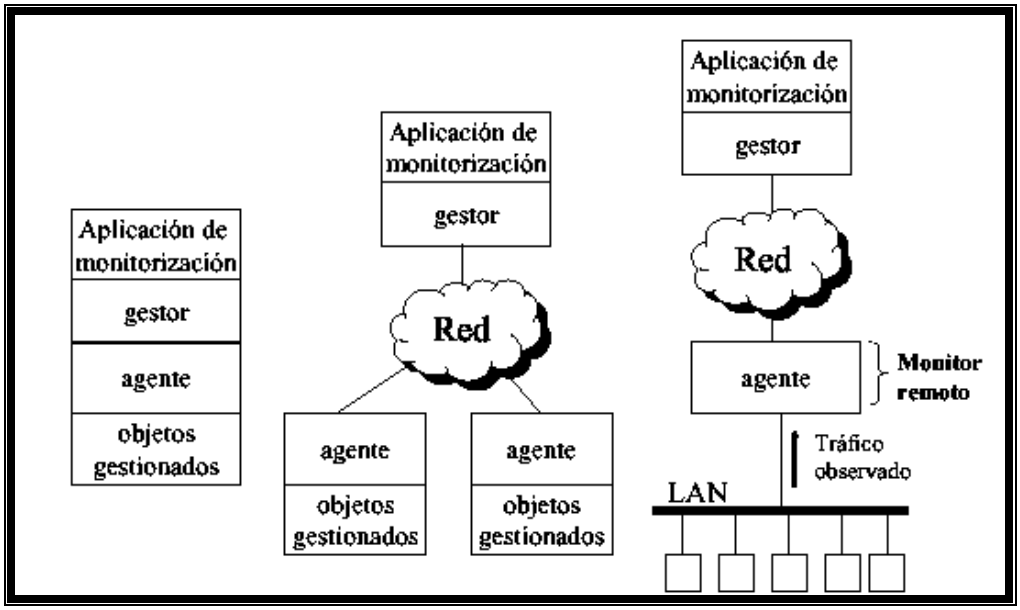


Figura 1.12.- Formas de Comunicación del control.

1.5.- LAS VLAN (*VIRTUAL LOCAL AREA NETWORK*).

Una VLAN es una agrupación lógica de dispositivos o usuarios que se pueden agrupar por función, departamento o aplicación, sin importar la ubicación física del segmento. La configuración de la VLAN se realiza en el conmutador a través del software (véase figura 1.13). **Las VLAN no están estandarizadas y requieren el uso de software propietario del fabricante del conmutador**.

1.5.1.- CONFIGURACIONES LAN COMPARTIDAS EXISTENTES.

Una LAN típica se configura según la infraestructura física que conecta. Los usuarios se agrupan según su ubicación en relación con el concentrador al que están conectados y según cómo el cable se tiende al centro del cableado. El router que interconecta cada concentrador compartido normalmente proporciona segmentación y puede actuar como corta fuegos de difusión (firewall broadcast). Los segmentos creados por los conmutadores no lo hacen. La segmentación tradicional de las LAN no agrupa a los usuarios según su asociación de grupo de trabajo o necesidad de ancho de banda. Por lo tanto, comparten el mismo segmento y ocupan el mismo ancho de banda, aunque los requisitos de ancho de banda varían enormemente por grupo de trabajo o departamento [1, 2, 3,23, 29].

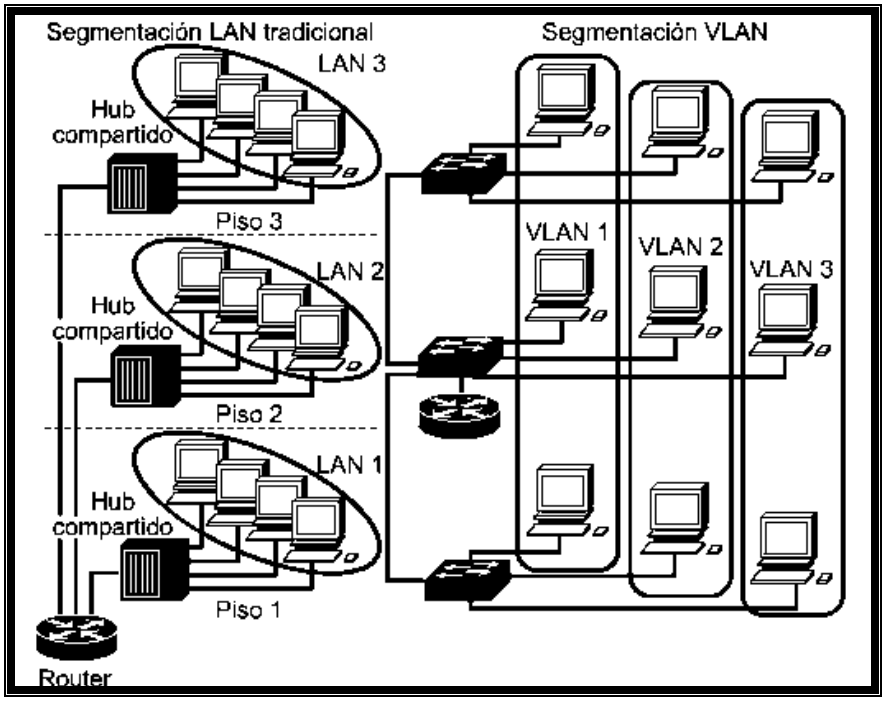


Figura 1.13.- Las VLAN y los límites físicos

1.5.2.- SEGMENTACIÓN CON ARQUITECTURAS DE CONMUTACIÓN

Las LAN se dividen cada vez con mayor frecuencia en grupos de trabajo conectados mediante la espina dorsal (backbone) común para formar topologías VLAN. Las VLAN segmentan lógicamente la infraestructura física de las LAN en diferentes subredes o dominios de difusión (broadcast) para Ethernet. Las tramas de difusión (broadcast) se conmutan sólo entre puertos dentro de la misma VLAN [4, 23,29].

Las primeras implementaciones de VLAN ofrecían una función de asignación de puertos que establecía un dominio de difusión (broadcast) entre un grupo de dispositivos por defecto. Los requisitos actuales de la red exigen la funcionalidad de VLAN que cubre toda la red. Este enfoque de las VLAN permite agrupar usuarios separados por grandes distancias físicas en topologías virtuales que abarcan toda la red. Las configuraciones VLAN agrupan a los usuarios por asociación lógica, en lugar de por ubicación física.

La mayoría de las redes actualmente instaladas ofrecen una segmentación lógica muy limitada. Los usuarios se agrupan normalmente según las conexiones al concentrador compartido y los puertos de router entre los concentradores. Esta topología brinda segmentación sólo entre los, que normalmente se ubican en pisos separados, y no entre usuarios conectados al mismo concentrador. Esto impone restricciones físicas en la red y limita la manera en que los usuarios se pueden agrupar. Algunas arquitecturas de concentrador compartido tienen cierta capacidad de agrupación, pero limitan la forma en que se pueden configurar los grupos de trabajo definidos lógicamente [4, 23, 29].

1.5.3.- DIFERENCIAS ENTRE LAS LAN CONMUTADAS TRADICIONALES Y LAS VLAN

En una LAN que utiliza dispositivos de conmutación LAN, la tecnología VLAN es una manera económica y eficiente de agrupar usuarios de la red en grupos de trabajo virtuales, más allá de su ubicación física en la red. Algunas de las diferencias principales son las siguientes [3, 4, 23, 29, 30]:

- Las VLAN funcionan a nivel de Capa 2 y Capa 3 del modelo de referencia OSI.
- La comunicación entre las VLAN es implementada por el enrutamiento de Capa 3.
- Las VLAN proporcionan un método para controlar la difusión (broadcast) de red.
- El administrador de la red asigna usuarios a una VLAN.
- Las VLAN pueden aumentar la seguridad de la red, definiendo cuáles son los nodos de red que se pueden comunicar entre sí.

Mediante la tecnología VLAN, se pueden agrupar los puertos de conmutador y sus usuarios conectados en grupos de trabajo lógicamente definidos, como los siguientes:

- Compañeros de trabajo en el mismo departamento
- Un equipo de producción interfuncional
- Diferentes grupos de usuarios que comparten la misma aplicación de red o software

Se pueden agrupar estos puertos y usuarios en grupos de trabajo con un solo conmutador o conmutador conectados. Al agrupar los puertos y los usuarios a través de múltiples conmutador, las VLAN pueden abarcar infraestructuras contenidas en un solo edificio, edificios conectados entre sí o aun redes de área amplia (WAN).

1.5.4.- TRANSPORTE DE LAS VLAN A TRAVÉS DE LAS ESPINAS DORSALES (BACKBONE)

Lo que es importante en cualquier arquitectura de VLAN es la capacidad para transportar información de la VLAN entre conmutador interconectados y los ruteador que residen en la espina dorsal (backbone) corporativa. Estas capacidades de transporte [1, 2, 3]:

- Eliminan las fronteras físicas entre los usuarios
- Aumentan la flexibilidad de la configuración de una solución de VLAN cuando los usuarios se desplazan
- Proporcionan mecanismos de interoperabilidad entre los componentes del sistema de espina dorsal (backbone).

La espina dorsal (backbone) normalmente funciona como el punto de reunión de grandes volúmenes de tráfico. También transporta información del usuario final de la VLAN y su identificación entre conmutador, ruteador y servidores directamente conectados. Dentro de la espina dorsal (backbone), los enlaces de alto ancho de banda y alta capacidad se seleccionan normalmente para transportar el tráfico en toda la empresa.

Los conmutadores y ruteadores son los componentes principales de las comunicaciones de VLAN (véase figura 1.14). Cada conmutador tiene la inteligencia de tomar decisiones de filtrado y envío por trama, basándose en las métricas de VLAN definidas por los administradores de red. El conmutador también puede comunicar esta información a otros conmutadores y ruteadores dentro de la red.

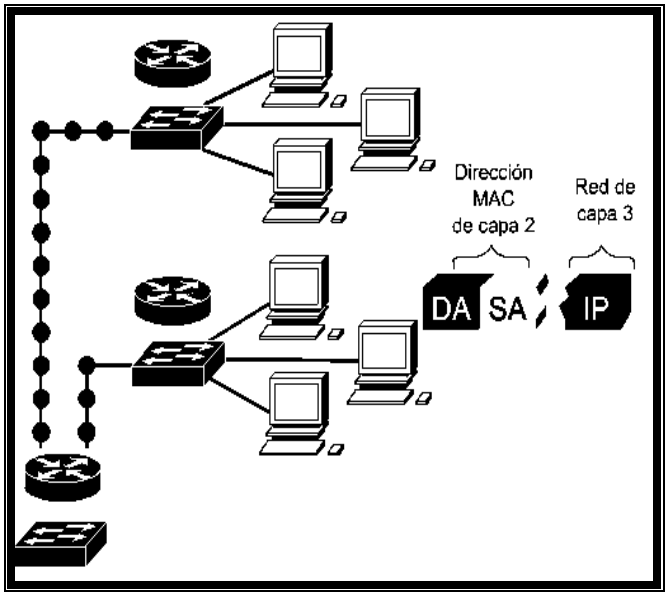


Figura.- 1.14 Conmutadores y Ruteadores en VLAN.

Los enfoques más comunes para agrupar lógicamente los usuarios en VLAN diferentes son el filtrado de trama y la identificación de trama (etiquetado de trama). Ambas técnicas examinan la trama cuando es recibida o enviada por el conmutador. Basadas en el conjunto de normas definidas por el administrador, estas técnicas determinan dónde la trama se debe enviar, filtrar o hacer difusión (broadcast). Estos mecanismos de control pueden ser administrados centralmente (con software de administración de red) y se implementan con facilidad en toda la red [3, 30].

El filtrado de trama examina información específica acerca de cada trama, se desarrolla una tabla de filtrado para cada conmutador; esto proporciona un alto nivel de control administrativo porque puede examinar muchos atributos de cada trama. Según la sofisticación del conmutador LAN, puede agrupar los usuarios según las direcciones de control de acceso al medio (MAC) de una estación o tipo de protocolo de capa de red. El conmutador compara las tramas que filtra con las entradas de tabla, y toma las medidas apropiadas según las entradas. Al principio, las VLAN eran basadas en filtros y agrupaban a los usuarios según una tabla de filtrado. Este modelo no era muy escalable, porque cada trama debía referirse a una tabla de filtrado. El etiquetado de trama asigna de forma exclusiva un identificador de VLAN a cada trama. Los identificadores de VLAN son asignados a cada VLAN en la configuración de conmutador por el administrador del conmutador. Esta técnica fue seleccionada por el grupo de estándares del Instituto de Ingeniería Eléctrica y Electrónica (IEEE) debido a su escalabilidad. El etiquetado de trama está obteniendo reconocimiento como el mecanismo troncal estándar. En comparación con el etiquetado de trama, puede ofrecer una solución más escalable para la implementación de las VLAN, que se puede aplicar a nivel de todo el campus. IEEE 802.1q establece que el etiquetado de trama es la manera de implementar las VLAN El etiquetado de trama de VLAN es un enfoque que se ha desarrollado específicamente para las comunicaciones conmutadas. El etiquetado de trama coloca un identificador único en el encabezado de cada trama a medida que se envía por todo la espina dorsal (backbone) de la red. El identificador es comprendido y examinado por cada conmutador antes de enviar cualquier difusión (broadcast) o transmisión a otros conmutadores, ruteadores o dispositivos de estación final. Cuando la trama sale de la espina dorsal (backbone) de la red, el conmutador elimina el identificador antes de que la trama se transmita a la estación final objetivo. La identificación de trama funciona a nivel de Capa de enlace de datos y requiere poco procesamiento o sobrecarga administrativa.

1.5.5.- IMPLEMENTACIÓN DE VLAN.

Una VLAN forma una red conmutada lógicamente segmentada por funciones, equipos de proyectos o aplicaciones, sin tener en cuenta la ubicación física de los usuarios. Cada puerto de conmutador se puede asignar a una VLAN. Los puertos asignados a la misma VLAN comparten difusión (broadcast). Los puertos que no pertenecen a esa VLAN no comparten esos dominios de difusión (broadcast). Esto mejora el rendimiento general de la red. Las siguientes secciones hacen referencia a tres métodos de implementación de VLAN que se pueden usar para asignar un puerto de conmutador a una VLAN. Ellos son [3, 4]:

- De puerto central
- Estática
- Dinámica

1.5.5.1.- LAS VLAN DE PUERTO CENTRAL FACILITAN EL TRABAJO DEL ADMINISTRADOR

En las VLAN de puerto central, a todos los nodos conectados a puertos en la misma VLAN se les asigna el mismo identificador de VLAN. El gráfico (véase figura 1.15) muestra la pertenencia a la VLAN por puerto, lo que facilita el trabajo del administrador y hace que la red sea más eficiente porque:

- Los usuarios se asignan por puerto.
- Las VLAN son de fácil administración.
- Proporciona mayor seguridad entre las VLAN.
- Los paquetes no se "filtran" a otros dominios.

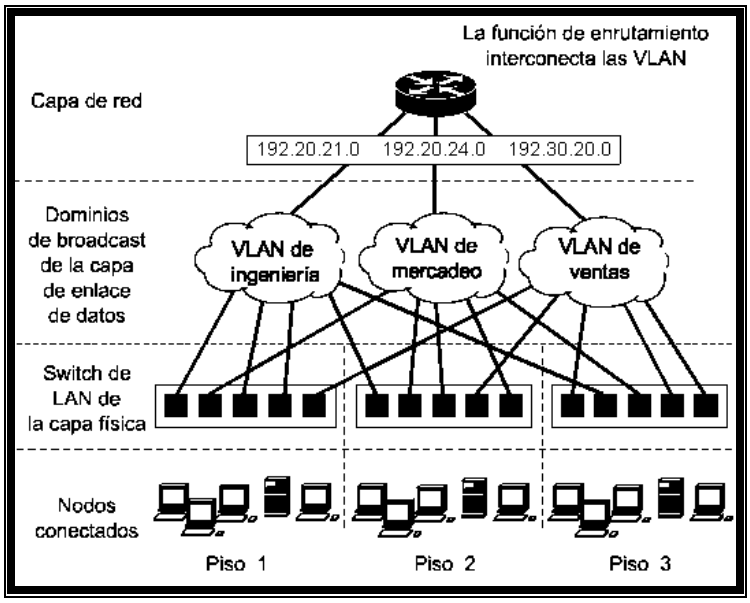


Figura 1.15.- VLAN de puerto central

1.5.5.2.-. VLAN ESTÁTICAS.

Las VLAN estáticas [1, 3, 23, 29] son puertos en un conmutador que se asignan estáticamente a una VLAN. Estos puertos mantienen sus configuraciones de VLAN asignadas hasta que se cambien. Aunque las VLAN estáticas requieren que el administrador haga los cambios, este tipo de red es segura, de fácil configuración y monitoreo. Las VLAN estáticas funcionan bien en las redes en las que el movimiento se encuentra controlado y administrado (véase figura 1.16).

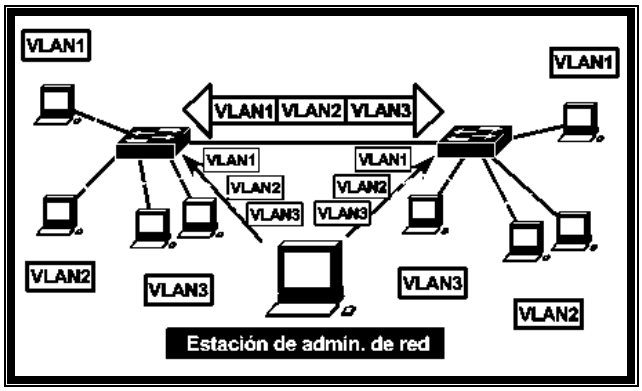


Figura 1.16.- VLAN estáticas

1.5.5.3.-VLAN DINÁMICAS.

Las VLAN dinámicas [1, 3, 23, 29] son puertos en un conmutador que pueden determinar automáticamente sus asignaciones de VLAN. Las funciones de las VLAN dinámicas se basan en las direcciones MAC, direccionamiento lógico o tipo de protocolo de los paquetes de datos. Cuando una estación se encuentra inicialmente conectada a un puerto de conmutador no asignado, el conmutador correspondiente verifica la entrada de direcciones MAC en la base de datos de administración de la VLAN y configura dinámicamente el puerto con la configuración de VLAN correspondiente. Los principales beneficios de este enfoque son una necesidad de administración menor en el centro de cableado, cuando se agrega o desplaza un usuario y la notificación centralizada cuando se agrega un usuario no reconocido en la red. Normalmente, se necesita una mayor cantidad de administración en un primer momento para configurar la base de datos dentro del software de administración de la VLAN y para mantener una base de datos exacta de todos los usuarios de la red (véase figura 1.17).

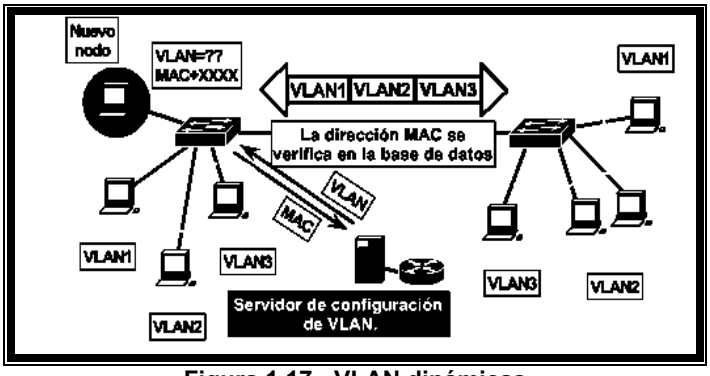


Figura 1.17.- VLAN dinámicas

RESUMEN CAPITULO I. LA GESTIÓN DE REDES.

Las empresas se reorganizan continuamente. Como promedio, de 20% a 40% de los empleados se trasladan físicamente todos los años. Esto hace que sea necesario implementar una serie de adiciones, desplazamientos y cambios que constituyen uno de los más grandes dolores de cabeza para el administrador de la red, y también uno de los más grandes gastos relacionados con la administración de redes. Muchos desplazamientos hacen que sea necesario rehacer el cableado, y casi todos requieren que se cambien las direcciones de las estaciones y se reconfiguren concentyradores y ruteador.

Las VLAN ofrecen un mecanismo efectivo para controlar estos cambios y reducir en gran parte el costo asociado con las reconfiguraciones de concentrador y ruteadores. Los usuarios en una VLAN pueden compartir el mismo espacio de dirección de red (es decir, la subred IP), sin importar su ubicación.

Cuando los usuarios en una VLAN se trasladan de una ubicación a otra, siempre que permanezcan en la misma VLAN y se conecten a un puerto de conmutador, sus direcciones de red no cambian. Un cambio de ubicación puede ser tan sencillo como conectar un usuario a un puerto en un conmutador adaptado para VLAN y configurar el puerto del conmutador a esa VLAN.

Las VLAN representan un importante progreso con respecto a las técnicas basadas en LAN que se usan en los centros del cableado, porque necesitan menos cambios en el cableado, configuración y depuración. La configuración del router queda intacta. Cuando simplemente se debe desplazar a un usuario de una ubicación a otra, esto no crea modificaciones en la configuración del router, si el usuario permanece en la misma VLAN.

El tráfico de difusión (broadcast) [1, 3, 23, 29] se produce en todas las redes. La frecuencia de difusión (broadcast) depende de los tipos de aplicaciones, los tipos de servidores, la cantidad de segmentación lógica y la manera en que se usan estos recursos de red. Aunque las aplicaciones se han perfeccionado durante los últimos años para reducir la cantidad de difusión (broadcast) que envían, se están desarrollando nuevas aplicaciones multimediales que producen gran cantidad de difusión (broadcast) y multidifusión (multicast).

Es necesario tomar medidas preventivas para evitar los problemas relacionados con la difusión (broadcast). Una de las medidas más efectivas es segmentar de manera adecuada la red, con corta fuegos (firewalls) de protección que, dentro de lo posible, eviten que los problemas de un segmento dañen otras partes de la red. Así, aunque un segmento puede presentar condiciones de difusión (broadcast) excesivas, el resto de la red se encuentra protegido con un firewall, normalmente proporcionado por un router. La segmentación con corta fuegos (firewalls) brinda confiabilidad y reduce al mínimo la sobrecarga de tráfico de difusión (broadcast), permitiendo un mayor rendimiento del tráfico de aplicaciones.

Cuando no se colocan ruteador entre los conmutador, los difusión (broadcast) se envían a cada puerto del conmutador. Esto normalmente se denomina red plana, donde hay un solo dominio de difusión (broadcast) para toda la red. La ventaja de una red plana es que proporciona baja latencia y alto rendimiento, y es fácil de administrar. La desventaja es que aumenta la vulnerabilidad al tráfico de difusión (broadcast) en todos los conmutadores, puertos, enlaces de espina dorsal (backbone) y usuarios.

Las VLAN son un mecanismo efectivo para extender los corta fuegos (firewalls) desde los ruteador a la estructura de los conmutador y proteger la red contra problemas de difusión

(broadcast) potencialmente peligrosos. Además, las VLAN conservan todas las ventajas de rendimiento de la conmutación.

Se pueden crear corta fuegos (firewalls) asignando puertos de conmutador o usuarios a grupos de VLAN específicos dentro de conmutador individuales y a través de múltiples conmutador conectados. El tráfico de difusión (broadcast) dentro de una VLAN no se transmite fuera de la VLAN. Por el contrario, los puertos adyacentes no reciben ningún tráfico de difusión (broadcast) generado desde otras VLAN. Este tipo de configuración reduce sustancialmente el tráfico total de difusión (broadcast), libera el ancho de banda para el tráfico real de usuarios, y reduce la vulnerabilidad general de la red a las tormentas de difusión (broadcast).

Cuanto menor sea el grupo de VLAN, menor será la cantidad de usuarios afectados por la actividad de tráfico de difusión (broadcast) dentro del grupo de VLAN. También se pueden asignar VLAN basadas en el tipo de aplicación y la cantidad de difusión (broadcast) de aplicaciones. Se pueden colocar usuarios que comparten una aplicación que produce difusión (broadcast) en el mismo grupo de VLAN y distribuir la aplicación a través del campus.

El uso de las LAN ha aumentado en forma notable durante los últimos años. Como resultado, en las LAN a menudo circulan datos confidenciales y fundamentales para el trabajo. Los datos confidenciales requieren seguridad implementada a través de limitación del acceso. Uno de los problemas de las LAN compartidas es que son relativamente fáciles de penetrar. Un intruso puede conectarse a un puerto activo y tener acceso a todo el tráfico dentro de un segmento. Cuanto mayor sea el grupo, mayores serán las posibilidades de acceso. Una técnica de administración económica y sencilla para aumentar la seguridad es segmentar la red en múltiples grupos de difusión (broadcast) que permitan que el administrador de red:

- Limite la cantidad de usuarios en un grupo de VLAN
- Evite que otro usuario se conecte sin recibir antes la aprobación de la aplicación de administración de red de la VLAN
- Configure todos los puertos no utilizados en una VLAN de bajo servicio por defecto

La implementación de este tipo de segmentación es relativamente simple. Los puertos de conmutador se agrupan según el tipo de aplicaciones y privilegios de acceso. Las aplicaciones y recursos de acceso restringido se ubican normalmente en un grupo seguro de VLAN. En la VLAN segura, el conmutador limita el acceso al grupo. Las restricciones se pueden implementar según las direcciones de estación, tipos de aplicación o tipos de protocolo.

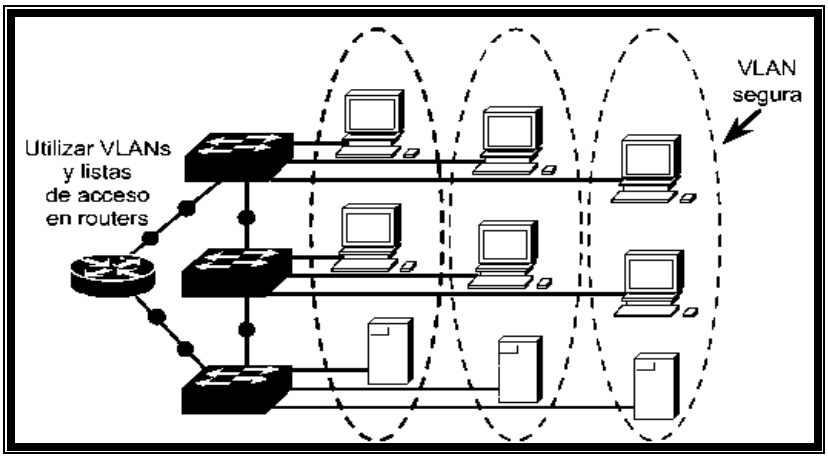


Figura 1.18.- Seguridad de VLAN.

CAPITULO II
PROTOCOLO DE ADMINISTRACIÓN DE RED SIMPLE SNMP (*SIMPLE NETWORK MANAGEMENT PROTOCOL*)

INTRODUCCIÓN

Los entornos de red, sus configuraciones y funcionamiento forman parte de la gestión de red, algunos de los elementos en los que se hace hincapié de forma determinantes en la configuración y el monitoreo; las VLAN's al ser redes de virtuales de área local, no quedan ajenas a estas funciones. Sin embargo surge la pregunta, ¿Cómo realizar la configuración a través de los variable principales que residen en la base de datos del ó los dispositivos que interviene en la red virtual de área local?; y ¿Cómo efectuar un monitoreo en este tipo de redes?. La respuesta a tales incógnitas se desglosan en el presente capitulo, que describe de manera detallada el protocolo de administración de red simple, SNMP (*Simple Network Management Protocol*). Este protocolo permite el acceso a las bases de datos del dispositivo y modificar parámetros asociados a las mismas, con lo que los cambios se reflejan de manera inmediata. SNMP es un protocolo que permite la gestión de los recursos que están disponibles en una red. Dentro de un entorno de red gestionado con SNMP habrá un conjunto de nodos de la red que se encarguen de la gestión y un conjunto de componentes de la red (hosts, concentradores, ruteadores, modems, etc.) que podrán ser gestionados por estas estaciones.

El fin del capitulo es dilucidar en detalle el protocolo de administración de red simple para aplicarlo al desarrollo de los módulos que se incorporan en la Herramienta de Gestión de Redes Virtuales.

2.1.- ELEMENTOS DE SNMP (SIMPLE NETWORK MANAGEMENT PROTOCOL)

Para que el software que se encarga de la gestión de la red en las estaciones de gestión pueda obtener información de los elementos de la red, es necesario que dichos elementos cuenten con un software que permita su comunicación con la estación de gestión. Este software se denomina agente [2].

Por último hay otra pieza importante en este entorno, y es la base de datos donde se encuentra toda la información que se gestiona. Esta base de datos se denomina MIB (*Management Information Base).*

Por lo tanto, se pueden distinguir los siguientes elementos [2]:

- Agente de gestión.
- Gestor
- Objeto gestionado
- Protocolo de gestión.

El agente de gestión, se encarga de supervisar un elemento de la red. Se comunica con el gestor para atender sus peticiones y para informarle de eventos acaecidos en el objeto gestionado. El agente de gestión suele residir físicamente en el elemento gestionado.

El gestor, es un software residente en una estación de gestión que se comunica con los agentes y que ofrece al usuario una interfaz a través de la cual comunicarse con los agentes de gestión

para obtener información de los recursos gestionados. Además recibirá las notificaciones enviadas por los agentes.

Los objetos gestionados, son las abstracciones de los elementos físicos de la red que se gestionan (tarjeta de red, concentrador, módem, router, etc.). Se pueden manejar los atributos y las operaciones que se pueden realizar sobre el objeto. De la misma forma, las notificaciones que dicho objeto puede generar así como las relaciones con otros objetos también son susceptibles de ser controladas. La base de datos de gestión (MIB) previamente mencionada está formada por todos los objetos gestionados.

Protocolo de gestión, es el protocolo que especifica cómo se realizará la comunicación entre los agentes de gestión y el gestor. En nuestro caso este protocolo es el SNMP. La comunicación se realiza en base a requerimientos, respuestas y notificaciones (véase figura. 2.1):

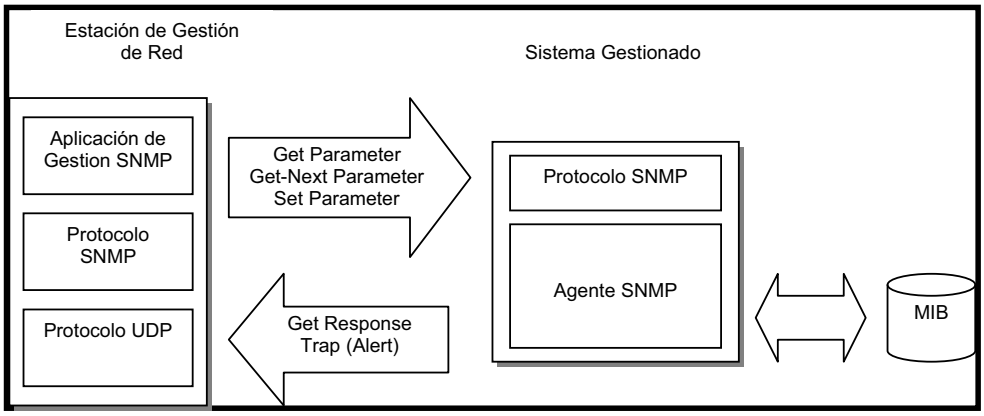


Figura 2.1.- Elementos de SNMP.

2.1.1.- AGENTES PROXY

Lo habitual es que el agente resida en el propio objeto gestionado, pero es posible que se requiera que la gestión de un objeto no incorpore el agente. Para realizar la gestión de este objeto es posible hacer uso de un intermediario que reciba los mensajes del gestor por un lado y se comunique con el objeto por otro [1] (véase figura. 2.2).

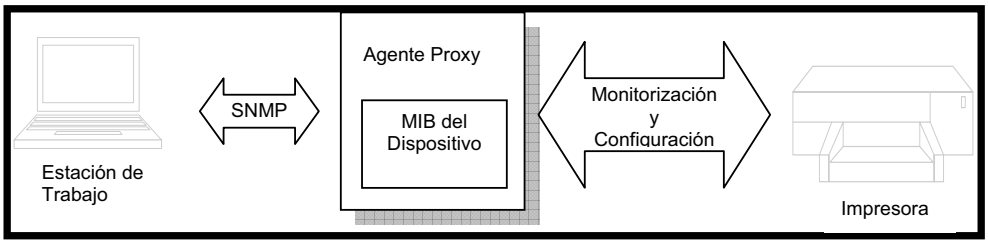


Figura 2.2.- Proxies.

2.2.- MENSAJES SNMP

SNMP define cinco tipos de mensajes [2]:

- Get Request: Para leer el valor de una o varias variables del MIB.
- Get Next Request: Para realizar lecturas secuenciales a través del Mffi.
- Get.Response: Es el mensaje de respuesta a un Set Request, Get Request o Get
- Next Request.
- Set Request: Mensaje enviado para establecer el valor de una variable.
- Trap: A través de este mensaje se hacen notificaciones de eventos.

Estos cinco tipos de mensajes SNMP son encapsulados en datagramas UDP. Los mensajes de petición y respuesta son enviados al puerto 161, mientras que las notificaciones de eventos usan el puerto 162.

2.2.1.- COMUNIDADES

Se denomina comunidad (community) a un conjunto de gestores y a los objetos gestionados. A las comunidades se les asignan nombres, de tal forma que este nombre junto con cierta información adicional sirva para validar un mensaje SNMP y al emisor del mismo.

Así, por ejemplo, si se tienen dos identificadores de comunidad "total" y "parcial", se podría definir que el gestor que use el identificador "total" tenga acceso de lectura y escritura a todas las variables de la base de administración de información, MIB (*Management Information Base*), mientras que el gestor con nombre de comunidad "parcial" sólo pueda acceder para lectura a ciertas variables del MIB.

2.2.2.- BASE DE ADMINISTRACIÓN DE INFORMACIÓN, MIB (*MANAGEMENT INFORMATION BASE*)

El MIB se organiza en forma de árbol. Cada nodo del árbol tiene asociado un número entero y una etiqueta de texto. Un nodo se identifica unívocamente con una secuencia de números enteros que identifican los nodos a través de los cuales hay que pasar para llegar desde la raíz al nodo que interese[2].

El siguiente gráfico muestra parte del árbol definido por ISO (*International Standards Organization*), (véase figura. 2.3):

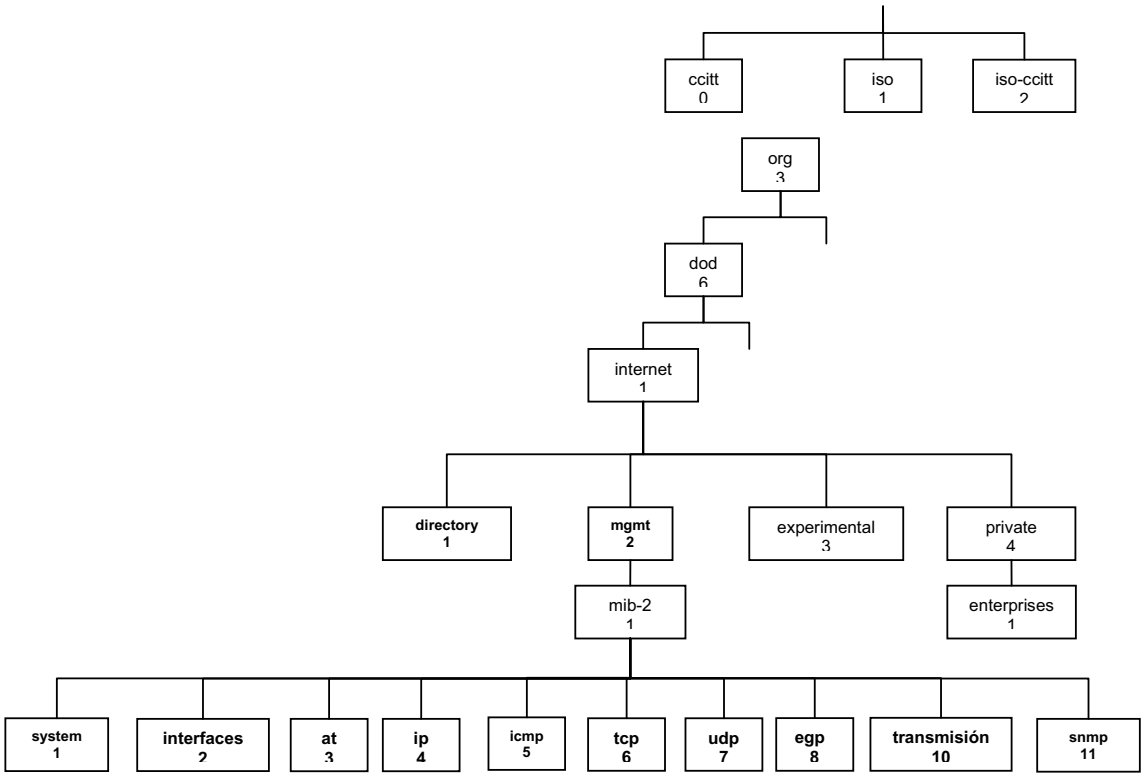


Figura 2.3.- Estructura de la MIB

Por ejemplo, el objeto sysName nos informa sobre el nombre del sistema, y puede ser alcanzado con la siguiente ruta: 1.3.6.1.2.1.1.5.

Hay que resaltar la existencia del nodo enterprises (1.3.6.1.4.1). Bajo este nodo, cada fabricante puede incorporar su propio subárbol que permita gestionar sus dispositivos.

Otro nodo especial es el nodo con etiqueta experimental (1.3.6.1.3), ya que bajo este nodo se colocan aquellos objetos que están en fase de pruebas y aún no son un estándar.

2.2.3.- NOTACION DE SINTAXIS ABSTRACTA, ASN.1 (*ABSTRACT SYNTAX NOTATION ONE*)

La notación de sintaxis abstracta, ASN.I (*Abstract Syntax Notation One*) es un lenguaje para la definición formal de tipos de datos desarrollado por ISO. El uso de ASN.I a la hora de definir las estructuras de datos permite [1,2]:

- Obtener definiciones claras, no ambiguas y uniformes a lo largo de toda la especificación.
- Reutilizar definiciones anteriores.
- Separar la definición de las estructuras de datos con la codificación física final que será la usada para la transmisión de los datos a través de la red.

SNMP hace uso de ASN.l para describir la estructura de los objetos del MIB. La definición de tipos mediante ASN.l se apoya en las siguientes estructuras (véase tabla. 2.1 a 2.5):

- Tipos simples que representan directamente valores numéricos, de cadena, lógicos.., etc.
- Tipos estructurados que se apoyan en los tipos simples para construir estructuras más complejas.
- Etiquetas que añaden información adicional a los tipos para facilitar su codificación.
- Módulos que agrupan definiciones de tipos.

Las siguientes tablas muestran los tipos con los que se cuenta [1,5]:

Tabla 2.1.- Tipos Básicos

Etiqueta	Nombre	Descripción
UNIVERSAL 1	Bolean	Verdadero o falso
UNIVERSAL 2	Integer	Numeros enterospositivos o negativos, sin restricción en el numero de digitos
UNIVERSAL 3	Bit String	Secuencia de bits, con posibilidad de hacer referencia a cada bit por su posición
UNIVERSAL 4	Octet String	Secuencia de octetos
UNIVERSAL 9	Real	Numero real
UNIVERSAL 10	Enumerated	Tipo enumerado

Tabla 2.2.- Cadenas de caracteres

Etiqueta	Nombre	Descripción
UNIVERSAL 22	IA5String	Alfabeto internacional 5
UNIVERSAL 26	VisibleString	Subconjunto de IA5 de 95 caracteres, unicamente se encuentran los caracteres visible y el espacio
UNIVERSAL 19	PrintableString	Subconjunto de IA5 de 74 caracteres, unicamente se encuentran algunos caracteres especiales
UNIVERSAL 18	NumericString	Subconjunto de IS5 compuesto de los digitos de 0 a 9 y el espacio
UNIVERSAL 25	GraphicString	Extensión de IA5 que añade caracteres graficos
UNIVERSAL 27	GeneralString	Extensión de IA5 que añade caracteres graficos y de control
UNIVERSAL 20	TeletexString	Subconjunto teletex del CCITT
UNIVERSAL 21	VideotexString	Subconjunto videotex del CCITT

Tabla 2.3.-Varios

Etiqueta	Nombre	Descripción
UNIVERSAL 5	NULL	Valor nulo
	ANY	Equivale a cualquier tipo
UNIVERSAL 8	EXTERNAL	Para incorporar tipos externos que no han sido definidos en ANS.1
UNIVERSAL 23	Hora UTC	Fecha en formato YYMMDDHHMMSS
UNIVERSAL 24	Hora General	Fecha en formato YYYYMMDDHHMMSS.S

Tabla 2.4-Objetos

Etiqueta	Nombre	Descripción
UNIVERSAL 6	OBJECT IDENTIFIER	Cadena de números que identifica a un objeto
UNIVERSAL 7	Object Designer	Texto informativo de un objeto

Tabla 2.5.-Constructores

Etiqueta	Nombre	Descripción
	CHOICE	Lista de tipos alternativos
UNIVERSAL 16	SEQUENCE SEQUENCE OF	Lista ordenada de identificadores de tipo. En caso de usar los mismo tipos se utiliza SEQUENCE Of sino SEQUENCE
UNIVERSAL 17	SET SET OF	Lista en la que no importa el orden de identificadores de tipos. Si todos tienen el mismo tipo se usa SET OF, en otro caso SET

2.2.3.1.- EJEMPLO DE CODIFICACIÓN EN ASN.1

Un ejemplo de codificación es:

```
Thost ::= SEQUENCE {
        Nombre      VisibleString,
        dirIP       NumericString,
        dirMAC      NumericString,
        Infomacion  VisibleString,
}
```

En este ejemplo se utilizan dos identificadores nuevos: OPTIONAL y DEFAULT Con OPTIONAL se puede indicar que un campo es opcional y con DEFAULIT es posible asignar un valor por defecto a un campo.

Se puede comprobar en las tablas, que todos los tipos excepto ANY y CHOICE tienen asignada una etiqueta. Hay cuatro tipos de etiquetas:

- UNIVERSAL. Identifican a los tipos de datos standard de ASN.1
- APPLICA TION: Etiquetas definidas en una aplicación normalizada
- PRIV ATE: Etiquetas definidas dentro del contexto de una aplicación particular.
- CONTEXTO: Etiquetas sin nombre utilizadas para resolver ambigüedades.

```
Ejemplo
Tejemplo ::= SEQUENCEI{
        Campo1      INTEGER OPTIONAL,
        Campo2      INTEGER OPTIONAL
}
```

En este caso campo1 y campo2 son ambos opcionales, por lo que no es posible identificar cuál de los dos se trata en el caso de que en un mensaje apareciese uno de ellos. Para resolver la ambigüedad se añaden etiquetas de contexto:

```
Tejemplo :: = SEQUENCE {
        CAMPO1          [0]     IMPLICIT INTEGER OPTIONAL
        CAMPO2          [1]     IMPLICIT INTEGER OPTIONAL
        }
```

Se ha incorporado una nueva palabra reservada: IMPLICIT. Con ella se indica que al realizar la codificación de este tipo no se codifique el tipo del campo, en este caso INTEGER, de esta manera se indica el campo por la etiqueta y, si se sabe el campo es, se deduce su tipo. La razón por la cual se realiza este tipo de codificación es debido a que el resultado que se obtiene es una codificación más compacta y, por lo tanto, un ahorro de bytes. Hay que confiar en que la aplicación receptora lleve a cabo, de manera correcta, la asociación Campo-Tipo de datos al identificarlo por su etiqueta. Por esta misma razón los tipos ANY y CHOICE nunca podrán usar IMPLICIT ya que, al no llevar etiquetas, la aplicación receptora del mensaje nunca podría deducir el tipo.

Una vez que se ha definido de manera abstracta la información, es necesario codificarla para que sea enviada por la red, esto se lleva a cabo mediante una sintaxis de transferencia o también conocida como BER (*Basic Encoding Rules*).

2.2.3.2.- SINTAXIS DE TRANSFERENCIA

ASN.1 propone una sintaxis de transferencia en la que todos los valores se codifican siguiendo el siguiente formato (véase figura. 2.4):

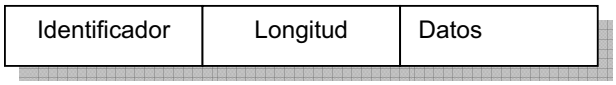


Figura 2.4.- Sintaxis de referencia.

IDENTIFICADOR

Identifica los datos que siguen a continuación. El formato del identificador es el siguiente(véase figura. 2.5):

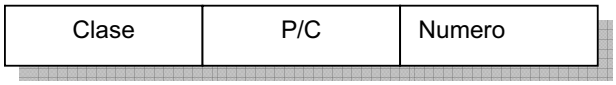


Figura 2.5.- Identificador.

El campo clase son 2 bits e indica el tipo de etiqueta. puede ser cualquiera de los cuatro tipos vistos, su codificación es la siguiente:

Codificacion	Descripción
00	Universal
01	Aplicación
10	Contexto
11	Privada

Figura 2.6.- Identificador

El campo P/C es un bit e indica si el contenido es primitivo, es decir, si se trata de un dato final, o es un constructor. Un valor de 1 indica que es un tipo primitivo, un valor de 0 indica que es un constructor.

El campo NÚMERO identifica una etiqueta concreta dentro de la clase indicada. El principio se cuenta con 5 bits para codificar el número de etiqueta, lo que proporciona un rango del 0 al 63. Si se necesita un número mayor se pueden usar más bytes pro codificar el número de etiqueta con el siguiente formato (véase figura. 2.7):

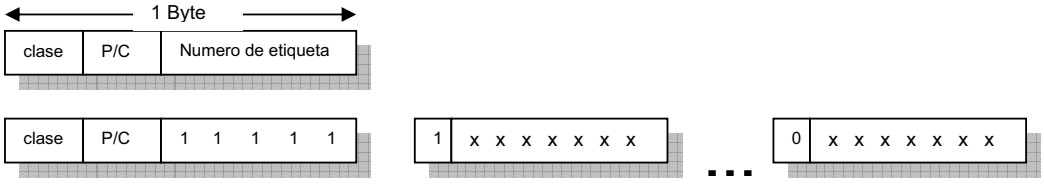


Figura 2.7.- Etiqueta

LONGITUD

La longitud de los datos indica cuántos bytes vienen a continuación, es decir, cuánto espacio ocupa la información que se envía (el contenido del campo que se es codificando). Se puede indicar una longitud concreta, o bien se puede indicar una longitud indefinida, marcando el final de los datos con dos bytes a 0 (véase figura. 2.8):

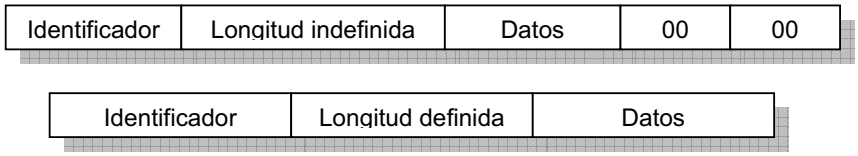


Figura 2.8.- formato longitud

Se puede comprobar que indicar la longitud es mucho más aconsejable, ya que se ahorran dos bytes en cada campo.

Hay tres formatos diferentes para codificar el campo de longitud (véase figura. 2.9):

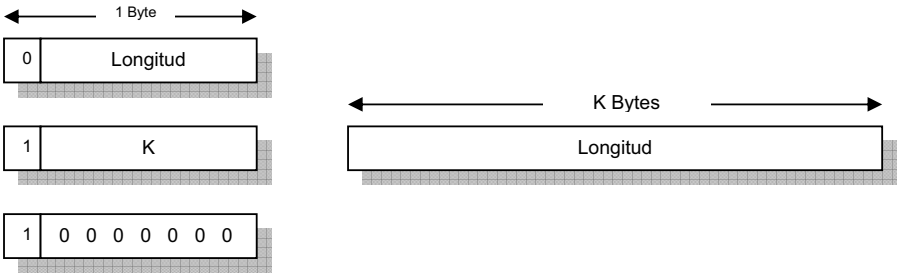


Figura 2.9.- Formatos de codificación

2.2.3.3.-MENSAJES SNMP

Un mensaje está compuesto por un identificador de versión, un nombre de comunidad y una PDU (*Protocol Data Unit*). La definición de un mensaje SNMP en ASN.1 es la siguiente [1,2]:

```
Message ::= SEQUENCE {
version INTEGER
community OCTET STRING,
data ANY
}
```

Se definen cinco unidades de datos de protocolo, PDU (*Protocol Data Unit*):

- GetRequest
- GetNextRequest
- GetResponse
- SetRequest
- Trap

que se pueden definir de la siguiente manera:

```
PDUs :: = CHOICE {
get-request:              GetReques t-PDU ,
get-next-request          GetNextRequest-PDU,
get-response    GetResponse-PDU,
set-request               SetRequest-PDU,
trap                      Trap-PDU
}
```

A continuación se describen cada una de las PDUs que pueden aparecer en un intercambio de mensajes entre un gestor y un agente. Antes de ello es conveniente definir algunos de los tipos que se verán en la definición de las PDUs:

```
RequestID ::= INTEGER

ErrorStatus :: = INTEGER {
noError(O) ,
tooBig(l) ,
noSuchName(2) ,
badValue(3) ,
readOnly(4) ,
genErr(5)
}

ErrorIndex ::= INTEGER

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

VarBindList ::= SEQUENCE OF VarBind
```

RequestID, es un identificador que irá en cada mensaje y que le permitirá a una aplicación asociar un mensaje de respuesta que le llegue con una petición que haya realizado.

ErrorIndex, indica qué variable de la lista de variables que se esté manejando fue la que causó un error. Dicho error viene identificado por ErrorStatus.

VarBind, hace referencia a un nombre de variable y su valor.

VarBindList, es una lista de variables con sus correspondientes valores.

2.2.3.3.1.- GETREQUEST

```
GetRequest-PDU ::= [0] IMPLICIT SEQUENCE {
request-id                RequestID,
error-status              ErrorStatus,
error-index               ErrorIndex,
variable-bindings         VarBindList
}
```

Con esta PDU se puede solicitar el valor de una o varias variables. Las variables de las cuales se requiere conocer su valor se listan en variable-bindings. Como respuesta se recibe una PDU de tipo GetResponse, con los valores de las variables solicitadas establecidos en variable-bindings o en caso de que hubiese algún error éste se identificaría con error-index para saber qué variable falló, y error-status para saber cuál fue el fallo. El campo request-id de la PDU GetResponse tendrá el mismo valor que en GetRequest, de esta manera la aplicación puede asociar la respuesta con la petición [2].

Posibles errores:

- En la petición se hace referencia a un nombre de variable que el receptor no conoce. En este caso el receptor indicaría en error-index qué variable causó el error, indicando en error-status "noSuchName".
- Si cuando se recibe una respuesta, ésta es demasiado grande para el sistema local, se devolverá el mensaje de respuesta con error-index puesto a 0 y error-status establecido a "tooBig".
- Si el agente no puede obtener el valor de una variable por alguna razón distinta a las previstas por el protocolo, entonces enviará el mensaje de respuesta con el campo error-index apuntando a la variable que causó el error y el campo error-status establecido a "genErr".

2.2.3.3.2.- GETNEXTREQUEST

```
NextRequest-PDU ::= [1] IMPLICIT SEQUENCE  {
request-id                RequestID,
cerror-status             ErrorStatus,
ror-index                 ErrorIndex,
riable-bindi-ngs          VarBindLfst
}
```

Con esta PDU se solicita el valor de la siguiente variable a la indicada o indicadas, suponiendo un orden léxico. Pueden darse las siguientes situaciones de error:

- No hay un sucesor léxico para alguna variable de las indicadas en variable-bindings. En este caso se devuelve en error-status el valor "noSuchName" y error-index indicará qué nombre de variable falló.
- La respuesta recibida es demasiado grande, como en la PDU anterior se devolverá la respuesta con el campo error-status indicando "tooBig" y error-index a 0.
- No se puede obtener el valor de la variable sucesora a alguna de las indicadas en variable-bindings. Se enviará la respuesta con error-index indicando qué variable

2.2.3.3.3.- GETRESPONSE

```
GetResponse-PDU ::= [2] IMPLICIT SEQUENCE {
request-id                  RequestID,
error-status                ErrorStatus,
error-index                 ErrorIndex,
variable-bindings           VarBindList
}
```

Esta PDU se genera como respuesta a las PDUs de tipo GetRequest, GetNextRequest y SetRequest.

2.2.3.3.4.- SETREQUEST

```
SetRequest-PDU ::= [3] IMPLICIT SEQUENCE {
request-id                  RequestID,
error-status                ErrorStatus,
error-index                 Error Index ,
variable-bindings           VarBindList
}
```

Con esta PDU se solicita el establecimiento del valor de la variable o variables que indiquemos en variable-bindings. En caso de no haber errores el receptor devuelve una PDU de tipo Response con el campo error-status establecido a NoError y error- index a 0.

Pueden darse los siguientes errores:

- El receptor no encuentra alguna variable de las indicadas en variable-bindings, en este caso indica en la respuesta el error "noSuchName" en el campo error-status e, indica qué variable falló en error-index.
- Si se intenta establecer un valor que no es correcto de acuerdo al tipo de la variable, entonces el receptor devolverá en la PDU de respuesta el error "badValue" en error-status e indicará qué variable falló en error-index. "
- Si al recibir la respuesta se ve que es demasiado grande y que no se puede tratar, entonces se devuelve con el error "tooBig" en error-status y error-index a O.
- Si ocurre algún error no especificado en el protocolo se devolverá una PDU de respuesta con el valor "genErr" en el campo error-status y un 0 en error-index.

2.2.3.3.5.- TRAP

```
Trap- PDU :: = [4] IMPLICIT SEQUENCE  { ,
Enterprise                  OBJECT IDENTIFIER,
agent-addr                  NetworkAddress,
generic-trap                INTEGER {
                                    coldStart (O),
                                    warmStart(l),
                                    authenticationFailure(4),
                                    egpNeighborLoss(5),
                                    enterpriseSpecific(6)
                                    } ,
specific-trap       INTEGER,
time-stamp          TimeTicks,
variable-bindings   VarBindList
}
```

Las PDU de tipo Trap permiten a los agentes comunicar de manera asíncrona a los gestores cualquier evento que haya sucedido al objeto gestionado y en el cual el gestor tiene interés de ser informado.
El campo generic-trap indica qué un evento ha ocurrido (véase figura. 2.10):

Evento	Significado
coldStart	El dispositivo se ha reiniciado, con lo que que la configuración del agente ha podido cambiar
WarmStart	El dispositivo se ha reiniciado, pero el agente sigue intacto
LinkDown	El dispositivo ha detectado un fallo en uno de sus enlaces de red. El enlace que falla es especificado en el campo variable-bindings
LinkUp	El dispositivo ha detectado que uno de sus enlaces con la red se ha activado. El nombre del enlace y el valor de la variable ifIndex aparecen en el campo variable-bindings
AuthenticationFailure	El agente ha detectado un fallo en la autenticación del mensaje
EgpNeighborLoss	Uno de los nodos colaboradores EGP se ha caido. El primer elemento del campo variable-bindings es el nombre y valor de la variable egpNeighAddr del nodo afectado
EnterprisesSpecific	Evento de un fabricante particular. El evento se identifica con el campo specific-trap

Figura 2.10.- Eventos asociados a SNMP

Los únicos campos que quedan sin comentar son agent-addr, que especifica la dirección de red del objeto que está generando la notificación, y time-stamp que indica el tiempo transcurrido desde la última inicialización de la entidad de red y la generación de la notificación.

3.- DOCUMENTACIÓN

La documentación básica de SNMP está descrita en siete *Request for Comments(RFCs)* publicados en Abril de 1999 y en Marzo de 2000. Estas RFCs se listan en la Tabla 2.6.

Tabla 2.6.- RFCs Asociadas a SNMP.

RFC	RFCS DE SNMPV3	FECHA	CATEGORÍA
RFC 2570	*Introduction to Version 3 of the Internet-standard*	Abril 1999	Informativo
RFC 2571	*An Architecture for Describing SNMP, Management Frameworks*	Abril 1999	Borrador de estándar
RFC 2572	*Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)*	Abril 1999	Borrador de estándar
RFC 2573	*SNMP Applications*	Abril 1999	Borrador de estándar
RFC 2574	*User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)*	Abril 1999	Borrador de estándar
RFC 2575	*View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)*	Abril 1999	Borrador de estándar
RFC 2576	*Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework*	Marzo 2000	Borrador de estándar

RFC 2570 [3] presenta una visión de conjunto sobre SNMP. Describe las relaciones entre las especificaciones de SNMP y las especificaciones de SNMPv1, SNMPv2 y SNMPv2c, SNMPv·3. También proporciona una guía de los documentos con especificaciones relevantes, y un resumen del contenido de cada uno de ellos.

RFC 257.- Define un vocabulario para los marcos de trabajo de administración de SNMP(que son SNMPv1, SNMPv2 y SNMPv3), así como una arquitectura modular común para la mayor parte deesos marcos de trabajo.

RFC 2572.- Describe el despacho *(dispatching)* y proceso de mensajes SNMP. Se definen los procedimientos para despachar mensajes SNMP de varias versiones al correspondiente modelo deproceso de mensajes, y para despachar *PDUs* (Unidades de Datos de Protocolo) a las aplicaciones SNMP. También describe el Modelo de Proceso de Mensajes de SNMPv3.

RFC 2573.- Describe cinco tipos de aplicaciones que usan el motor SNMP descrito en RFC 2571. Los tipos de aplicaciones son: generadores de comandos, respondedores de comandos, originadotes de notificaciones, receptores de notificaciones, y *proxy forwarder*s. También define los módulos MIB para especificar destinos de las operaciones de administración, para filtrado de notificaciones ,y para el uso de *proxy forwarder*s.

RFC 2574.- Describe el Modelo de Seguridad basado en Usuario (USM) para SNMPv3 y especifica procedimientos para proporcionar niveles de seguridad a los mensajes SNMP. También incluye una MIB para administración y monitorización de los parámetros de configuración USM.

RFC 2575.- Describe el Modelo de Control de Acceso basado en Vistas (VACM), y define los elementos del procedimiento para controlar el acceso a la información de administración. También incluye una MIB para administración remota de los parámetros de configuración.

RFC 2576.- Describe la coexistencia entre los tres marcos de trabajo de administración: **SNMPv1**, **SNMPv2** y **SNMPv3**.

RESUMEN CAPITULO II. SNMP(SIMPLE NETWORK MANAGEMENT PROTOCOL).

SNMP (Simple Network Management Protocol) es un protocolo que permite la gestión de los recursos que están disponibles en una red.

Dentro de un entorno de red gestionado con SNMP habrá un conjunto de nodos de la red que se encarguen de la gestión y un conjunto de componentes de la red (hosts, concentradores, ruteadores, modems, etc.) que podrán ser gestionados por estas estaciones.

El agente de gestión se encarga de supervisar un elemento de la red. Se comunica con el gestor para atender sus peticiones y para informarle de eventos acaecidos en el objeto gestionado. El agente de gestión suele residir físicamente en el elemento gestionado.

El gestor es un software residente en una estación de gestión que se comunica con los agentes y que ofrece al usuario una interfaz a través de la cual comunicarse con los agentes de gestión para obtener información de los recursos gestionados. Además recibirá las notificaciones enviadas por los agentes.

Los objetos gestionados; son las abstracciones de los elementos físicos de la red que se gestionan (tarjeta de red, concentrador, módem, ruteador, etc.). Se pueden manejar los atributos y las operaciones que se pueden realizar sobre el objeto. De la misma forma, las notificaciones que dicho objeto puede generar así como las relaciones con otros objetos son susceptibles de ser controladas. La base de datos de gestión (MIB) previamente mencionada está formada por todos los objetos gestionados.

Protocolo de gestión, es el protocolo que especifica cómo se realizará la comunicación entre los agentes de gestión y el gestor. En nuestro caso este protocolo es el SNMP. La comunicación se realiza en base a requerimientos, respuestas y notificaciones.

La Herramienta de Gestión de Redes Virtuales, usa SNMP como medio para configurar los parámetros asociados a los equipos, así también, cada una de las primitivas asociadas permite obtener determinada información del equipo que es necesaria para formarse un criterio de decisión, acerca del funcionamiento de la red.

CAPITULO III
HERRAMIENTA DE GESTIÓN DE REDES VIRTUALES.

INTRODUCCIÓN

La herramienta de Gestión de redes virtuales se basa en los fundamentos teóricos de la gestión de redes particularmente los aspectos de monitorización (lectura) y control (escritura), así como en el análisis de protocolos principalemente a nivel de capa de red; con respecto a la configuración se retoma el estándar de facto SNMP y las primitivas que el protocolo define para obtener la información de los objetos getionados. Con respecto a la monitorización, la herramienta permite observar, analizar el estado y el comportamiento de la configuración de red y sus componentes. En lo correspondiente al control, permite alterar parámetros de los componentes de la red. Para cumplir con estos fines la herramienta se integra de forma modular de tal manera que sea posible incorporar elementos de programación sin afectar a los que ya estan en buen funcionamiento. Algunos de los Módulos son: SNMPGET, SNMPGET-NEXT, SNMPSET, SNMPTRAPS, SNMPSTATISTICS, SNMPVIEWER y ANALIZADOR. El capitulo trata, en consecuencia, con el desarrollo de la Herramienta de Gestión de Redes Virtuales, se profundiza en los componentes modulares, la explicación a detalle de la codificación asociada a cada uno de estos, así como también, la explicación referente a el funcionamiento de cada una de las librerías que se emplearon en el desarrollo de la herramienta.

3.1 LA HERRAMIENTA DE GESTIÓN DE REDES VIRTUALES (ALCANCES).

Los alcances de la herramienta estan relacionados con la monitorizaición y el control. En lo que respecta a la información que monitoriza está se encuentra en una de las siguientes categorías: **1.- Estática**: **2.- Dinámica 3.- Estadística.** Esto conlleva a los objetivos de: a) **Identificación de la información**; b) **Diseño de mecanismos de monitorización** y c) **Utilización de la información.**

La herramienta monitoriza diferentes recursos de red en los que destacan:

- **Recursos de comunicación**: LANs, WANs
- **Hardware de computación**: Servidores, estaciones de trabajo, dispositivos de conectividad tales como ruteadores, conmutadores, concentradores, etc.

Para cumplir con esta función, la de monitorizar, la herramienta emplea el análisis de protocolos y analiza a detalle el datagrama IPv4 en cada uno de sus elementos de interés: versión, largo del encabezado, tipo de servicio, largo del datagrama, identificador del paquete, banderas, ofsett de fragmento, tiempo de vida, protocolo de capa 4, checksum de encabezado, dirección IP origen, dirección IP Destino opciones y datos.

EN lo que respecta al control, la herramienta incorpora un estándar de facto para tal objetivo, el SNMP, para lo cual la herramienta incorpora las primitivas de : Get, Get-Next, Trap, Set. Con los correspondientes módulos SNMPGET, SNMPSET, SNMPTRAP, SNMPSTATISTICS, SNMPGETNEXT y SNMPVIEWER.

Estas primitivas leen la base de datos asociada al objeto gestionado, la MIB (*Management Information Base*), esta base de objetos estructurada en forma arbórea permite identificar ciertos

objetos de interés por ejemplo numero de interfaces y sus correspondientes estados, entre otros. Por lo que la herramienta proporciona un mecanismo de lectura de la MIB en formato de Identificación de Objeto, OID (*Object Idetification*) o en texto.

Al modificar un atributo la información de configuración de un **agente cambia**. Las modificaciones que la proporciona son:**1.- Actualización de la base de datos; 2.- Actualización de la base de datos que implica modificación del recurso y 3.- Actualización de la base de datos que implica una acción**

En cuanto a la infromación estadística la herramienta permite obtener datos realcionados a las siguientes categorias:

- **SYSTEM**: Informaciones sobre el equipamiento.
- **INTERFACES**: Informaciones sobre las interfases del equipamiento.
- **IP**: información sobre el protocolo IP.
- **TCP**: Informaciones sobre el protocolo TCP.

Además de que con una interfaz gráfica amigable y menús de opciones se facilita la seleccion las de las opciones correspondientes dependiendo de la función a realizar: monitorizar y/o el control.

3.2.- DESCRIPCIÓN TÉCNICA

Atendiendo a la necesidad de gestión de la red y a las areas de monitorización y control, la herramienta de gestión de redes virtuales consta de los siguientes programas:

1.- Programa de configuración/Reconfig de la red virtual
2.- Programa de monitoreo e informe de la disponibilidad y la utilización de la red

3.2.1.- PROGRAMA DE CONFIGURACIÓN DE LA RED VIRTUAL.

Dado que los sistemas distribuidos y las redes tienen un carácter abierto, es necesario definir una arquitectura de gestión de red para el programa de configuración. Esta arquitectura permite la gestión de los elementos heterogéneos de múltiples proveedores. **La arquitectura que se emplea se basa en el estándar de facto que opera sobre TCP/IP, el SNMP *(Simple Network Management Protocol).***

3.2.1.1.- PROTOCOLO DE ADMINISTRACIÓN DE RED SIMPLE, SNMP *(SIMPLE NETWORK MANAGEMENT PROTOCOL).*

El protocolo de administración de red simple, SNMP (*Simple Network Management Protocol*), es un protocolo que permite la gestión de los recursos que están disponibles en la red.

Dentro de un entorno de red gestionado con *SNMP* hay un conjunto de nodos de la red que se encargan de la gestión y un conjunto de componentes de la red (hosts, concentradores, ruteadores, modems, etc.) que pueden ser gestionados por estas estaciones.

Para que el software que se encarga de la gestión de la red en las estaciones de gestión pueda obtener información de los elementos de la red, es necesario que dichos elementos cuenten con un software que permita su comunicación con la estación de gestión. Este software se denomina agente *SNMP*.

Por último hay otra pieza importante en este entorno, y es la base de datos dónde se almacena toda la información que se gestiona. Esta base de datos se denomina **MIB *(Management Information Base)***. La figura 3.1 muestra la interacción de estos elementos.

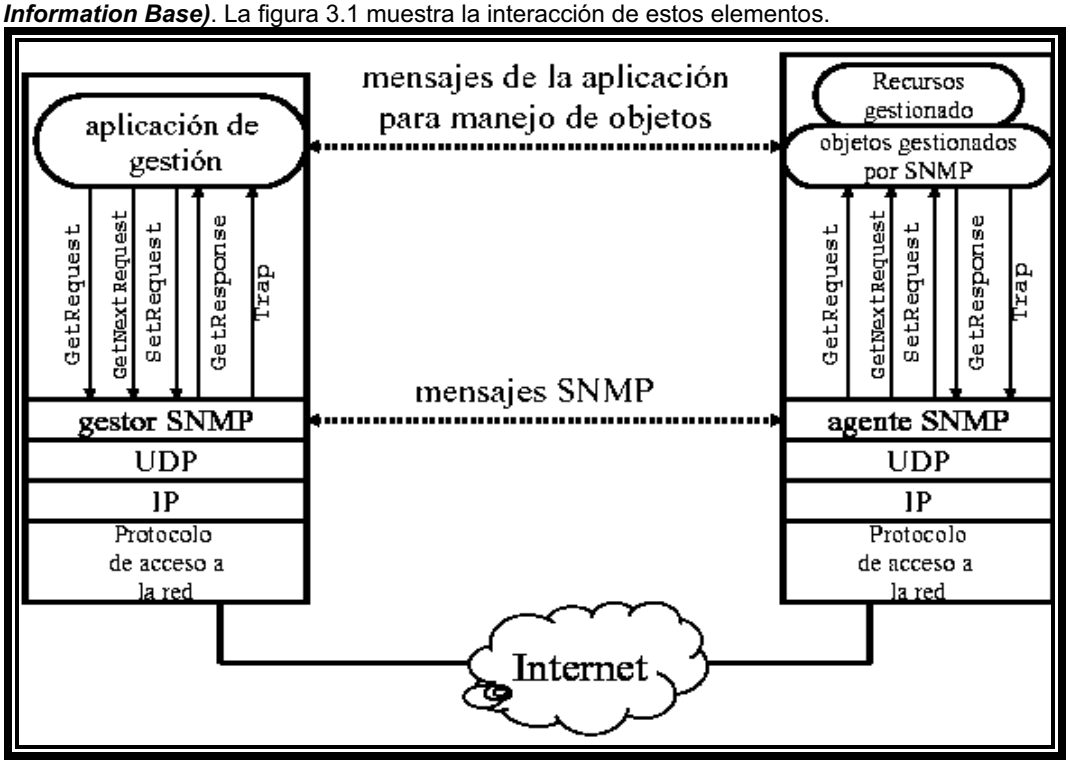


Figura 3.1. Elementos dentro de un entorno gestionado por *SNMP*.

3.2.1.2.- ESTRUCTURA DEL PROGRAMA DE CONFIGURACIÓN PARA REDES VIRTUALES

Si se desea construir un programa que se adapte a los elementos dentro de un entorno gestionado por *SNMP*, el programa debe de estar formado de acuerdo a la RFC 2571, RFC 2572 y RFC 2573 por cierta cantidad de componentes (como se muestra en la figura 3.2). En ésta figura, la parte sombreada de gris es lo que se implementó en el programa de configuración.

- **AGENTE DE GESTIÓN.**
  Se encarga de supervisar un elemento de la red. Se comunica con el sistema gestor para atender las peticiones y para informarle de eventos sucedidos en el objeto gestionado, **éste agente de gestión es parte del dispositivo**.
- **GESTOR.**
  El gestor es el software residente en una estación de gestión que se comunica con los agentes y que ofrece al usuario una interfaz a través de la cual comunicarse con los elementos de gestión para obtener información de los recursos gestionados. Además recibe las notificaciones enviadas por los agentes, éste componente se programa y forma parte de del programa de configuración.
- **OBJETOS GESTIONADOS**
  Los objetos gestionados son las abstracciones de los elementos físicos de la red que se gestionan (tarjeta de red, hub, módem, router, etc.). Se pueden manejar los atributos y las operaciones que se puedan realizar sobre el objeto. De la misma forma, las

notificaciones que dicho objeto puede generar así como las relaciones con otros objetos de la red también son susceptibles de ser controladas.
La base de datos de gestión (*MIB*) previamente, mencionada esta formada por todos los objetos gestionados, **ésta *MIB* es parte del dispositivo gestionado.**

- **PROTOCOLO DE GESTIÓN.**
  Es el protocolo que específica como se realiza la comunicación entre los agentes de gestión y el gestor. **Es nuestro caso este protocolo es el *SNMP*, por ser el más conocido y usado actualmente.**

3.3.- LOS MANDATOS *SNMP* EN EL PROGRAMA DE CONFIGURACIÓN

Una de las claves de la flexibilidad del *SNMP* es el uso de "variables" como forma de representación de los recursos, tanto físicos como lógicos, en los sistemas gestionados.

3.3.1.- MANDATOS *SNMP*.

En cada nodo gestionado, el agente *SNMP* proporciona una base de datos llamada *MIB (Management Information Base)*, que contiene objetos de datos, más conocidos como variables MIB. La monitorización del nodo la lleva a cabo la estación gestora, la cual, periódicamente, lee los valores de estas variables. El control del nodo se realiza mediante el cambio de los valores de las variables. Adicionalmente, existe una operación ***trap*** para permitir al nodo gestionado informar una estación gestora sobre determinadas condiciones o eventos inusuales.

Los mandatos *SNMP* que emplea el programa de configuración son:

- **Get**: Enviado por la estación gestora para obtener las variables *MIB* específicas de un nodo gestionado.
- **Get-next**: Es enviado por la estación gestora a la gestionada para obtener la variable *MIB* siguiente a la gestionada.
- **Get-response**: es enviado por el nodo gestionado a la estación gestora como respuesta a mandatos get, get-next o set.
- **Set**: Enviado por la estación gestora para dar un valor determinado a una variable *MIB* de un nodo gestionado.

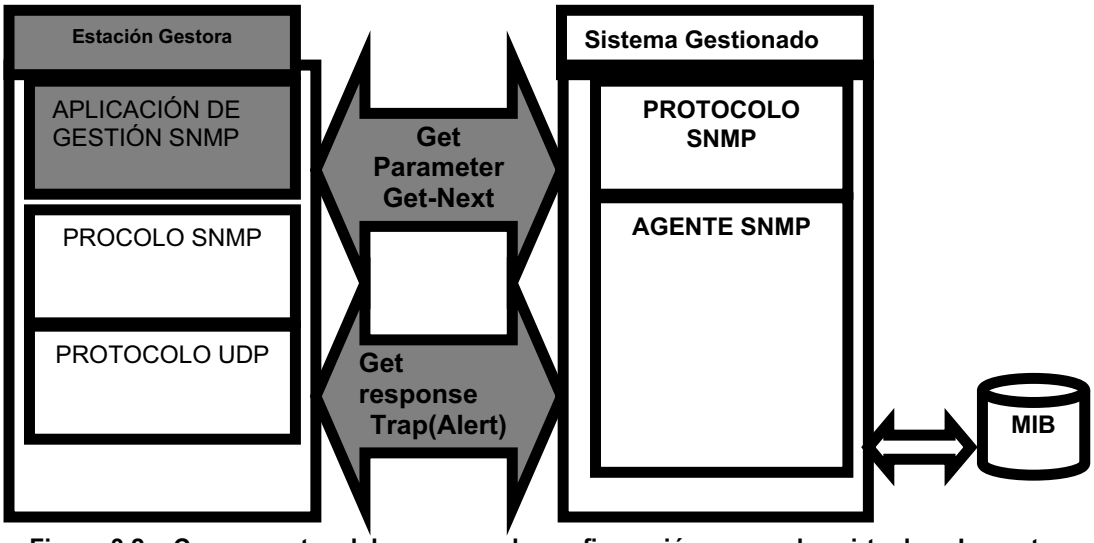


Figura 3.2.- Componentes del programa de configuración para redes virtuales. La parte sombreada representa lo que implementa el programa de configuración.

3.3.2.- MODELO DE CONFIGURACIÓN ESTACIÓN GESTORA Y SISTEMAS GESTIONADOS

El modelo final que se utiliza en el programa de configuración para cumplir su objetivo es el mostrado en la figura 3.3 y 3.4.

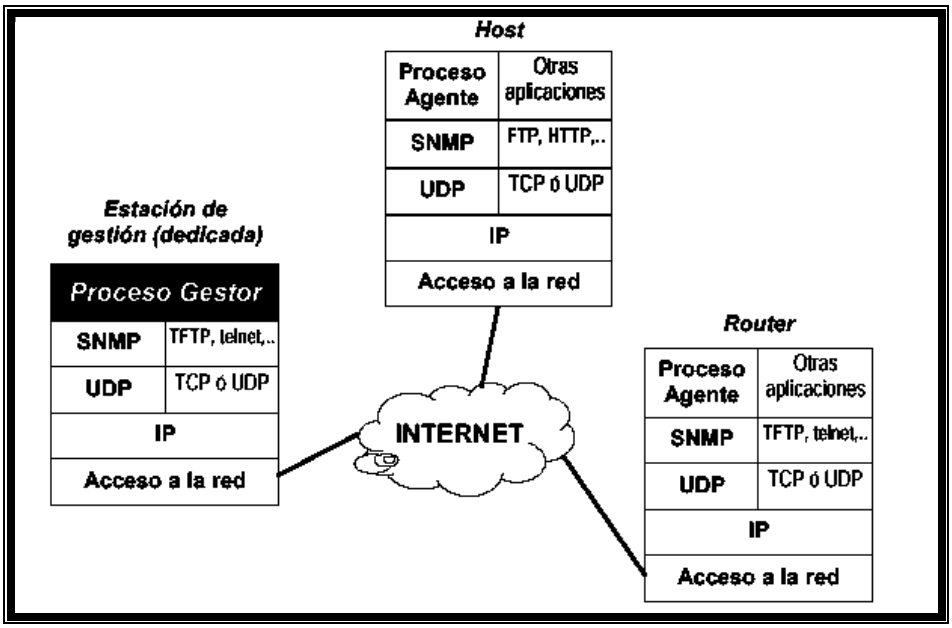


Figura 3.3.- Modelo de configuración estación Gestora y sistemas Gestionados.

La interacción entre la estación de administración de red(*NMS*) y los dispositivos administrados se puede dar bajo 4 tipos de comandos, que llevan asociadas las primitivas de gestión de *SNMP*:

- **Leer**.- Para monitorear el dispositivo administrado, la *NMS* lee las variables mantenidas por los dispositivos.
- **Escribir**.- Para controlar los dispositivos administrados, la *NMS* escribe en las variables almacenadas en el dispositivo.
- **Operaciones Transversales**.- *NMS* usa estas operaciones para determinar cuales variables son soportadas por un dispositivo.
- **Traps**.- para permitir al nodo gestionado informar a una estación gestora sobre determinadas condiciones o eventos inusuales.

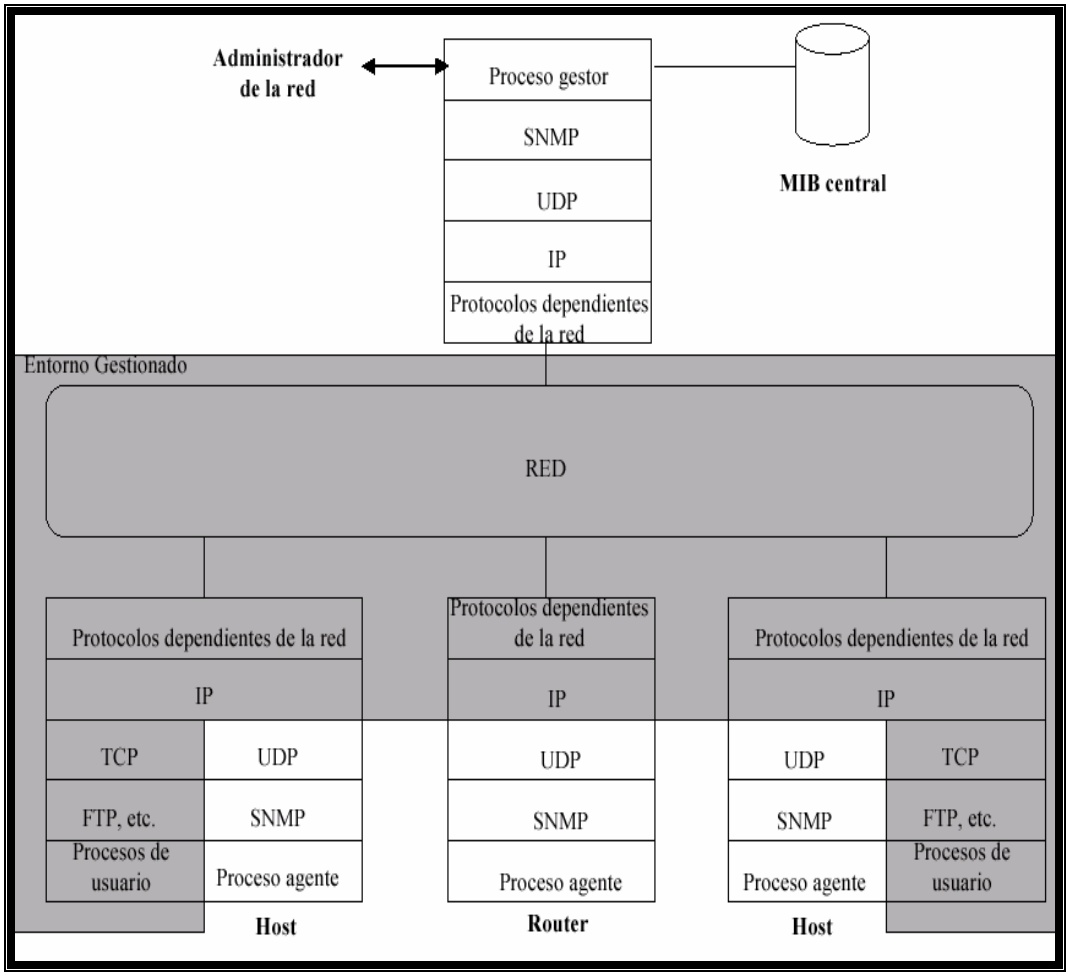


Figura 3.4.- Modelo de configuración estación SNMP Gestor y Agente.

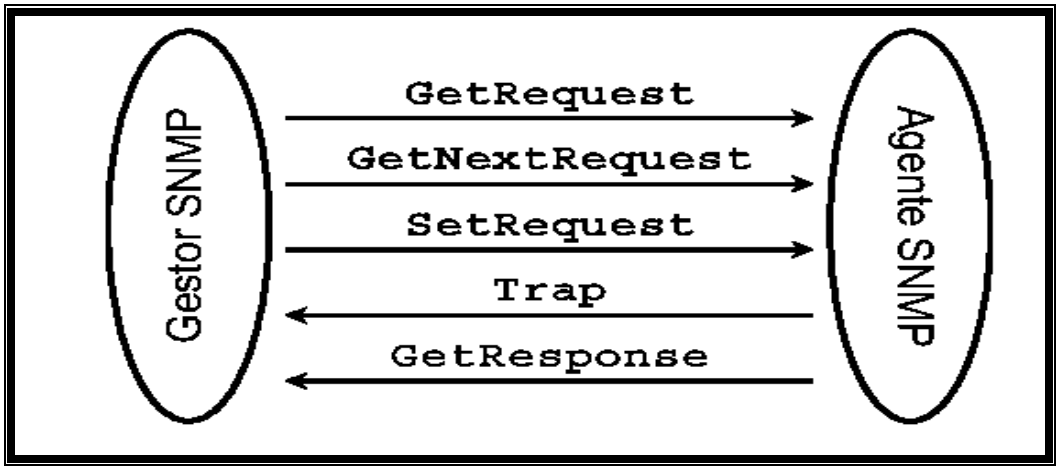


Figura 3.4.- Modelo de configuración estación SNMP Gestor y Agente(cont).

3.3.3.- ACCESO AL ÀRBOL DE VARIABLES DE LOS SISTEMAS GESTIONADOS.

El programa de configuración accede a la *MIB* del dispositivo gestionado; la *MIB* se organiza en forma de árbol; cada nodo del árbol tiene asociado un número entero y una etiqueta de texto. Un nodo se identifica unívocamente con una secuencia de números enteros que identifican los nodos a través de los cuales hay que pasar para llegar desde la raíz al nodo que nos interese (véase figura 3.5).

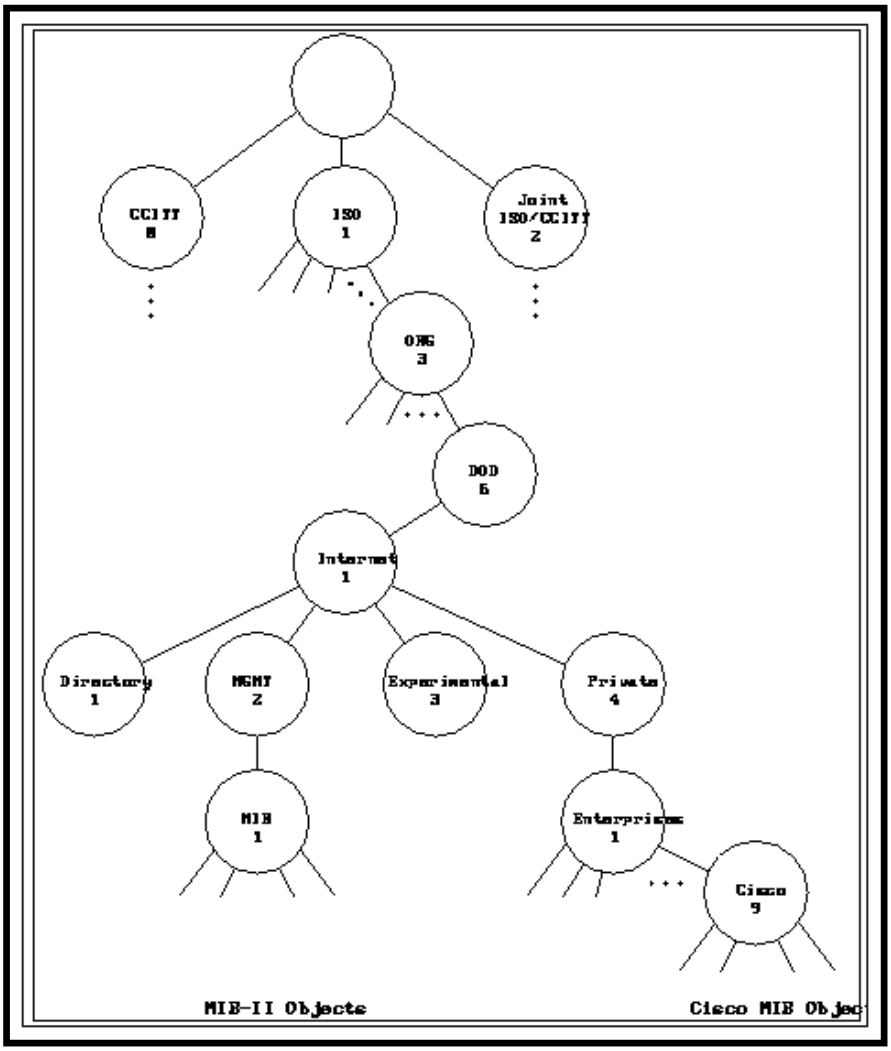


Figura 3.5.- Estructura de la MIB

Un ejemplo de este tipo de MIB's es el siguiente, se toman los puntos de definición y tipos de datos definidos en el capitulo cuatro referente a la implementación de SNMP:

```
RFC1213-MIB DEFINITIONS ::= BEGIN

        IMPORTS
              mgmt, NetworkAddress, IpAddress, Counter, Gauge,
                    TimeTicks
                 FROM RFC1155-SMI
              OBJECT-TYPE
                    FROM RFC-1212;

        -- This MIB module uses the extended OBJECT-TYPE macro as
        -- defined in [14];

        -- MIB-II (same prefix as MIB-I)

        mib-2     OBJECT IDENTIFIER ::= { mgmt 1 }

        -- textual conventions

        DisplayString ::=
           OCTET STRING
        -- This data type is used to model textual information taken
        -- from the NVT ASCII character set.  By convention, objects
        -- with this syntax are declared as having


        --
        --     SIZE (0..255)

        PhysAddress ::=
           OCTET STRING
        -- This data type is used to model media addresses.  For many
        -- types of media, this will be in a binary representation.
        -- For example, an ethernet address would be represented as
        -- a string of 6 octets.

        -- groups in MIB-II

        system      OBJECT IDENTIFIER ::= { mib-2 1 }

        interfaces   OBJECT IDENTIFIER ::= { mib-2 2 }

        at          OBJECT IDENTIFIER ::= { mib-2 3 }

        ip          OBJECT IDENTIFIER ::= { mib-2 4 }

        icmp         OBJECT IDENTIFIER ::= { mib-2 5 }

        tcp         OBJECT IDENTIFIER ::= { mib-2 6 }

        udp          OBJECT IDENTIFIER ::= { mib-2 7 }

        egp          OBJECT IDENTIFIER ::= { mib-2 8 }
……
```

3.4.- ESTRUCTURA MODULAR DEL PROGRAMA DE CONFIGURACIÓN

De está manera los módulos que integran el programa de configuración son (véase fig. 3.6):

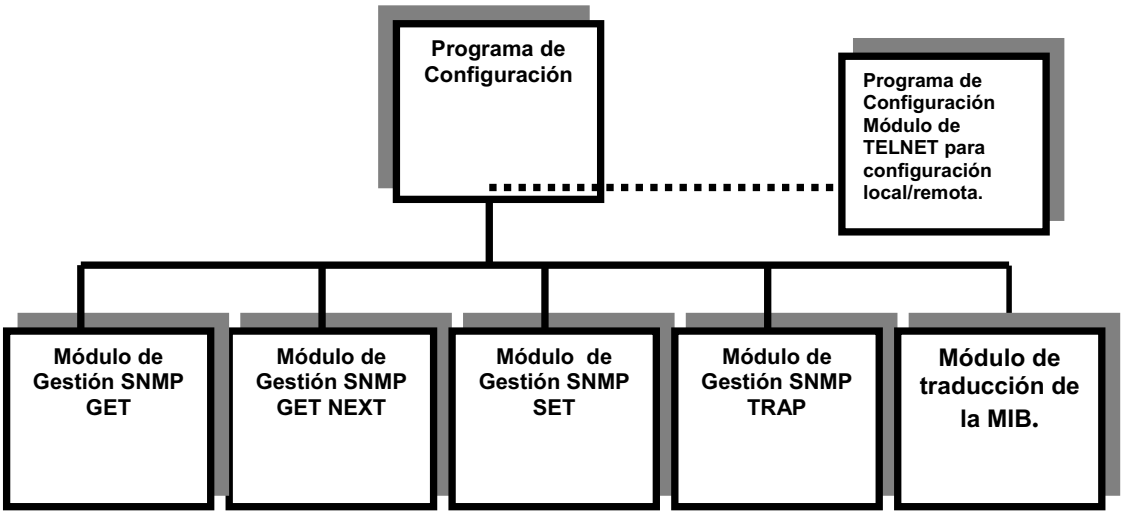


Figura 3.6.- Módulos del programa de configuración de la herramienta de gestión de redes virtuales.

3.4.1 EL MÓDULO DE GESTIÓN SNMP GET

Se encarga de leer el valor de una ó mas variables de la *MIB (Management Information Base).*

Ejemplos de uso por la herramienta de gestión de redes virtuales:

```
% snmpget 10.0.0.180 public system.sysUpTime.0
system.sysUpTime.0 = Timeticks: (586731977) 67 days, 21:48:39.77
```

donde:
10.0.0.180: Representa el nodo a monitorear
public : la comunidad
system.sysUpTime.0 : la variable dentro de la MIB a obtener

3.4.2.- EL MÓDULO DE GESTIÓN SNMP GET NEXT

Se encarga de realizar lecturas secuenciales de la *MIB.*

Ejemplos de uso por la herramienta de gestión de redes virtuales:

```
% snmpgetnext 10.0.0.180 public system.sysUpTime.0
```

3.4.2.- EL MÓDULO DE GESTIÓN SNMP SET

Se encarga de establecer el valor de una variable de la *MIB.*

Ejemplos de uso por la herramienta de gestión de redes virtuales:

```
% snmpget 10.0.0.180 public PublicString.0
% snmpget 10.0.0.180 public PublicString.0
```

3.4.3.- EL MÓDULO DE GESTIÓN SNMP TRAP

Se encarga del manejo de notificaciones de eventos.

Módulo de traducción de la MIB.- Se encarga de la traducción de variables de la *MIB.*

Ejemplos de uso por la herramienta de gestión de redes virtuales:

```
% snmptranslate system.sysUpTime.0
.1.3.6.1.2.1.1.3.0

% snmptranslate -Td -Ib 'sys.*ime'
.1.3.6.1.2.1.1.3
sysUpTime OBJECT-TYPE
  -- FROM SNMPv2-MIB, RFC1213-MIB
  SYNTAX TimeTicks
  MAX-ACCESS     read-only
  STATUS current
  DESCRIPTION    "The time (in hundredths of a second) since the
network
           management portion of the system was last re-
initialized."
::= { iso(1) org(3) dod(6) internet(1) mgmt(2) mib-2(1) system(1)
3 }

% snmptranslate -Tp system
+--system(1)
   |
   +-- -R-- String    sysDescr(1)
   |        Textual Convention: DisplayString
   +-- -R-- ObjID     sysObjectID(2)
   +-- -R-- TimeTicks sysUpTime(3)
   +-- -RW- String    sysContact(4)
   |        Textual Convention: DisplayString
   +-- -RW- String    sysName(5)
   |        Textual Convention: DisplayString
   +-- -RW- String    sysLocation(6)
   |        Textual Convention: DisplayString
   +-- -R-- Integer   sysServices(7)
   +-- -R-- TimeTicks sysORLastChange(8)
   |        Textual Convention: TimeStamp
   |
   +--sysORTable(9)
      |
      +--sysOREntry(1)
         |
```

```
+-- ---- Integer   sysORIndex(1)
+-- -R-- ObjID     sysORID(2)
+-- -R-- String    sysORDescr(3)
|        Textual Convention: DisplayString
+-- -R-- TimeTicks sysORUpTime(4)
         Textual Convention: TimeStamp
```

3.4.4.- MÓDULO DE TELNET PARA CONFIGURACIÓN

Se encarga de la configuración remota del dispositivo, en el caso de que el dispositivo no soporte la configuración vía *SNMP.*

3.5.- DESARROLLO DE LOS MODULOS ASOCIADOS A LAS PRIMITIVAS GETREQUEST, GETNEXTREQUEST, GETRESPONSE, SETREQUEST, TRAP

Con el fin de poner en práctica el uso de SNMP e incorporarlo en la Herramienta de Gestión de Redes Virtuales. Se presenta el desarrollo de la primitiva "SNMPGET". La estructura asociada a SNMP, la programación asociada a la primitiva y las sesiones de apertura; son aplicables a cada una de las primitivas de SNMP, sin embargo es importante destacar que los resultados que arrojan cada una de estas se particularizan, y hay que tomarlos con reserva, debido a que cada una de ellas cumple con una función específica dentro del protocolo SNMP. El programa permite consultar el valor de cualquier objeto que se encuentre en la base de datos MIB de un agente SNMP, con el uso de la biblioteca de funciones SNMP Windows/Linux [2].

A continuación se presenta el listado de la primitiva "SNMPGET" junto con toda la información asociada, que explica paso a paso la funcionalidad de la codificación.

```
#include <sys/types.h>
#include <snmp/snmp.h>
#include <string.h>
#include <stdio.h>

int main(int argc, char *argv[]) {
```

La estructura snmp_session permite almacenar la dirección del agente, los puertos UDP que se usan, los datos para la autenticación, etc.

```
struct snmp_session sesion;
struct snmp_session *sp;
```

La estructura snmp-pdu define una PDU indicando la dirección del destinatario, el tipo de PDU que es, la lista de variables que se manejan, los campos errindex y errstat, etc.

```
struct snmp-pdu *getpdu;
struct snmp-pdu *respdu;
```

Para la autentificación se usan comunidades de SNMP v 1. El nombre de la comunidad se torna del primer argumento.

```
,char *comunidad = argv[l];
```

El nombre o dirección lP del host donde reside el agente al cual se hace la consulta se toma del segundo argumento.

```
char *host = argv[2];
```

La variable que se desea consultar es el tercer argumento.

```
char *variable = argv[3];
cid var [MAX_NAME_LEN];
int nvar;
```

El primer paso es cargar los datos del MIB.

```
init_mib();
```

Se rellenan los datos de la sesión SNMP .

```
memset ((char *) &sesion, O, sizeof (sesioE! )-;
sesion.retries = SNMP_DEFAULT_RETRIES;
sesion.timeout = SNMP_DEFAULT_TIMEOUT;
sesion.peername = argv[2];
sesion.remote-port = SNMP_DEFAULT_REMPORT;
sesion.local-port = o;
sesion.community = argv[1];
sesion.community_len = strlen(argv[l]);
```

Se establece la sesión con los datos proporcionados anteriormente.

```
snmp_synch_setup(&sesion) ,
```

Se conecta con el agente tomando los datos de la sesión establecida.

```
if (! (sp = snmp_open(&sesion») {
fprintf(stderr, 'No se pudo abrir la sesion SNMP\n');
exit(1) ;
}
```

Se crea la PDU de petición del valor de la variable que se indica a continuación.

```
if (! (getpdu = snmp_pdu_create(SNMP_PDU_GET))) {
        fprintf(stderr, 'Error al crear la PDU\n');
        exit(1);
}
```

Se toma la variable que se quiere consultar, ya sea en formato de etiquetas (iso.org.doclntemet) o de números (1.3.6.1), y se crea un identificador de objeto valido para ser incluido en la lista de variables de la PDU.

```
nvar = MAX_NAME_LEN;
if (!read_objid(argv[3].. var, &nvar)) {
        fprintf (gtderr, 'read_objid');
        exit(1);
}
```

Se incorpora a la lista de variables de la PDU.

```
snmp_add_null_var(getpdu, var, nvar);
```

Se envía la PDU y se espera el mensaje de respuesta. Si no hay ningún error se imprime la información que llegue en la PDU de respuesta referente a la lista de variables.

```
if(snmp_synch_response (sp, getpdu, &respdu) = = STAT_SUCCES,S) {
        if (respdu->errstat == SNMP_ERR_NOERROR) {
        print_variable_list (respdu->variables);
        } else {
        fprintf(stderr, 'errstat=u, errindex=%u \n', respdu->errstat,
        respdu->errindex)
}

snmp_free_pdu(respdu);                  /* Se libera la PDU de respuesta */
else {
                fprintf(stderr, "snmp_synch_response\n');
}'

snmp_close(sp); /* Se cierra la sesion */
```

El desarrollo pude ser en Windows o en Linux, ya que las bibliotecas son portables y transparentes , con esto es posible el desarrollo de cada una de las primitivas SNMP. Para el caso que nos compete se empleo el Sistemas Operativo Windows/Linux, como lenguaje de programación se utilizó Visual C++ 6.0.(véase figura 3.7).

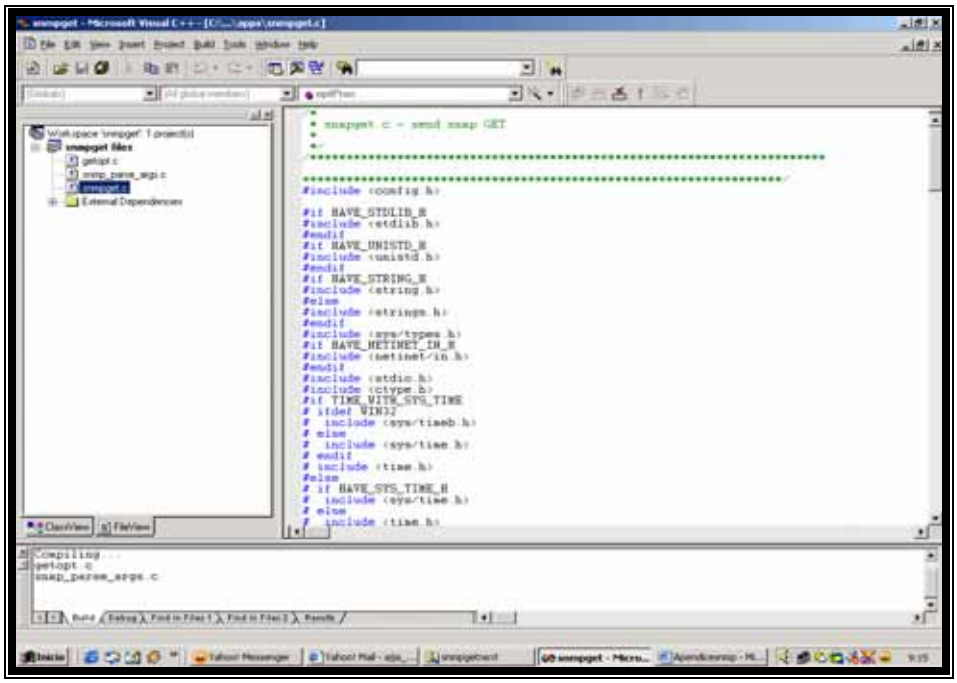


Figura 3.7.- Módulos del programa de configuración de la herramienta de gestión de redes virtuales SNMPGET.

Para mostrar en ejemplo de la compilación se presenta el caso de la primitiva SNMPGET (véase figura 3.8).

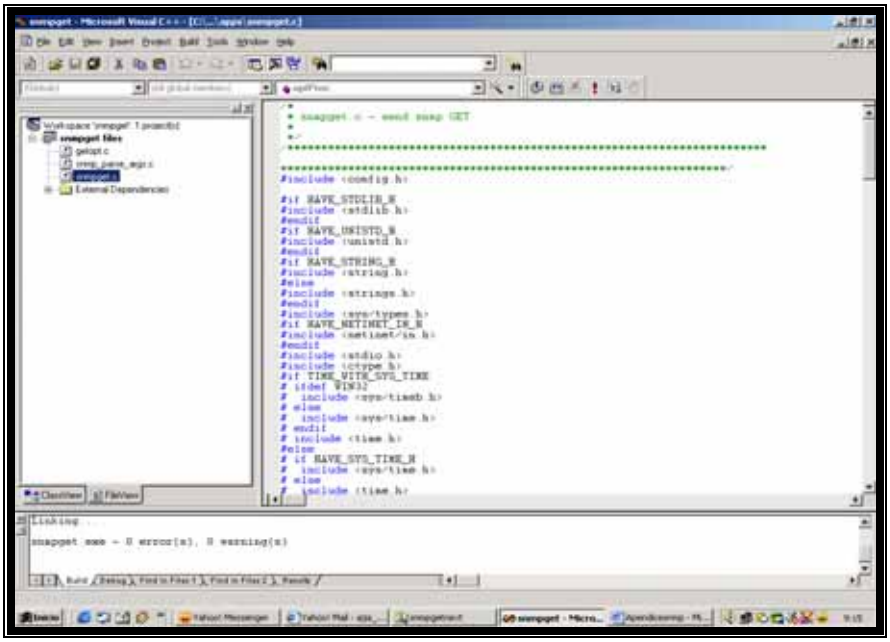


Figura 3.8.- Módulos del programa de configuración de la herramienta de gestión de redes virtuales Compilación.

Cada una de las primitivas tiene su programación asociada, pero, el basamento inminentemente es similar de acuerdo a la explicación a detalle proporcionada ya en párrafos anteriores.

3.6.- PROGRAMA DE MONITOREO DE LA HERRAMIENTA DE GESTION DE REDES VIRTUALES

En la figura 3.9, se observa la diferencia entre una LAN normal y una VLAN. La diferencia más destacable es la existencia de la segmentación física en las redes tradicionales y la segmentación lógica en las VLAN. En una VLAN la segmentación lógica se realiza, tomando en cuenta criterios tales como: las funciones, las aplicaciones, los protocolos, etc. Por lo que en sitios físicos diferentes los nodos pueden pertenecer a la misma VLAN.

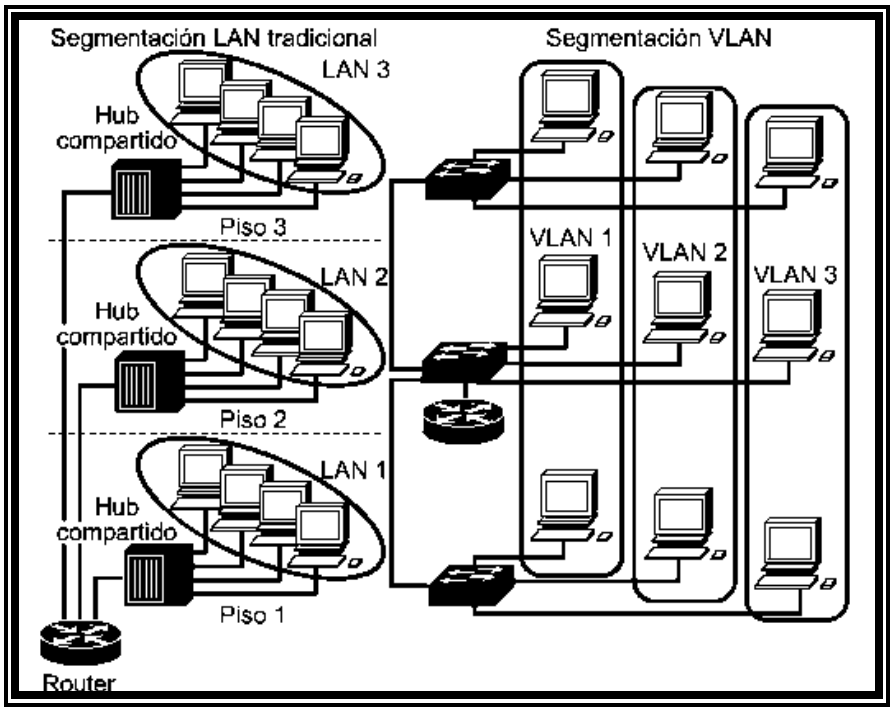


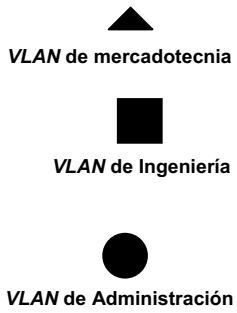
Figura 3.9.- Segmentación en una LAN tradicional y segmentación VLAN.

Así de ésta manera, una VLAN es una red computada, lógicamente segmentada con base en las funciones de los nodos, sin importar su ubicación física.

3.6.1.- ESTRUCTURA DEL PROGRAMA DE MONITOREO

Para que el programa de monitoreo lleve a cabo sus funciones en la herramienta de gestión de VLAN y atendiendo a que diferentes nodos en distintos lugares físicos pueden pertenecer a la misma VLAN, la arquitectura que se emplea para la monitorización consiste en:

Un Servidor Miltihomedhost, el cual permite conectarse a diferentes *VLAN´s*, a través de diferentes interfaces de red.



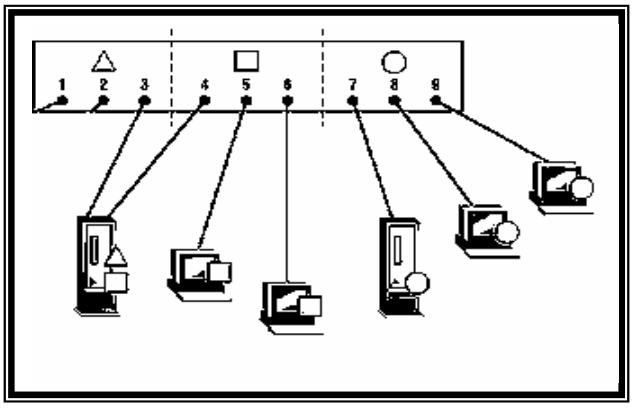


Fig.3.10 Conexión de un servidor de monioreo a la infraestructura de VLAN.

El programa de monitorización por lo tanto reside en el servidor multihomedhost, de tal manera que es posible seleccionar la VLAN la cual se desea monitorizar (véase fig. 3.11). Con este diseño de monitor, es posible atender a cualquier nodo en cualquier lugar físico.

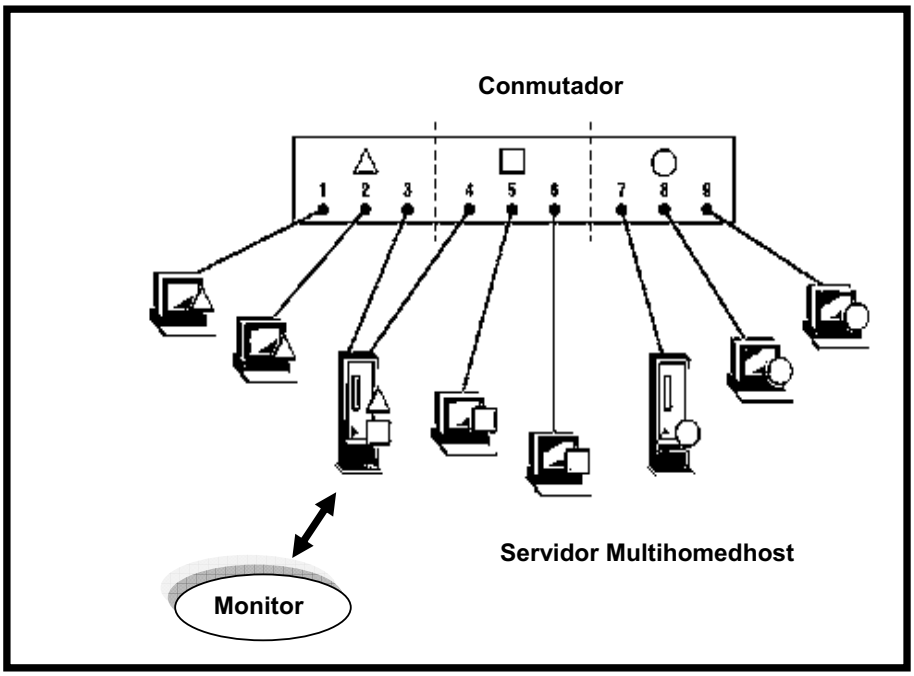


Figura 3.11.- Localización del programa de monitorización en las VLAN´s

Para poder monitorear la VLAN, el programa monitor toma como base la RFC 3069 y la RFC 2643, en la que se determina que para poder atender la comunicación entre los nodos de la VLAN es necesario que estos pertenezcan a la misma subred, por lo que también se debe de asignar a cada VLAN una subred. Con esto se garantiza la localización de cualquier nodo en la VLAN. Para esto se usa cualquier clase de dirección IP, destacando las clases C y B respectivamente (véase fig. 3.12).

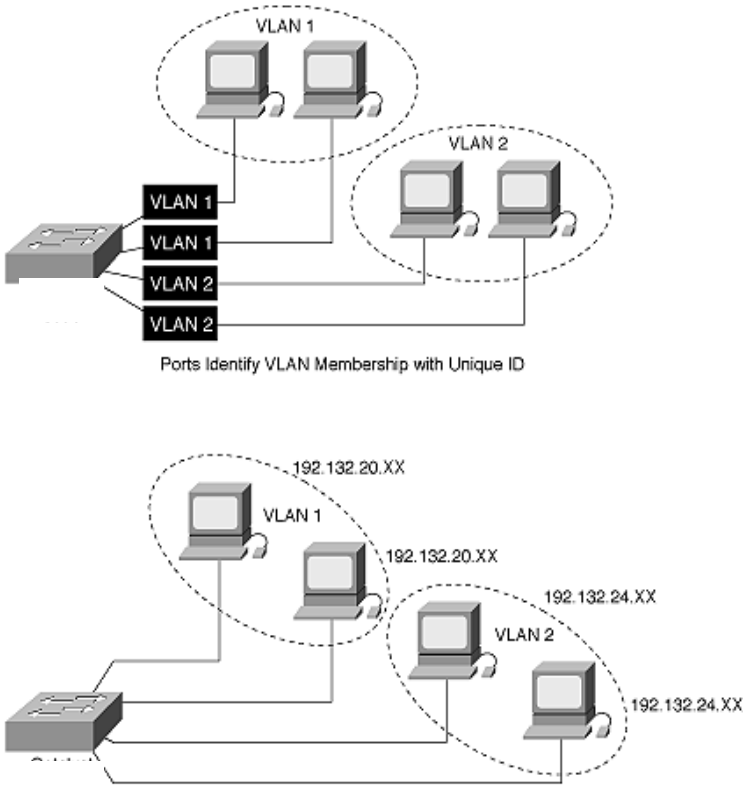


Figura 3.12.- Asociación de una dirección de red a las VLAN's.

Por lo que para esto, se hace necesario el análisis de las tramas. Hay varios tipos distintos de tramas que se describen en diversos estándares. Una trama genérica única tiene secciones denominadas campos, y cada campo está formado por bytes. Los nombres de los *campos* son los siguientes (véase figura 3.13):

- campo de inicio de trama
- campo de dirección
- campo de longitud/tipo/control
- campo de datos
- campo de secuencia de verificación de trama
- campo de fin de trama

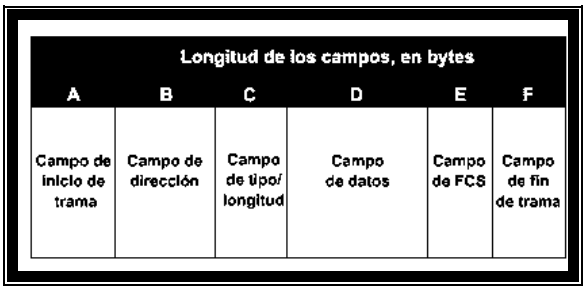


Figura 3.13.- Formato de Trama Genérica.

Para el caso que nos compete el análisis se efectua en los campos de dirección de capa dos y tres; ya que indican la dirección fuente y la dirección destino de los datos/información; basados en estos elementos es posible detectar de que subred se está tratando, y en ese camino la correspondiente VLAN monitoreada (véase figura 3.14).

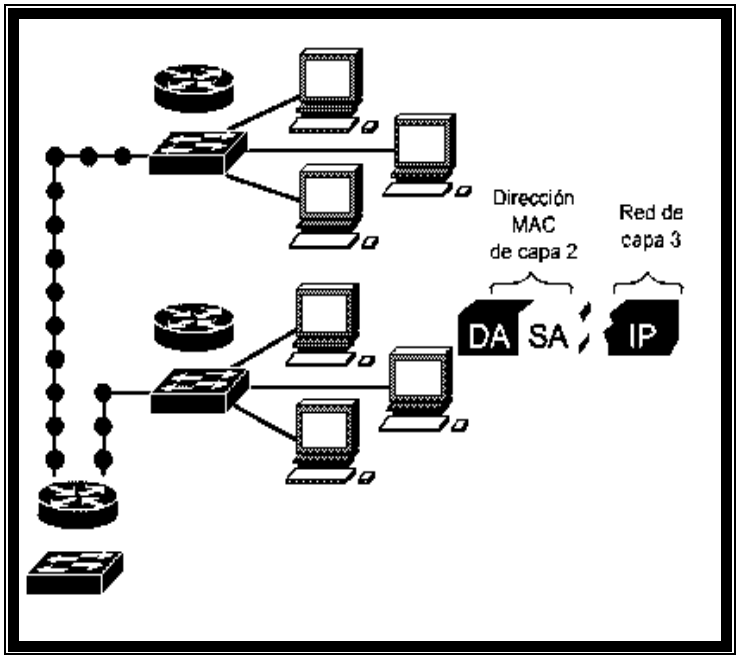


Figura 3.14.- Direcciones a tomar en cuenta en el monitoreo de *VLAN's*, destaca en importancia la dirección de capa tres de acuerdo al estándar RFC 3069 y RFC 2643.

El programa de monitorización permite entonces monitorear:

- **Paquetes, cuyo origen o destino sea un host determinado.**
- **Paquetes, cuya dirección origen o destino corresponda a una red determinada.**
- **Paquetes, cuyo puerto de origen sea algún número de puerto.**
- **Paquetes, TCP, UDP y los puertos correspondientes.**
- **Paquetes, cuya dirección destino corresponda a una red determinada.**
- **Cualquier combinación de los anteriores.**

Con esto es posible un monitoreo a nivel empresarial, como el que se destaca en la figura 3.15, en esta figura se aprecia la funcionalidad del servidor multihomehost y el programa de monitoreo.

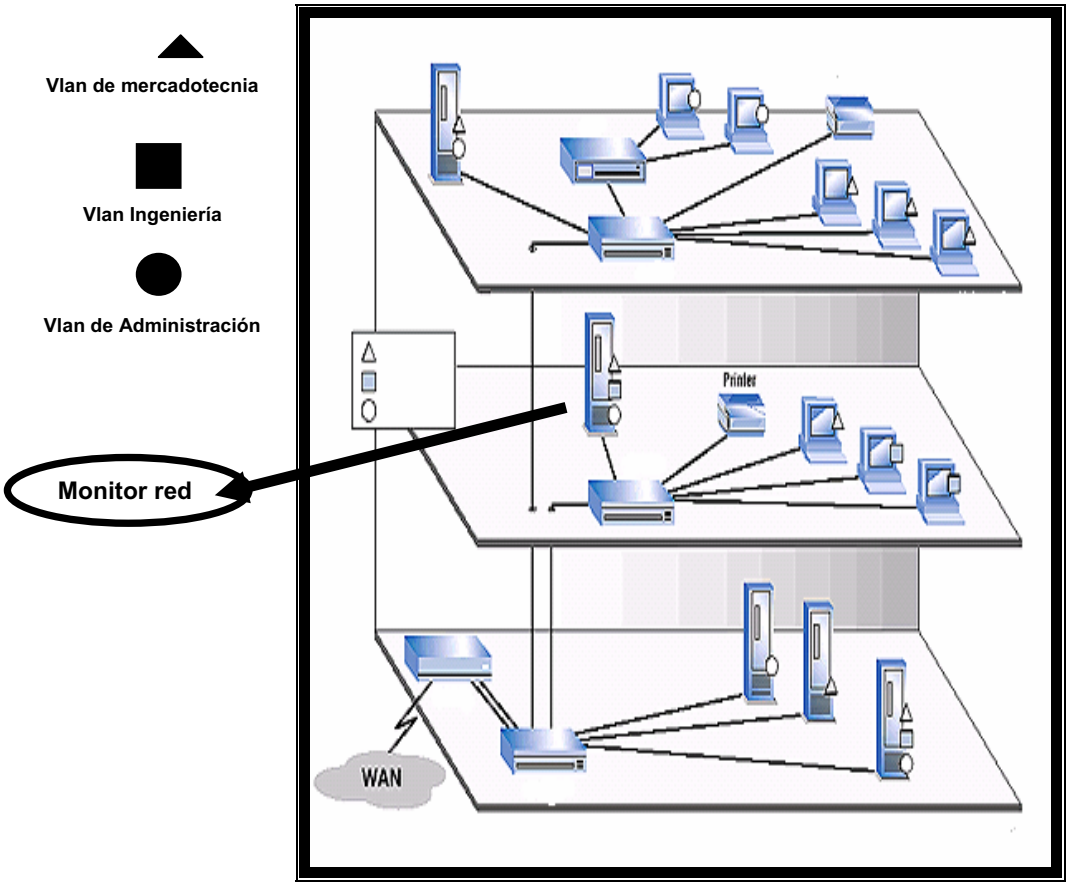


Figura 3.15. Monitoreo en una empresa con *VLAN's*

3.7.- ESTRUCTURA MODULAR DEL PROGRAMA DE MONITOREO.

De está manera los módulos que integran el programa de monitoreo son (véase fig. 3.16):

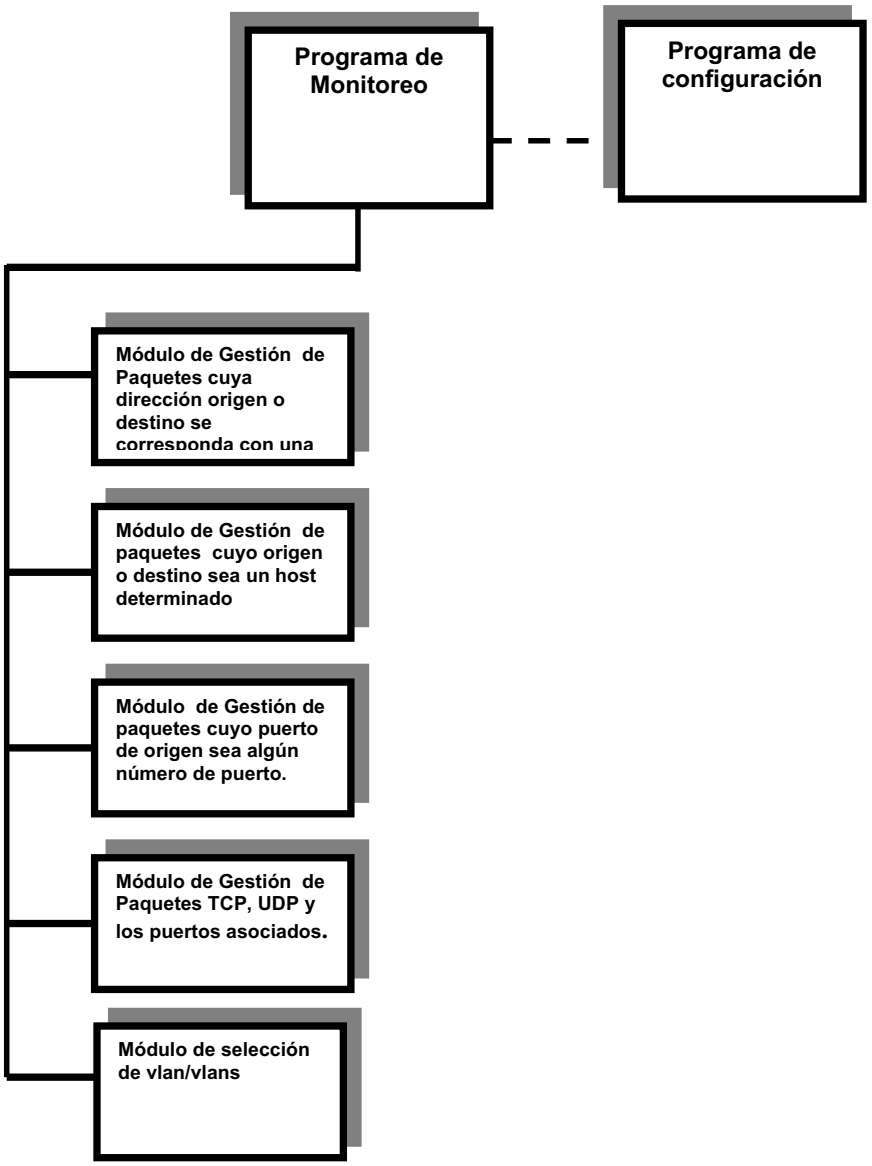


Figura 3.16. Módulos del programa de monitoreo

3.7.1 PROTOCOLO IP

Para cada uno de los módulos que integran el programa de monitoreo, se hace necesario analizar el paquete de información para analizar las direcciones fuente y destino que son la fuente de información para el programa. El protocolo IP (INTERNET Protocol) es la pieza fundamental en la que se sustenta el sistema TCP/IP y por tanto todo el funcionamiento de INTERNET. Su especificación está recogida en **[RFC791]**. La unidad de datos del protocolo IP es el *datagrama*, un esquema del cual puede verse en la figura 3.17.

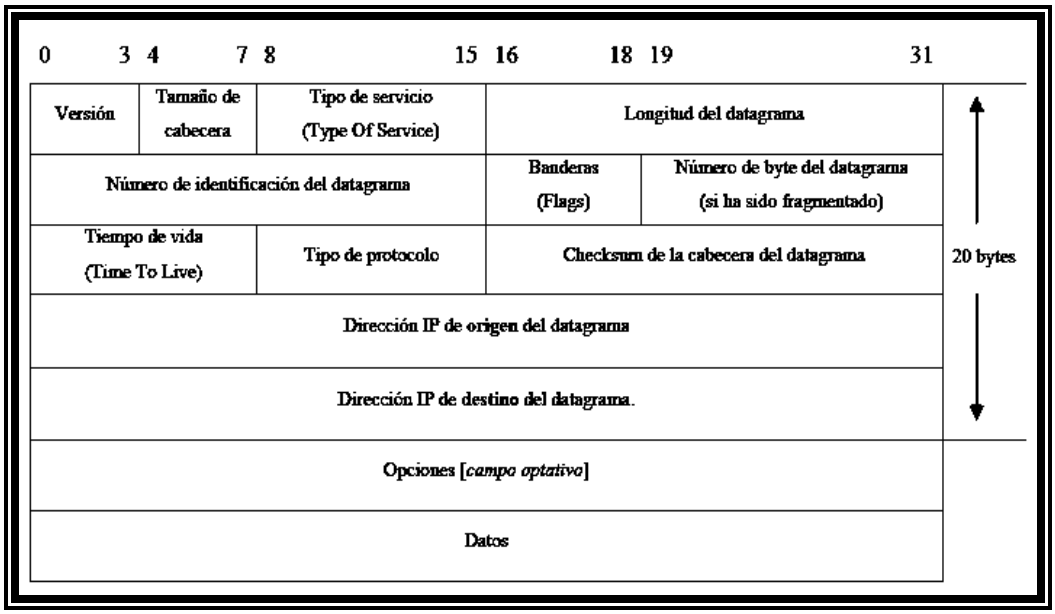


Figura 3.17.- Estructura de un datagrama IP v4.

El protocolo IP facilita un sistema **sin conexión** (connectionless) y **no fiable** (unreliable) de entrega de datagramas entre dos ordenadores cualesquiera conectados a INTERNET. IP da un servicio de entrega basado en el mejor intento (*best effort*).

Esto implica que cuando hay algún funcionamiento anómalo de INTERNET, como podría ser un *router* colapsado, se contempla un sistema muy simple de tratamiento de errores. Este mecanismo de control de errores viene regulado por el ICMP (*INTERNET Control Message Protocol*). En nuestro caso, el router colapsado descartaría el datagrama y enviaría un mensaje de error ICMP al ordenador de origen sin encargarse de la retransmisión del datagrama, lo que **no implica fiabilidad**. Además, no mantiene ningún tipo de información referente al estado de las conexiones. Cada datagrama es encaminado de forma independiente. Esto lo convierte en un **protocolo sin conexión**.

Debido a estas particulares características, puede pasar que se pierdan datagramas y/o que estos no lleguen en orden. De esta manera, cualquier fiabilidad que se necesite, deberá ser realizada por las capas superiores (TCP...).

En la figura 3-18 se pude ver cómo la estructura de un datagrama IP está estructurada en bloques de 32 bits (4 bytes). El datagrama IP se transmite enviando primero el bit 0, luego el bit 1, 2, 3... y así sucesivamente hasta finalizar el datagrama. Este orden se denomina **network byte order**. El orden es muy importante, puesto que los diferentes ordenadores tienen diferentes sistemas de almacenamiento de bits en memoria. El formato *little endian*, consite en almacenar los bits en orden inverso al network byte order (usando por ejemplo en los procesadores Intel), mientras que la otra posibilidad se denomina Big endian (usado por ejemplo en los procesadores Motorola).

La *versión* (4 bits), sirve para identificar a que versión específica (RFC) hace referencia el formato del datagrama. Esta información sólo es utilizada por los routers y capa IP de origen y final del datagrama. Esto permite la coexistencia de diferentes versiones del protocolo IP de una forma transparente al usuario. La versión actual es la 4 (conocida también cómo IPv4, véase figura 3.18).

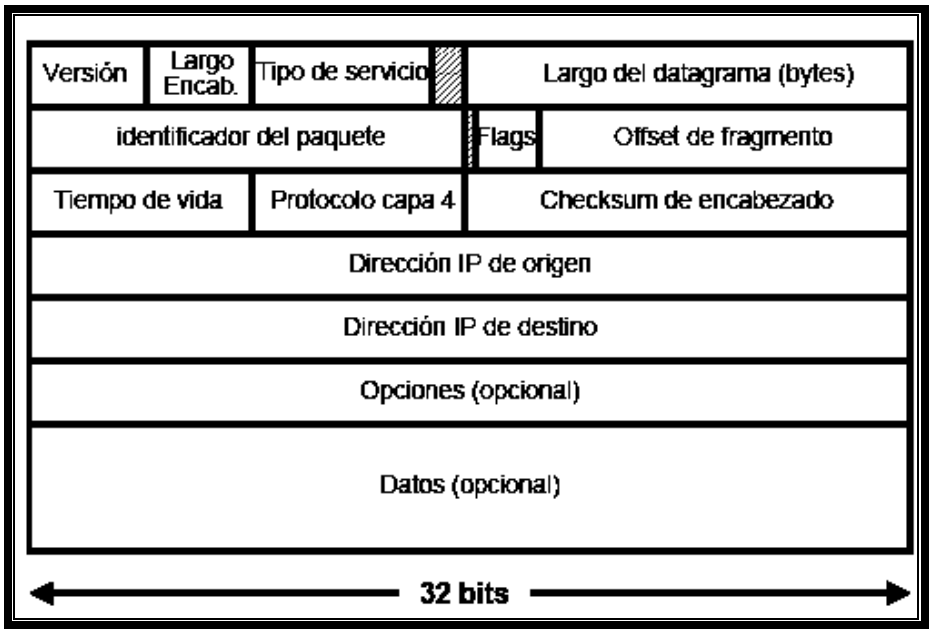


Figura 3.18.- Estructura de un datagrama IP v4, campo versión.

El ***tamaño de la cabecera*** (Header Length), son 4 bits (2^4 = 16 posiciones,...15) que indican el número de palabras de 32 bits que ocupa la cabecera. Estos 4 bits de tamaño máximo, nos limitan a un tamaño de cabecera máximo de 60 bytes (15 * 32 bits = 60 bytes). No obstante, el valor usual de este campo es 5 (5 * 32 bits = 20 bytes), veáse figura 3.19.

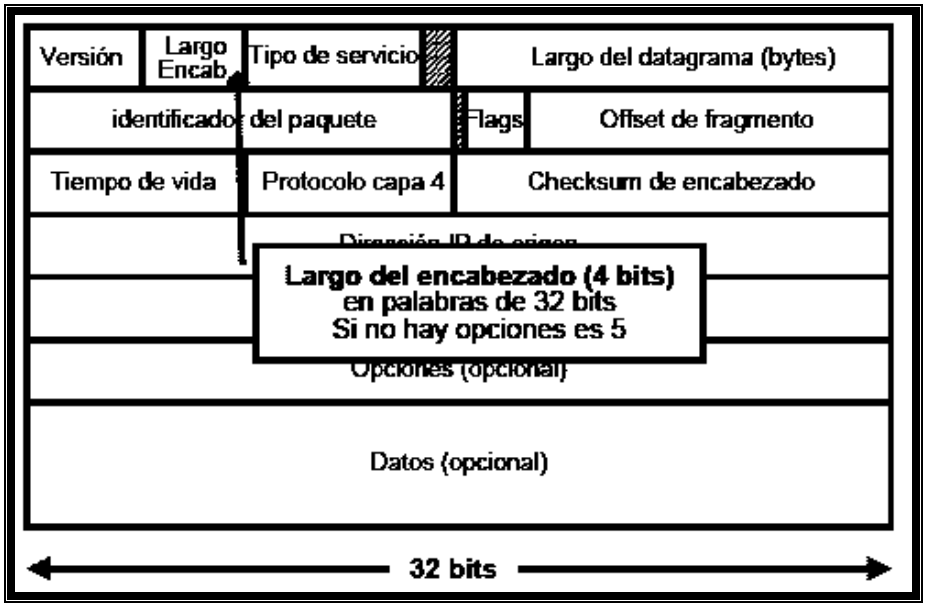


Figura 3.19.- Estructura de un datagrama IP v4, largo del encabezado.

El ***campo del tipo de servicio*** (Type Of Service), se compone de 8 bits. Los primeros 3 bits tienen una función obsoleta y no se contemplan actualmente. Los 4 bits siguientes definen el tipo de servicio (ver figura 2-12). Y el último bit no se utiliza actualmente y debe tener valor 0. Solo 1 de los 4 bits del tipo de servicio puede estar activo a la vez, véase figura 3.20.

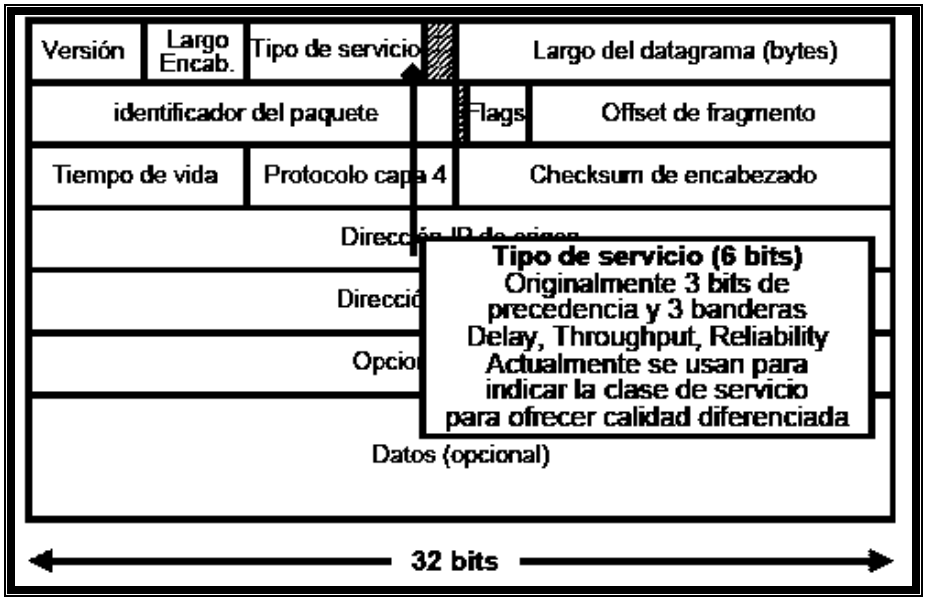


Figura 3.20.- Estructura de un datagrama IP v4, tipo de servicio.

El tipo de servicio determina la política a seguir en el envío del datagrama por INTERNET. Las opciones posibles son *minimizar el retraso* (minimize delay), *maximizar el rendimiento* (maximize

throughput), *maximizar la fiabilidad del transporte* (maximize reliability) y *minimizar el coste económico* del transporte (minimize monetary cost), véase tabla 3.1.

Tabla 3.1.- Valores típicos del tipo se servicio según la aplicación.

Tipo de aplicación	Minimizar retraso	Maximizar rendimiento	Maximizar fiabilidad	Minimizar coste	Valor en hexadecimal
TELNET	1	0	0	0	0x10
FTP	0	1	0	0	0x08
SMTP	0	1	0	0	0x08
DNS (UDP)	1	0	0	0	0x10
DNS (TCP)	0	0	0	0	0x00
ICMP	0	0	0	0	0x00
BOOTP	0	0	0	0	0x00

La ***longitud del datagrama*** (Total Length), es un número de 16 bits (2^16 = 65536, 0..65535) que indica la longitud total del datagrama. Este valor es muy importante, ya que nos permite saber que tamaño de memoria debemos reservar para la recepción del datagrama. Además, nos indica el número de bytes a leer, lo que nos permite un simple control de error. De esta forma, si el valor es incorrecto, el número de bytes leídos será como máximo de 65535, acotando el error. Además nos limita el número de bytes a enviar en un datagrama (Maximum Transfer Unit, MTU) a 65535 – 20 (tamaño típico de la cabecera) = 65515 bytes, véase figura 3.21.

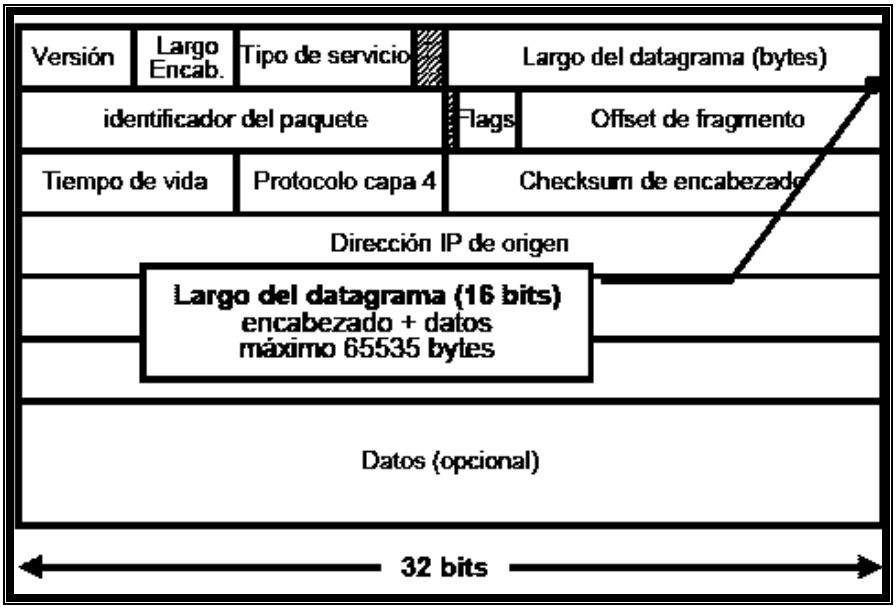


Figura 3.21.- Estructura de un datagrama IP v4, largo del datagrama.

Si el tamaño del datagrama, es mayor que el tamaño máximo del paquete de red (Ej. Datagrama de 32000 bytes enviado sobre una ethernet, que tiene un tamaño máximo de paquete de 1500 bytes), se fragmenta en N trozos.

El ***número de identificación del datagrama*** (Identificartion Field), es un número de 16 bits que en caso de fragmentación de un datagrama nos indica su posición en el datagrama original. Esto nos permite recomponer el datagrama original en la máquina de destino. Este valor nos indica que un datagrama puede ser fragmentado en un máximo de 65535 fragmentos, véase figura 3.22.

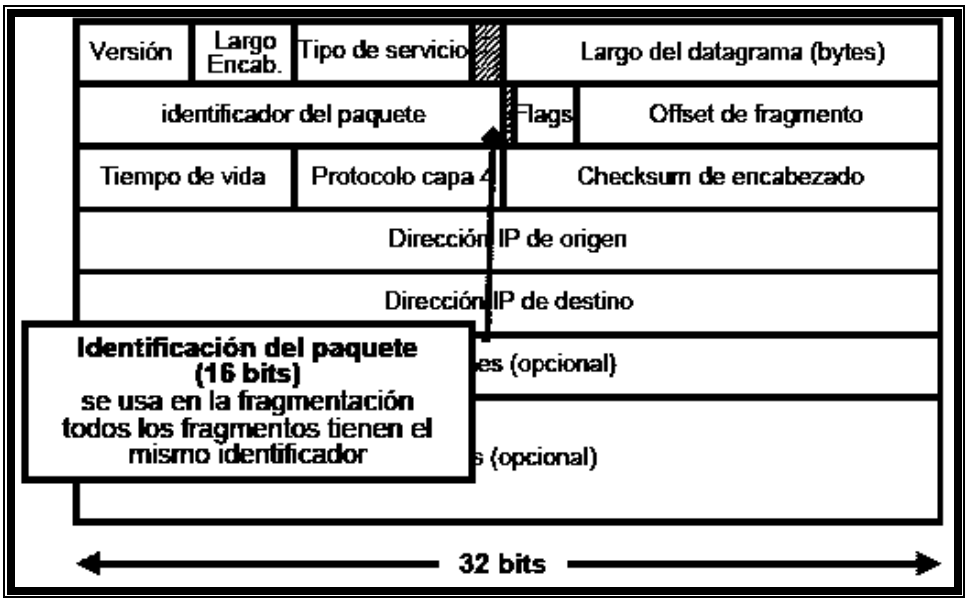


Figura 3.22- Estructura de un datagrama IP v4, identificación del paquete.

Las ***banderas*** (Flags) son 3 bits. El primero permiten señalar si el datagrama recibido es un fragmento de un datagrama mayor, bit M (More) activado. El segundo especifica si el datagrama no debe fragmentarse, bit DF (Don't fragment) activado. Y el tercero no se utiliza actualmente, asignándole el valor 0, véase figura 3.23.

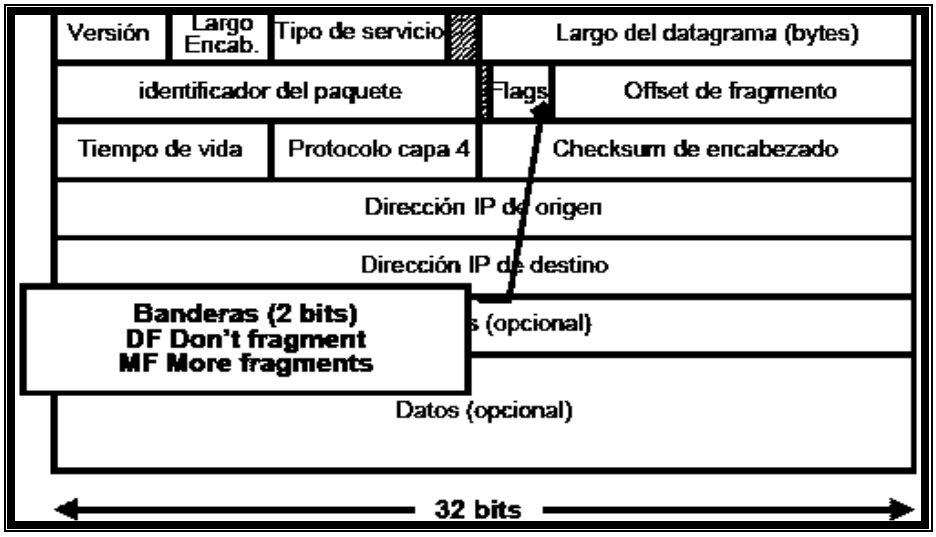


Figura 3.23.- Estructura de un datagrama IP v4, Banderas.

El ***número de byte en el datagrama*** (Fragmentation Offset), nos indica la posición en bytes que ocupan los datos en el datagrama original. Sólo tiene sentido si el datagrama forma parte de uno mayor que ha sido fragmentado. Este campo tiene un máximo de 13 bits (2^13 = 8192, como nos indica el desplazamiento en bytes 8192 * 8 bits = 65536). De esta forma, podemos reconstruir el datagrama original con los fragmentos, veáse figura 3.24.

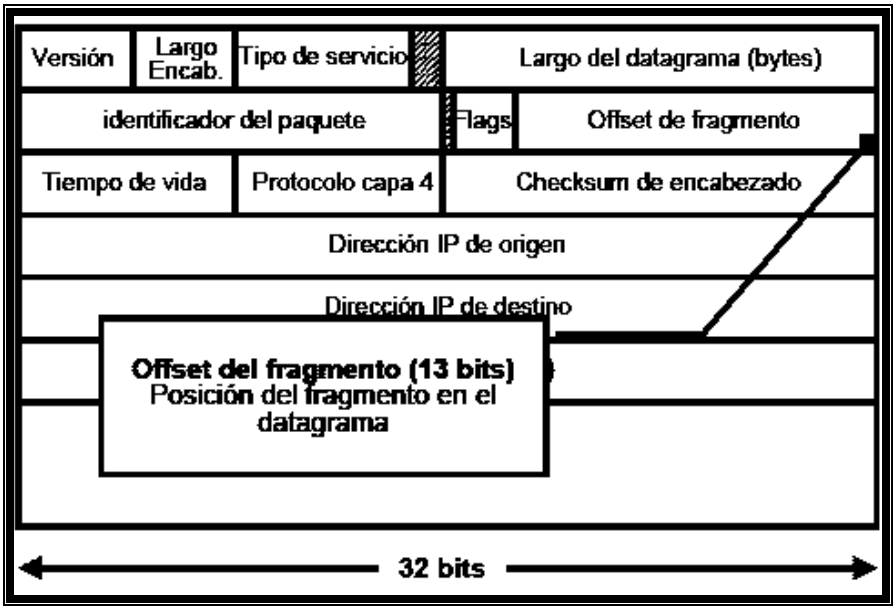


Figura 3.24.- Estructura de un datagrama IP v4, offset del fragmento.

El ***tiempo de vida*** (Time To Live), es un campo de 8 bits que indica el tiempo máximo que el datagrama será válido y podrá ser transmitido por la red. Esto permite un mecanismo de control para evitar datagramas que circulen eternamente por la red (por ejemplo en el caso de bucles). Este campo se inicializa en el ordenador de origen a un valor (máximo 2^8 = 256) y se va decrementando en una unidad cada vez que atraviesa un router. De esta forma, si se produce un bucle y/o no alcanza su destino en un máximo de 255 “saltos”, es descartado. Entonces se envía un datagrama ICMP de error al ordenador de origen para avisar de su pérdida, véase figura 3.25.

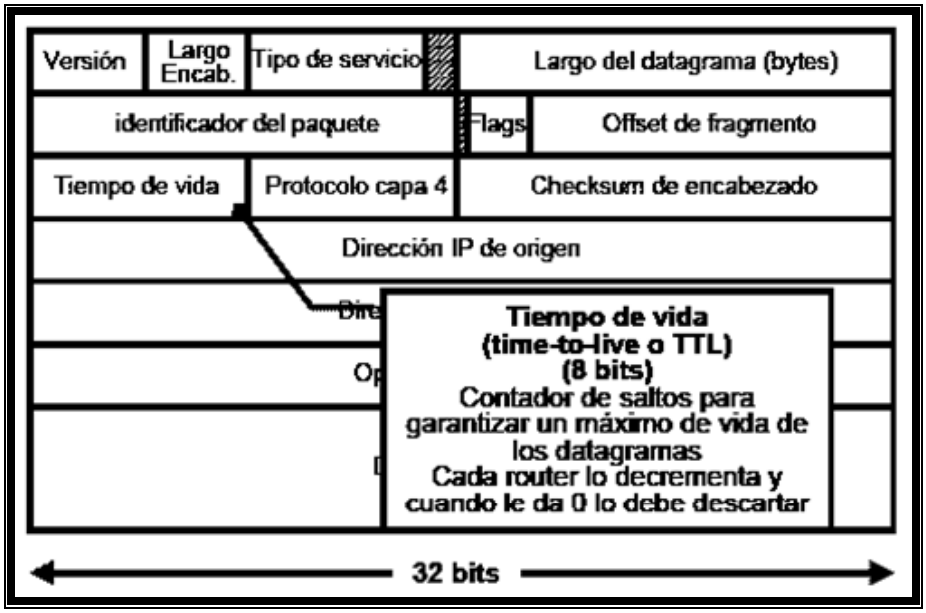


Figura 3.25.- Estructura de un datagrama IP v4, TTL.

El ***tipo de protocolo*** (Protocol), es un valor que indica a que protocolo pertenece el datagrama (TCP, UDP, ICMP...). Es necesario debido a que todos los servicios de INTERNET utilizan IP como transporte, lo cual hace necesario un mecanismo de discriminación entre los diferentes protocolos, véase figura 3.26.

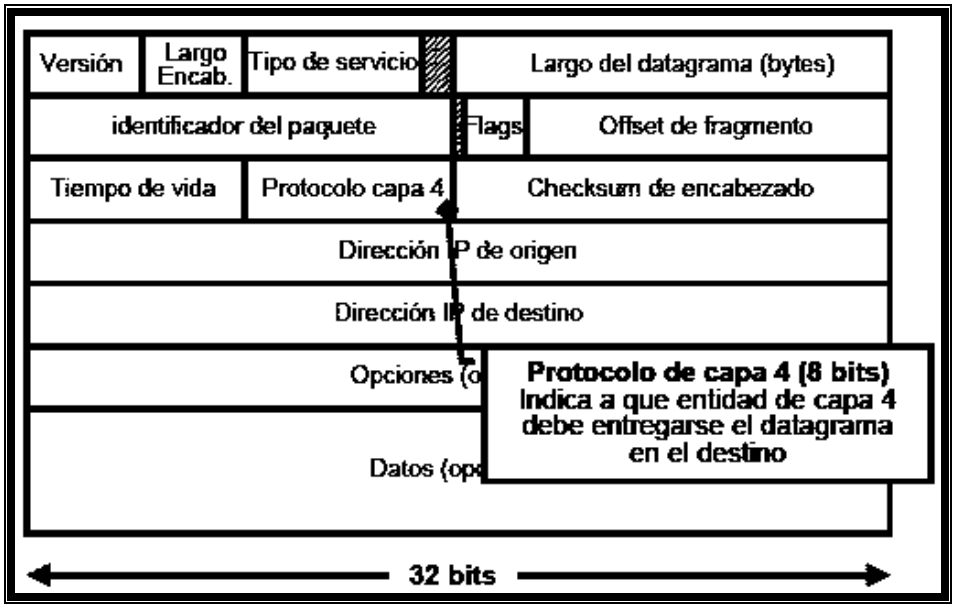


Figura 3.26.- Estructura de un datagrama IP v4, Protocolo de capa 4.

El campo de protocolo permite identificar que tipo de datos está trasportando IP en su campo de información(Datos). Puede ser los siguientes, véase tabla 3.2:

Tabla 3.2.- Valores del campo Protocolo.

Número	Tipo	Descripción
1	ICMP	Transmite Mensajes de Error
2	IGMP	Para Grupos Multienvío
6	TCP	Protocolo de Control de Transmisión
8	EGP	Utilizado para encaminar Datagramas a redes externas
17	UDP	Protocolo de Datagrama de Usuario
88	IGRP	Protocolo de intercambio de encaminamiento

El ***checksum de la cabecera del datagrama*** (Header Checksum), es una suma de comprobación que afecta sólo a la cabecera del datagrama IP. El resto de protocolos TCP, UDP, IGMP... tienen su propia cabecera y checksum. Su función es simplemente la de un mecanismo de control de errores. De esta forma, si se encuentra un error en el checksum de un datagrama IP, este es simplemente descartado y no se genera ningún mensaje de error. Esto implica que es deber de las capas superiores el control del flujo de los datagramas. Asegurándose que estos lleguen correctamente al destino, ya sea utilizando un protocolo fiable (TCP) o implementando internamente algún tipo de control, véase figura 3.27.

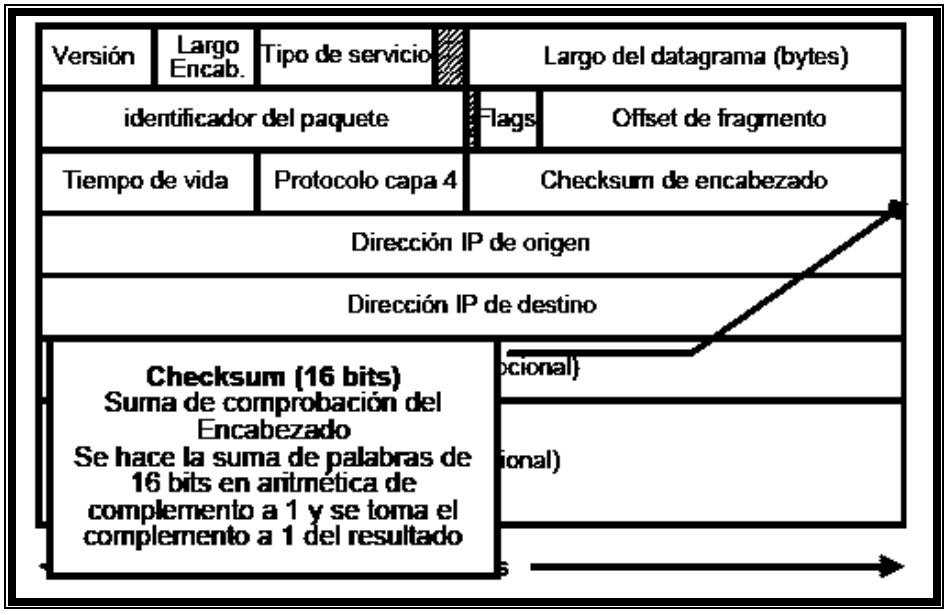


Figura 3.27.- Estructura de un datagrama IP v4, Checksum.

Tanto la ***dirección IP de origen como la de destino*** (IP address), están formadas por dos números de 32 bits. Estas direcciones se corresponden a una distribución según la figura 3.28, 3.29 y 3.30.

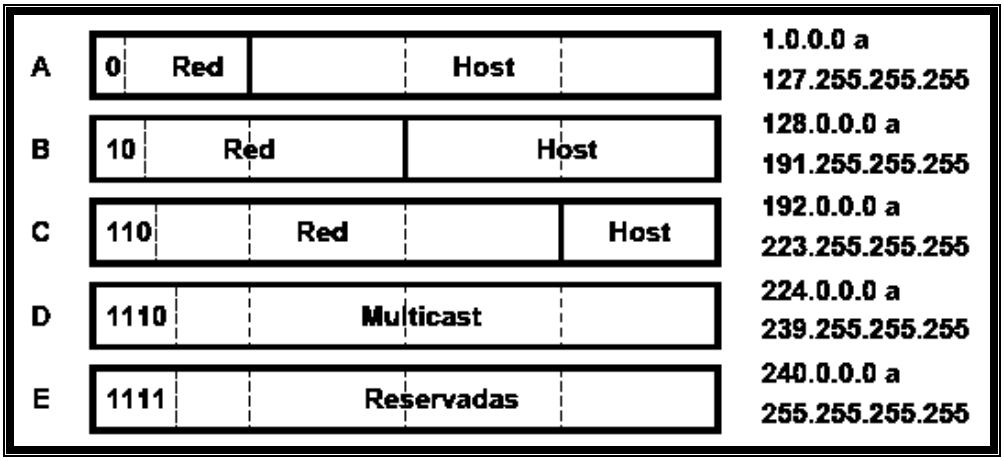


Figura 3.28.- Formato de Direcciones IP.

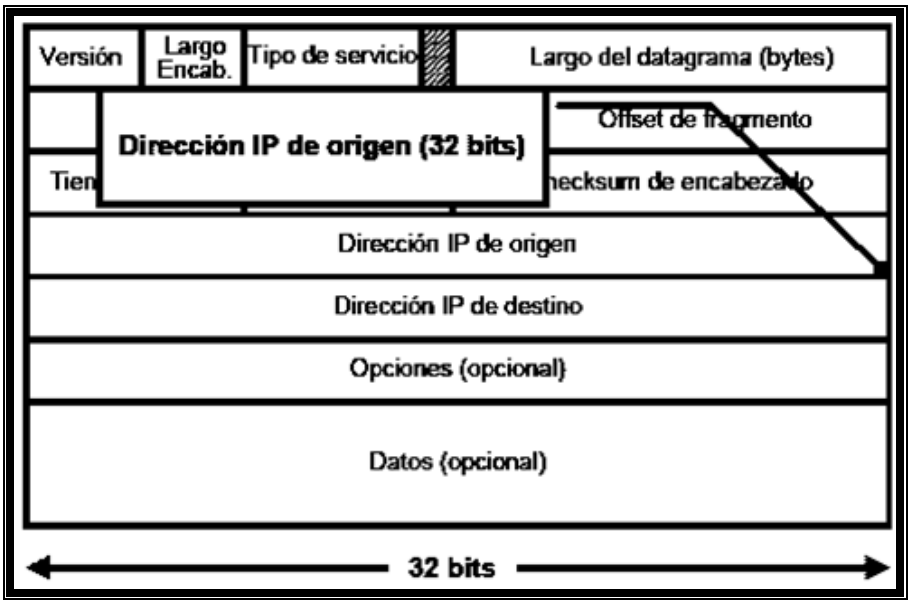


Figura 3.29.- Estructura de un datagrama IP v4, direcciones IP Origen.

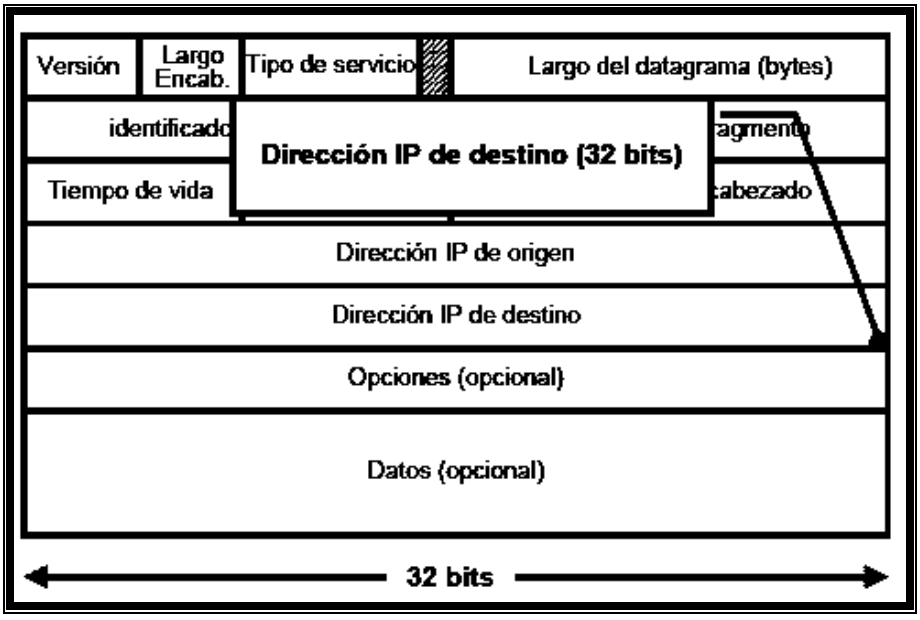


Figura 3.30.- Estructura de un datagrama IP v4, direcciones IP Destino.

El campo de ***opciones*** de un largo de 0 y 40 bytes que indican determinadas reglas de seguridad representado en dos formas:
1.- Un único byte indica el tipo de opción
2.- Un byte que indica el tipo de opción, un byte que indica la longitud de la opción y los bytes relacionados con la opción en sí, véase figura 3.31.

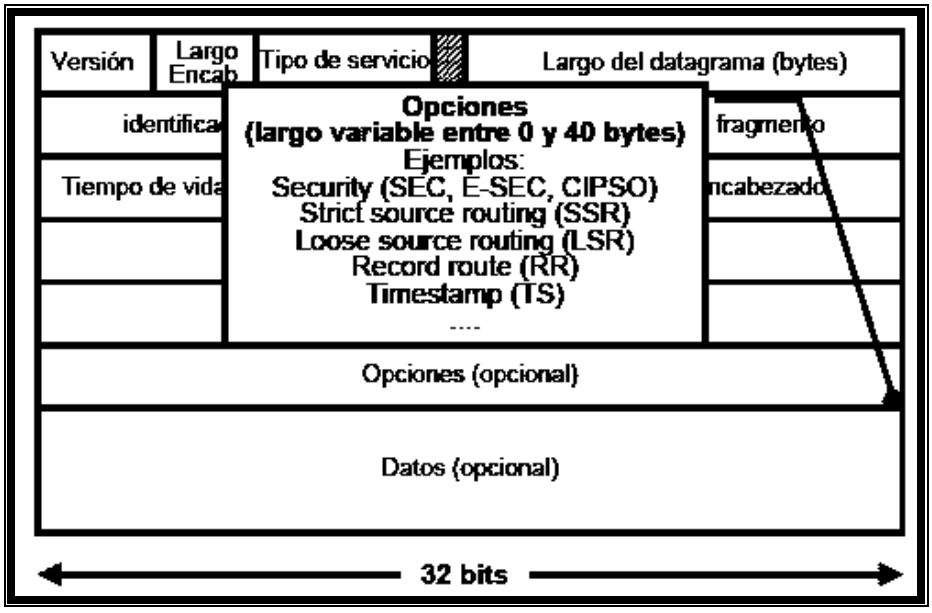


Figura 3.31.- Estructura de un datagrama IP v4, opciones.

Fianalmene El campo de datos esta formado por una serie de información que dependiendo del tipo del protocolo se trasnporta por el datagrama IP, véase figura 3.31.

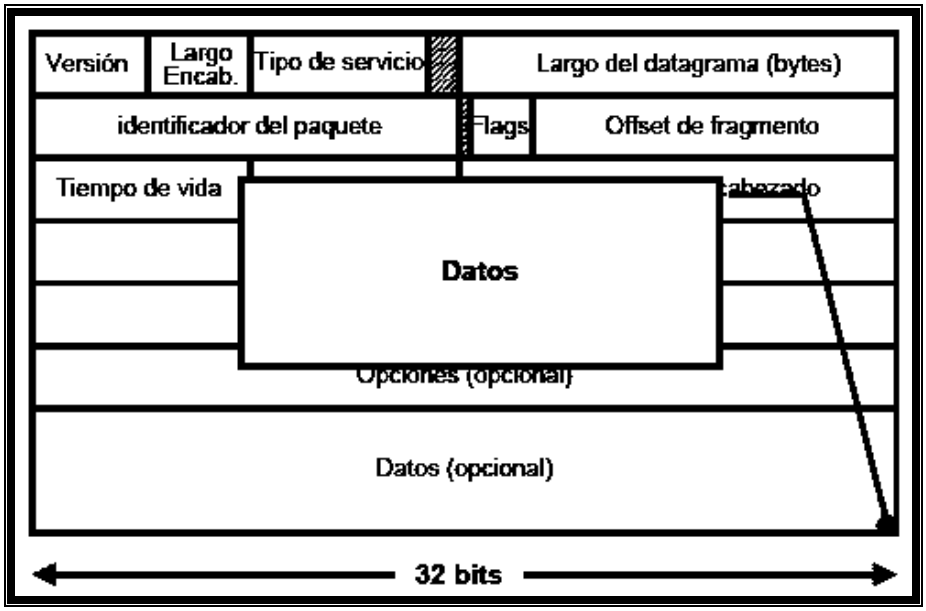


Figura 3.32.- Estructura de un datagrama IP v4, Datos.

RESUMEN CAPITULO III. HERRAMIENTA DE GESTIÓN DE REDES VIRTUALES.

La herramienta de gestión de red permite al través de diferentes módulos: la monitorización, configuración y el informe de la disponibilidad y la utilización de la red. Estos datos pueden usarse para la detección de problemas (p. ej. los cuellos de botella), determinar los umbrales de operatividad (necesarios para la determinación de los acuerdos de nivel de servicio), y la planificación de la capacidad de la *VLAN*. Además, se tiene la capacidad de poder modificar los parámetros asociados a los dispositivos de configuración de la red virtual con el fin de proporcionar los servicios para la adaptación de los recursos de la red.

La Herramienta empleo la biblioteca de captura LIBCAP y el API asociado a la misma, se desarrollo bajo plataforma Windows con portabilidad con el sistema operativo Linux. Se empleo C++ para la programación efectiva de cada uno de los módulos que componen la herramienta. El siguiente capitulo detalla los resultados alcanzados con el empleo de la Herramienta de Gestión de Redes Virtuales.

Para la parte de monitoreo es necesario considerar los elementos de red para una forma sustentable de obtener la información, para esto se considero:

- Protocolo de red: Internet Protocol (IP)
- Esquema de direccionamiento
- Formato de la PDU
- Encaminamiento de paquetes
- Función de ruteo: determina el contenido de la tabla de ruteo
- Función de forwarding: decide en base al contenido de la tabla de ruteo el próximo salto de cada paquete
- Protocolo de control ICMP
- Mensajes de reporte de “errores” e información adicional sobre la red

Esto nos permite vislumbrar que la capa de red cumple una función de direccionamiento que es propicia para la herramienta de gestión de red, véase figura 3.33.

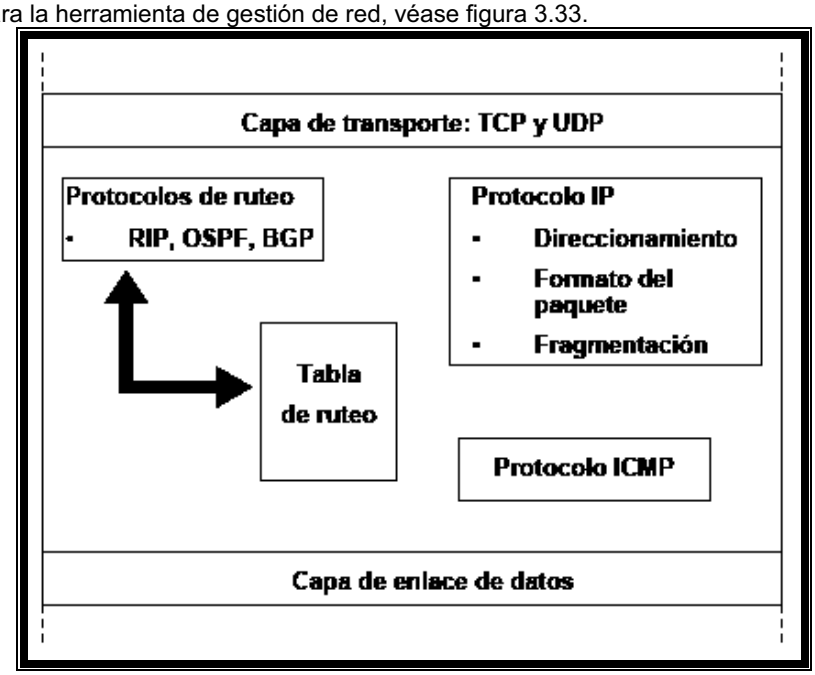


Figura 3.33.- La capa de red en la herramienta de gestión de redes virtuales

CAPITULO IV
PRUEBAS Y RESULTADOS DE LA HERRAMIENTA DE GESTIÓN DE REDES VIRTUALES.

INTRODUCCIÓN

En el campo de las tecnologías de información la tendencia más importante en este momento la constituyen los sistemas distribuidos y las redes de computadoras. Siendo así, la mayoría de las computadoras trabajan conectados a una red a través de la cual los usuarios pueden acceder a recursos remotos, comunicarse, trabajar en grupo, etc. En este capitulo se presentan los resultados de las pruebas aplicadas a los módulos que integran la Herramienta de Gestión de Redes Virtuales, así como, la topología que se empleo para el monitoreo e información de la red virtual de área local(VLAN). Se empleo la red 10.0.0.0/24, junto con los dispositivos de conectividad que permiten el buen funcionamiento de la red.

4.1 TOPOLOGÍA PARA LA PRUEBA DE LA HERRAMIENTA DE GESTIÓN DE REDES VIRTUALES.

Para aplicar los módulos de la Herramienta de Gestión de redes Virtuales se emplearon los siguientes equipos:

1.- Dos ruteadores CISCO 2600 con sistema Operativo 12.3, con soporte de agente SNMP.

2.- Un conmutador CISCO 2950 con sistema Operativo 12.3, con soporte de agente SNMP.

3.- Ocho computadoras Pentium Cuatro, con 128 Megabytes, disco duro de 40 Gigabytes y sistema operativo Windows 2000.

4.- Se crearon las siguientes VLAN`s: VLAN de Ingeniería, VLAN de Mercadotecnia.

5.- Se coloco la estación de monitoreo, así también, se instalo la herramienta de Gestión de Redes Virtuales.

6.- Se configuraron los agentes en los Ruteadores Cisco 2600 y en los Conmutadores CISCO 2950 para poder acceder a las variables MIB de la VLAN, así como, a las estadísticas de uso de la red virtual de área local (véase figura 4.1).

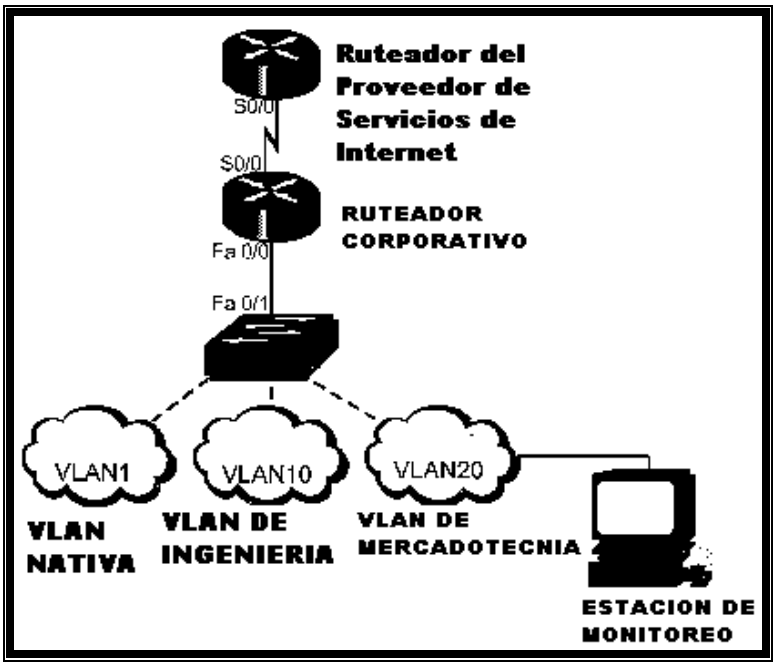


Figura 4.1.- Topología de Prueba para la Herramienta de Gestión de Redes Virtuales.

4.2- PROGRAMA DE CONFIGURACIÓN/RECONFIGURACIÓN DE LA RED VIRTUAL.

El programa de configuración hace uso del estándar de FACTO SNMP, por lo que incorpora (véase figura 4.2):

a) Un agente de gestión.
b) Un gestor.
c) Objetos gestionados.
d) Protocolo de gestión.

El gestor es el software residente en una estación de gestión que se comunica con los agentes y que ofrece al usuario una interfaz a través de la cual comunicarse con los elementos de gestión para obtener información de los recursos gestionados. Además recibe las notificaciones enviadas por los agentes, éste componente programado forma parte de del programa de configuración.

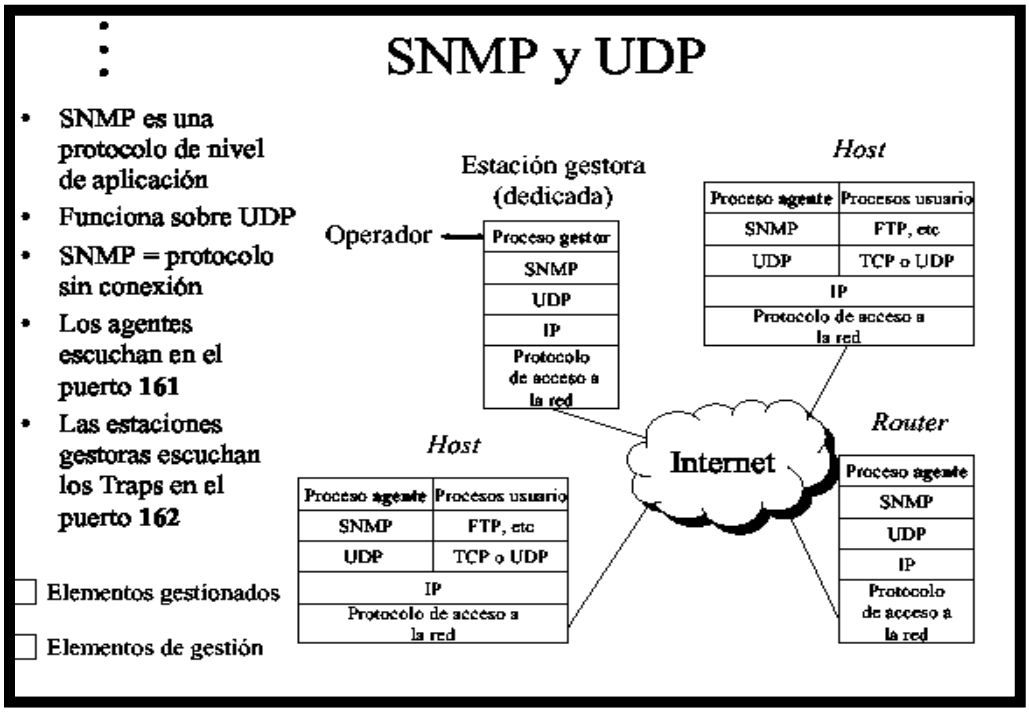


Figura 4.2.- Modelo de configuración estación Gestora y sistemas Gestionados.

4.3.- ACCESO A LOS MANDATOS SNMP EN EL PROGRAMA DE CONFIGURACIÓN.

Una de las claves de la flexibilidad del SNMP es el uso de "variables" como forma de representación de los recursos, tanto físicos como lógicos, en los sistemas gestionados. En cada nodo gestionado, el agente SNMP proporciona una base de datos llamada MIB *(Management Information Base)*, que contiene objetos de datos, más conocidos como variables MIB.

Los mandatos SNMP que emplea el programa de configuración son:

- **Get**: Enviado por la estación gestora para obtener las variables MIB específicas de un nodo gestionado.
- **Get-next**: Es enviado por la estación gestora a la gestionada para obtener la variable MIB siguiente a la gestionada.
- **Get-response**: es enviado por el nodo gestionado a la estación gestora como respuesta a mandatos get, get-next o set.
- **Set**: Enviado por la estación gestora para dar un valor determinado a una variable MIB de un nodo gestionado.

Para acceder a estos elementos de configuración es necesario entrar a la Herramienta de Gestión de Redes Virtuales, con lo que se debe de localizar y ejecutar (véase figura 4.3).

Figura 4.3.- Pantalla de presentación de la herramienta de gestión de redes virtuales.

La herramienta consta de un menú, en dónde es posible seleccionar: Configuración de VLAN a través del uso de las primitivas **SNMP*;*** El monitor de red; Un visualizador de las variables MIB de cualquier dispositivo; Un modulo de ayuda que describe que se necesita realizar para usar la herramienta de gestión de redes virtuales (véase figura 4.4).

Figura 4.4.- Pantalla de Menús de la herramienta de gestión de redes virtuales.

La opción **opciones** permite salir de la herramienta de Gestión de Redes Virtuales (Véase Figura 4.5).

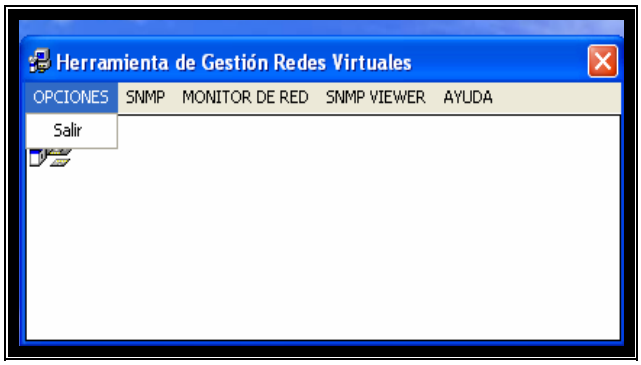


Figura 4.5.-Menú opciones de la Herramienta de Gestión de Redes Virtuales

EL menú SNMP *(SIMPLE NETWORK MANAGEMENT PROTOCOL),* tiene las principales primitivas para manipular la MIB *(MANAGEMENT INFORMATION BASE),* además de un modulo para las estadísticas del sistema (véase figura 4.6).

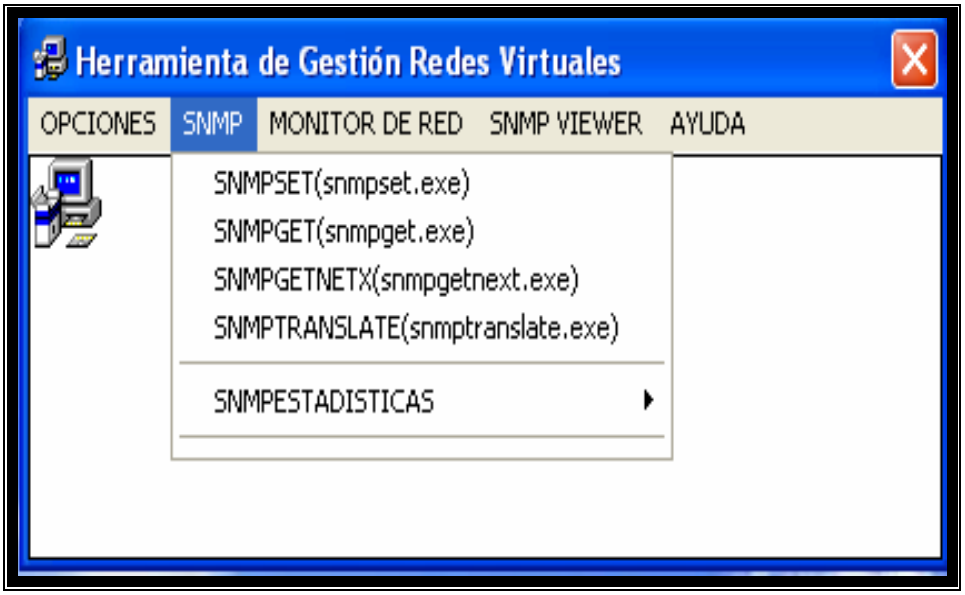


Figura 4.6.-Menú opciones SNMP de la Herramienta de Gestión de Redes Virtuales.

4.4 CORRIDAS EJEMPLO DE LOS MANDATOS SNMP.

Cada opción que se elija de los mandatos SNMP tiene un editor de ayuda para poder imprimirla o adicionarle nueva información, así como una ventana de consola para poder ejecutar el programa respectivamente (véanse figuras 4.7 y 4.8).

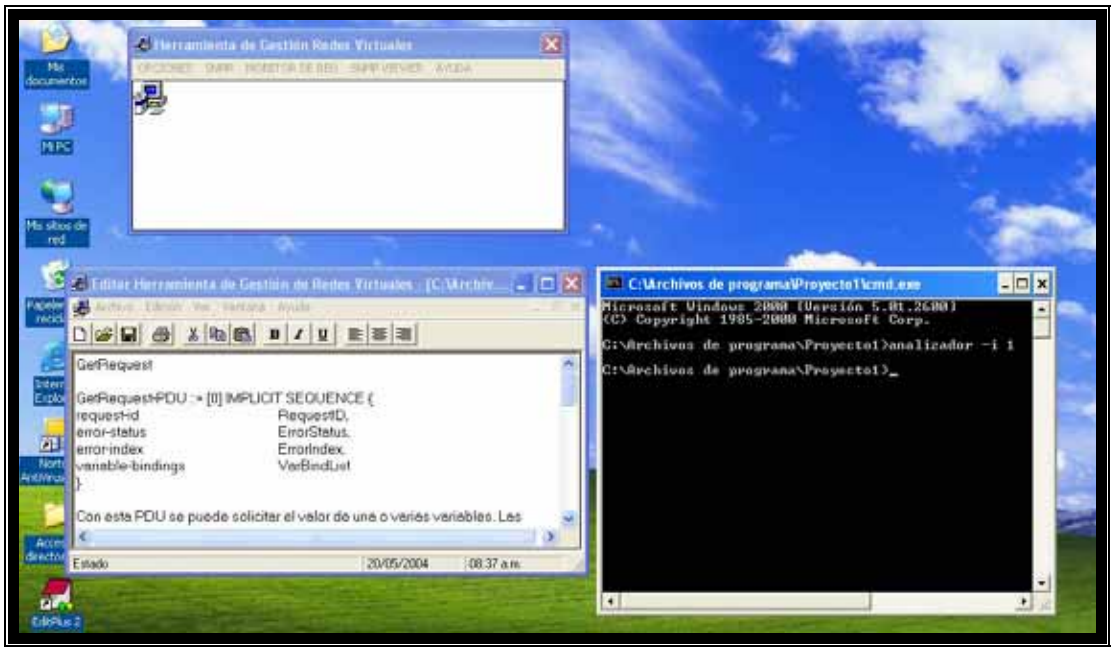


Figura 4.7.-Ejecución de la Herramienta de Gestión de Redes Virtuales.

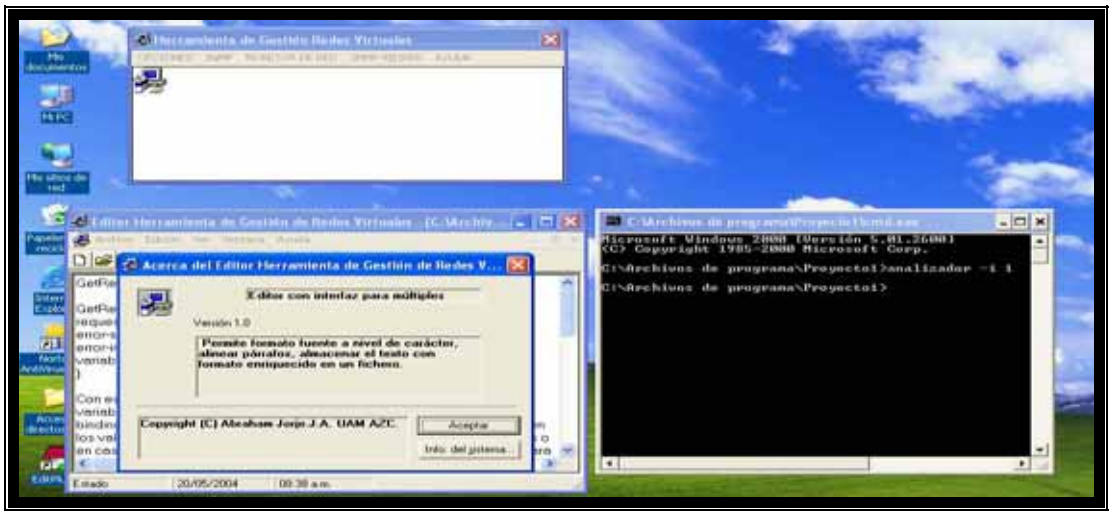


Figura 4.8.-Ejecución de la Herramienta de Gestión de Redes Virtuales, Editor.

Como ejemplo de las corridas se tienen las siguientes pantallas (véanse figuras 4.9-4.11):

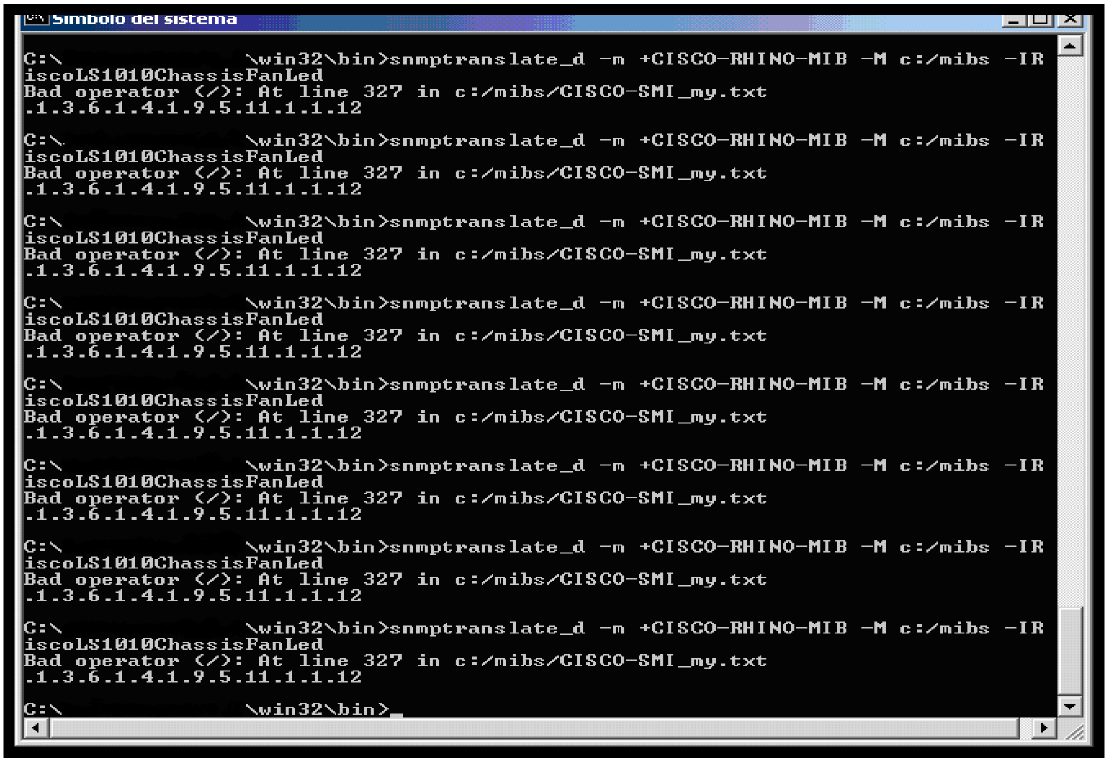


Figura 4.9.- Uso de snmptranslate

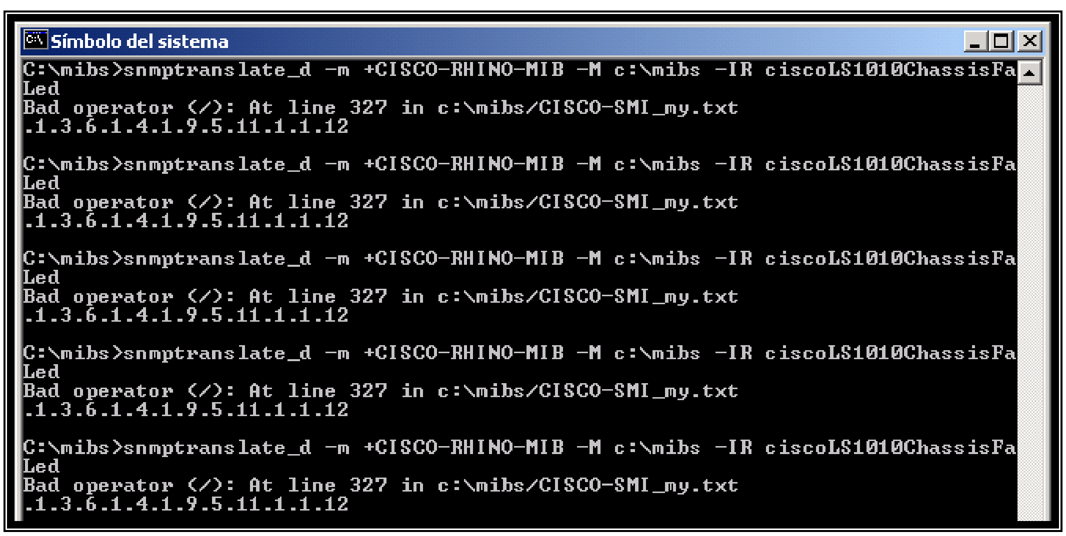


Figura 4.10.- Uso de snmptranslate

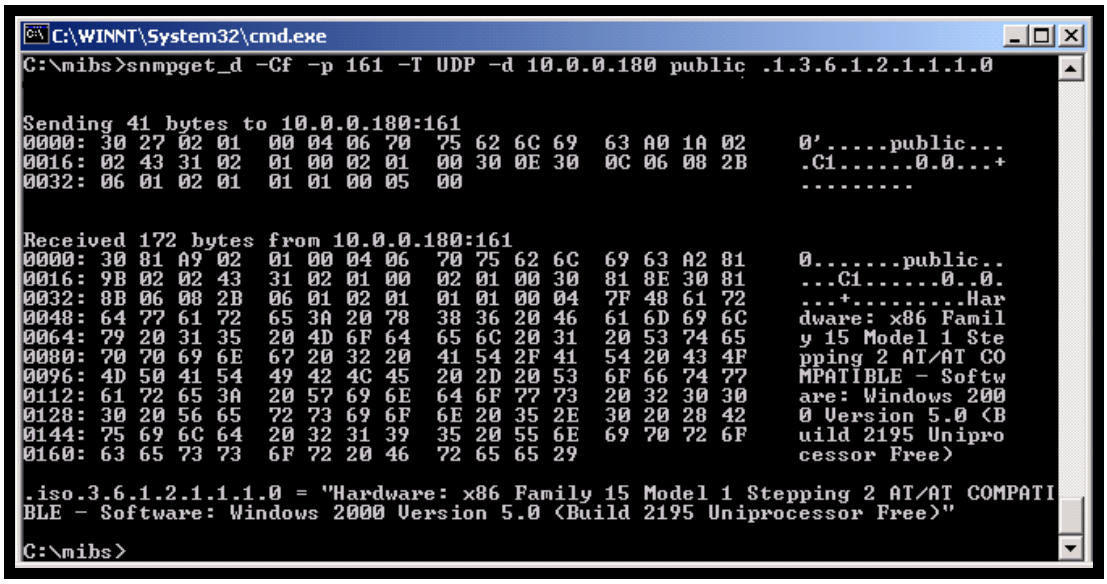


Figura 4.11- Uso de snmpget.

4.4.1.- GRUPOS DE LA MIB.

Es posible observar de los grupos que forman la MIB, en orden correspondiente los siguientes:

- system: Informaciones sobre el equipamiento.
- interfaces: Informaciones sobre las interfases del equipamiento.
- ip: información sobre el protocolo IP.
- tcp: Informaciones sobre el protocolo TCP.

Para acceder a estos grupos, se debe seguir los siguientes pasos: Primero se selecciona la MIB RFC1213, y a continuación, se contacta con el agente SNMP.

4.4.1.1.- GRUPO SYSTEM.

El grupo system es el grupo que contiene información del sistema, algunos de sus objetos son los siguientes:

- sysDescr - Descripción completa del sistema (version, HW, OS).
- sysObjectID - Identificación que da el distribuidor al objeto.
- sysUpTime - Tiempo desde la última reinicialización.
- sysContact - Nombre de la persona que hace de contacto.
- sysName - Nombre del dispositivo.
- sysLocation – Ubicación del dispositivo.
- sysServices - Servicios que ofrece el dispositivo

4.4.1.2.- GRUPO INTERFACES.

El grupo de interfaces tiene la información sobre los interfaces presentes. Como puede haber varios, el grupo de interfaces contiene dos objetos de nivel superior: el número de interfaces del objeto *(ifNumbe*r) y una tabla con información de cada interfaz *(ifTabl*e). Cada entrada de la tabla *(ifEntr*y) contiene información relativa a esa interfaz. Algunos de los objetos que contiene esta tabla son.

- ifType - Tipo de la interfaz
- ifMtu - Tamaño máximo del datagrama IP
- ifInUCastPkts - Nº de paquetes “unicast” recibidos.
- ifInNUCastPkts - Nº de paquetes NO “unicast” recibidos.

4.4.1.3.- GRUPO IP.

Los objetos del grupo es posible observar los siguientes:

- ipForwarding - Indicación de si la entidad es un router IP
- ipInHdrErrors - Número de datagramas de entrada desechados debido a errores en sus cabeceras IP
- ipInAddrErrors - Número de datagramas de entrada desechados debido a errores en sus direcciones IP
- ipInUnknownProtos - Número de datagramas de entrada desechados debido a protocolos desconocidos o no soportados
- ipReasmOKs - Número de datagramas IP reensamblados con éxito.
- ipInReceives - Número de paquetes IP recibidos.
- ipInDelivers - Número total de los datagramas de entrada de información que se han entregado con éxito a la capa de transporte.
- ipOutRequests - Número total de los datagramas IP que se han suministrado desde la capa de transporte.

4.4.1.4.- GRUPO TCP.

Del grupo TCP se observan los siguientes objetos

- tcpRtoAlgorithm - Algoritmo que determina el timeout para retransmitir octetos para los que no se ha recibido reconocimiento.
- tcpMaxConn - Límite en el número de conexiones TCP que puede soportar.
- tcpActiveOpens - Número de veces que las conexiones TCP han efectuado una transición directa del estado CLOSED al estado SYN-SENT.
- tcpInSegs - Número de segmentos recibidos, incluyendo aquellos con error.
- tcpConnRemAddress - La dirección IP remota para esta conexión TCP.
- tcpInErrs - Número de segmentos desechados debido a errores de formato.
- tcpOutRsts - Número de resets generados.

Una vez que se tienen las primitivas asociadas a SNMP, se tiene que probar que es posible modificar las VLANS asociadas.

Para esto se debe de entrar al conmutador y observar los puertos asociados (véase figura 4.12).

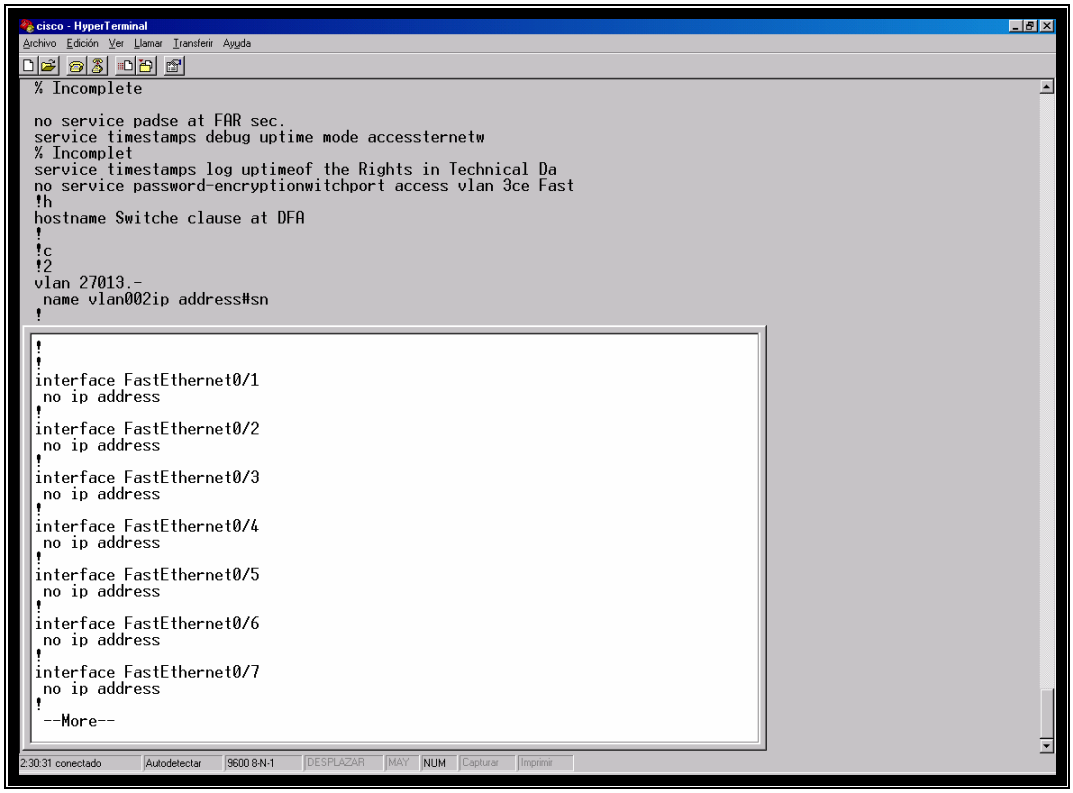


Figura 4.12.- Ningún puerto se encuentra asignado a alguna VLAN.

Se desea que al conmutador se incorporen los siguientes puertos a las VLANS respectivas, de acuerdo a la configuración siguiente:

!
version 12.3
no service single-slot-reload-enable
no service pad
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname Switch
!
!
!INDICAMOS LAS VLANS QUE SE DESEAN MANIPULAR
vlan 2
!
vlan 3
!
ip subnet-zero

vtp mode transparent
!

```
spanning-tree extend system-id
!
!INDICAMOS QUE PUERTOS ASOCIAR A LAS VLANS
interface FastEthernet0/1
 switchport access vlan 2
 switchport mode access
 no ip address
!
interface FastEthernet0/2
 switchport access vlan 3
 switchport mode access
 no ip address

!
interface FastEthernet0/3
 switchport access vlan 2
 switchport mode access
 no ip address
!
interface FastEthernet0/4
 switchport access vlan 3
 switchport mode access
 no ip address
!
interface FastEthernet0/5
 switchport access vlan 2
 switchport mode access
 no ip address
!
interface FastEthernet0/6
 no ip address
!
interface FastEthernet0/7
 no ip address
!
interface FastEthernet0/8
 no ip address
!
interface FastEthernet0/9
 no ip address
!
interface FastEthernet0/10
 no ip address
!
interface FastEthernet0/11
 no ip address
!
interface FastEthernet0/12
 no ip address
!
interface Vlan1
 ip address 10.0.0.170 255.255.255.0
 no ip route-cache
!
ip http server
!
```

!
line con 0
line vty 5 15
!
end

Se verifica que no exista configuración activa para incorporar las VLAN's y los puertos respectivos (véase figura 4.13).

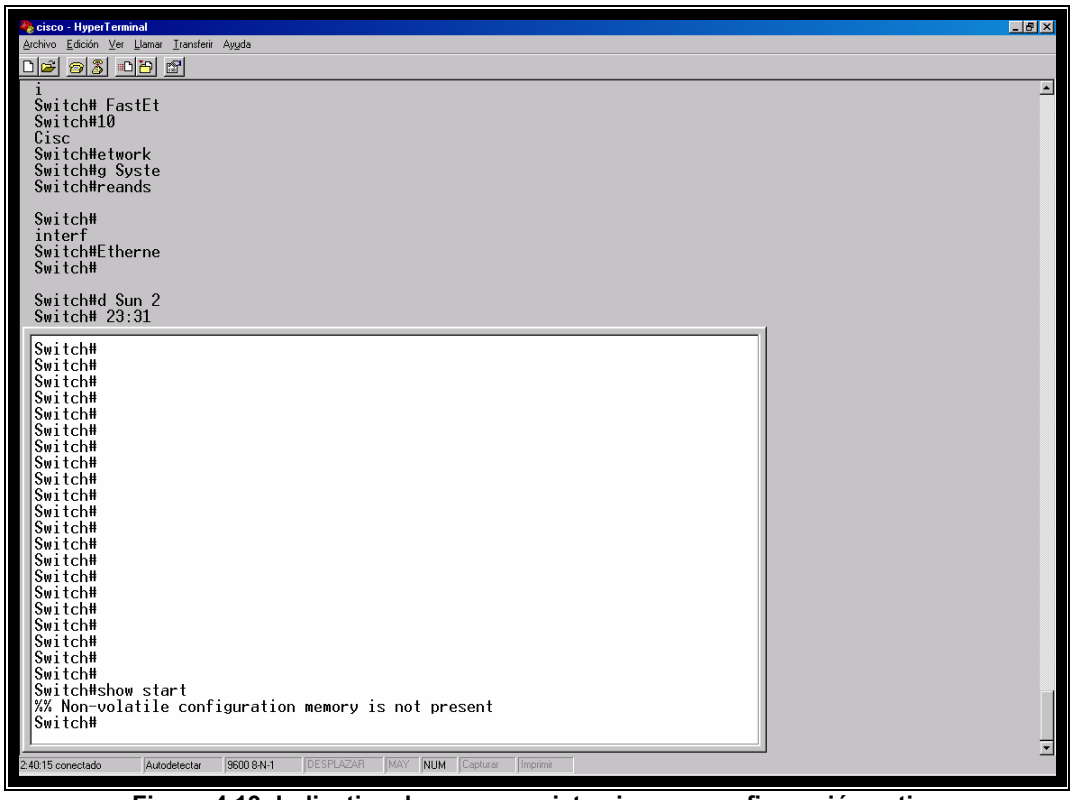


Figura 4.13- Indicativo de que no existe ninguna configuración activa

Seguidamente se modifica la configuración del conmutador y hay que reiniciarlo para que acepte los nuevos cambios (véanse figuras 4.14, 4.15).

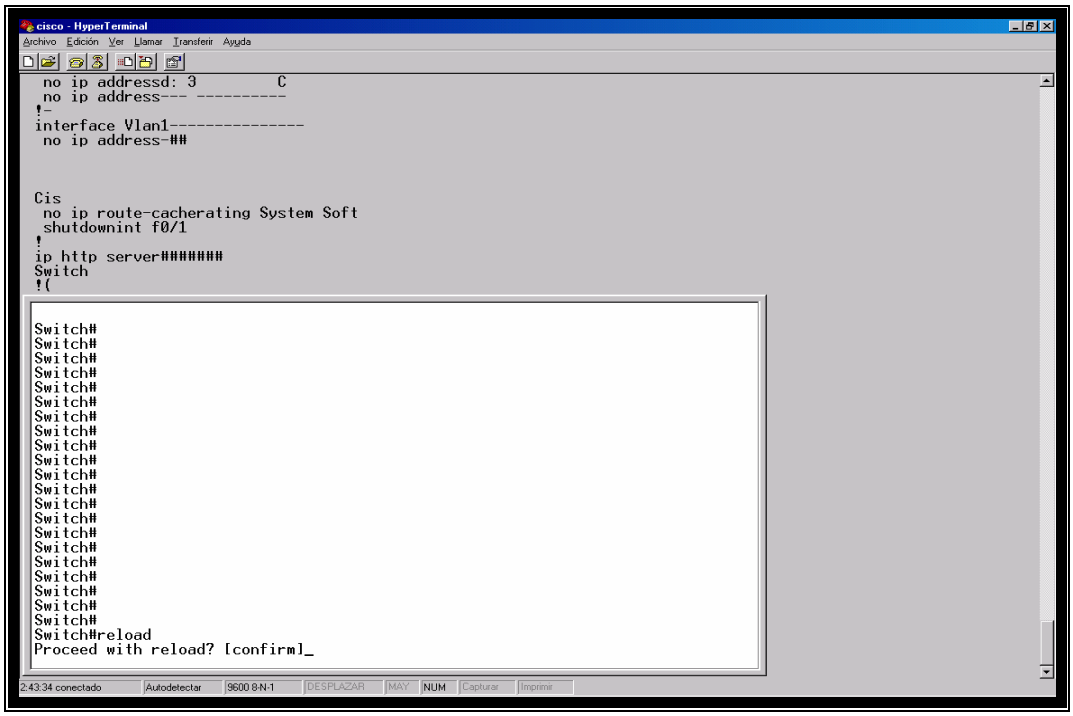


Fig. 4.14.- Reinicio al conmutador para que acepte las nuevas configuraciones.

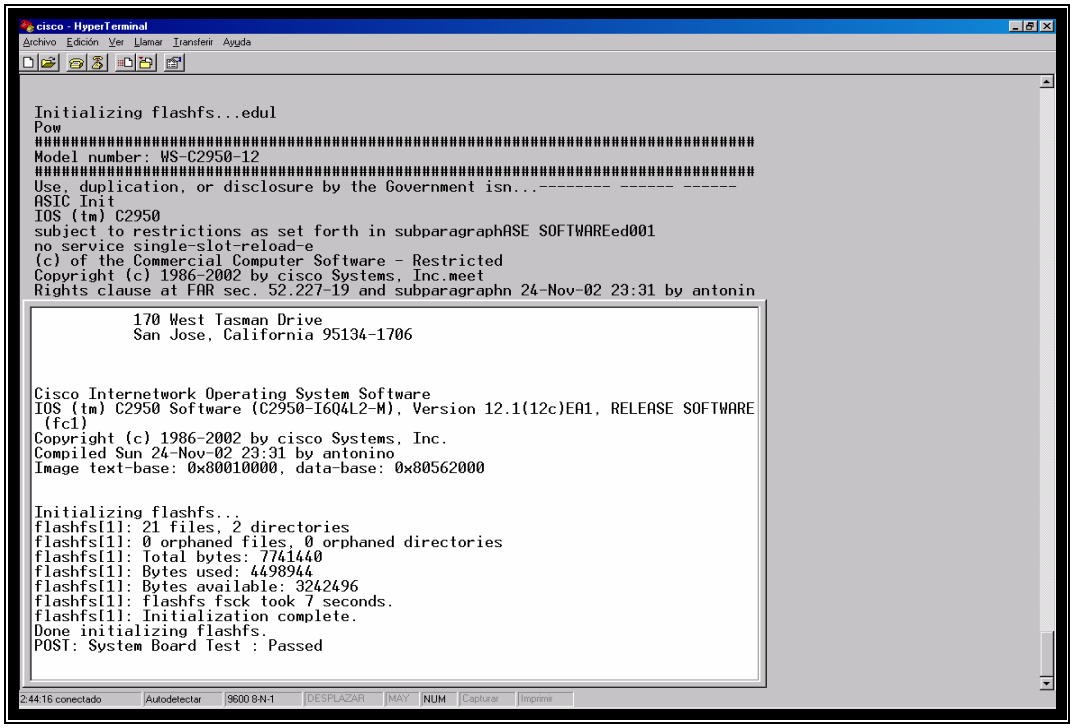


Fig. 4.15.- Reinicio al conmutador para que acepte las nuevas configuraciones.

Ahora se muestran los nuevos puertos y las VLAN's asociadas (véanse figuras 4.16, 4.17).

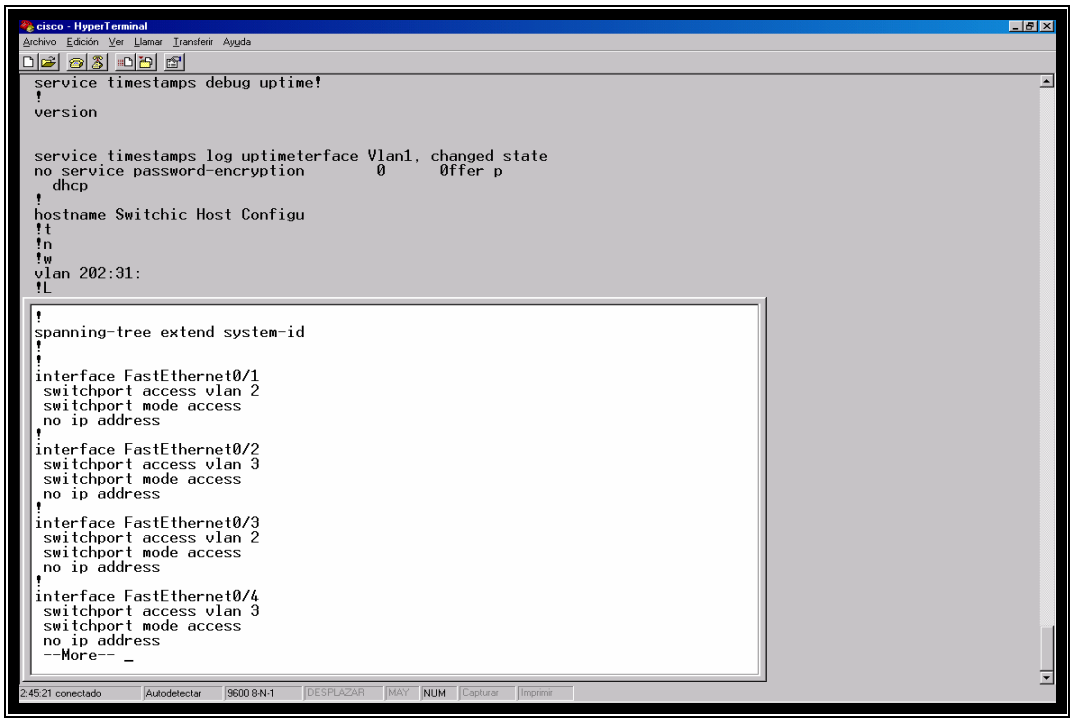


Fig. 4.16.- Las VLAN's y sus puertos asociados.

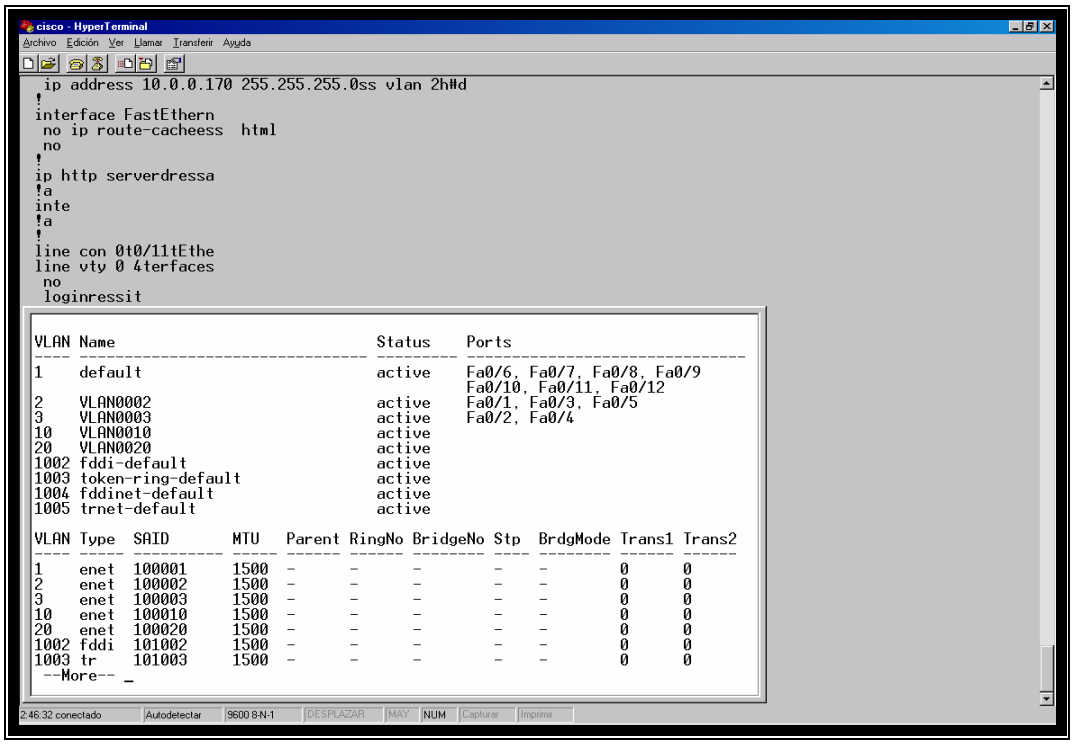


Fig. 4.17.- Las VLAN's y sus puertos asociados(cont..).

4.5.- PROGRAMA DE MONITOREO DE LA HERRAMIENTA DE GESTIÓN DE REDES VIRTUALES.

4.5.1- ESTRUCTURA DEL PROGRAMA DE MONITOREO.

Para que el programa de monitoreo lleve al cabo sus funciones en la herramienta de gestión de VLAN y debido a que diferentes nodos en distintos lugares físicos pueden pertenecer a la misma VLAN, la arquitectura que se emplea para la monitorización consiste en:

4.5.1.1.- UNA ESTACIÓN QUE FUNCIONA COMO GESTOR Y QUE SE CONECTA A LAS VLAN'S.

El programa de monitorización por lo tanto reside en la estación gestora, de tal manera que es posible seleccionar la VLAN la cual se desea monitorizar (véase fig. 4.18,4.19). Con este diseño de monitor, es posible atender a cualquier nodo en cualquier lugar físico.

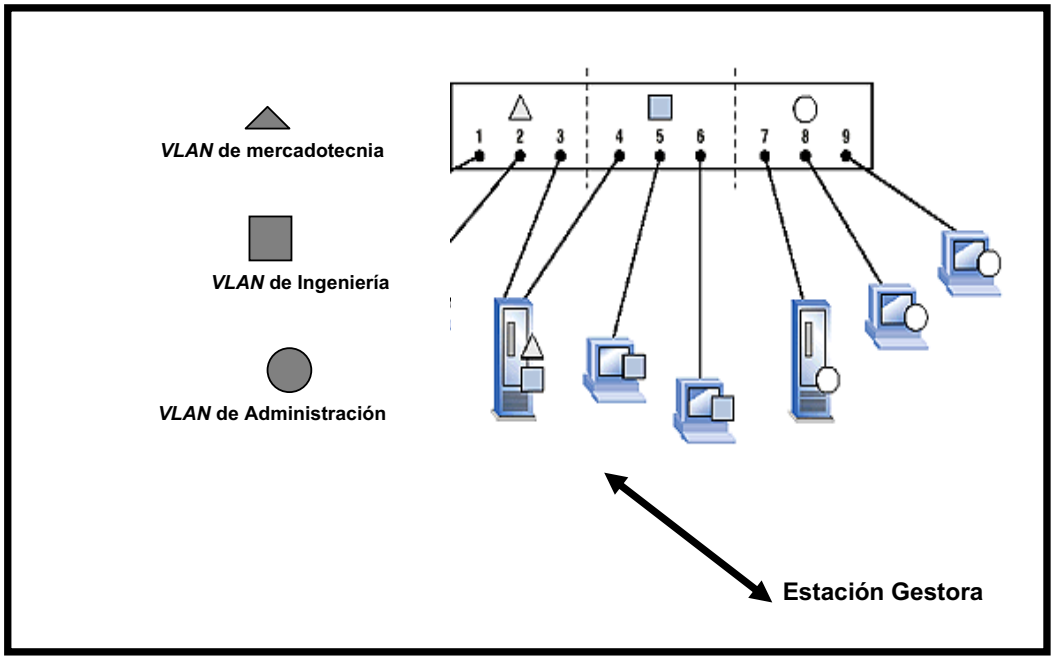


Figura 4.18.- Conexión de la estación gestora para el monitoreo de la VLAN.

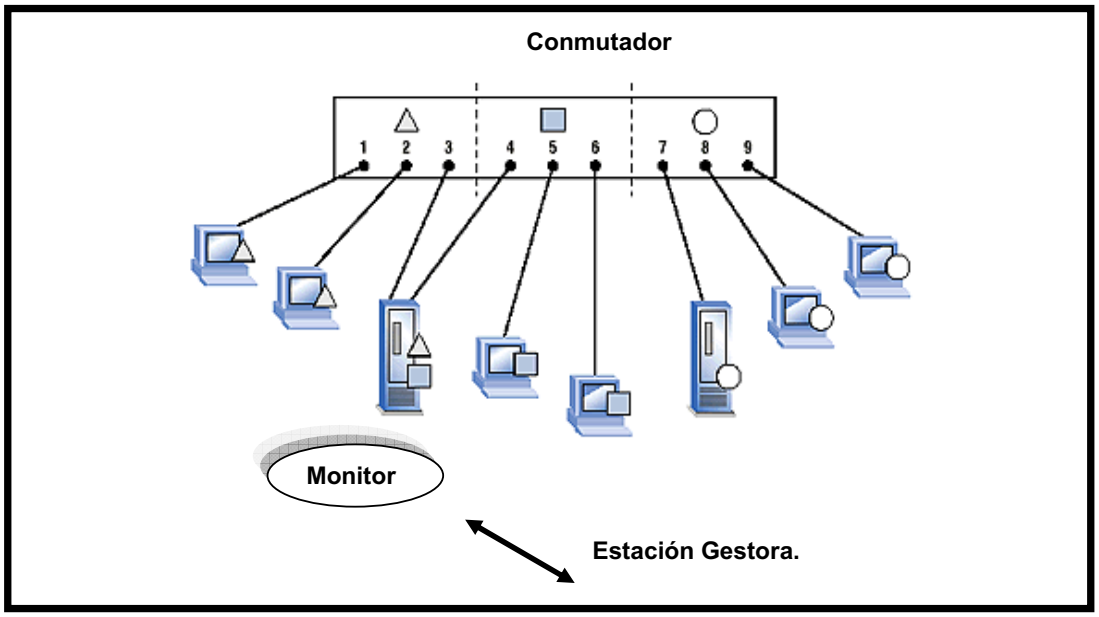


Figura 4.19.- Localización del programa de monitorización en las VLAN´s

Para poder monitorear la VLAN, el programa monitor tomará como base la RFC 3069 y la RFC 2643, en la que se determina que para poder atender la comunicación entre los nodos de la VLAN es necesario que estos pertenezcan a la misma subred, por lo que también se debe de asignar a cada VLAN una subred. Con esto se garantiza la localización de cualquier nodo en la VLAN. Para esto se usará cualquier clase de dirección IP, destacando las clases C y B respectivamente (véase figura.4.1). Para el monitor de Red, es necesario instalar la librería de captura (Wincap3.0.exe) que es necesaria para capturar los paquetes; una vez instalada la librería es posible usar el monitor de red(véase figura 4.20).

Figura 4.20.-Menú opciones MONITOR DE RED de la Herramienta de Gestión de Redes Virtuales.

Una vez que se activa la opción de monitor de red, es posible aplicar los conceptos de filtrado y demás opciones, que se tratan en el capítulo cuatro referente al desarrollo de la Herramienta de Gestión de Redes Virtuales. En principio es posible correr el monitor de red de VLAN y automáticamente detectará el dispositivo y el proceso de escucha en la red comienza (véase figura 4.21). La parte izquierda corresponde a la sección fuente, y la derecha al destino de los datos e información respectivamente(véase figura 4.22).

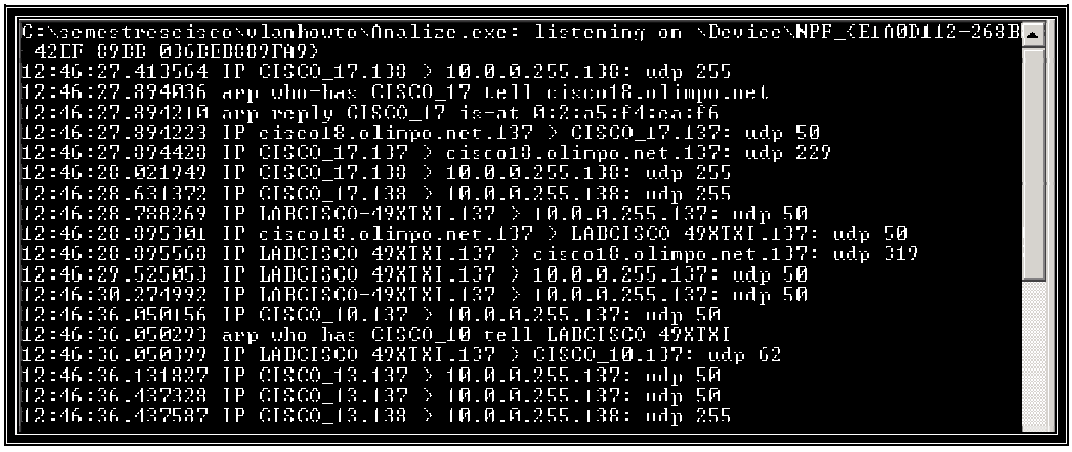


Figura 4.21.- Escuchar algún dispositivo de Capa de de la red.

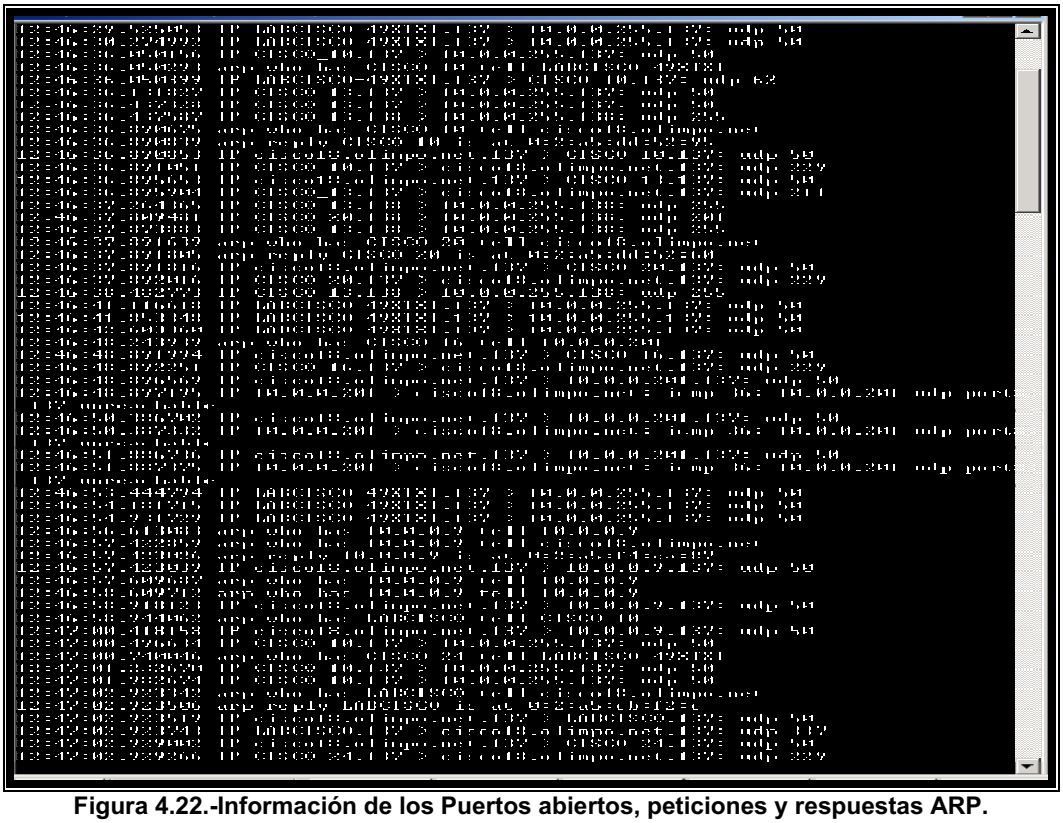


Figura 4.22.-Información de los Puertos abiertos, peticiones y respuestas ARP.

En las siguientes figuras se muestra el análisis de parámetros de filtro que son posibles usar en el analizador (véanse figuras 4.23 a 4.28):

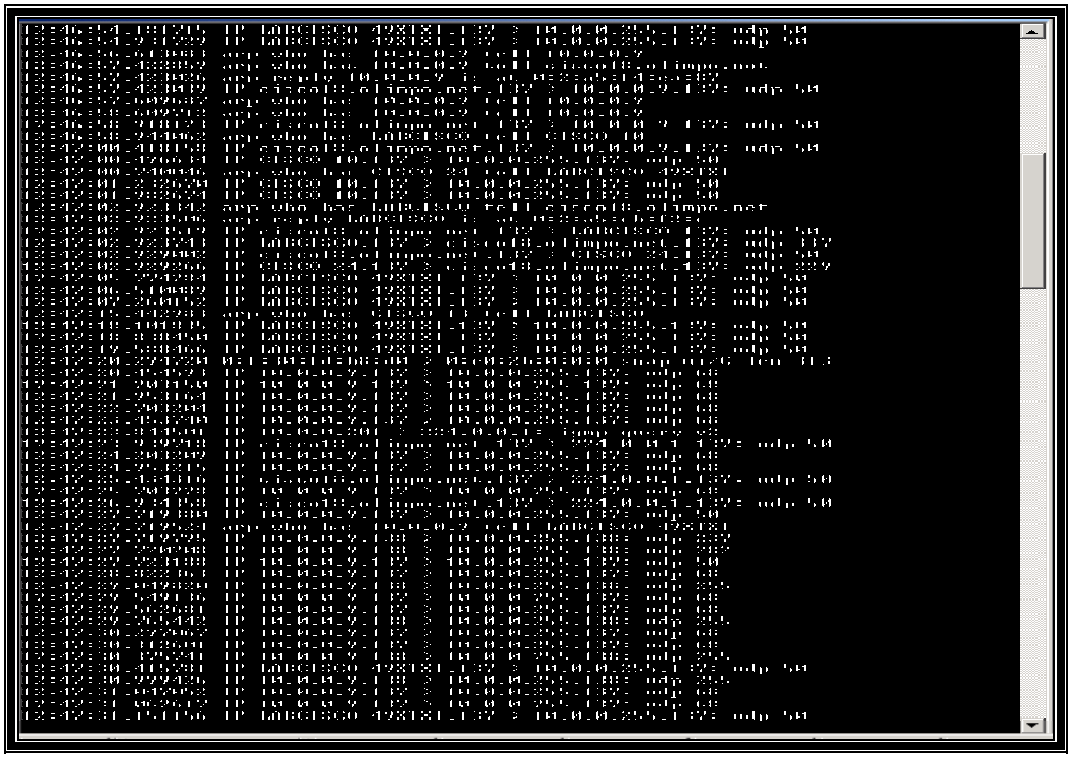


Figura 4.23.-Información de los Puertos abiertos , así como, dirección Fuente y Destino.

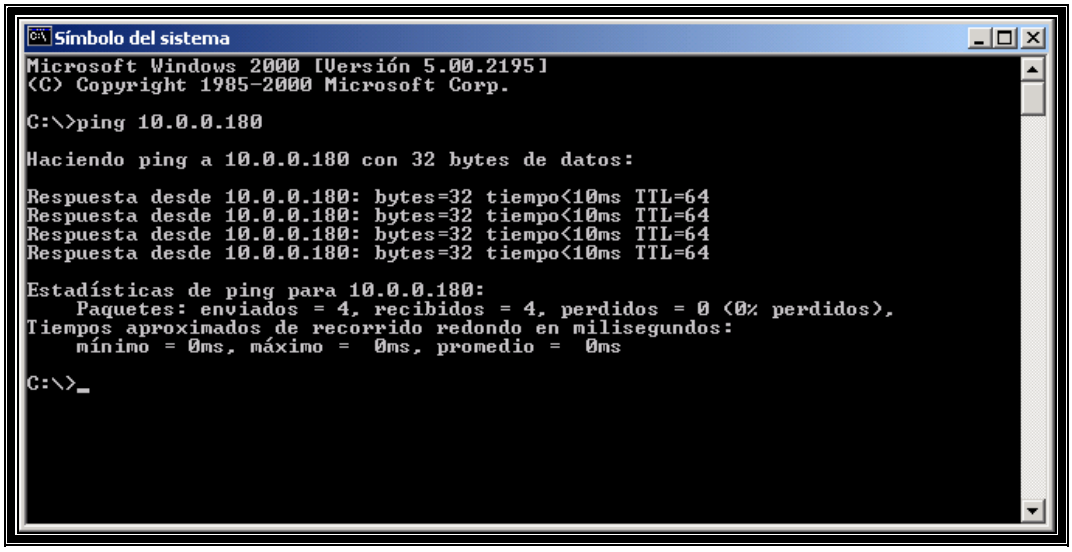


Figura 4.24.- Escaneo aun host especifico.

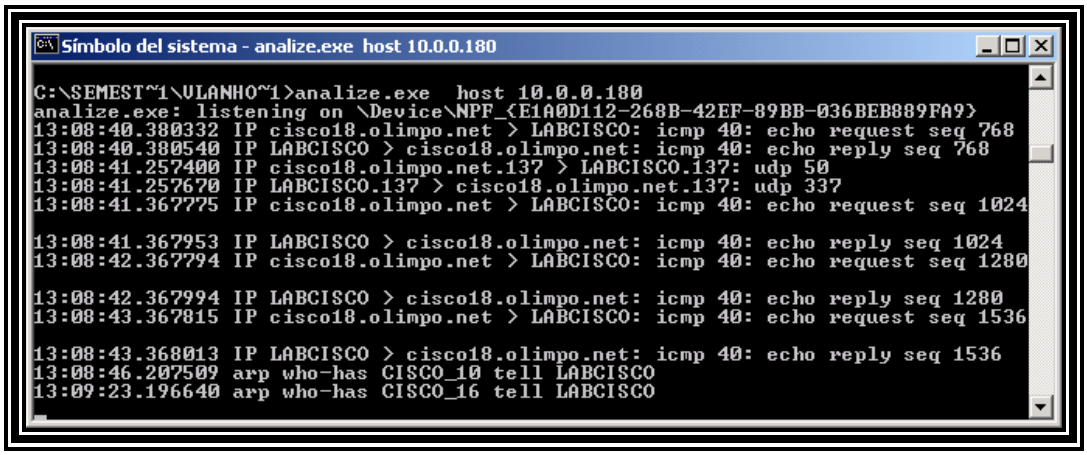


Figura 4.25- Escaneo a un host especifico.

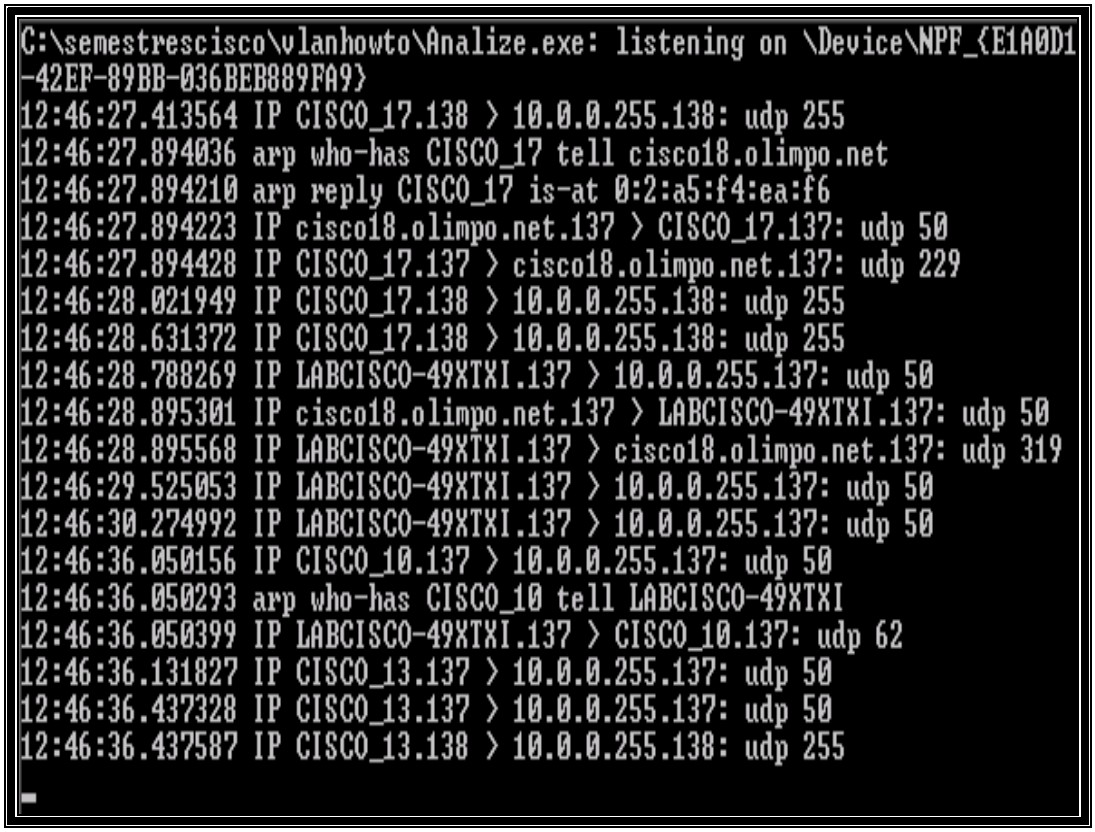


Figura 4.26.-Análisis de una red o subred especifica.

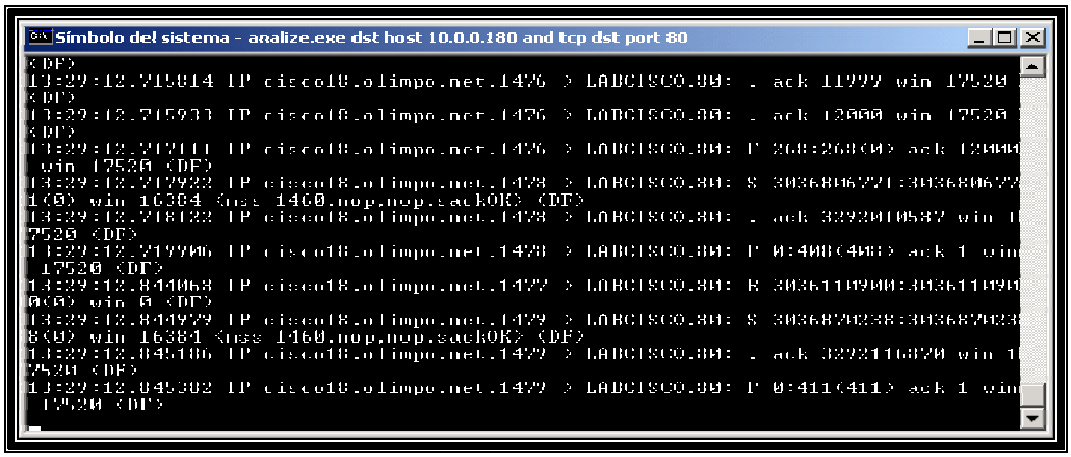


Figura 4.27.-Un puerto o lugar en especifico

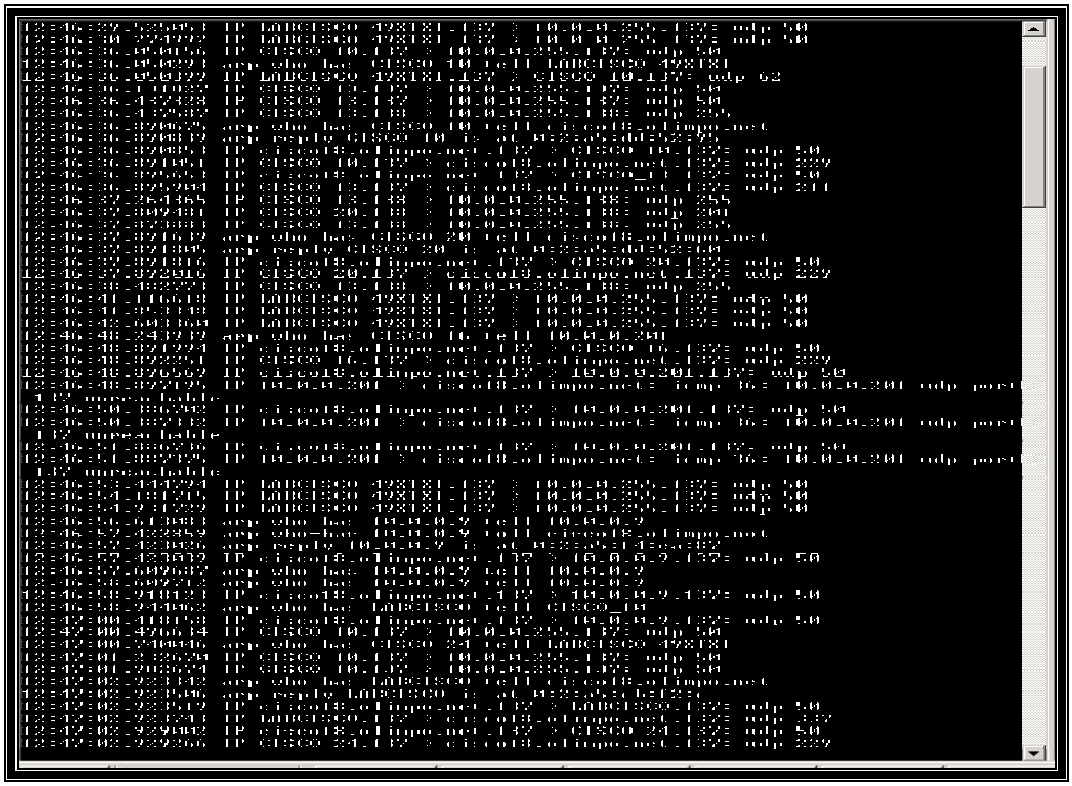


Figura 4.28.- Escuchar algún dispositivo de Capa de la red(cont.).

Una vez que se quiere terminar de monitorizar el programa emite una estadística de paquetes enviados, recibidos y perdidos respectivamente.

Finalmente la opción SNMP VIEWER permite consultar la MIB de cualquier agente de forma gráfica (véase figura 4.29).

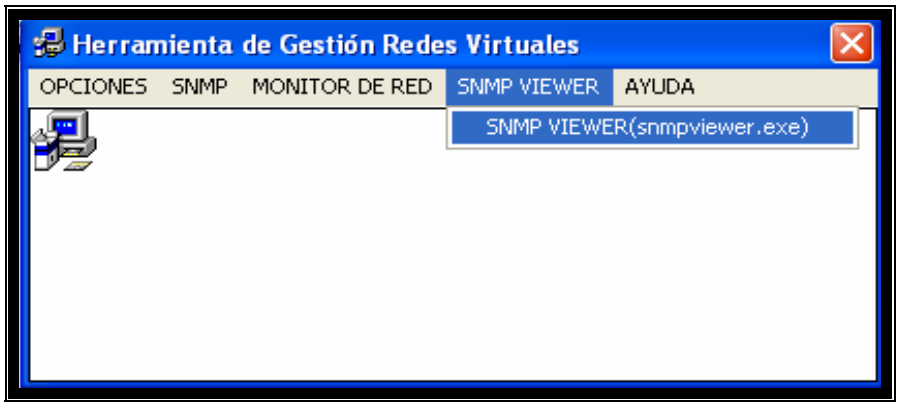


Figura 4.29.- Menú opciones *SNMP VIEWER* de la Herramienta de Gestión de Redes Virtuales.

El menú de ayuda permite revisar lo que se necesita para la puesta a punto de la herramienta de gestión (véase figura 4.30).

Figura 4.30.-Menú opciones AYUDA de la Herramienta de Gestión de Redes Virtuales.

RESUMEN CAPÍTULO IV. PRUEBAS Y RESULTADOS.

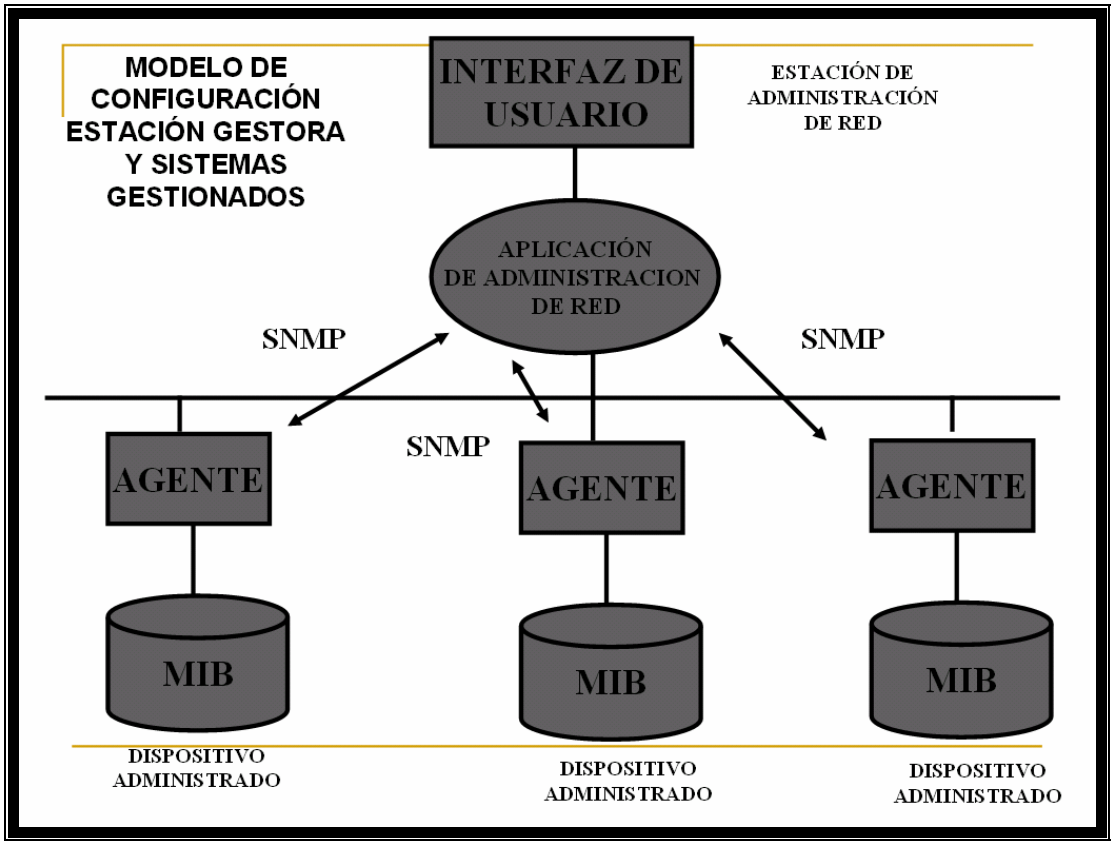
Los resultados de las pruebas aplicadas a los modulo que integran la herramienta de gestión de redes virtuales se realizaron en un entorno de trabajo de red virtual. El entorno de trabajo consistió de una red VLAN la 10.0.0.0/24, junto con los dispositivos de conectividad asociados tales como ruteadores, conmutadores, nodos y demás dispositivos de comunicación.

Los resultados obtenidos muestran que es posible monitorear y acceder a la configurar de cualquier nodo de la red; siempre y cuando exista la posibilidad de encontrarlo como un elemento activo de la misma y se disponga del agente en el respectivo dispositivo.

Es necesario activar el SNMP en los dispositivos para acceder a sus comunidades de lo contrario no es posible obtener ninguna información. Los puestos deben de permanecer abiertos para el intercambio de información (Puertos 161 y 162 asociados a SNMP)

Los módulos funcionaron sin ningún error de codificación ni de ejecución, la prueba de instalación se ejecuto en ambientes Windows 2000/LINUX, Cable aclarar que es posible incrementar la potencialidad de la herramienta incorporando nuevas funcionalidades.

El modelo genérico para el uso de la herramienta de gestión de redes virtuales es en un entorno de procesamiento distribuido lo que implica de manera determinante el que los dispositivos se encuentren conectados (véase figura 4.31).



4.31.- Entorno de Gestión de SNMP y Monitorización en la Herramienta de Gestión.

CONCLUSIONES

A partir del análisis realizado en el presente trabajo, en torno a la herramienta de gestión de redes virtuales se llego al planteamiento de que ésta, es producto de dos grandes vertientes de la gestión de redes, la monitorización y el control. La herramienta de gestión incorpora la monitorización para las funciones de lectura con lo que permite observar, analizar el estado y el comportamiento de la configuración de red y sus componentes. El lo correspondiente al control la herramienta de gestión tiene como objeto la funciones de escritura con lo que permite alterar parámetros de los componentes de la red.

La información que monitoriza la herramienta de gestión de redes virtuales se encuentra en una de las siguientes categorías: **1.- Estática**: Caracterizada por la configuración actual de la red y de sus elementos (p.e. El número de puerto dispositivo de conectividad ó nodo); **2.- Dinámica**: Información relacionada con eventos en la red (p.e. La transmisión de un paquete por la red) ; **3.- Estadística**: Información que puede ser derivada de la información dinámica (p.e. El número medio de paquetes transmitidos por unidad de tiempo por un sistema final).

Con estos tipos de información la herramienta cubre los objetivos de: a) **Identificación de la información** a monitorizar; b) **Diseño de mecanismos de monitorización** para obtener la información y c) **Utilización de la información** obtenida dentro de las distintas áreas funcionales de gestión de red.

La herramienta permite monitorizar diferentes recursos de red destacan:

- **Recursos de comunicación**: LANs, WANs
- **Hardware de computación**: Servidores, estaciones de trabajo, dispositivos de conectividad tales como ruteadores, conmutadores, concentradores, etc.

La información que proporciona la herramienta es recogida y supeditada a las necesidades de la organización. Son algunos ejemplos de la información que se maneja: a) **Identificación de usuario; b) Emisor-Receptor; c) Número de Paquetes; c) Inicio y fin de un proceso de transmisión; protocolos asociados a la capa de transporte del modelo TCP/IP, y otros.**

Para cumplir con esta función, la de monitorizar, la herramienta emplea el análisis de protocolos y analiza a detalle el datagrama IPv4 en cada uno de sus elementos de interés: versión, largo del encabezado, tipo de servicio, largo del datagrama, identificador del paquete, banderas, ofsett de fragmento, tiempo de vida, protocolo de capa de transporte, checksum de encabezado, dirección IP origen, dirección IP Destino, opciones y datos.

Con esta información que obtiene de las tramas que circulan por la red el programa de monitorización emite los resultados en pantalla que son útiles para quien los analiza. Con esto en mente es posible aplicar los filtros de información correspondientes a:

- **Paquetes, cuyo origen o destino sea un host determinado.**
- **Paquetes, cuya dirección origen o destino corresponda a una red determinada.**
- **Paquetes, cuyo puerto de origen sea algún número de puerto.**
- **Paquetes, TCP, UDP y los puertos correspondientes.**
- **Paquetes, cuya dirección destino corresponda a una red determinada.**
- **Cualquier combinación de los anteriores.**

EN lo que respecta al control, la herramienta incorpora un estándar de facto para tal objetivo, el SNMP, que se describe a detalle en el capitulo dos de la tesis. Tomando en consideración el estándar la herramienta tiene un programa gestor, que reside en una estación de gestión que se comunica con los agentes, además de que ofrece al usuario una interfaz gráfica para facilitar el proceso de comunicación con los agentes de gestión para obtener información de los recursos gestionados.

Las primitivas que incorpora la herramienta son: Get, Get-Next, Trap, Set. Con los correspondientes módulos SNMPGET, SNMPSET, SNMPTRAP, SNMPSTATISTICS, SNMPGETNEXT y SNMPVIEWER.

Estas primitivas leen la base de datos asociada al objeto gestionado, la MIB (*Management Information Base*), esta base de objetos se estructura en forma arbórea y permite identificar ciertos objetos de interés por ejemplo número de interfaces y sus correspondientes estados, entre otros. La herramienta lee la MIB en formato de Identificación de Objeto, OID (*Object Idetification*) o en texto, lo que le da una gran flexibilidad en caso de no saber concretamente el OID.

La modificación de un atributo, obviamente, modifica la información de configuración de un **agente**. Las modificaciones que la herramienta realiza a los objetos gestionados se clasifican en tres categorías:

1.- Actualización de la base de datos: Se cambia un valor en la base de datos del **agente** sin que este cambio afecte en nada más a dicho **agente**

2.- Actualización de la base de datos que implica modificación del recurso: Además de cambiar un valor en la base de datos del **agente** la modificación tiene un efecto sobre el recurso.

3.- Actualización de la base de datos que implica una acción: En algunos sistemas de gestión el **gestor** no dispone de “comandos de acción” directos. En vez de estos comandos, disponen de determinados atributos en la base de datos de los **agentes** que cuando se establecen, hacen que el **agente** inicie cierta acción.

Con respecto a la información estadística del dispositivo gestionado el modulo de estadísticas de la herramienta permite obtener información en la categorización siguiente:

- **SYSTEM**: Informaciones sobre el equipamiento.
- **INTERFACES**: Informaciones sobre las interfases del equipamiento.
- **IP**: información sobre el protocolo IP.
- **TCP**: Informaciones sobre el protocolo TCP.

La interfaz gráfica de la herramienta permite un uso fácil de la misma, ya que a través de los menús, se seleccionan las opciones correspondientes dependiendo de la función a realizar: Monitorizar y/o Control. Para facilitar su uso la herramienta incorpora una ayuda por cada una de las opciones de que consta la herramienta, con la posibilidad de enriquecerla y guardar esas adiciones e imprimirla. Así también, la ayuda incorpora ejemplos de uso de cada una de las primitivas o del monitoreo.

Las pruebas de la herramienta se realizaron considerando la VLAN y la división en subredes que marca la RFC 3069 y RFC 2643, en la que se determina que para poder atender la comunicación entre los nodos de la VLAN se hace necesario que estos pertenezcan a la misma subred, por lo que también se debe de asignar a cada VLAN una subred. El capitulo cuatro de la tesis describe a detalle las pruebas realizadas. Cabe destacar que es necesario para las funciones de control activar el agente en los dispositivos de los que se requiere su gestión. Para el caso de la topología de prueba se contó con: un dos ruteadores, un conmutador y las máquinas correspondientes. Con estos dispositivos se puso en funcionamiento la VLAN y se procedió a obtener información de la VLAN.

Los resultados que arroja la herramienta cumplen con los criterios de filtro que el usuario desea emplear, así como que información desea obtener del dispositivo gestionado.

Los resultados obtenidos muestran que es posible monitorear cualquier dispositivo de conectividad que este en la VLAN y acceder a la configuración correspondiente. Los módulos de la herramienta se programaron en C/C++ para facilitar su portabilidad ya que de manera indistinta se emplearon los ambientes Windows 2000 y LINUX. El modelo genérico para el uso de la herramienta de gestión de redes virtuales es en un entorno de procesamiento distribuido lo que implica de manera determinante el que los dispositivos se encuentren conectados.

Es posible Instalar la herramienta sin necesidad de otras configuraciones adicionales, el sistema configura lo necesario para poder ejecutarse. La aportación más significativa en el desarrollo de la herramienta fue el de crear un producto terminado que pude ser de libre uso sin necesidad de comprar licencias ni membresías, además de que es independiente de los mecanismos privativos de protocolos y etiquetados de los proveedores de servicio. En lo que respecta a los objetivos para el caso que nos competen se materializaron en un producto terminado denominado **HERRAMIENTA DE GESTIÓN DE REDES VIRTUALES.** Con respecto al **objetivo general** de desarrollar una herramienta de gestión que permita la configuración y monitorización de una red virtual de área local. La herramienta permite la configuración de la red virtual de área local vía SNMP, junto con toda la programación asociada con respecto a sus primitivas, así como, la monitorización de la misma. Con respecto a los **objetivos específicos**:

1.-Crear una VLAN experimental, de tamaño reducido, para su gestión vía la herramienta de gestión que se pretende desarrollar. La VLAN se armó considerando equipos de conectividad como: conmutadores, ruteadores, interfaces seriales de conectividad de los ruteadores y estaciones de trabajo; así como, las configuraciones asociadas a cada dispositivo.

2.- Diseñar y desarrollar una herramienta para gestionar redes virtuales

> 2.1.- Crear un programa de monitoreo que permita observar las disponibilidades y el uso de la red virtual.
> 2.2.- Crear un programa que permita la configuración-reconfiguración, el control y la consulta de información de la red virtual.

La herramienta se describe en detalle en el capítulo tres, así como una explicación que profundiza en los elementos de programación asociados a la herramienta, que se presenta en el Apéndice A.

3.- Implantar la herramienta de gestión en la red virtual (VLAN) experimental para demostrar su funcionamiento.

Los resultados de la implantación de la herramienta se muestra en el capitulo cuatro de la tesis referente a las Pruebas y Resultados de la herramienta de gestión de redes virtuales

TRABAJOS FUTUROS.

La herramienta puede incorporar módulos extras que permitan de forma gráfica la elección de algún nodo, así como integrar un generador de expresiones para los elementos que se desean monitorear y/o configurar.

En la versión actual de la Herramienta se consideró el IPv4, con un formato de 32 bits, sin embargo, debe de adaptarse a los nuevos retos, con lo que hay que visualizar la posibilidad de incorporar también al IPv6, con 128 bits de formato. Con esta incorporación, la herramienta brindará mayor capacidad de monitorización, extendiendo su alcance a distintos dispositivos sin importar el formato del nivel de capa de red.

Del capitulo uno de la tesis se mencionan otras funciones de la gestión de red, la herramienta brinda la flexibilidad de incorporar nuevos módulos, destaca en importancia el módulo de seguridad que puede ser el modulo de firewall ó corta fuegos, este módulo permitirá que la información que retorna la herramienta no se vea violentada por programas promiscuos, escucha, en la red o que se haga un mal uso de la información de la herramienta.

La herramienta también puede potencializarse si se incorpora un módulo de auditoria para el control de versiones del software que se esta empleando en una organización, con lo que es posible además detectar software sin registro y tomar las medidas precautorias correspondientes.

Es posible en cuanto a la programación generar librerías y componentes que sean reutilizables en cualquier programa que necesite monitorear o controlar. Esto aportara el desarrollo de aplicaciones rápido, ya que como cualquier objeto que se inserta en un lenguaje, el componente tendrá la capacidad de autoconfigurarse en base a una lista de propiedades y métodos que pueden ser invocados para las funciones propias de la gestión de red.

BIBLIOGRAFÍA

LIBROS

[1] García Tomás Jesús. Redes y Procesos Distribuidos. Edit. Prentice-Hall, México,2003.

[2] López Angel. Protocolos de Internet. Edit. Alfaomega, México, 2003.

[3] Stallings W. Comunicaciones y redes de computadoras. Prentice-Hall, 2003.

[4] Stallings W. SNMP, SNMPv2, SNMPv3, RMON 1 y 2. Prentice-Hall, 2003.

[5] Liu, Cricket et all. Managing Internet Information Services. O'Relly Associates, Inc. Sebastopol, CA 95472. (2000)

[6]Stevens, W. Richard. TCP/IP Illustrated Vol.1. Addison-Wesley. (2000)

[7] Kirch, Olaf. The Linux Network Administrators Guide. (2001).

[8] Brandan P. Kehoe. Zen and the Art of the Internet. Dpto. Publicaciones E.U.I. Madrid (2003).

[9] E. Thomsen. OLAP Solutions: Building Multidimensional Information Systems. John Wiley & Sons, 2000.

[10] Liu, Cricket et all. Managing Internet Information Services. O'Relly Associates, Inc. Sebastopol, CA 95472. (2000)

[11] Craig Hunt. Networking Personal Computers, whit TCP/IP. O'Relly Associates, Inc. Sebastopol, CA 95472. (2003)

[12] Huitema, Christian. Routing in the Internet. Prentice Hall: Englewood Cliffs, NJ (2001).

[13] Cheswick, William R. and Steven M. Bellovin. Firewalls and Internet Security. Addison-Wesley Publishing Company. (2003)

[14] Daniel Calzada del Fresno. Servicios de Internet. Dpto. Publicaciones E.U.I Madrid. (2000)

[15] Comer, Douglas E . Internetworking With TCP/IP. Prentice Hall: Englewood Cliffs, NJ. (2000).

RFC'S

[16] RFC 3069. VLAN Aggregation for Efficient IP Address Allocation D. McPherson, B. Dykes [February 2001]

[17] RFC 2643. Cabletron's Secure Fast VLAN Operational Model D. Ruffen, T. Len, J. Yanacek [August 1999]

[18] RFC 2641. Cabletron's Vlan Hello Protocol Specification Version 4 D. Hamilton, D. Ruffen [August 1999]

[19] RFCs 1155, 1157 y 1212 para SNMPv1.

[20] RFCs 1441 a 1452 para SNMPv2.

[21] RFC 1213 para MIB y MIB-II.

ARTICULOS

[22] David Passmore y John Freeman, "*The Virtual LAN Technological Report",* http://www.3com.com/nsc/200374.html. 2002

[23] "3com *Transcend* VLANs", http://www.3com.com/nsc/200375.html.2001

[24] "Cisco VLAN *Readmap*", http://www.cisco.com/warp/public/538/7.html. 2001

[25] Kahle, B. Overview of Wide Area Information Servers. Thinking Machines. <URL:ftp://ftp.wais.com/>. 2001.

[26] G. COLOURIS, J. DOLLIMORE., T. KINDBERG. "Distributed Systems Concepts and Design". http://www.eui.upm.es/jesteinv/asign/sis_dis.htm. 2001

[27] Cisco - Configuring EtherChannel and 802_1q Trunking Between Catalyst 2950 and Catalyst Switches Running CatOS, *www.cisco.com/soporte/redes.html. 2002*

[28] Cisco - IEEE 802_10 VLAN Encapsulation, *www.cisco.com/soporte/redes.html. 2002*

[29] Cisco - VLAN Standardization via IEEE 802_10_archivos, *www.cisco.com/soporte/redes.html. 2002*

[30] Virtual LANs & Trunking 802_1Q, *www.cisco.com/soporte/redes.html.2002*

[31] LAN Switching, *www.cisco.com/soporte/redes.html. 2002*

[32] Cisco - Configuring InterVLAN Routing and ISL-802_1Q Trunking on a Catalyst 2900XL-3500XL-2950 Switch Using An External Router_archivos

INTERNET

[33] www-uk/~geiweb/fjmm/vlans.html

[34] www.uk/~geiweb/fjmm/tesauro/vlans.html

[35] www.uk/~geiweb/fjmm/tesauro/vlans.html

[36] http://infase.es/FORMACION/INTERNET/tcpip.html#RTFToC13ma

[37] www.info.com/soporte/redes.html

APÉNDICE A.
CODIGO FUENTE DE LA HERRAMIENTA DE GESTION DE REDES VIRTUALES

A.1. DESARROLLO DEL PROGRAMA DE MONITOREO (BIBLIOTECAS DE DESARROLLO PARA LA HERRAMIENTA DE GESTIÓN DE REDES VIRTUALES).

La biblioteca libnet (Network Library) simplifica la creación y envío de datagramas IP, mensajes ICMP, paquetes UDP y tramas TCP. Las funciones de interfaz de programación utilizadas en la programación de la herramienta de gestión son:

1.- u_char * host_lookup (u_long in, u_short use_name);

Convierte la dirección de red Ipv4 pasada a través del argumento en formato numérico interno a una cadena de texto. Si el argumento "use_name" es un 1 se intenta devolver un nombre, si no es posible realizar la resolución del nombre se devuelve la representación numérica de la dirección IP.

2.- void host_lookup_r(u_long in, u_short use_name, u_char *buf};

Esta función cumple la misma tarea que la anterior, con la diferencia de que devuelve la cadena en un buffer pasado por el usuario como argumento buf, en vez de devolver la cadena de texto en un buffer estático como en el caso anterior.

3.- u_long name_resolve(u_char *hostname, u_short use_name);

Devuelve la representación numérica interna de la dirección de red pasada como argumento en hostname. Esta dirección de red puede ser una cadena que represente una dirección IP numérica, o bien puede ser un nombre de host. En este último caso se debe indicar que se está pasando un nombre de host mediante un valor 1 en el argumento use_name.

4.- int open_raw _sock(int protocol);

Abre un socket raw para el envío de información mediante el protocolo indicado como parámetro. Esta función establece en el socket creado la opción IP _HDRINCL, la cual evita que la pila de protocolos TCP/IP del sistema incluya la cabecera IP a la información que se envíe por este socket, ya que se supone que el usuario ha confeccionado la cabecera IP y esta incluida en la información enviada.

5.-struct link_int *open_link_interface(char *device, char *ebuf);

Cuando se deseen enviar tramas del nivel de enlace se debe usar esta función para abrir una interfaz a nivel de enlace. El parámetro "device" indica el nombre del dispositivo de red que se usa, y el parámetro "ebuf" es un buffer donde se almacenan los mensajes de error provocados por la función.

```
structlink_int
{
int id; /*link layer file descriptor */
int linktype; /* link type */
int linkoffset; /* link header size */
u_char *device; /* device name */
} ;
```

6.- void build_tcp(u_short sport, u_short dport, u_long seq, u_long ack, u_char control, u_short win, u_short urg, const u_char *payload, int payload_s, u_char *buf);

Construye un segmento TCP. Los argumentos pasados a la función corresponden a los diferentes campos que constituyen el segmento:

- Puerto de origen (sport)
- Puerto de destino (dport)
- Número de secuencia (seq)
- Número de aceptación (ack)
- Flags (control)
- Ventana (win)
- Puntero a datos urgentes (urg)
- Datos (payload)
- Tamaño de los datos (payload_s)

Por último, el parámetro buf es un puntero a un buffer de memoria donde la función almacena el segmento TCP una vez construido.

7.- void build_udp(u_short sport, u_short dport, const u_char *payload, int payload_s, u_char *buf);

Construye un paquete UDP. Cada argumento corresponde a los siguientes campos del paquete:

- Puerto origen (sport)
- Puerto destino (dport)
- Datos (payload)
- Longitud de los datos (payload_s)

El parámetro buf, como en el caso anterior, es un puntero a un buffer de memoria que contiene el paquete UDP una vez construido.

8.- void build_icmp_echo(u_char type, u_char code, u_short id, u_short seq, const u_char *payload, int payload_s, u_char *buf);

Permite la construcción de mensajes ICMP del tipo ECHO y ECHOREPL. Los argumentos son los siguientes:

- Tipo de mensaje ICMP (type)
- Código identificador dentro del tipo de mensaje ICMP (code)
- Identificador de mensaje (id)
- Número de secuencia (seq)
- Datos (payload)
- Longitud de los datos (payload_s)

El parámetro buf es el puntero al buffer donde se almacena el mensaje.

9.- void build_igmp(u_char type, u_char code, u_long ip, u_char *buf);

Construye un paquete del protocolo de de administración del grupo de internet, IGMP(*Internet Group Management Protocol*). Los parámetros son:

- Tipo de paquete (type)
- Código (code)
- Dirección de clase D (ip)

El parámetro buf es un puntero al buffer donde se almacena el paquete.

10.- void build_ip(u_short leo, u_char tos, u_short id, u_short frag, u_char ttl, u_char prot, u_long saddr, u_long daddr, const u_char *payload, int payload_s, u_char *buf);

Construye un datagrama IP. Los parámetros de la función se corresponden con los diferentes campos del datagrama:

- Longitud del datagrama sin incluir la cabecera (len)
- Flags identificativos del tipo de servicio (tos)
- Identificador del datagrama (id)
- Flags de fragmentación y desplazamiento del fragmento (frag)
- Tiempo de vida del datagrama (ttl)
- Protocolo de nivel superior transportado (prot)
- Dirección de red origen (saddr)
- Dirección de red destino (daddr)
- Datos (payload)
- Longitud de los datos (payload_s)

El parámetro buf es un puntero al buffer que contiene el datagrama IP una vez creado.

Los campos payload y payload_s no deben usarse al construir datagramas que transporten paquetes UDP, segmentos TCP o mensajes ICMP.

11.- void build_ethernet(u_char *daddr, u_char *saddr, u_short id, const u_char *payload, int payload_s, u_char *buf);

Construye una trama Ethernet. Se deben pasar como parámetros a la función los siguientes datos para rellenar los campos de la trama:

- Dirección MAC destino (daddr)
- Dirección MAC origen (saddr)
- Tipo de trama (id)
- Datos (payload)
- Longitud de los datos (payload_s)

El parámetro buf; como en todos los casos anteriores, es un puntero a un buffer donde se almacena la trama construida.

12.- void build_arp(u_short hrd, u_short pro, u_short hln, u_short pln, u_short op, u_char *sha, u_char *spa, u_char *tha, u_char *tpa, const u_char *payload, int payload_s, u_char*buf);

Construye un mensaje ARP. Los parámetros a indicar para construir el mensaje son los siguientes:

- Tipo de dirección hardware (hrd)
- Tipo de dirección de red (pro)
- Longitud de las direcciones hardware (hln)
- Longitud de las direcciones de red (pln)
- Tipo de mensaje ARP (op)
- Dirección hardware (MAC) de origen (sha)
- Dirección de red (IP) de origen (spa)
- Dirección hardware (MAC) de destino (tha)
- Dirección de red (IP) de destino (tpa)

- Datos (payload)
- Longitud de los datos (payload_s)

El parámetro buf es un puntero al buffer donde se almacena el mensaje ARP una vez construido.

13.- int insert_ipo(struct ipoption *opt, u_char opt_len, u_char *buf);

Inserta opciones en un datagrama IP ya creado. El parámetro opt es un puntero a un estructura especial ipoption. El parámetro len indica el tamaño de la lista de opciones. El parámetro buf es un puntero al buffer que contiene el datagrama IP al que se le añaden las opciones. Si al añadir las opciones, el tamaño total del datagrama excede de 65535 bytes, que es el tamaño máximo de un datagrama IP, la función retorna con un valor de -1 indicando el error.

```
struct ipoption
{
struct in_addr ipopt_dst;
char ipopt_list [MAX_IPOPTLEN];
} ;
```

14.- int insert_tcpo(struct tcpoption *opt, u_char opt_len, u_char *buf);

Inserta opciones en un segmento TCP ya creado. El parámetro opt es un puntero a una estructura tcpoption donde se indican las opciones. La longitud de la lista de opciones se indica por opt_len. El parámetro buf es un puntero al buffer que contiene el segmento TCP ya creado.

```
struct tcpoption
{
        u_char tcpopt_list[MAX_IPOPTLEN];
};
```

15.- int write_ip(mt SOCk, const U_Char "'pak, mt len);

Envía un datagrama IP a la red. El parámetro sock hace referencia al socket a través del cual se envia el datagrama. Este socket ha sido creado con anterioridad mediante la función open_raw_sock(). El parámetro pak es un puntero al datagrama que se envia. Este datagrama ha sido creado con la función build_ip(). El tamaño total del datagrama se indica con el parámetro len.

La función devuelve el total de bytes escritos.

16.- int write_link_layer(struct link_int *1, const u_char *device, const u_char *buf, int len);

Envía una trama de nivel de enlace a la red. El primer parámetro es un puntero a una estructura link_int que se debe haber obtenido con anterioridad mediante la función open_link_interface(). El parámetro device es el nombre del dispositivo de red por el cual se envia la trama, la cual se indica mediante el parámetro buf que es un puntero al buffer que contiene la trama a enviar. El tamaño total de la trama se indica con el parámetro len.

La función devuelve el total de byte s escritos o un valor de -1 en caso de error.

17.- void do_checksum(u_char *buf, int protocol, int len);

Realiza el cálculo del campo de checksum que aparece en las cabeceras de segmentos TCP y paquetes UDP.

El primer parámetro es un puntero al buffer que contiene los datos del paquete sobre el cual se calcula la suma de control. El tamaño del paquete se indica con el parámetro len. El tipo de paquete (el protocolo de transporte) se indica con el parámetro protocol.

18.- u_short ip_check(u_short *buf, int len)

Calcula el checksum de un datagrama. El parámetro buf es un puntero al buffer que contiene los datos. La longitud de los datos se indica con el parámetro len.

19.- u_short tcp_check(struct tcphdr *th, int len, u_long src, u_long dst);

Calcula el checksum de la cabecera TCP. En el parámetro th se le pasa un puntero a la cabecera y datos del segmento TCP. La longitud del segmento se indica con len. La dirección de red (IP) origen y destino se indican con los parámetros src y dst respectivamente.

La función devuelve el checksum.

20- int seed_prand();

Alimenta la semilla para la generación de números pseudoaleatorios.

21.- u_long get_prand(int type);

Devuelve un número aleatorio. El parámetro type indica el tipo de número que se desea obtener (véase figura. 3.17):

- Un 1 o un 0 (PR2)
- Un byte (PR8)
- Entero corto con signo (0 a 32767) (PR16)
- Entero corto sin signo (0 a 65535) (PRu16)
- Entero largo con signo (0 a 2147483647) (PR32)
- Entero largo sin signo (0 a 4294967295) (PRu32)

Longitud de las Cabeceras	
Constante	**Descripcion**
ICMP_H	Cabecera de mensajes ICMP
ICMP_ECHO_H	Cabecera de los mensajes ICMP de tipo ECHO y ECHOREPLY
TCP_H	Cabecera de los segmentos TCP
UDP_H	Cabecera de los paquetes UDP
IP_H	cabecera de datagrama IP
ARP_H	Cabecera de mensaje ARP
ETH_H	Cabecera de trama Ethernet
P_H	Pseudo-cabecera TCP/UDP. Esta pseudo-cabecera se usa para calcular el check sum de los segmentos TCP y de los paquetes UDP. La pseudo-cabecera consta de la direccion IP de origen y destino, y el campo de longitud de la cabecera TCP o del paquete UDP.

Figura 3.17.- Constantes Definidas en Libnet

#define ARP_H 0xlc
#define ETH_H 0xe
#define UDP_H 0x8

```
#define TCP H 0x14
#define IP_H 0x14
#define P_H 0xc
#define ICMP_H 0x4
```

Pseudo-cabecera TCP, UDP(véase figura. 3.18):

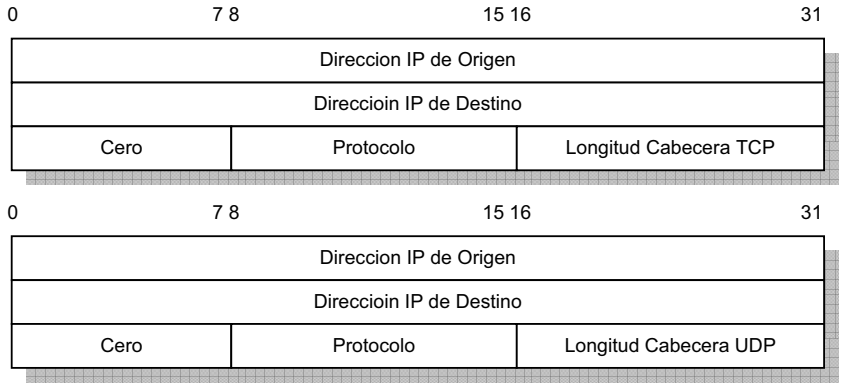


Figura 3.18.- Pseudo-cabecera TCP, UDP

Pseudo-cabecera UDP:

Definición de la pseudo-cabecera consta de(véase figura. 3.19 a 3.20):

```
struct psuedohdr
{
u_lon9 ip_src; /* source IP address */
u_lon9 ip_dst; /* destlnation IP address */
u_char null; /* p:adding octect */
u_char protocol; /* TCP or UDP */
u_short len; /* packet size */
} ;
```

TAMAÑOS MAS COMUNES	
Constante	**Descripción**
PACKET	Tamaño estandar de paquete PACKET = IP_H + TCP_H
OPTS	Tamaño maximo de la lista de opciones en el datagrama IP
MAX_PACKET	Tamaño maximo de un datagrama IP en IPv4

Figura 3.19.- Tamaños Comunes

```
#define PACKET  IP_H + TCP_H
#define OPTS      MAX_IPOTLEN
#define MAX_PACKET  IP_MAXPACKET
```

TIPO DE SERVICIO EN LA CABECERA DEL DATAGRAMA IP	
Constante	**Descripción**
IPTOS_LOWDELAY	Minimizar el retardo
IPTOS_THROUGHPUT	Maximizar el rendimiento
IPTOS_RELIABILITY	Maximizar la fiabilidad
IPTOS_MINCOST	Minimizar el coste

Figura 3.20.- Tipo de servicio en la cabecera del datagrama IP

```
#define IPTOS_TOS_MASK                  0x1E
#define IPTOS_TOS                       ((tos)   & IPTOS_TOS~SK)
#define IPTOS_LOWDELAY                  0xf0
#define IPTOS_THROUGHPUT                        0x0B
#define IPTOS_RELIABILITY               0x04
#define IPTOS_COST                      0x02
#define IPTOS_MINOST                    I       PTOS_LOWCOST

#define rpTOS_PREC_MASK                 0xe0
#define IPTOS_PREC (tos)                ((tos) & IPTOS_PRECJ'lASK)
#define IPTOS_PREC_ETCONTROL            0xe0
#define IPTOS_PREC_INTERNETCONTROL      0xc0
#define IPTOS_PREC_CRITIC_ECP           0xa0
#define IPTOS_PREC_FLASHOVERRIDE        0x80
#define IPTOS_PREC_FLASH                0x60
#define IPTOS_PREC_IMMEDIATE            0x40
#define IPTOS_PREC_PRIORITY             0x20
#define IPTOS_PREC _ROUTINE             0x00
```

FLAGS DE FRAGMENTACION EN LA CABECERA DEL DATAGRAMA IP	
Constante	**Descripcion**
IP_DF	No fragmentar este datagrama
IP_MF	Hay mas fragmentos del mensaje

Figura 3.21.- Flags de Fragmentación

```
#defineIP_DF 0x4000
#defineIP_MF 0x2000
```

FLAGS EN LA CABECERA DEL SEGMENTO TCP	
Constante	**Descripción**
TH_URG	Se transportan datos urgentes
TH_ACK	El campo de numero de secuencia ACK debe tenerse en cuenta
TCP_PSH	Se desea que la pila TCP/IP remota facilite los datos al nivel de aplicación con la mayor prioridad posible
TH_RST	Abortar la conexión
TH_SYN	Sincronizacion de la conexión. Enviando normalmente en los primeros segmentos TCP para el establecimiento de la conexión
TH_FIN	Finalizar la conexión. Enviando normalmente en el ultimo segmento TCP para la liberación de la conexion

Figura 3.22.- Flags en cabecera de segmento TCP.

```
#define TH_FIN          0x0l
```

```
#define TH_SYN      0x02
#define TH_RST      0x04
#define TH_PUSH     0x08
#define TH_ACK      0x10
#define TH_URG      0x20
```

A.2.- EL MONITOR DE LA HERRAMIENTA DE GESTIÓN DE REDES VIRTUALES

La captura y análisis de todos los paquetes transmitidos puede ser, en ocasiones, necesaria si se quiere monitorizar la red para comprobar su carga y realizar estadísticas de uso, o bien cuando se desee comprobar que un funcionamiento incorrecto de la red no se debe a la transmisión de tramas erróneas por parte de un programa o dispositivo mal configurado.

Cuando se desee realizar una captura de paquetes a bajo nivel, se deben usar las facilidades que proporcione el sistema operativo para tal fin. Algunos ejemplos son (véase figura 3.23):

Método	Sistema Operativo
BPF (Berkeley Packet Filter)	Variantes de BSD
DLPI (Data Link Provider Interface)	Solaris, HP-UX, SCO Openserver
NIT (Network Interface Tap)	SunOS
SNOOP	Iris
SNIT (Streams Networks Interface Tap)	
SOCKET_PACKET	Linux/Windows
LSF (Linux Socket Filter)	Linux/Windows

Figura 3.23.- Métodos de acceso por Sistema Operativo

En el programa de monitoreo utiliza SOCK_PACKET, ya que la plataforma de desarrollo principal en la que se ha centrado el desarrollo es Linux/Windows. Si se desea tener una mayor portabilidad entre plataformas de este tipo de programas, se usa una biblioteca que aísle la aplicación de las particularidades de cada sistema operativo a la hora de capturar paquetes. Ésta es la función de libpcap, la biblioteca de uso de la Herramienta de Gestión de Redes Virtuales.

22.- Función pcap _open_live()
pcap_t* pcap_open_live(char *dev, int slen, int prm, int to~s, char *ebuf)

La función pcap_open_live() prepara las estructuras de datos necesarias para comenzar la captura de paquetes devolviendo un descriptor que puede ser utilizado en las restantes operaciones de captura. En caso de error se devuelve NULL.

- dev indica el dispositivo sobre el cual se hace la captura. El dispositivo se identifica por una cadena de caracteres, algunos ejemplos de dispositivos válidos pueden ser "eth0" o "ppp0".
- slen indica el número máximo de bytes que se pueden capturar. Este valor depende de la tecnología de red que se esta usando, ya que según sea ésta así varian los tamaños de la trama.
- prm actúa como un flag lógico, indicando si el dispositivo de captura se colocan en modo promiscuo o no.
- to_ms es un valor entero que indica un timeout de lectura en milisegundos.
- ebuf hace referencia a un buffer donde en caso de error se guarda en una cadena explicativa del problema surgido.

El descriptor devuelto es realmente una estructura con el siguiente formato:

```
struct pcap_sf {
FILE *rfile;
int swapped;
int version_major;
int version_minor;
u_char *base;
};

struct pcap_md {
struct pcap_stat stat;
/*XXX*/
int use_bpf;
u_log TotPkts;                  /* can't oflow for 79hrs on ether */
u_long TotAccepted;             /* count accepted by filter*/
u_long TotDrops;                /* count of dropped packets*/
long TotMissed;                 /* missed by i/f during this run*/
long OrigMissed;                /* missed by i/f before this run */
#ifdef linux
        int pad;
        int skip;
        char *device;
#endif
} ;

struct pcap {
        int fd;
        int snapshot;
        int linktype;
        int tzoff;          /* timezone offset */
        int offset;             /* offset for proper alignment */

struct pcap_sf sf;
struct pcap_md md;

/*
* Read buffer.
*/

int bufsize;
u_char *buffer;
u_char *bp;
int cc;

/*
* Place holder for pcap_next().
*/

u_char *pkt;

/*
* Placeholder for filter code if bfp not in kernel.
*/

strúct bpf_program fcode;
```

char errbuf[PCAP_ERRBUF_SIZE];
} ;

23.- Función pcap _open_offline()
pcap_t* pcap_open~offline(char * fname , char *ebuf)

La función pcap _open_offline() permite abrir un archivo en lugar de un dispositivo de red como en el caso anterior. Esta función devuelve un descriptor o NULL en caso de error.

La verdadera intención de esta función es obtener los datos de una captura anterior que se han almacenado en un archivo determinado. A continuación se ven las funciones que permiten abrir el archivo y almacenar los datos que se van capturando.

- fname es el nombre del archivo que se abre para leer los paquetes.
- ebuf es un buffer donde guardar en caso de error un mensaje explicativo.

24.- Función pcap _dump _open()
pcap_dumper_t* pcap_dump_open(pcap_t *p, char *fname)

La función pcap_dump_open() abre un archivo para volcar en él los paquetes capturados. Este archivo puede ser abierto posteriormente con la función pcap_open_offline() para leer su contenido. En caso de error se devuelve NULL.

- p es un descriptor devuelto por las funciones pcap _open_live() o pcap _open_offline( ).
- fname es el nombre del archivo que se quiere abrir para escritura.

25.- Función pcap _lookupdev()
char* pcap_lookupdev(char *errbuf)

La función pcap_lookupdev() devuelve una cadena de caracteres que identifica un dispositivo del sistema válido para realizar la captura de paquetes. La cadena devuelta por esta función puede ser usada como parámetro dev en la función pcap_open_live(). Si hay algún error se devuelve NULL.

- errbuf es un buffer donde, en caso de error, la función devuelve una cadena explicativa del error.

***26.- Función* pcap _lookupnet()**
int pcap_lookupnet(char; *dev, bpf_u_int32 *netp , bpf_u_int32 *maskp, char *errbuf)

La función pcap _lookupnet() obtiene la red y la máscara de red asociadas con un dispositivo de captura. En caso de error la función devuelve -1.

- dev es una cadena de texto representativa del dispositivo.
- netp recibie el valor de la red.
- maskp recie el valor de la máscara de red.
- errbuf en ese buffer se almacena el mensaje de error.

***27.- Función* pcap _dispatch()**
int pcap_dispatch(pcap_t *p, int cnt, pcap_handler callback, u_char *user)

La función pcap_dispatch() comienza la captura de paquetes. Devuelve el número de paquetes recibidos, O en caso de alcanzar un EOF ó -1 en caso de error.

- p es un descriptor obtenido mediante pcap_open_live() o pcap_open_offline().
- cnt indica el número máximo de paquetes que se procesan antes de que la función retorne. Un valor de -1 procesa todos los paquetes recibidos en un buffer. Un valor de O procesa todos los paquetes hasta que ocurre un error o expira el timeout de lectura establecido.
- callback hace referencia a una rutina que es llamada para analizar el paquete recibido. Esta rutina tiene la siguiente signatura:

void callback(u_char *ud, pcap-pkthdr *pkthdr, u_char *pd)

- ud hace referencia a datos de usuario que son pasados desde pcap _dispatch() a través del parámetro user de esta función.
- pkthdr es una cabecera adicional incorporada por la biblioteca, precede a los datos y cabeceras propios de los protocolos de red. Su formato es el siguiente:

```
struct pcap_pkthdr {
struct timeval ts;          /* time stamp */
bpf_u_int32 caplen;         /* length of portion present */
bpf_u_int32 len;            /* length this packet (off wire) */
} ;
```

- pd es un puntero a través del cual están accesibles los datos del paquete capturado.
- user es un puntero a datos de usuario.

28.- Función pcap _loop()
int pcap_loop(pcap_t *p, int cnt, pcap_handler callback, u_char *user)

La función pcap_loop() es similar a pcap_dispatch(), esta función procesa tantos paquetes como se indiquen en el parámetro cnt, o retorna antes, en caso de ocurrir algún error, pero nunca retorna por causa del timeout de lectura. Un valor negativo en el parámetro cnt provoca un bucle infinito, procesando todos los paquetes recibidos y parando únicamente en caso de error. El resto de parámetros tienen el mismo significado que en la función pcap_dispatch().

29.- Función pcap _dump()
voidpcap_dump(u_char *user, struct pcap-pkthdr *h, u_char *sp)

Vuelca un paquete a un archivo abierto previamente con la función pcap_dump_open(). Los parámetros de esta función son los mismos que los de la rutina de callback indicada enpcap_dispatch() y pcap_loop().

30.- Función pcap _compile()
int pcap_compile(pcap_t *p, struct bpf_program *fp, char *str, int optimize, bpf_u_int32 netmask);

La función pcap_compile() toma una cadena de texto y la interpreta para crear un filtro sobre el cual basarse a la hora de elegir qué paquetes toma para su análisis.

La cadena de texto se indica en el parámetro str. Esta cadena de texto se interpreta creando un filtro que se guarda en la estructura indicada por el parámetro fp. El filtro almacenado en fp puede optimizarse o no, dependiendo del valor del parámetro optimize. Por último el parámetro netmask toma la máscara de red asociada al dispositivo de captura, este dato puede obtenerse mediante la función pcap _lookupnet( ).

Las estructuras relativas al filtro compilado presentes en <net/bpf.h> son las siguientes:

```
struct bpf_insn {
u_short code;
u_char jt;
u_char jf;
bpf_int32 k;
};

struct bpf_program {
u_int bf_len;
struct bpf_isn *bf_insns;
} ;
```

31.- Función pcap _setfilter()
int pcap_setfilter(pcap_t *p, struct bpf_program *fp)

La función pcap_setfilter() activa el filtro indicado en el parámetro fp. Este filtro ha sido creado anteriormente mediante la función pcap_compile(). Si el filtro se ha establecido correctamente, la función pcap_setfilter() devuelve 0; en caso de error devuelve -1.

32.- Función pcap _next()
u_char* pcap_next(pcap_t *p, struct pcap-pkthdr *h)

La función pcap_next() devuelve un puntero al siguiente paquete que se haya capturado y se tenga almacenado en el buffer de entrada.

33.- Función pcap _datalink()
int pcap_datalink(pcap_t *p)

La función pcap_datalink() devuelve el tipo de enlace asociado al descriptor p indicado como parámetro. En el archivo de cabecera <net/bpf.h> hay definidas constantes para identificar los posibles tipos de enlace devueltos por esta función:

```
/*
* Data link level type codes.
*/
#define DLT_NULL;                0      /*no link-layer encapsulation */
#define DLT_EN10MB       1       / * Ethernet (10Mb) * /
#define DLT_EN3MB        2       /* Experimental Ethernet (3Mb)*/
#define DLT_AX25                 3      / * Amateúr Radio AX. 25 * /
#define DLT_ PRONET              4      /* Proteon ProNET Token Ring * I
#define DLT_CHAOS        5       /* Chaos */
#define DLT_IEEE802      6       /* IEEE 802 Networks */

#define DLT_ARCNET       7       /* ARCNET */
#define DLT_SLIP                 8      /* Serial Line IP */
#define DLT_PPP                  9      /* Point to point Protocol */
#define DLT_FDDI                 10     /* FDDI */
#define DLT_ATM_RFC1483  11      /* LLC/SNAP encapsulated atm */
#define DLT_RAW                  12     /* raw IP*/
#define DLT_SLIP_BSOOS   13      /* BSD/OS Serial Line IP */
#define DLT_PPP_BSDOS    14      /* BSD/OS Point-to-point protocol */
```

34.- Función pcap _snapshot()
Init cap_is_swapped(pcap_t *p)

Devuelve el tamaño máximo de paquete que se indicó en la función pcap_open_live() mediante el parámetro sIen.

35.- Función pcap _is _swapped()
int pcap_is_swapped(pcap_t *p)

Esta función devuelve verdadero si el sistema local usa un orden de byte s (little indian o big indian) diferente al archivo de captura que se esté usando.

36.- Funciones pcap _major _versionO y pcap _minor _version()
int pcap_major_version(pcap_t *p)
int pcap_minor_version(pcap_t *p)
Las funciones pcap_major_version() y pcap_minor_version() devuelven el número de versión major (número de versión) y minor (número de revisión) respectivamente.

37.- Función pcap _stats()
int pcap_stats(pcap_t *p, struct pcap_stat *ps)

Esta función coloca en la estructura apuntada ps datos estadísticos de la captura de paquetes. Si hay algún error la función devuelve -1. El formato de la estructura pcap_stat tal y como está definido en el archivo <pcap.h> es el siguiente:

```
/*
* As returned by the pcap_stats()
*/
struct pcap_stat {
u_int ps_recv;          /* number of packets received */
u_int ps_drop;          /* number of packets dropped */
u_int ps_ifdrop;        /* drops by interface XXX not yet supported*/
};
```

38.- Función pcap_file()
FILE *pcap_file(pcap_t *p)

La funciónpcap-file() devuelve el nombre del archivo de captura que se esté usando.

39.- Función pcap_fileno()
int pcap_fileno(pcap_t *p)

Devuelve el descriptor numérico del archivo de captura que se esté usando.

40.- Función pcap_perror()
void pcap_perror(pcap_t *p, char *prefix)

La función pcap-perror() es similar a la función perror(). Muestra por stderr una cadena explicativa del último error ocurrido con alguna función de la biblioteca pcap. El mensaje es precedido por la cadena prefix.

41.- Función pcap_geterr()
char *pcap geterr (pcap t *p)

Esta función obtiene la cadena asociada al último error

42.- Función pcap _close()
void pcap_close(pcap_t *p)

La función pcap_close() libera todos los recursos asociados con el dispositivo de captura referenciado por el puntero p.

43.- Función pcap _dump _close()
void pcap_dump_close(pcap_dumper_t *p)

Esta función cierra el archivo de captura asociado ap.

A.2.1 SINTAXIS DE LOS FILTROS

Es posible indicar un filtro que seleccione los paquetes que se desean tratar de entre todos aquellos que se reciban. Este filtro se indica en forma de cadena de texto a la función pcap_compile(), activándolo posteriormente con la función pcap_setfilter().

El formato de la cadena de texto que se le puede indicar a la funciónpcap_compile() a través del parámetro str es lo que se ve a continuación.

La expresión que constituye el filtro consiste en una o más primitivas. Cada primitiva está formada por uno o más cualificado res a los cuales sigue un identificador.

Hay tres tipos básicos de cualificadores:

- Cualificadores de tipo: pueden ser las palabras reservadas host, net y port. Para hacer referencia a un host, una red o un puerto respectivamente. Permiten especificar a cuál de estos tres tipos corresponde el identificador al que se hace referencia. Si no se indica ninguno de los tres, se asume host por defecto.

- Cualificadores de dirección: Indican el sentido de la transferencia que interesa. Se representan por las palabras reservadas src, dst, src or dst y src and dst, para indicar que el identificador referenciado es el origen del paquete, el destino, el origen o destino o por último el origen y destino respectivamente. Si no se especifica ninguno se asume un cualificador src or dst por defecto.

- Cualificadores de protocolo: Permiten especificar paquetes que transporten un cierto protocolo. Algunas palabras reservadas válidas para hacer referencia a protocolos son ether,fddi, ip, alp, tcp, udp.

Con lo visto hasta ahora ya se pueden especificar primitivas simples que seleccionen tipos de paquetes concretos, algunos ejemplos de primitivas válidas serían:

Primitiva	Significado
host asimov	Paquetes cuyo origen o destino sea el host de nombre asimov
net 192.168	Paquetes cuya direccion de origen o destino se corresponda con la red 192.168
port 80	Paquetes cuyo puerto de origen o destino sea 80
dst net 192.18	Paquetes cuya direccion destino corresponda con al red 192.18
tcp dst port 21	Paqutes TCP cuyo puerto de destino sea el 21

El siguiente paso es combinar varias primitivas simples para crear un filtro más complejo. Se pueden combinar primitivas con las palabras reservadas and, or y no. Un ejemplo es el siguiente:

host asimov and not port 21 and not port 23

Este filtro toma todos aquellos paquetes que tengan como origen o destino al host asimov, y los puertos de origen o destino no sean ni el 21 ni el 23.

A continuación se indican algunas palabras reservadas especiales que pueden ser de utilidad para completar los filtros que se pude indicar.

La primera de ellas es gateway. Mediante esta sentencia se pueden filtrar aquellos paquetes que hayan usado el host indicado como gateway. Por ejemplo, si se indica el siguiente filtro:

gateway asimov

Simplemente se está indicando que se desean obtener todos aquellos paquetes que tengan como dirección MAC de origen aquella que corresponda al host asimov, pero la dirección IP de origen no es la del host asimov. Este es la prueba típica del uso de un gateway.

A la hora de indicar redes como origen o destino de un paquete, se puede ampliar su sintaxis indicando también su máscara de red. Se puede indicar explícitamente la máscara, o bien se pueden indicar los bits de la máscara. Se tiene un ejemplo de cada tipo:

net 192.168.100.0 mask 255.255.255.0
net 192.168.100.0/24

Otra opción disponible es filtrar teniendo en cuenta el tamaño del paquete. Se pueden seleccionar aquellos paquetes con tamaño menor o mayor que uno indicado. Algunos ejemplos son:

src host asimov and less 2048
dst host tolkienand port 80 and greater 128

Si nos interesase seleccionar únicamente los paquetes broadcast o multicast se puede hacer uso de las palabras reservadas correspondientes:

ether broadcast
ip broadcast
ether multicast
ip multicast

4Se pueden seleccionar tramas ethernet o paquetes IP, y una vez que se tengan seleccionados filtrar por el tipo de protocolo que encapsulan, de tal manera que se puedan tener expresiones como las siguientes:

ip proto icmp
ip pproto tcp
ip proto udp

Para seleccionar los datagramas IP que transporten paquetes ICMP, TCP o DP
respectivamente. Se puede hacer lo mismo pero a nivel de trama ethernet con la siguiente sintaxis:

ether proto ip
ether proto arp
ether proto rarp

Se pueden usar también expresiones aritméticas con operadores relacionales >, <, >=, <=, = y !=.

Se permite el acceso a los datos dentro de los paquetes para realizar comparaciones más detalladas. Para acceder a los datos del paquete se sigue la siguiente sintaxis:

Protocolo [expresión : tamaño]

Protocolo puede ser el nombre de cualquier protocolo reconocido: ether,fddi, ip, arp, rarp, tcp, udp, icmp. De esta manera se indica la capa donde se realiza la operación. Una vez indicada la capa, se consulta la expresión, la cual da el offset dentro de la capa seleccionada donde se quiera posicionar. El campo tamaño indica el número de bytes que son de interés a partir del offset seleccionado. El tamaño es opcional, de indicarse puede tomar un valor de 1,2 ó 4, y por defecto toma el valor 1.

Por ejemplo, la siguiente expresión:

IP[8] < 7

Selecciona únicamente aquellos paquetes en donde el octavo byte de la cabecera IP tuviese un valor menor que 7. Si se comprueba el formato de la cabecera IP se puede ver que el octavo byte corresponde con el TTL del datagrama, por tanto únicamente se seleccionan aquellos datagramas con un TTL menor que 7.

tcp[13] & 0x20 != 0

Esta expresión selecciona aquellos paquetes con el bit URG activado. Por último indicar que las primitivas pueden agruparse usando paréntesis.

A.2.2.- PROGRAMA DE MONITOREO.

```
#include <net/ethernet.h>
#include <netinet/ip.h>
#include <signal.h>
#include <stdio.h>
#include <pcap.h>
```

El descriptor del dispositivo de captura se declara como una variable global para que pueda ser usado desde la rutina que trata la señal SIGINT.

```
pcap_t *pc;
```

Se establece una rutina que se ejecuta cuando el usuario interrumpa el programa con CTRL+C. En esta rutina se muestran las estadísticas almacenadas sobre los paquetes capturados. Estas estadísticas se obtienen mediante la función pcap _stats(). Por último se pasa a cerrar correctamente el dispositivo de captura con la funciónpcap_close().

```
void fin (int s) {
        struct pcap_stat ps;

        pcap_stats(pc, &ps);
        fprintf(stdout, “paquetes recibidos: %i \n”, ps.ps_recv);
        fprintf(stdout, “paquetes descartados: %i \n”, ps.ps_drop);
        pcap_close(pc);
}
```

La función ethernet() es la rutina de callback ejecutada cada vez que llegue un paquete y éste cumpla el filtro establecido. Esta función se encarga de mostrar por pantalla los campos de información indicados.

```
void ethernet(u_char *u, const struct pcap_pkthdr *h, const u_char *d ) {
        struct ether_header eth;
struct iphdr  ip;
int i;

memcpy(&eth, d, sizeof(struct ether_header);
memcpy(&ip, d + ETH_HLEN, sizeof(struct iphdr);

for (i = 0; i < ETH ALEN;  i++)
{
        fprintf(stdout, “%02X”, eth.ether_shost[i]);
                if (i != ETH_LEN-1) fprintf(stdout, “:”);
}

fprintf(stdout, “ -> ”);
for (i = o; i < ETH_ALEN; i++)
{
        fprintf(stdout, “%02X”, eth.ether_dhost[i]);
        if (i != ETH_ALEN-1) fprintf(stdout, ':');
}

if (ntohs(eth.ether_type) == ETHERTYPE_IP) {
fprintf(stdout, “%s ->”, inet_ntoa(ip.saddr));
fprintf(stdout. “%s”,  inet_ntoa(ip.daddr));

switch (ip.protocol){
case IPPROTO_rCMP:
        fprintf(stdout,” [ICMP]”);
        break;
case IPPROTO_TCP:
        fprintf(stdout, “ [TCP] ”);
        break;
case IPPROTO_UDP:
        fprintf (stdout, “[UDP]”);
        break;
        }
}

fprintf(stdout, '\n');
}
```

En la función main del programa se obtiene un dispositivo de captura con pcap_lookupdev(), se abre con pcap_open_live() y se obtiene el tipo de enlace de red del que se dispone mediante pcap_datalink(). Dependiendo del tipo de enlace se elige una rutina de callback u otra, ya que los campos de las tramas y la forma de tratar los datos puede ser diferente. En nuestro caso únicamente se define una rutina para un enlace ethemet. A continuación se obtiene la red y máscara de red configuradas para el dispositivo seleccionado mediante pcap_lookupnet(), se compila el filtro pasado, opcionalmente como primer parámetro al programa utilizando pcap_compile() y se activa el filtro con pcap_setfilter(). Por último se entra en un bucle infinito para la captura de todos los paquetes de la red. De este bucle se sale cuando surja algún error o al pulsar el usuario CTRL+C.

```
int main(int argc, char **argv) {
pcap_handler rutina;
char *dev;
bpf_u_int32 red, mascara;
struct bpf_program filtro;
int datalink;
char pbuf[ETHER_MAX_LEN];
char ebuf[PCAP_ERRBUF_SIZEj;

if (! (dev = pcap_lookupdev(ebuf))) {
fprintf(stderr, "%s\n", ebuf);
exit(1) ;
}

fprintf(stdout, "Dispositivo: %s \n", dev);
if (! (pc = pcap_open_live(dev, ETHER_MAX_LEN, 1, 0, ebuf))) {
fprintf(stderr, "%s\n", ebuf);
exit(1);
}
if ((datalink = pcap_datalink(pc)) < 0) {
fprintf(stderr, "pcap_datalink: %s\n", pcap_geterr(pc));
exit(1) ;
}
switch (datalink) {
case DLT_EN10MB:
        fprintf(stdout, "Datalink: ethernet\n");
        rutina = ethernet;
        break;
default:
        fprintf(stderr, "(%i) No se puede manejar este tipo de tramas\n", datalink);
        exit(1) ;
}

if (pcap_lookupnet(dev, &red, &mascara, ebuf) < 0) {
fprintf(stderr, "%s\n", ebuf);
exit(1);
}
fprintf(stdout, "Red: %s\n", inet_ntoa(red);
fprintf(stdout, "Mascara: %s\n", inet_ntoa(mascara));

if (argv[1]) (
        if (pcap_compile(pc, &filtro, argv[l] , 0, mascara) < 0) {
                fprintf(stderr, "pcap_compile: %s \n", pcap_geterr(pc));
exit(1);
}

if (pcap_setfilter(pc, &filtro) < 0) {
fprintf(stderr, "pcap_setfilter: %s\n", pcap_geterr(pc));
exit(1);
}
}
signal(SIGINT, fin);
pcap_loop(pc, -1, rutina, NULL);}
```

A.2.3.- PROGRAMA DE MONITOREO (CÓDIGO).

```
#include <sys/time.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/if_ether.h>
#include <netinet/ip.h>
#include <netinet/tcp.h>
#include <net/if.h>
#include <string.h>
#include <unistd.h>
#include <stdio.h>

struct paquete {
struct ethhdr eth;
struct iphdr ip;
struct tcphdr tcp;
unsigned char datos[8192];
};

int main(int argc, char *argv[]) {

int ifd, offset, tam, lineas, resto, i, j, sigue, bloque;
struct ifreq ifr;
struct paquete pqt;
struct iphdr *ip;
struct tcphdr *tcp;
fd_set  rfd;
unsigned char *p;
char buffer[17];

if ((ifd=socket(AF_INET, SOCK_PACKET, htons(ETH_P_IP)))<0) {
perror ("No se ha podido obtener un socket");
return(-1);
}

strcpy(ifr.ifr_name,argv[1]);
if (ioctl(ifd, SIOCGIFFLAGS, &ifr) < 0) {
close(ifd);
perror("No se han podido obtener los flags de la interfaz de red PRIMERO");
return(-1);
}

ifr.ifr_flags =IFF_PROMISC;

if (ioctl(ifd, SIOCGIFFLAGS, &ifr) <0 ) {
close(ifd);
perror("No se han podido obtener los flags de la interfaz de red SEGUNDO");
return(-1);
}

while(1)
{
```

```
FD_ZERO(&rfd);
FD_SET(ifd, &rfd);
select(20, &rfd, NULL, NULL, NULL);
recvfrom(ifd, &pqt, sizeof(pqt),0,NULL,NULL);
ip=(struct iphdr *) (((unsigned long)&pqt.ip)-2);
tcp=(struct tcphdr *)(((unsigned long)&pqt.tcp)-2);

fprintf(stdout, "Origen :[%02:%02:%02:%02:%02:%02] %s:%i\n",
pqt.eth.h_source[0],pqt.eth.h_source[1],pqt.eth.h_source[2],
pqt.eth.h_source[3],pqt.eth.h_source[4],pqt.eth.h_source[5],
inet_ntoa(ip->saddr),ntohs(tcp->source));


fprintf(stdout, "Destino :[%02:%02:%02:%02:%02:%02] %s:%i\n",
pqt.eth.h_dest[0],pqt.eth.h_dest[1],pqt.eth.h_dest[2],
pqt.eth.h_dest[3],pqt.eth.h_dest[4],pqt.eth.h_dest[5],
inet_ntoa(ip->daddr), ntohs(tcp->dest));

offset= 0;

if (tam =htons(ip->tot_len)-sizeof(pqt.ip)-sizeof(pqt.tcp)) {
p=(unsigned char *)(((unsigned long) &pqt.datos)-2);
lineas=tam/16;
resto=tam % 16;
buffer[16]=0;
for (i=0;i< lineas;i++) {

fprintf(stdout, "%04X ", offset);
            for(j=0;j<16;j++) {
            fprintf(stdout, "%02X ", *(p+j) & 0xff);
            if ((*(p+j)>31) && (*(p+j) < 127)){
            buffer[j]=*(p+j);
            }else {
             buffer[j]='.';
            }

    }
fprintf(stdout, " %s\n", buffer);
offset += 16;
p +=16;
}

fprintf(stdout,"%04X ", offset);
for (i=0; i<resto; i++) {
fprintf(stdout, "%02X ", *(p+i) & 0xff);
if ((*(p+i) >31) && (*(p+i) < 127)) {
            buffer[i]= *(p+i);

   } else {
    buffer[i]='.';
   }
}

for( i=0; i<(16-resto); i++) {
fprintf(stdout, "    ");
```

```
}
buffer[resto]=0;
fprintf(stdout, " %s\n", buffer);
}
fprintf(stdout, "\n");
p=pqt.datos-2;
fprintf(stdout, "opciones de negociaciòn del telnet en este paquete:\n");

sigue=1,
bloque=0;

for (i=0; i< tam; i++){
if (bloque){
   if (*(p+1) == 0xff){
      if (*(p+i+1)== 0xff) {
         fprintf(stdout, “%02X”, *(p+i));
         i++;
         continue;
}
} else{
   fprintf(stdout, “%02x”, *(p+i));
   continue;
}
}
If(*(p+i) == 0xff){
   i++;
   switch (-(p+i)){
            case 240:
                     fprintf(stdout, “ } SE “);
                     bloque=0;
                     sigue=0;
                     break
            case 241:
                     fprintf(stdout, “NOP”);
                     break;
            case 242:
                     fprintf(stdout, “DATA MARK”);
                     break;
            case 243:
                     fprintf(stdout, “BREAK”);
                     break;
            case 244:
                     fprintf(stdout, “INTERRUPT PROCESS”);
                     break;
            case 245:
                     fprintf(stdout, “ABORT OUTPUT”);
                     break;
            case 246:
                     fprintf(stdout, “ARE YOU THERE ”);
                     break;
            case 247:
                     fprintf(stdout, “ERASE CHARACTER”);
                     break;
            case 248:
                     fprintf(stdout, “ERASE LINE”);
                     break;
```

```
            case 249:
                        fprintf(stdout, “GO AHEAD ”);
                        break;
            case 250:
                        fprintf(stdout, “SB”);
                        bloque =1;
                        break;
            case 251:
                        fprintf(stdout, “WILL”);
                        break;
            case 252:
                        fprintf(stdout, “WONT”);
                        break;
            case 253:
                        fprintf(stdout, “DO”);
                        break;
            case 254:
                        fprintf(stdout, “DON’T ”);
                        break;
            case 255:
                        break;
            sigue=0;
            default:
                        fprintf(stdout, “Codigo %02X desonocido”, *(p+i));
            }
if(sigue){
i++;
switch(-(p+i)){
case 0:
            fprintf(stdout, “BINARY TRANSMISSION”);
            break;
case 1:
            fprintf(stdout, “ECHO”);
            break;
case 2:
            fprintf(stdout, “RECONNECTIONS ”);
            break;
case 3:
            fprintf(stdout, “SUPPRESS GO AHEAD”);
            break;
case 4:
            fprintf(stdout, “APPROC MESSAGE SIZE NEGOTIATION ”);
            break;
case 5:
            fprintf(stdout, “STATUS ”);
            break;
case 6:
            fprintf(stdout, “TIMING MARK ”);
            break;
case 7:
            fprintf(stdout, “REMOTE CONTROLLED TRANS AND ECHO ”);
            break;
case 8:
            fprintf(stdout, “OUTPUT LINE WIDTH ”);
            break;
case 9:
```

```
            fprintf(stdout, “OUTPUT PAGE SIZE ”);
            break;
case 10:
            fprintf(stdout, “OUTPUT CARRIAGE-RETURN DISPOSITION ”);
            break;
case 11:
            fprintf(stdout, “OUTPUT HORIZONTAL TAB STOPS ”);
            break;
case 12:
            fprintf(stdout, “OUTPUT HORIZONTAL TAB DISPOSITION ”);
            break;
case 13:
            fprintf(stdout, “OUTUP FORMFEED DISPOSITION”);
            break;
case 14:
            fprintf(stdout, “OUTPUT VERTICAL TAB STOPS”);
            break;
case 15:
            fprintf(stdout, “OUTPUT VERTICAL TAB DISPOSITION ”);
            break;
case 16:
            fprintf(stdout, “OUTPUT LINEFEED DISPOSITION ”);
            break;
case 17:
            fprintf(stdout, “ ESTENDED ASCII ”);
            break;
case 18:
            fprintf(stdout, “LOGOUT ”);
            break;
case 19:
            fprintf(stdout, “BYTE MACRO”);
            break;
case 20:
            fprintf(stdout, “DATA ENTRY TERMINAL”);
            break;
case 22:
            fprintf(stdout, “SUPDUP”);
            break;
case 23:
            fprintf(stdout, “SEND LOCATION”);
            break;
case 24:
            fprintf(stdout, “TERMINAL TYPE”);
            break;
case 25:
            fprintf(stdout, “END OF RECORD”);
            break;
case 26:
            fprintf(stdout, “TATCS USER IDENTIFICATION”);
            break;
case 27:
            fprintf(stdout, “OUTPUT MARKING”);
            break;
case 28:
            fprintf(stdout, “TERMINAL LOCATION NUMBER”);
            break;
```

```
case 29:
        fprintf(stdout, “LNET 3270 REGIME”);
        break;
case 30:
        fprintf(stdout, “x.3 PAD”);
        break;
case 31:
        fprintf(stdout, “WINDOWS SIZE”);
        break;
case 32:
        fprintf(stdout, “TERMINAL SPEED”);
        break;
case 33:
        fprintf(stdout, “REMOTE FLOW CONTROL”);
        break;
case 34:
        fprintf(stdout, “LINE MODE”);
        break;
case 35:
        fprintf(stdout, “ X DISPLAY LOCATION”);
        break;
case 36:
        fprintf(stdout, “ENVIROMENT OPTION”);
        break;
case 37:
        fprintf(stdout, “AUTHENTICATION OPTION”);
        break;
case 38:
        fprintf(stdout, “ENCRYPTION OPTION”);
        break;
case 39:
        fprintf(stdout, “NEW ENVIRONMENT OPTION”);
        break;
case 40:
        fprintf(stdout, “TN3270E”);
        break;
case 255:
        fprintf(stdout, “EXTENDED OPTION LIST”);
        break;
default:
        fprintf(stdout, “Opcion %02X desconocida \n”,*(p+i));
}
}
Sigue =1;
if(bloque){
        fprintf(stdout, “[“);
        } else {
                Fprintf(stdout, “\n”);
                }
        }
}
        fprintf(stdout,”\n\n\n”);
        }
}
```

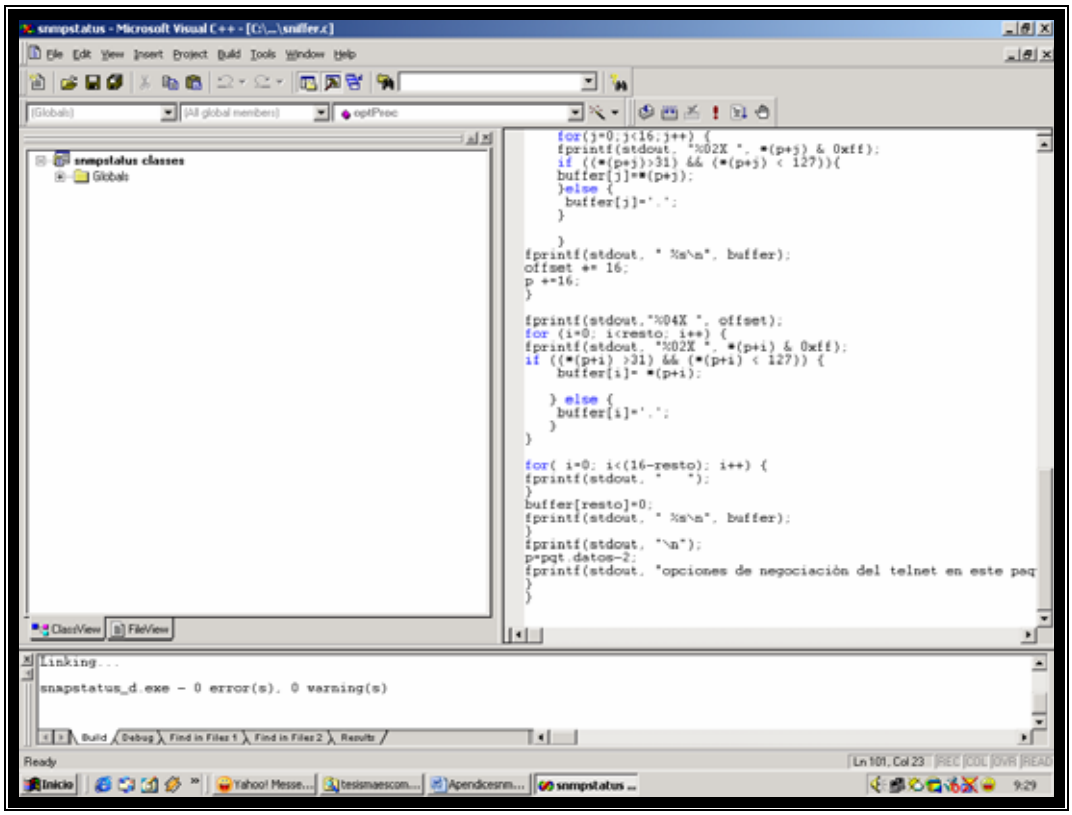


Figura A.1.- La compilación del MONITOR.

A.3.- PROGRAMA DE CONFIGURACIÓN (CÓDIGO).

Con el fin de poner en práctica el uso de SNMP e incorporarlo en la Herramienta de Gestión de Redes Virtuales. Se presenta el desarrollo de la primitiva “SNMPGET”. La estructura asociada a SNMP, la programación asociada a la primitiva y las sesiones de apertura; son aplicables a cada una de las primitivas de SNMP, sin embargo es importante destacar que los resultados que arrojan cada una de estas se particularizan, y hay que tomarlos con reserva, debido a que cada una de ellas cumple con una función específica dentro del protocolo SNMP. El programa permite consultar el valor de cualquier objeto que se encuentre en la base de datos MIB de un agente SNMP, con el uso de la biblioteca de funciones SNMP Windows/Linux [2].

A continuación se presenta el listado de la primitiva “SNMPGET” junto con toda la información asociada, que explica paso a paso la funcionalidad de la codificación.

```
#include <sys/types.h>
#include <snmp/snmp.h>
#include <string.h>
#include <stdio.h>
```

```
int main(int argc, char *argv[]) {
```

La estructura snmp_session permite almacenar la dirección del agente, los puertos UDP que se usan, los datos para la autenticación, etc.

```
struct snmp_session sesion;
struct snmp_session *sp;
```

La estructura snmp-pdu define una PDU indicando la dirección del destinatario, el tipo de PDU que es, la lista de variables que se manejan, los campos errindex y errstat, etc.

```
struct snmp-pdu *getpdu;
struct snmp-pdu *respdu;
```

Para la autentificación se usan comunidades de SNMP v 1. El nombre de la comunidad se torna del primer argumento.

```
,char *comunidad = argv[l];
```

El nombre o dirección IP del host donde reside el agente al cual se hace la consulta se toma del segundo argumento.

```
char *host = argv[2];
```

La variable que se desea consultar es el tercer argumento.

```
char *variable = argv[3];
cid var [MAX_NAME_LEN];
int nvar;
```

El primer paso es cargar los datos del MIB.

```
init_mib();
```

Se rellenan los datos de la sesión SNMP .

```
memset ((char *) &sesion, O, sizeof (sesioE! )-;
sesion.retries = SNMP_DEFAULT_RETRIES;
sesion.timeout = SNMP_DEFAULT_TIMEOUT;
sesion.peername = argv[2];
sesion.remote-port = SNMP_DEFAULT_REMPORT;
sesion.local-port = o;
sesion.community = argv[1];
sesion.community_len = strlen(argv[l]);
```

Se establece la sesión con los datos proporcionados anteriormente.

```
snmp_synch_setup(&sesion) ,
```

Se conecta con el agente tomando los datos de la sesión establecida.

```
if (! (sp = snmp_open(&sesion») {
fprintf(stderr, 'No se pudo abrir la sesion SNMP\n');
exit(1) ;
}
```

Se crea la PDU de petición del valor de la variable que se indica a continuación.

```
if (! (getpdu = snmp_pdu_create(SNMP_PDU_GET))) {
        fprintf(stderr, 'Error al crear la PDU\n');
        exit(1);
}
```

Se toma la variable que se quiere consultar, ya sea en formato de etiquetas (iso.org.doclntemet) o de números (1.3.6.1), y se crea un identificador de objeto valido para ser incluido en la lista de variables de la PDU.

```
nvar = MAX_NAME_LEN;
if (!read_objid(argv[3].. var, &nvar)) {
        fprintf (gtderr, 'read_objid');
        exit(1);
}
```

Se incorpora a la lista de variables de la PDU.

```
snmp_add_null_var(getpdu, var, nvar);
```

Se envía la PDU y se espera el mensaje de respuesta. Si no hay ningún error se imprime la información que llegue en la PDU de respuesta referente a la lista de variables.

```
if(snmp_synch_response (sp, getpdu, &respdu) = = STAT_SUCCES,S) {
        if (respdu->errstat == SNMP_ERR_NOERROR) {
        print_variable_list (respdu->variables);
        } else {
        fprintf(stderr, 'errstat=u, errindex=%u \n', respdu->errstat,
        respdu->errindex)
}

snmp_free_pdu(respdu);                  /* Se libera la PDU de respuesta */
else {
                fprintf(stderr, "snmp_synch_response\n');
}'
snmp_close(sp); /* Se cierra la sesion */

/* CODIGO FUENTE PARA SNMP_PARSE_ARGS * SNMP_PARSE_ARGS.C
Programa generico de uso por todos las demas primitivas de SNMP*/

#include <config.h>
#if HAVE_STDLIB_H
#include <stdlib.h>
#endif
#if HAVE_UNISTD_H
#include <unistd.h>
#endif
#if HAVE_STRING_H
#include <string.h>
#else
#include <strings.h>
#endif
#include <sys/types.h>
#include <stdio.h>
#if HAVE_UNISTD_H
```

```
#include <unistd.h>
#endif
#include <ctype.h>
#if HAVE_NETINET_IN_H
#include <netinet/in.h>
#endif
#if TIME_WITH_SYS_TIME
# ifdef WIN32
#  include <sys/timeb.h>
# else
#  include <sys/time.h>
# endif
# include <time.h>
#else
# if HAVE_SYS_TIME_H
#  include <sys/time.h>
# else
#  include <time.h>
# endif
#endif
#if HAVE_SYS_SELECT_H
#include <sys/select.h>
#endif
#if HAVE_WINSOCK_H
#include <winsock.h>
#endif
#if HAVE_NETDB_H
#include <netdb.h>
#endif
#if HAVE_ARPA_INET_H
#include <arpa/inet.h>
#endif

#include <getopt.h>

#include "asn1.h"
#include "snmp_api.h"
#include "snmp_impl.h"
#include "snmp_client.h"
#include "mib.h"
#include "snmp.h"
#include "scapi.h"
#include "keytools.h"

#include "snmp_parse_args.h"
#include "snmp_logging.h"
#include "version.h"
#include "system.h"
#include "parse.h"
#include "read_config.h"
#include "snmp_debug.h"
#include "snmpv3.h"
#include "default_store.h"

int random_access = 0;
```

```
#define USM_AUTH_PROTO_MD5_LEN 10
static oid usmHMACMD5AuthProtocol[]  = { 1,3,6,1,6,3,10,1,1,2 };
#define USM_AUTH_PROTO_SHA_LEN 10
static oid usmHMACSHA1AuthProtocol[] = { 1,3,6,1,6,3,10,1,1,3 };
#define USM_PRIV_PROTO_DES_LEN 10
static oid usmDESPrivProtocol[]      = { 1,3,6,1,6,3,10,1,2,2 };

void
snmp_parse_args_usage(FILE *outf)
{
  fprintf(outf, "[options...] <hostname> {<community>}");
}

void
snmp_parse_args_descriptions(FILE *outf)
{
/****/
}

#define BUF_SIZE 512

int
snmp_parse_args(int argc,
                    char *const *argv,
                    struct snmp_session *session, const char *localOpts,
                    void(* proc)(int, char *const *, int))
{
  int arg;
  char *cp;
  char *Apsz = NULL;
  char *Xpsz = NULL;
  char *Cpsz = NULL;
  u_char buf[BUF_SIZE];
  int bsize;
  int tmp_port;
  char Opts[BUF_SIZE];

  /* initialize session to default values */
  snmp_sess_init( session );
  strcpy(Opts, "VhHm:M:O:l:P:D:dv:p:r:t:c:Z:e:E:n:u:l:x:X:a:A:T:");
#ifndef DEPRECATED_CLI_OPTIONS
  strcat(Opts, "fsSqR");
#endif
  if (localOpts) strcat(Opts, localOpts);

  /* get the options */
  DEBUGMSGTL(("snmp_parse_args","starting: %d/%d\n", optind, argc));
  for(arg=0; arg < argc; arg++) {
     DEBUGMSGTL(("snmp_parse_args"," arg %d = %s\n", arg, argv[arg]));
  }

  optind = 1;
  while ((arg = getopt(argc, argv, Opts)) != EOF) {
    DEBUGMSGTL(("snmp_parse_args","handling (#%d): %c\n", optind, arg));
    switch(arg){
      case 'V':
```

```
      fprintf(stderr,"UCD-snmp version: %s\n", VersionInfo);
      return(-2);

    case 'h':
      return(-1);
      break;

    case 'H':
      init_snmp("snmpapp");
      fprintf(stderr, "Configuration directives understood:\n");
      read_config_print_usage("  ");
      return(-2);

    case 'm':
      setenv("MIBS", optarg, 1);
      break;

    case 'M':
      setenv("MIBDIRS", optarg, 1);
      break;

#ifndef DEPRECATED_CLI_OPTIONS
    case 'f':
        fprintf(stderr, "Warning: -f option is deprecated - use -Of\n");
        snmp_set_full_objid(1);
        break;

    case 's':
        fprintf(stderr, "Warning: -s option is deprecated - use -Os\n");
        snmp_set_suffix_only(1);
        break;

    case 'S':
        fprintf(stderr, "Warning: -S option is deprecated - use -OS\n");
        snmp_set_suffix_only(2);
        break;

    case 'q':
        fprintf(stderr, "Warning: -q option is deprecated - use -Oq\n");
        snmp_set_quick_print(1);
        break;

    case 'R':
        fprintf(stderr, "Warning: -R option is deprecated - use -IR\n");
      snmp_set_random_access(1);
      break;
#endif /* DEPRECATED_CLI_OPTIONS */

    case 'O':
      cp = snmp_out_toggle_options(optarg);
      if (cp != NULL) {
        fprintf(stderr,"Unknown output option passed to -O: %c.\n", *cp);
        return(-1);
      }
      break;
```

```
case 'I':
  cp = snmp_in_toggle_options(optarg);
  if (cp != NULL) {
    fprintf(stderr,"Unknown input option passed to -I: %c.\n", *cp);
    return(-1);
  }
  break;

case 'P':
  cp = snmp_mib_toggle_options(optarg);
  if (cp != NULL) {
    fprintf(stderr,"Unknown parsing option passed to -P: %c.\n", *cp);
    return(-1);
  }
  break;

case 'D':
  debug_register_tokens(optarg);
  snmp_set_do_debugging(1);
  break;

case 'd':
  snmp_set_dump_packet(1);
  break;

case 'v':
  if (!strcmp(optarg,"1")) {
    session->version = SNMP_VERSION_1;
  } else if (!strcasecmp(optarg,"2c")) {
    session->version = SNMP_VERSION_2c;
  } else if (!strcasecmp(optarg,"3")) {
    session->version = SNMP_VERSION_3;
  } else {
    fprintf(stderr,"Invalid version specified after -v flag: %s\n", optarg);
    return(-1);
  }
  break;

case 'p':
  tmp_port = atoi(optarg);
  if ((tmp_port < 1) || (tmp_port > 65535)) {
    fprintf(stderr,"Invalid port number after -p flag.\n");
    return(-1);
  }
  session->remote_port = (u_short)tmp_port;
  break;

case 't':
  session->timeout = atoi(optarg) * 1000000L;
  if (session->timeout < 0 || !isdigit(optarg[0])) {
    fprintf(stderr,"Invalid timeout in seconds after -t flag.\n");
    return(-1);
  }
  break;

case 'r':
```

```
  session->retries = atoi(optarg);
  if (session->retries < 0 || !isdigit(optarg[0])) {
    fprintf(stderr,"Invalid number of retries after -r flag.\n");
    return(-1);
  }
  break;

case 'T':
    if (strcasecmp(optarg,"TCP") == 0) {
        session->flags |= SNMP_FLAGS_STREAM_SOCKET;
    } else if (strcasecmp(optarg,"UDP") == 0) {
        /* default, do nothing */
    } else {
        fprintf(stderr,"Unknown transport \"%s\" after -T flag.\n", optarg);
        return(-1);
    }
    break;

case 'c':
       Cpsz = optarg;
       break;

case 'Z':
  session->engineBoots = strtoul(optarg, NULL, 10);
  if (session->engineBoots == 0 || !isdigit(optarg[0])) {
    fprintf(stderr,"Need engine boots value after -Z flag.\n");
    return(-1);
  }
  cp = strchr(optarg,',');
  if (cp && *(++cp) && isdigit(*cp))
    session->engineTime = strtoul(cp, NULL, 10);
      else if ((optind<argc) && isdigit(argv[optind][0]))
        session->engineTime = strtoul(argv[optind], NULL, 10);
  else {
    fprintf(stderr,"Need engine time value after -Z flag.\n");
    return(-1);
  }
  break;

case 'e':
       if ((bsize = hex_to_binary(optarg,buf)) <= 0) {
    fprintf(stderr,"Bad engine ID value after -e flag. \n");
    return(-1);
       }
       session->securityEngineID = (u_char *)malloc(bsize);
       memcpy(session->securityEngineID, buf, bsize);
       session->securityEngineIDLen = bsize;
  break;

case 'E':
       if ((bsize = hex_to_binary(optarg,buf)) <= 0) {
    fprintf(stderr,"Bad engine ID value after -E flag. \n");
    return(-1);
       }
       session->contextEngineID = (u_char *)malloc(bsize);
       memcpy(session->contextEngineID, buf, bsize);
```

```
        session->contextEngineIDLen = bsize;
    break;

  case 'n':
        session->contextName = strdup(optarg);
        session->contextNameLen = strlen(optarg);
    break;

  case 'u':
        session->securityName = strdup(optarg);
        session->securityNameLen = strlen(optarg);
    break;

  case 'l':
    if (!strcasecmp(optarg,"noAuthNoPriv") || !strcmp(optarg,"1") ||
        !strcasecmp(optarg,"nanp")) {
      session->securityLevel = SNMP_SEC_LEVEL_NOAUTH;
    } else if (!strcasecmp(optarg,"authNoPriv") || !strcmp(optarg,"2") ||
        !strcasecmp(optarg,"anp")) {
      session->securityLevel = SNMP_SEC_LEVEL_AUTHNOPRIV;
    } else if (!strcasecmp(optarg,"authPriv") || !strcmp(optarg,"3") ||
        !strcasecmp(optarg,"ap")) {
      session->securityLevel = SNMP_SEC_LEVEL_AUTHPRIV;
    } else {
      fprintf(stderr,"Invalid security level specified after -l flag: %s\n", optarg);
      return(-1);
    }

    break;

  case 'a':
    if (!strcasecmp(optarg,"MD5")) {
      session->securityAuthProto = usmHMACMD5AuthProtocol;
      session->securityAuthProtoLen = USM_AUTH_PROTO_MD5_LEN;
    } else if (!strcasecmp(optarg,"SHA")) {
      session->securityAuthProto = usmHMACSHA1AuthProtocol;
      session->securityAuthProtoLen = USM_AUTH_PROTO_SHA_LEN;
    } else {
      fprintf(stderr,"Invalid authentication protocol specified after -a flag: %s\n", optarg);
      return(-1);
    }
    break;

  case 'x':
    if (!strcasecmp(optarg,"DES")) {
      session->securityPrivProto = usmDESPrivProtocol;
      session->securityPrivProtoLen = USM_PRIV_PROTO_DES_LEN;
    } else {
      fprintf(stderr,"Invalid privacy protocol specified after -x flag: %s\n", optarg);
      return(-1);
    }
    break;

  case 'A':
        Apsz = optarg;
        break;
```

```
    case 'X':
      Xpsz = optarg;
      break;

    case '?':
      return(-1);
      break;

    default:
          proc(argc, argv, arg);
          break;
  }
}
DEBUGMSGTL(("snmp_parse_args","finished: %d/%d\n", optind, argc));

/* lee en la base de datos de MIB */
init_snmp("snmpapp");

if (session->version == SNMP_DEFAULT_VERSION) {
  session->version = ds_get_int(DS_LIBRARY_ID, DS_LIB_SNMPVERSION);
}

if (Apsz) {
    session->securityAuthKeyLen = USM_AUTH_KU_LEN;
    if (session->securityAuthProto == NULL) {
        /* get .conf set default */
        session->securityAuthProto =
            get_default_authtype(&session->securityAuthProtoLen);
    }
    if (session->securityAuthProto == NULL) {
        /* assume MD5 */
        session->securityAuthProto = usmHMACMD5AuthProtocol;
        session->securityAuthProtoLen = USM_AUTH_PROTO_MD5_LEN;
    }
    if (generate_Ku(session->securityAuthProto,
                    session->securityAuthProtoLen,
                    (u_char *)Apsz, strlen(Apsz),
                    session->securityAuthKey,
                    &session->securityAuthKeyLen) != SNMPERR_SUCCESS) {
        snmp_perror(argv[0]);
        fprintf(stderr,"Error generating Ku from authentication pass phrase. \n");
        return(-2);
    }
}
if (Xpsz) {
    session->securityPrivKeyLen = USM_PRIV_KU_LEN;
    if (session->securityPrivProto == NULL) {
        session->securityPrivProto =
            get_default_privtype(&session->securityPrivProtoLen);
    }
    if (session->securityPrivProto == NULL) {
        session->securityPrivProto = usmDESPrivProtocol;
        session->securityPrivProtoLen = USM_PRIV_PROTO_DES_LEN;
    }
    if (generate_Ku(session->securityAuthProto,
```

```
                session->securityAuthProtoLen,
                (u_char *)Xpsz, strlen(Xpsz),
                session->securityPrivKey,
                &session->securityPrivKeyLen) != SNMPERR_SUCCESS) {
        snmp_perror(argv[0]);
        fprintf(stderr,"Error generating Ku from privacy pass phrase. \n");
        return(-2);
    }
  }
  /*obtien el nombre del host*/
  if (optind == argc) {
    fprintf(stderr,"No hostname specified.\n");
    return(-1);
  }
  session->peername = argv[optind++];     /* hostname */

  /* get community */
  if ((session->version == SNMP_VERSION_1) ||
      (session->version == SNMP_VERSION_2c)) {
    /* v1 and v2c - so get community string */
    if (!Cpsz) {
      if ((Cpsz = ds_get_string(DS_LIBRARY_ID, DS_LIB_COMMUNITY)) != NULL)
            ;
      else if (optind == argc) {
        fprintf(stderr,"No community name specified.\n");
        return(-1);
      }
      else
            Cpsz = argv[optind++];
    }
    session->community = (unsigned char *)Cpsz;
    session->community_len = strlen(Cpsz);
  }
  return optind;
}
```

/* snmp_parse_args.c(Fin) */

Las pantallas de compilación para cada una de estas primitivas se muestran a continuación.

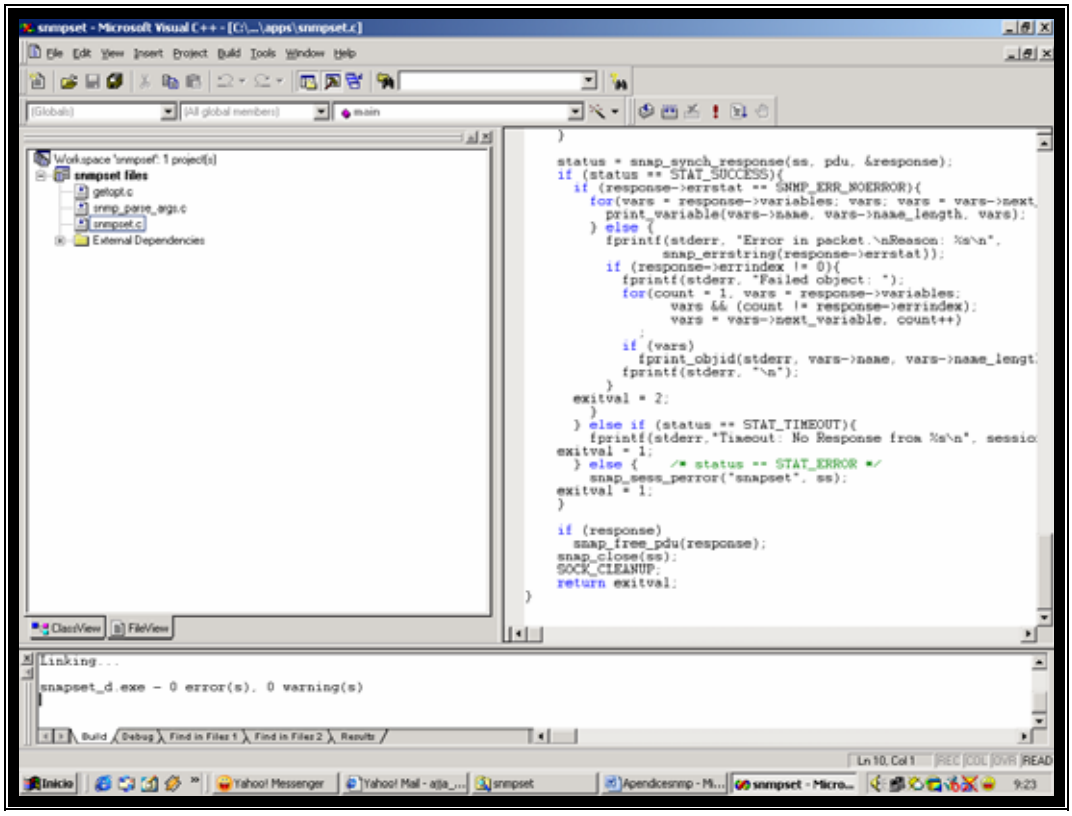


Figura A.2.- Compilación, Primitiva SNMP SET.

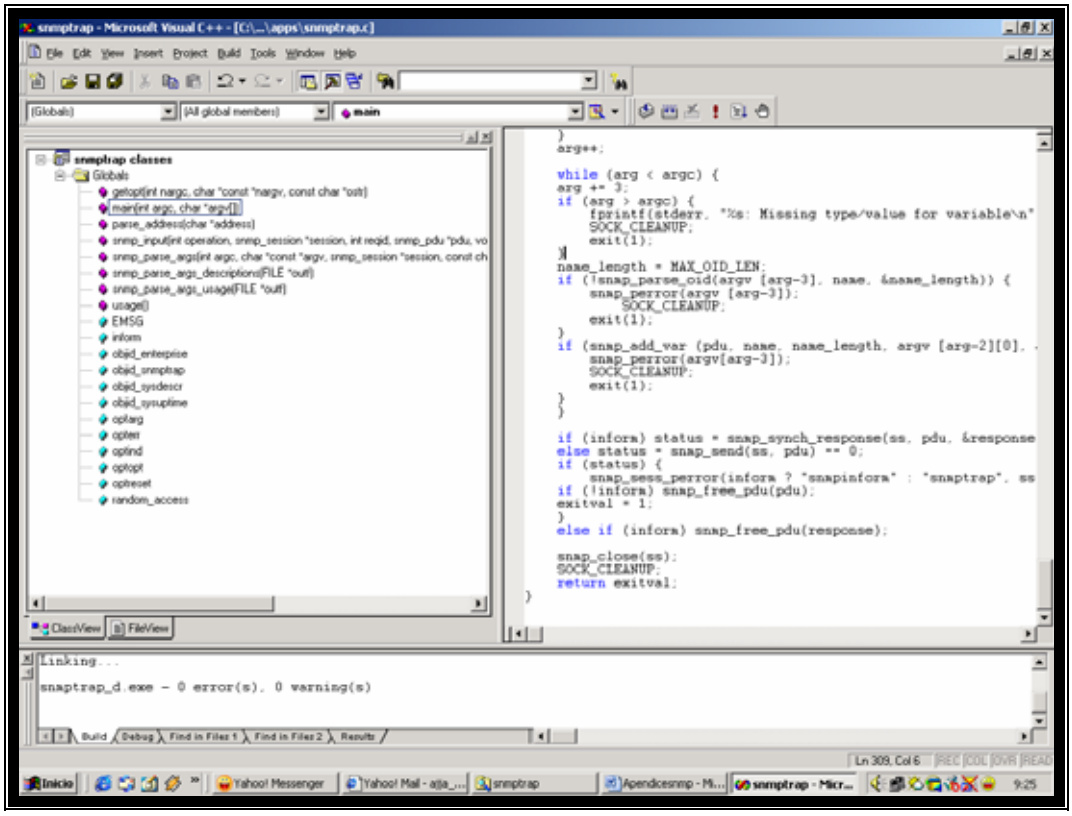


Figura A.3 compilación, Primitiva SNMP TRAPS

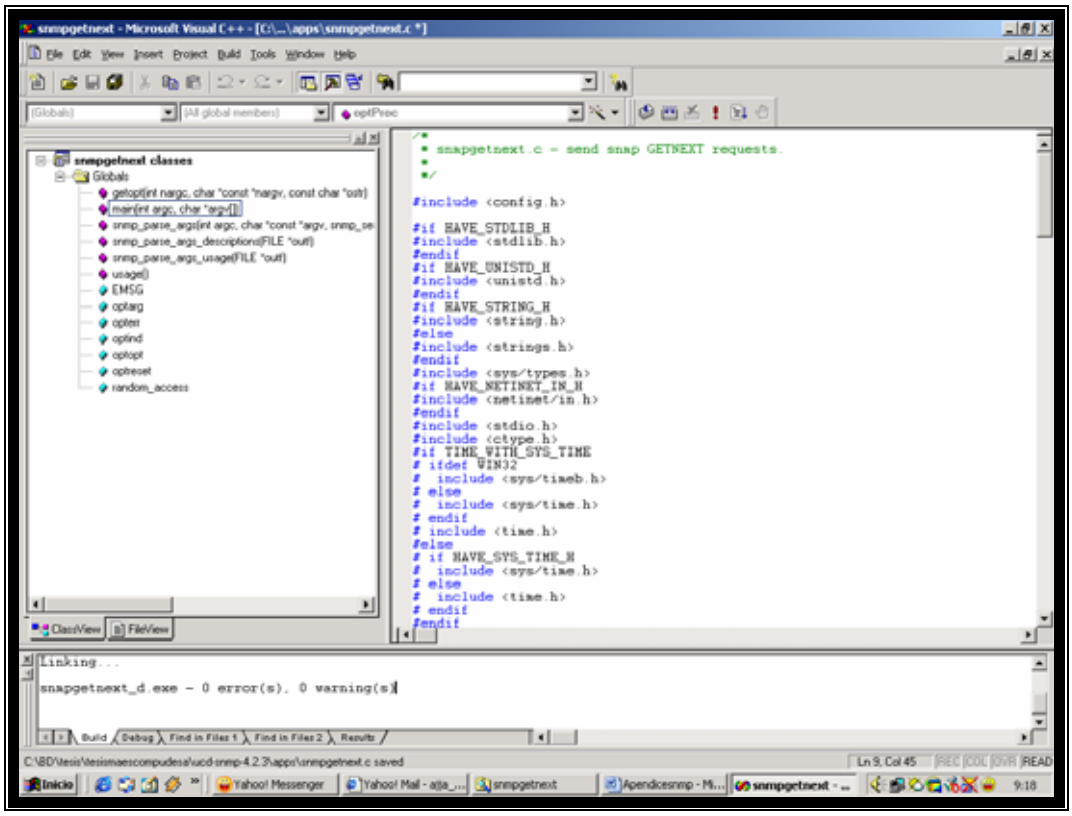


Figura A.4 Primitiva SNMP GETNEXT

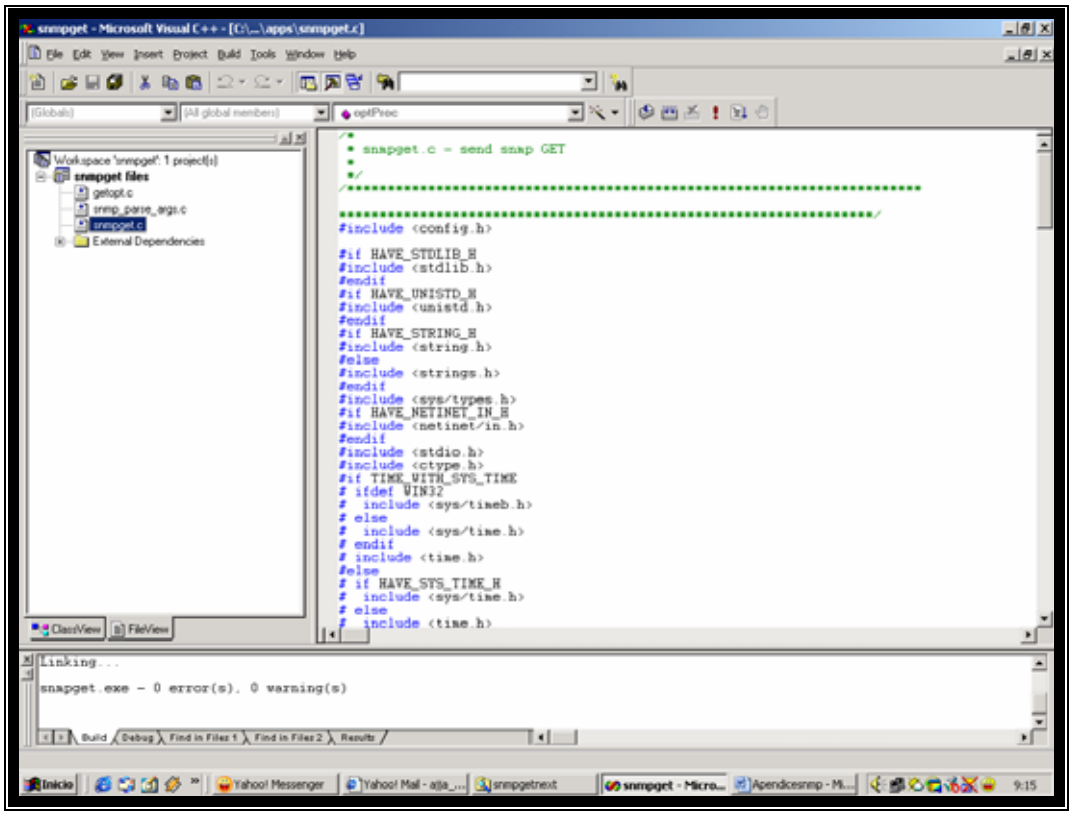


Figura A.5.- Compilación, Primitiva SNMP GET