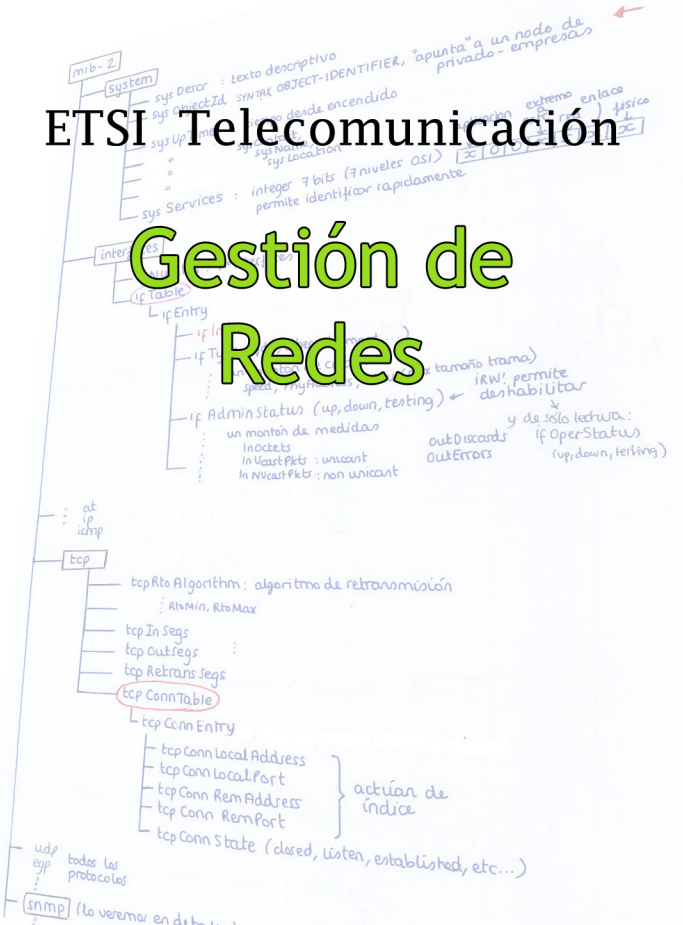
ETSI Telecomunicación

Gestión de Redes



Apuntes de Pak

Gestión de Redes

Apuntes de Pak (Fco. J. Rodríguez Fortuño)
ETSI Telecomunicación. Universidad Politécnica de Valencia.
Segundo cuatrimestre de 4º curso
Curso 2006/2007

Contenido:

- Referencia rápida
- Apuntes

Fecha de última actualización: 29 Febrero 2008

Gestión de Redes – Referencia Rápida

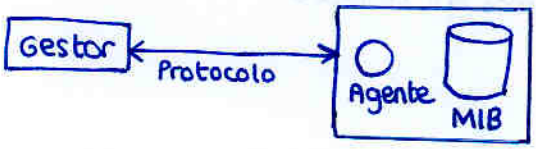
Tema 1. Introducción a la gestión de red

Definición: gestión red: conjunto de funciones → permiten intercambio y procesamiento de información → para controlar y monitorizar una red → para uso eficiente y al menor coste

elementos de red: NE (network element): dispositivo
OS (operating system): gestor

Sistemas Propietarios
(-) solución parcial (un OS para cada fabricante)
(-) Complejas
(-) caras
(-) Incompatibles

Sistemas Estándar



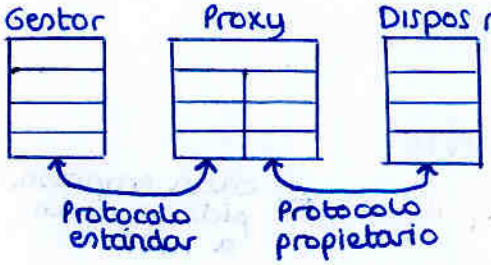
Tipo de info:
- estática: info de config ej (nº interfaces)
- dinámica: info de red
- estadística: info procesada (lo suele hacer el gestor, no el agente)

Métodos de acceso:

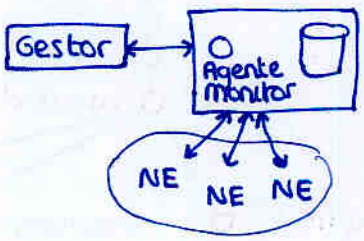
- Polling (petición respuesta)
  (+) Sencillo
  (-) Frecuencia de polling (según lo crítica q sea la variable)
  Varios dispositivos → AB! → polling dirigido por trap
- Event Report (alarmas)
  - comunic. asíncrona
  - Hay que configurarlas

  (+) Uso eficiente AB
  (+) conocimiento inmediato
  (-) No nos enteramos si cae el enlace gestor-disp.
  Hay que combinarlas con polling

Proxy = traductor entre protocolo estándar y propietario

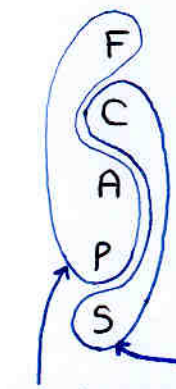


Agente monitor:



Recoge info de toda una subred almacenándola en su propia base de datos

Areas Funcionales



- Faults
- Configuration (inicializar y actualizar dispositivos)
- Accounting (facturación, análisis)
- Performance (throughput, utilización, disponibilidad)
- security (claves cifrado, mecanismos seguridad)

Estándares de gestión

SNMP: - redes TCP/IP
- desarrollado por IETF

TMN: - grandes redes públicas de comunic.
- desarrollado por UIT-T

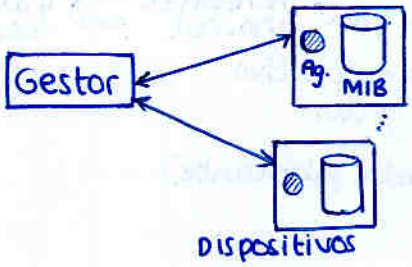
ISO tiene su modelo que nadie usa pero 'tomar ideas'

Tema 2. Modelos de Gestión de Red SNMP

RFC - 1155 SMI
RFC - 1213 MIB-2
RFC - 1157 SNMPv1

1. Conceptos Básicos

- Arquitectura del sistema



- Arquitectura de protocolos

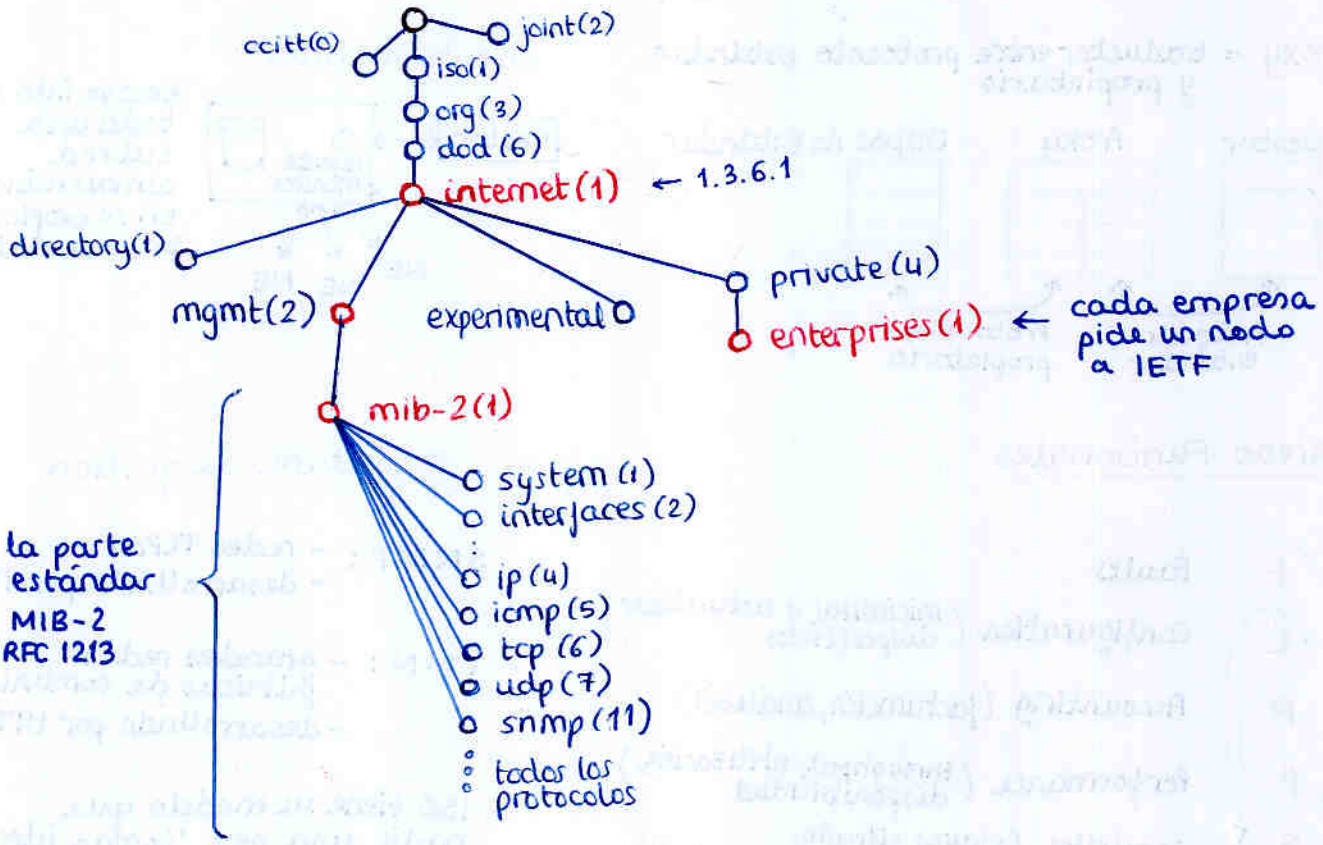
SNMP → nivel aplicación sobre UDP

¿Porque no TCP?

TCP:
- orientado a la conexión
  - ↳ sobrecarga tráfico por establecimiento conexiones
  - ↳ perder tiempo
  - ↳ sobrecarga en gestor para mantener las conexiones
- fiable mediante retransmisiones
  - ↳ empeora situación congestión que tratamos de resolver

- Polling dirigido por TRAP
  - Frecuencia de polling baja por defecto
  - Si recibes un TRAP: - aumentas la frecuencia de polling
    - interrogas al dispositivo y sus vecinos
  - Filtros de TRAP: para ignorar alarmas de los vecinos, que serán consecuencia de lo ya sabido

2. Estructura de información de gestión (SMI)



Tipos de datos

- **Simples** (universales: subconjunto de la especificación formal SMI)
  - INTEGER (entero 32 bits)
  - OCTET STRING (cadena caracteres)
  - OBJECT IDENTIFIER (ejemplo 1.3.6.1)
  - NULL
  - SEQUENCE OF ..... (array)
  - SEQUENCE { (estructura)

```
nombre1  INTEGER,
nombre2  OCTET STRING,
nombre3  INTEGER (0..65535)
}
```

- **SNMP**
  - Network Address
  - IpAddress ↔ OCTET STRING (SIZE(4))
  - Counter
    - entero ≥ 0 $[0, 2^{32}-1]$
    - sólo incrementar
    - al alcanzar máximo se resetea
      - ↳ max puedo especificarlo
  - Gauge
    - entero ≥ 0 $[0, 2^{32}-1]$
    - se puede incrementar y decrementar
    - se detiene al alcanzar el máximo
  - TimeTicks
    - entero ≥ 0
    - tiempo en centésimas de segundo desde evento

Cosas que se pueden hacer al definir estos tipos:

```
INTEGER
INTEGER  (0..127)                    ← rango
INTEGER { other(1),
          yes(2),                    ← numerado
          no(3)
        }

OCTET STRING
OCTET STRING  (SIZE (0..10))         ← rango de tamaños
OCTET STRING  (SIZE (4))             ← tamaño fijo
```

Definición de objetos

- Objetos identificadores (nodos)

```
iso OBJECT IDENTIFIER ::= { 1 }
org OBJECT IDENTIFIER ::= { iso 3 }
dod OBJECT IDENTIFIER ::= { org 6 }
internet OBJECT IDENTIFIER ::= { dod 1 }
```

- Definición de nuevos tipos de datos:

tipo de datos estructura

```
DisplayString ::= OCTET STRING
TcpConnEntry ::= SEQUENCE { nombre1 tipo1, nombre2 tipo2, ... }
```

- Objetos escalares

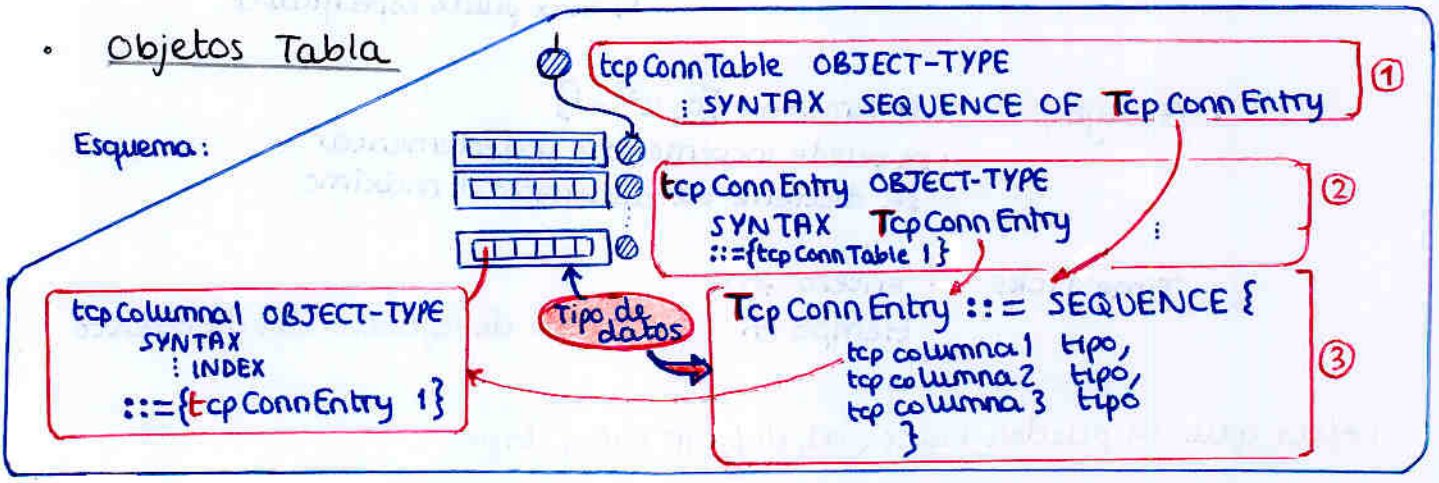
```
sysUpTime OBJECT-TYPE
    SYNTAX TimeTicks
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION "The time...."
    ::= { system 3 }
```

← es de tipo objeto

ACCESS: read-only | write-only | read-write | not-accesible

STATUS: mandatory | optional | obsolete | deprecated

- Objetos Tabla



index	unidad	capacidad

ejemplo:

```
discoTable OBJECT-TYPE
    SYNTAX SEQUENCE OF DiscoEntry
    ACCESS not-accesible
    STATUS mandatory
    DESCRIPTION "Tabla discos"
    ::= { demo 1 }

discoEntry OBJECT-TYPE
    SYNTAX DiscoEntry
    ACCESS not-accesible
    STATUS mandatory
    DESCRIPTION "Info de un disco"
    INDEX { index }
    ::= { discoTable 1 }
```

① ②

columnas que hacen de índice

```
DiscoEntry ::= SEQUENCE {
    index INTEGER,
    unidad OCTET STRING,
    capacidad INTEGER
    }
```

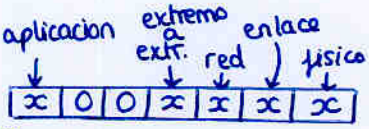
③ aún no definir tamaños

```
index OBJECT-TYPE
    SYNTAX INTEGER (0..10)
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION "indice"
    ::= { discoEntry 1 }

    : resto de
    : columnas
    :
```

obligado: nº máximo de filas !!!

3. Contenido importante del nodo mib-2

- **mib-2**
  - **system**
    - sysDescr : texto descriptivo
    - sysObjectId SYNTAX OBJECT-IDENTIFIER, "apunta" a un nodo de privado-empresas
    - sysUpTime : tiempo desde encendido
    - ⋮ sysContact, sysName, sysLocation
    - sysServices : integer 7 bits (7 niveles OSI) permite identificar rápidamente

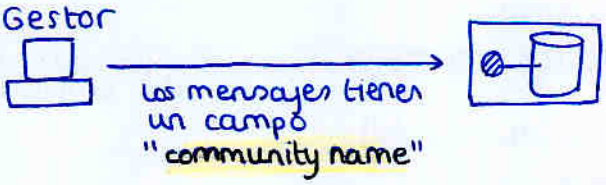


  - **interfaces**
    - ifNumber : nº interfaces
    - **ifTable**
      - ifEntry
        - ifIndex
        - ifType (fddi, ethernetCsmacd, ...)
        - ⋮ un montón de cosas. speed, PhyAddress, Mtu (max tamaño trama)
        - ifAdminStatus (up, down, testing) ← iRW: permite deshabilitar
        - ⋮ un montón de medidas: InOctets, InUcastPkts : unicast, InNUcastPkts : non unicast, OutDiscards, OutErrors
        - y de sólo lectura: ifOperStatus (up, down, testing)
  - ⋮ at, ip, icmp
  - **tcp**
    - tcpRtoAlgorithm : algoritmo de retransmisión
    - ⋮ RtoMin, RtoMax
    - tcpInSegs
    - tcpOutSegs
    - tcpRetransSegs
    - **tcpConnTable**
      - tcpConnEntry
        - tcpConnLocalAddress
        - tcpConnLocalPort
        - tcpConnRemAddress
        - tcpConnRemPort

        } actúan de índice
        - tcpConnState (closed, listen, established, etc...)
  - udp, egp, ⋮ todos los protocolos
  - **snmp** (lo veremos en detalle)
  - ⋮

4. SNMPv1

• Política de acceso

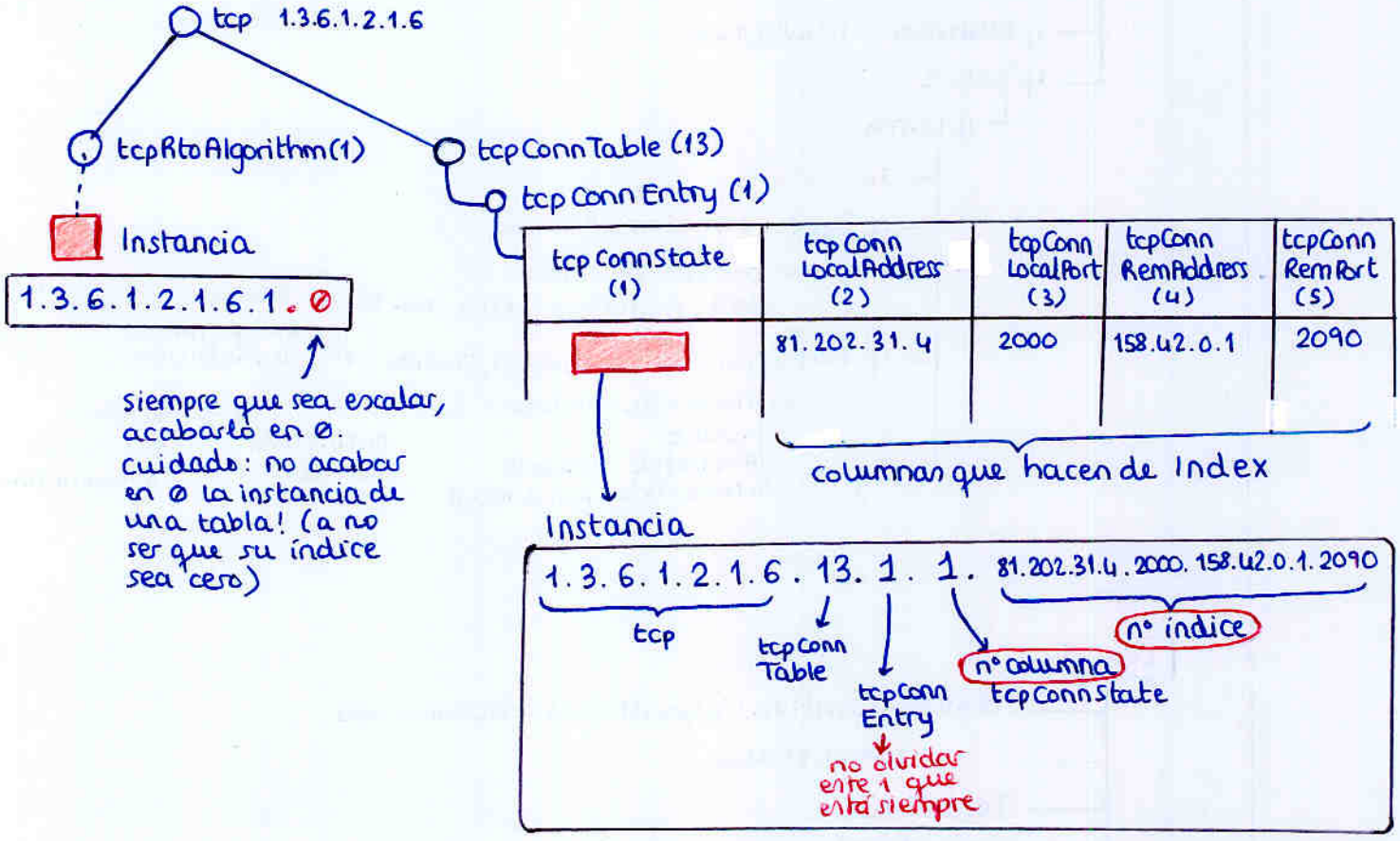


En el dispositivo definimos un **Perfil de acceso**:

community name	Vista MIB	Modo acceso
public	Toda MIB	RO
private	Toda MIB	RW

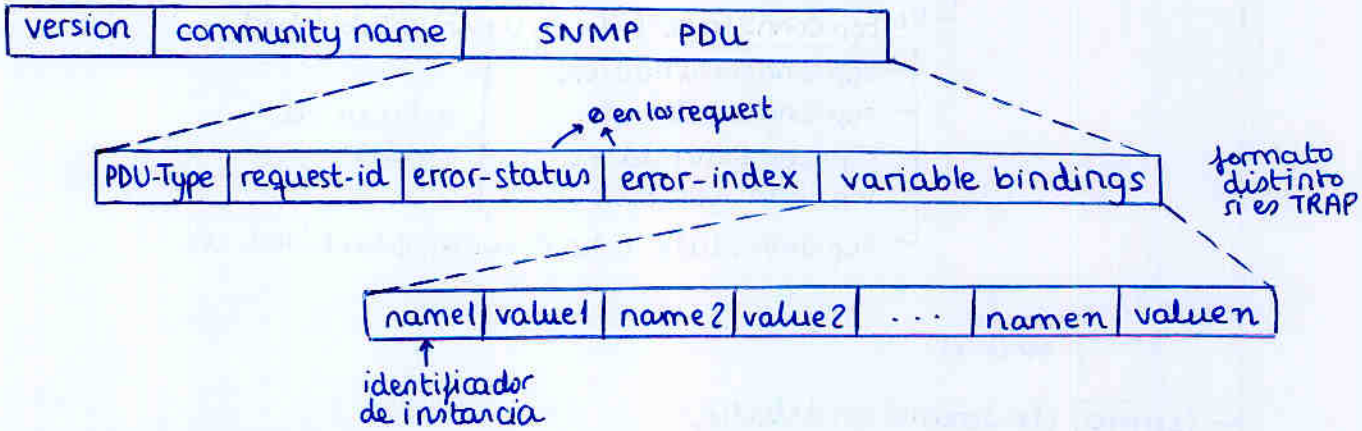
↑ las que vienen por defecto nosotros configuramos las que queramos

• Identificación de instancias

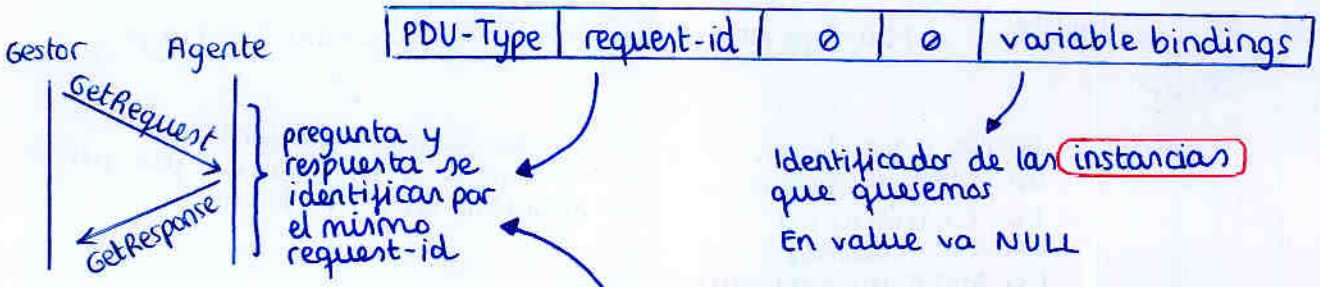


• Mensajes SNMP

Viaja en UDP (puertos 161, 162)

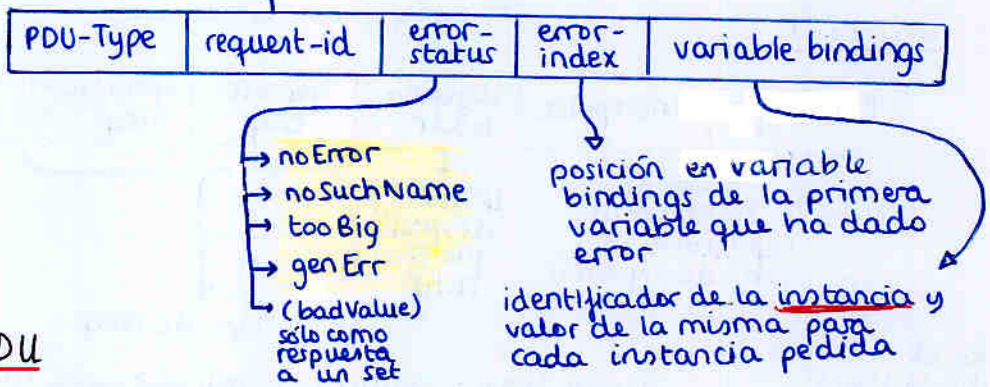


- **Get Request PDU**

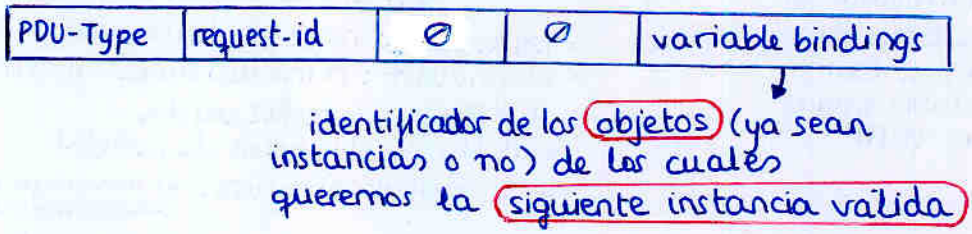


- **GetResponse PDU**

operación "atómica" (indivisible)
↓
Devuelvo todas o ninguna



- **GetNextRequest PDU**



ejemplo: GetRequest simple

- GetRequest (tcpInSegs.0, tcpOutSegs.0)
  GetResponse((tcpInSegs.0 = 1986), (tcpOutSegs.0 = 56))

ejemplos: GetNextRequest

- GetNextRequest (tcp)
  GetResponse ((tcpRtoAlgorithm.0 = vanj))
- GetNextRequest (tcpInSegs, tcpOutSegs)
  GetResponse ((tcpInSegs.0 = 1986), (tcpOutSegs.0 = 56))
- GetNextRequest (tcpInSegs.2) identif erróneo! le da igual, él busca la siguiente instancia válida
  GetResponse ((tcpOutSegs.0 = 56))

ejemplo: Leer tabla con GetNextRequest

ifTable

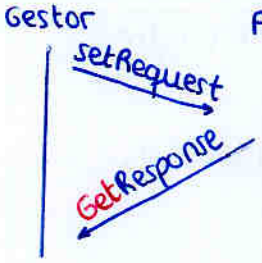
ifEntry

ifIndex	ifType	ifSpeed
1	fddi	10
2	fddi	100

- GetNextRequest(ifIndex, ifType, ifSpeed)
  GetResponse (ifIndex.1 = 1, ifType.1 = fddi, ifSpeed.1 = 10)
- GetNextRequest (ifIndex.1, ifType.1, ifSped.1)
  GetResponse (ifIndex.2 = 2, ifType.2 = fddi, ifSpeed.2 = 100)
- GetNextRequest (ifIndex.2, ifType.2, ifSpeed.2)
  GetResponse (ifType.1 = fddi, ifSpeed.1 = 10, ifPhyAddress.1 = ..)

el gestor ve que se han acabado las filas

· SetRequest PDU



PDU-Type	request-id	0	0	variable bindings

variable bindings → lista de identificadores de instancia y los valores que quiero ESCRIBIR en la MIB

Agente responde con un GetResponse con las instancias y nuevos valores
i.e. MISMO mensaje que antes → PERO permite nuevo código error-status: badValue

nota: código 'error-status = read-only' está definido, pero por error no lo utilizan y se usa 'noSuchName'

· Trap PDU

PDU Type	enterprise	agent-addr	generic-trap	specific-trap	time stamp	variable bindings

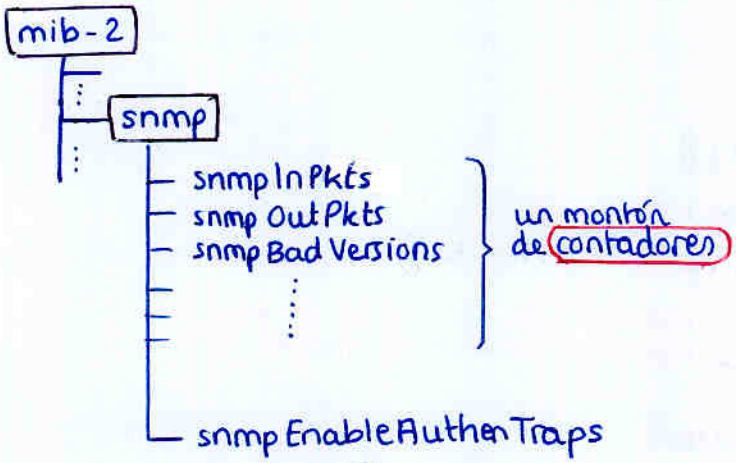
- enterprise → valor de sysObjectId (apunta a un nodo de empresas)
- agent-addr → IP del dispositivo que envía TRAP
- generic-trap + specific-trap → Tipo de trap
- variable bindings → información adicional

En el dispositivo configuramos la lista de IP's a las cuales enviar los TRAP

generic trap: condiciones estándar ante las cuales todos los dispositivos envían trap

- coldstart: reset por fallo grave
- warmstart: reinicialización intencionada
- linkDown: interfaz caída
- linkUp: activación de interfaz
- authenticationFailure: el mensaje ha recibido mensaje con community name erróneo
- enterpriseSpecific: evento configurado por el fabricante (campo specific-trap) → opcional configurable

· Grupo SNMP



un montón de contadores

- InTooBigs
- InNoSuchNames
- InGenErrs
- OutTooBigs
- OutGetNexts
- OutGetResponses
- OutTraps

Recuerda: la MIB puede estar también en un gestor (que a fin de cuentas es tb un dispositivo) por eso hay variables que sólo tienen sentido en el gestor y otras que sólo tienen sentido en el agente

snmpEnableAuthenTraps ↑ la única variable que no es un contador de sólo lectura
Configura si envías TRAP bajo la condición authenticationFailure o no: enabled(1) disabled(2)

· Limitaciones SNMPv1

- baja eficiencia de acceso a tablas → SNMPv2
- seguridad → SNMPv3
- orientado sólo a dispositivos (no redes) → RMON
- Alarmas muy limitadas → RMON

5. SNMPv2

Nota: estudiamos SNMPv2 V.R. (versión revisada) tb llamada SNMPv2C (community name), ya que la V.O. incluía seguridad pero no se extendió por perder sencillez

- **SMIv2** cambia la forma de definir objetos

- Objetos escalares:

```
nombre OBJECT-TYPE
    SYNTAX Counter64        → nuevos tipos de datos para más o menos bits
                              Unsigned32, Counter64, Gauge32
    UNITS "packets"         → descriptivo, no obligatorio
    MAX-ACCESS read-only    → MAX enfatiza q está por encima de permisos:
    STATUS current
    DESCRIPTION "..."
    DEFVAL {30}             → valor por def. al arrancar
    ::= { demo 1}
```

STATUS current → sólo 3 opciones:
- current
- obsolete
- deprecated

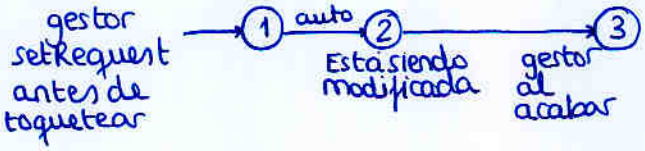
Añade opciones nuevas y quita innecesarias, dejando:
- not-accesible
- read-only
- read-write
- read-create (para la columna STATUS de las tablas)
- accesible-for-notify (no accesible (ni en lectura) pero el agente las puede usar para enviar en el campo variablebindings de un TRAP)

- Objetos Tabla

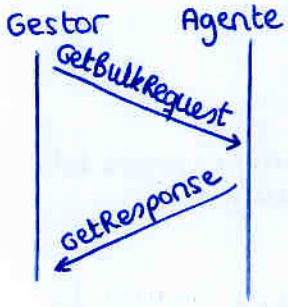
Se definen igual pero tienen una nueva columna

```
nombretablaStatus OBJECT-TYPE
    SYNTAX RowStatus        → nuevo tipo de datos (integer numerado)
      ⋮
    MAX-ACCESS read-create
    ::= {nombretablaEntry 6}
```

Es un integer numerado que permite implementar un semáforo para añadir y quitar filas



- **Protocolo** Dos nuevas PDU
- **GetBulkRequest** Permite coger grandes cantidades de datos



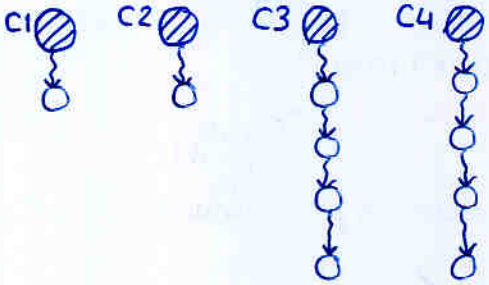
PDU Type	request id	non-repeaters	max-repetitions	variable bindings

nº de variables (de las primeras de var. bind.) de las que solicitamos sólo la siguiente instancia válida

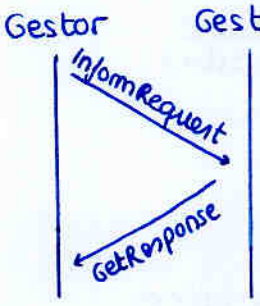
Para el resto de variables del campo var. bind. indica la cantidad de sucesivas instancias válidas que queremos

ejemplo

GetBulkRequest [non-repeaters=2, max-repetitions=4] (C1, C2, C3, C4)



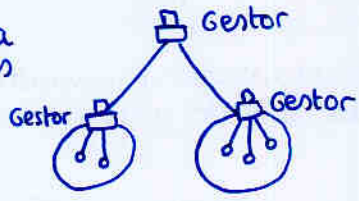
- **Inform Request** es como un trap confirmado entre gestores
  - ↳ pero el PDU no se parece al trap, sino al GetRequest



PDU Type	request-id	∅	∅	variable bindings

↑ lleva info de gestión

Permite construir una jerarquía de gestores



6. SNMPv3

→ Arquitectura modular (guía para implementar gestores y agentes)

→ Seguridad (cifrado, autenticación, temporización)

PDU SNMPv1, v2

Mensaje SNMPv3

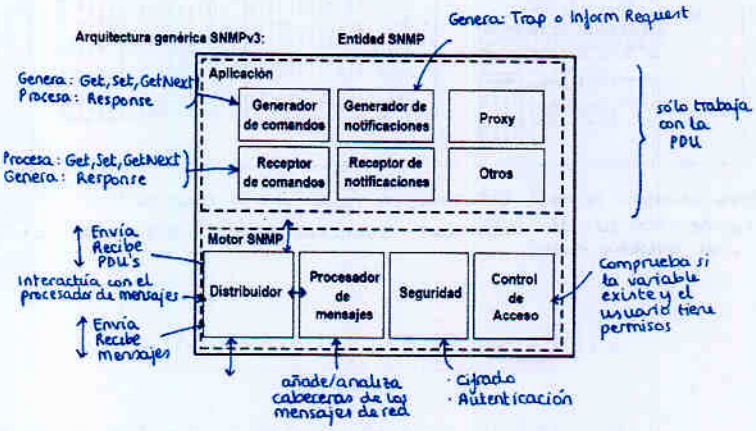
• Arquitectura modular

Entidad SNMP ≡ Agente ó Gestor →

(+) Optimización en implementación

(+) Mejora de módulos independientemente

· mensajes entre módulos (primitivas)



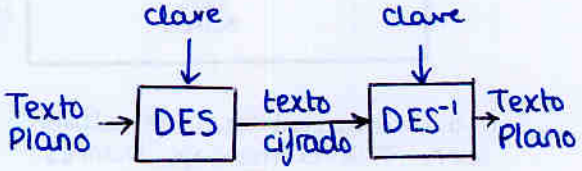
• Seguridad

USM User security module

- → Cifrado: DES (Data Encryption Standard)
- → Autenticación: HMAC (Hash Message Authentication Code) verifica si el mensaje es auténtico
- → Temporización: Tiempo máximo de validez de un mensaje a partir del cual lo ignora (para que un atacante, a pesar de no saber descifrar el mensaje, no pueda guardarse el mensaje para utilizarlo cuando quiera)

→ DES (Data Encription Standard)

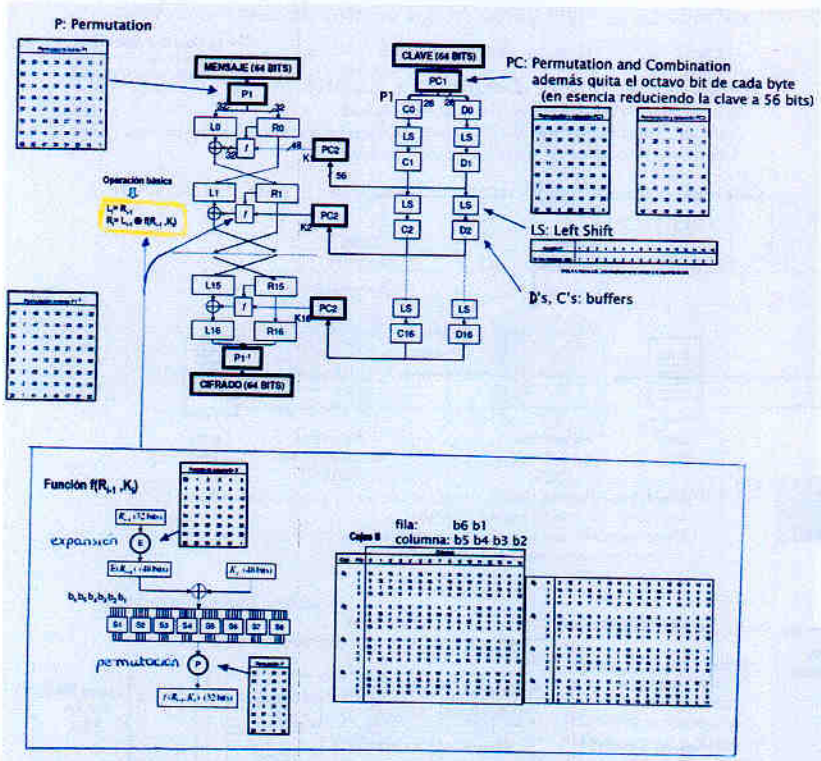
Cifrado simétrico y de una sola clave (igual en tx y rx)



Tipos de ataque:

- Fuerza bruta
- Criptoanálisis

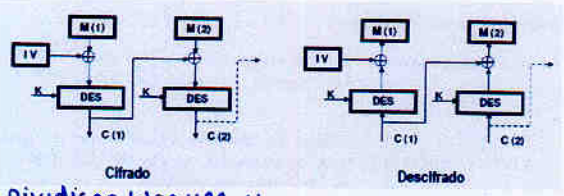
Algortimo de cifrado DES



Para descifrar se hace EXACTAMENTE igual pero de abajo arriba
i.e. necesito calcular todas las subclaves K1, K2, ... K16 antes de empezar nada.

Cipher Block Chaining (CBC)

para mensajes de más de 64 bits

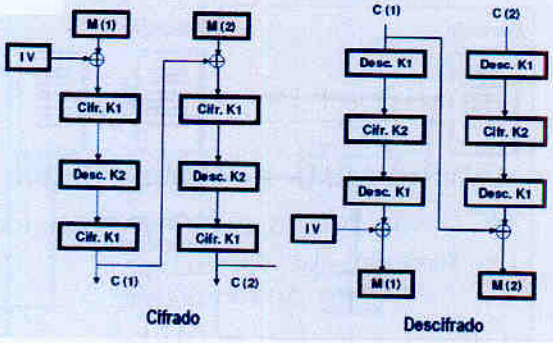


Dividir en bloques, y en lugar de cifrar cada uno: antes se suma con el bloque cifrado anterior

Triple DES

para mayor seguridad

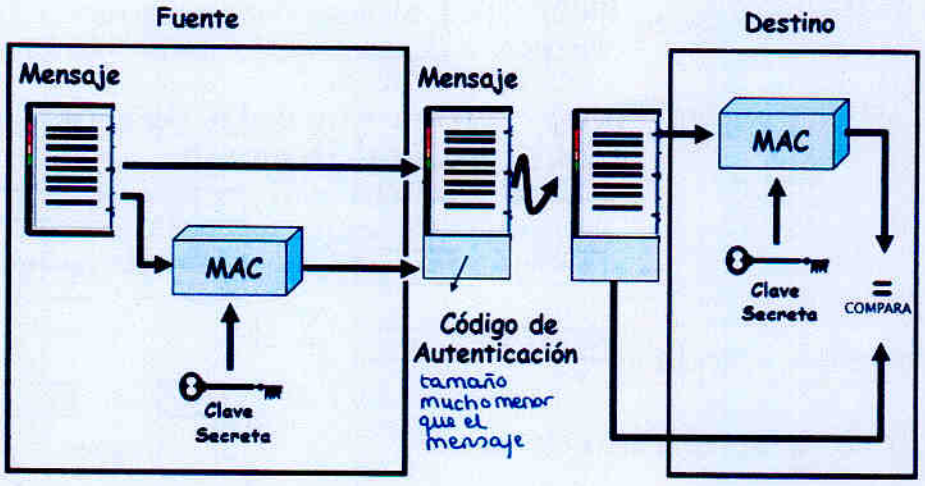
Como ves requiere 2 claves



→ Autenticación HMAC

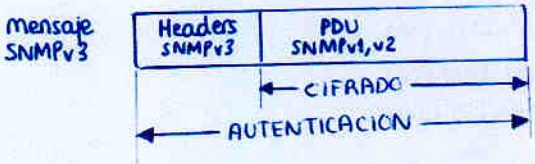
Lo usan varios protocolos ej: SNMPv3, IP, TLS, etc...

Garantiza que el mensaje no ha sido alterado y que la fuente es la que dice ser



Nota: la autenticación se calcula sobre TODO el mensaje SNMPv3

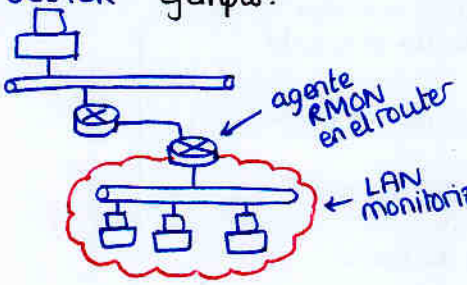
Para ello se considera siempre (tanto en tx como rx) que el campo "código de autenticación" son todo ceros, antes de pasar el mensaje al bloque MAC



7. Monitorización de red remota RMON

- Extensión de SNMP
- Un agente RMON en un dispositivo (por ej un router) escucha de sus interfaces y almacena información y estadísticas (las mismas que guardaría un analizador de protocolos) en la MIB RMON. ← nuevo nodo dentro de mib-2
- Un gestor puede recuperar la información de esa MIB por SNMP como si el agente RMON fuera un agente SNMP normal.

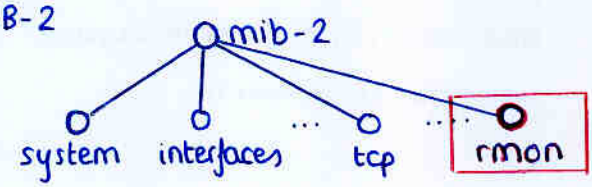
GESTOR ejemplo:



(+) Captura todo el tráfico (filtros de captura)
(+) Almacena info
(+) Estadísticas
} igual que un analiz. de protoc.

(+) Monitorización remota desde gestor (no requiere desplazarse)
(+) Comunicación con gestor por protocolo estándar SNMP
} ventajas sobre un analizador de protocolos

Nuevo nodo en MIB-2



los nodos dentro de RMON (los veremos) contienen únicamente tablas de control y de datos (o única tabla con ambas)

---

ejemplo: nodo history

Tabla de control (SIEMPRE ESTAN en tablas de datos)

índice	parámetros de control	_owner	_status
1		monitor	valid
2		pepe	valid

ej: → origen de datos (ifIndex de interfaz)
→ intervalo captura
→ max nº muestras

Columna de tipo ownerString para saber quien ha configurado la fila (para respetar, no para seguridad)

el fabricante deja algunas config. por defecto con owner = "monitor"

Nota: al BORRAR una fila de la tabla de control, autom. se borran las asociadas de datos

Columna de tipo entryStatus ::=
INTEGER {
valid(1)
createRequest(2)
underCreation(3)
invalid(4)
}

Implementa un semáforo para evitar problemas de concurrencia
- gestor escribe un 2 cuando quiera configurar fila
- automaticamente pasa a 3
- cuando acabe gestor escribe 1

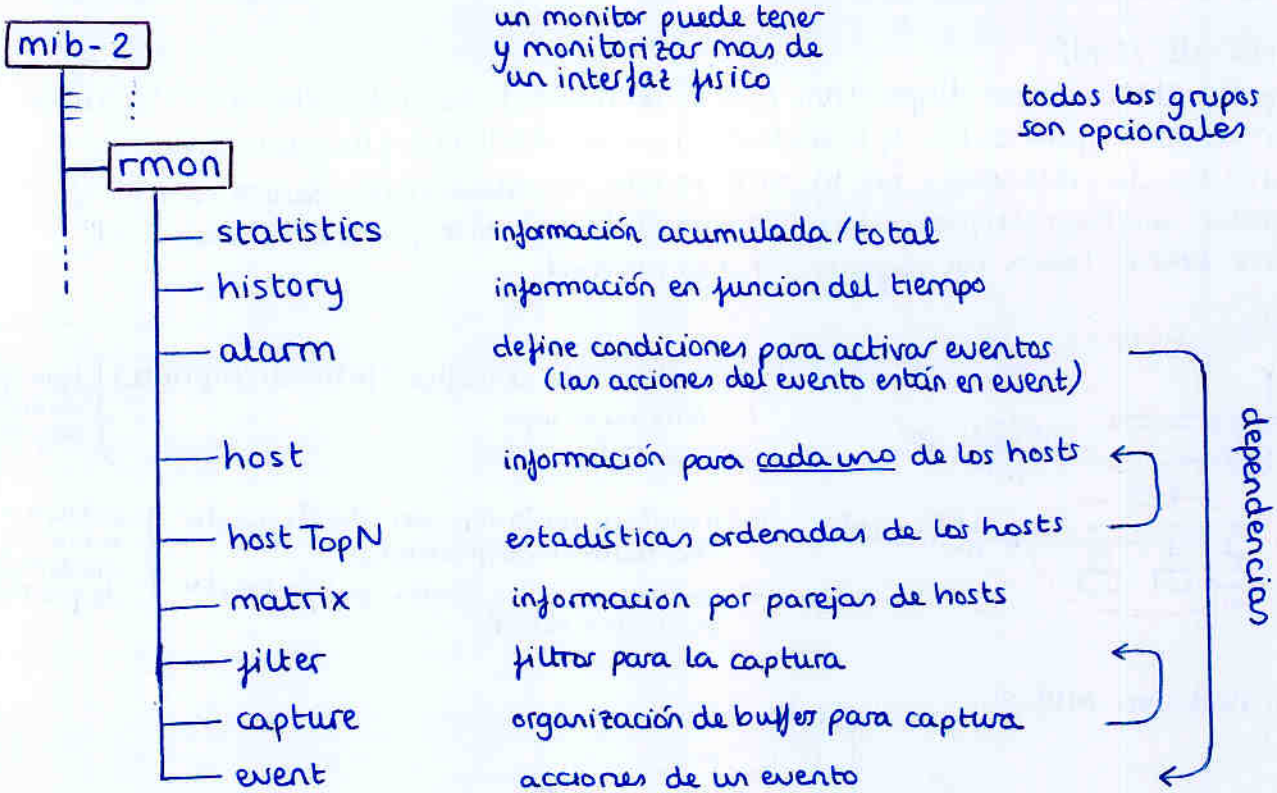
Tabla de datos

índice	Sample Index	datos
1	29	...
1	30	
1	26	...
1	27	
1	28	
2	5	
2	2	...
2	3	
2	4	

identifica a qué fila de control corresponde

Nada más se pone una fila de control a 'valid', el agente se pone a almacenar datos y a guardarlos aquí para cada intervalo; para diferenciar las distintas medidas va incrementando índice2 para cada sucesiva medida, así cuando sobreescriba circularmente las medidas por haber llegado al máximo nº de muestras, podremos saber cual es la entrada más vieja

- **MIB RMON**



Nota: RMON-2

Incluye más grupos

↳ para información de protocolos a nivel de aplicación

↳ información en función de la dirección IP

- **Grupo statistics**

- actualmente sólo contiene objetos para la interfaz ethernet, aunque hay extensiones

- Una UNICA TABLA (control + datos)

→ Ésto es posible ya que la tabla de datos no requiere filas, simplemente añade columnas a la de control → contadores para esa interfaz

→ Hay una única fila para cada interfaz monitorizada

etherStatsTable
↳ etherStatsEntry

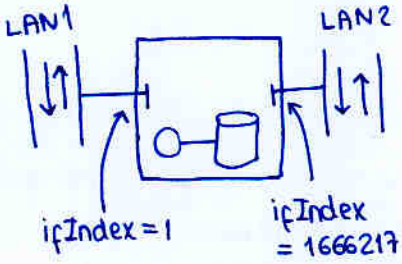
escribo '_' para no repetir 'etherstats'

_Index	_Data Source	···· Datos ···	_Owner	_Status
1	1		admin	1
2	1666217		admin	1

valor de ifIndex de la interfaz a monitorizar

cada columna es un contador de algo para la fila / interfaz correspondiente

- _DropEvents
- Octets
- Pkts
- BroadcastPkts
- MulticastPkts
- CRCAlignErrors
- UndersizePkts (<64)
- OversizePkts (>1518)
- Fragments (<64 & CRCerroneo)
- Jabbers (>1518 & CRC erroneo)
- Collisions
- ⋮ contadores de paquetes para distintos tamaños de paquete



- **Grupo history** captura "a lo largo del tiempo"

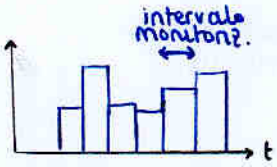
Consta de 2 tablas:

CONTROL : historyControlTable
DATOS : etherHistoryTable

- para una misma interfaz pueden haber más de un proceso de muestreo, pero deben tener periodo de muestreo diferente
- la especificación recomienda dos entradas por interfaz, monitorizando 30seg y 30min de periodo de muestreo

Parámetros de control:

- Interfaz a escuchar
- Intervalo de monitorización (tiempo durante el cual cuento para almacenar la siguiente muestra)
- máximo número de muestras (buckets requested)
  - por razones de memoria puede no tener espacio y pone un número máximo menor (buckets granted)



ej: intervalo 30s, max nº muestras 120 → tengo información histórica de la última hora

ejemplo:

historyControlTable

_Index	_DataSource	_BktsReq	_BktsGrntd	_Interval	_Owner	_Status
1	1	120	120	30s	admin	1
2	1	96	96	30min	admin	1
3	1666217	120	120	30s	admin	1
4	1666217	96	96	30min	admin	1

etherHistoryTable

_Index	_SampleIndex	_IntervalStart	...
1 ... 1	1 ... 120		
2 ... 2	1 ... 96		
3 ... 3	1 ... 120		
4 ... 4	1 ... 96		

columnas:
- _DropEvents → eventos que no he tenido tiempo de registrar en contadores
- Octets
- Pkts
- Broadcast Pkts
- Multicast Pkts
- CRC Align Errors
- Undersize Pkts
- Oversize Pkts
- Fragments
- Jabbers
- Collisions
- Utilization

_IntervalStart: sysUpTime de cuando empezó la captura

Tema 1. Introducción a la Gestión de Red

1. Introducción
2. Sistemas Propietarios
3. Sistema de Gestión de Red estándar
4. Areas Funcionales
5. Estándares de Gestión

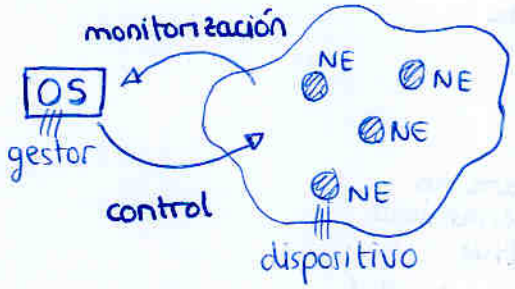
1. Introducción

Red de comunicaciones
- Complejas → Diferentes Servicios en la misma infraestructura (internet, teléfono, TV, ...)
- Heterogéneas → Equipos de diferentes fabricantes

Sistemas de Gestión

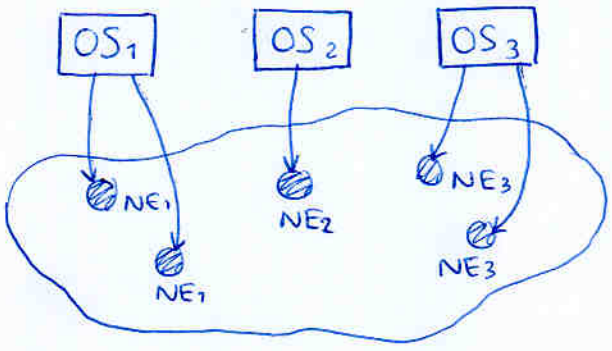
Definición: Gestión de red
El conjunto de funciones que permiten el intercambio y procesamiento de información con el objetivo de monitorizar y controlar una red de comunicaciones para su uso de la forma más eficiente posible y al menor coste.

Definición: elementos de red
- NE (Network Element) son los dispositivos ≡ recursos que queremos gestionar (recursos tanto HW como SW)
- OS (operating system) → Gestor : gestionan los NE



2. Sistemas Propietarios

"Legacy Systems"

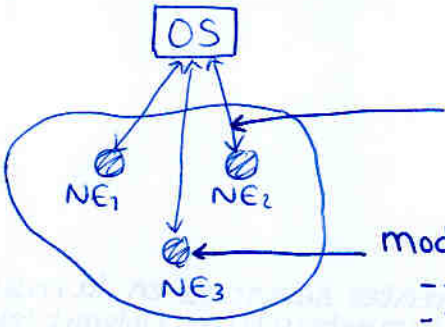


Al darte el dispositivo te obligan a comprar también el OS del fabricante para poder gestionarlo

Al crecer la red, crece el número de OS necesarios

Inconvenientes:
(-): Solución Parcial
(-): Complejas (hay que conocerla a fondo)
(-): Caras (formación de personas)
(-): Incompatibilidad (no pueden compartir resultados) (no pueden "colaborar")

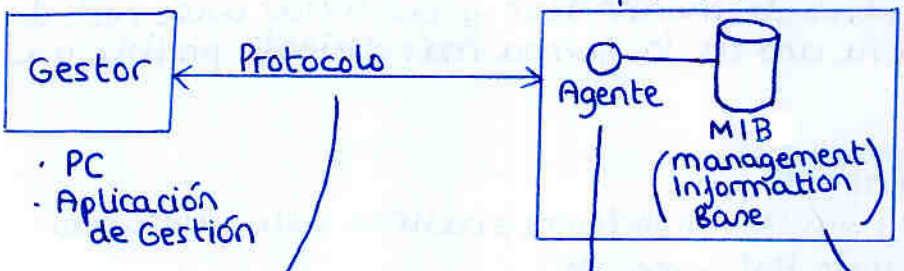
Se reunieron todas las empresas y organismos de estandarización para poder lograr:



Protocolo de comunicaciones

Modelo de información
- estructura de la base de datos
- variables
- lenguaje de especificación

3 Sistemas de Gestión de Red Estándar



- PC
- Aplicación de Gestión

Protocolo de nivel de aplicación que funciona sobre el nivel de transporte

→ proceso software que se ejecuta en el dispositivo.

→ En tiempo real obtiene información del dispositivo y la guarda en la base de datos

→ Debe ser eficiente para no degradar el funcionamiento del propio dispositivo

→ Capaz de recibir mensajes del gestor, leer/escribir en la MIB y contestar al gestor

Es una representación como variables del dispositivo

Tipo de información

- Estática ej: información de configuración (ej nº interfaces)
- Dinámica ej: información de red (ej: nº paquetes recibidos)
- Estadística ej: información procesada (ej: nº medio de paquetes/segundo)

↓ Este procesamiento no lo suele hacer el agente, sino el gestor

Métodos de acceso

- Polling (petición/respuesta)

  agente recibe la petición
  consulta la base de datos
  responde a la petición

  (+) Sencillo
  - La frecuencia del polling depende de:
    - lo crítica que sea la variable
    - el número de dispositivos que tengo que gestionar
  - ¿Cuántos dispositivos puede gestionar un gestor?
    depende de
    - prestaciones del gestor
    - AB en el enlace del gestor

- Event Report (alarmas)

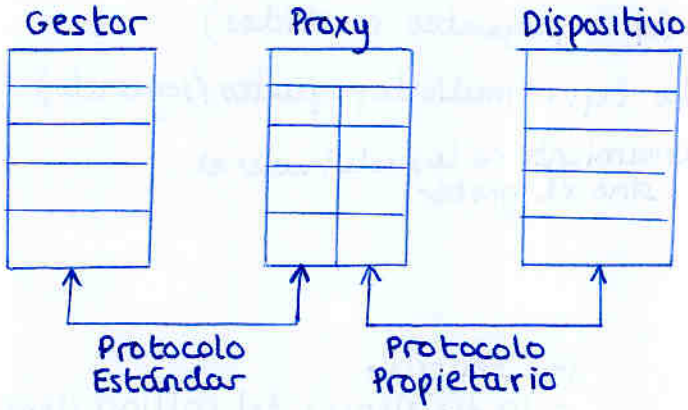
  Es el agente el que bajo determinada condición (umbral de una variable) envía un mensaje de event report

  - Comunicación asíncrona
  - Hay que configurar las alarmas

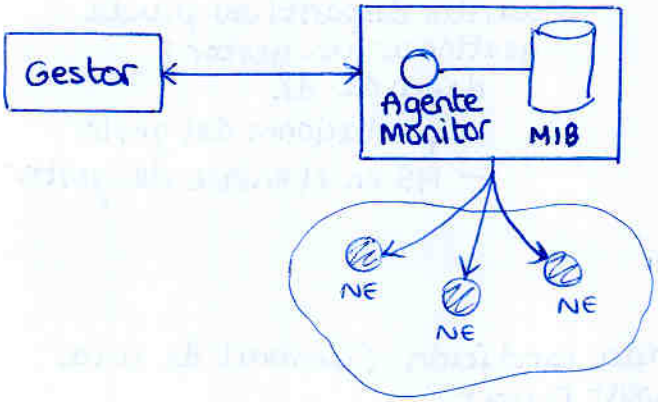
  (+) Permite un uso eficiente del AB (no hay que estar haciendo polling)
  (+) Conocimiento inmediato de la situación (no hay que esperar al próximo polling)

  Si sólo utilizásemos alarmas no podríamos enterarnos de si la comunicación dispositivo-gestor ha caído; hay que usarlas junto al polling

- Proxy ≡ Traductor

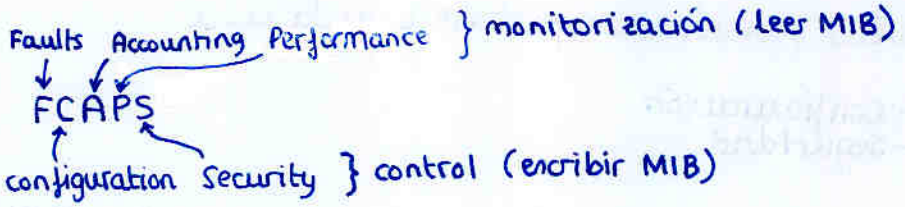


- Agente monitor



El agente monitor del dispositivo, además de recopilar su información, recopila la de toda una subred almacenándola en su propia base de datos

4. Areas Funcionales



- Gestión de prestaciones: evaluar los dispositivos y la red (throughput, utilización, disponibilidad, tiempo respuesta, tasa errores)
- Gestión de fallos: localizar problemas, aislar los dispositivos y resolver el problema
- Gestión de contabilidad: establecer parámetros por los que cobrar (ej: por imprimir) intervalos de tiempo, método de facturación o análisis
- Gestión de configuración: inicializar y actualizar información en los dispositivos (facilitan inventario y actualización)
- Gestión de seguridad: gestión de claves para cifrar, mecanismos de seguridad, ...

5. Estándares de gestión

Basicamente existen dos:

- SNMP: - gestion de redes TCP/IP
  - desarrollado por IETF
- TMN: - grandes redes públicas de comunicaciones
  - desarrollado por UIT-T

Además ISO tiene en su modelo OSI incluida la gestión de red para el modelo (que en realidad nadie utiliza, pero se 'cogen ideas')

Cuestiones

- Indicar a qué areas funcionales pertenece cada cosa:
  - Antivirus: – configuración
    – seguridad
  - Estado dispositivos de interconexión: – Prestaciones (disponibilidad)
    – Fallos
  - Registro de accesos a servidor: – seguridad
    – contabilidad
  - Monitorización uso impresora: – Contabilidad
    – Prestaciones
- Indicar qué accion se hace en la base de datos cuando se hace:
  - Monitorización: Leer base de datos
  - Control: Escribir base de datos

Tema 2. Modelos de Gestión de Red SNMP

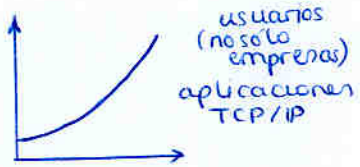
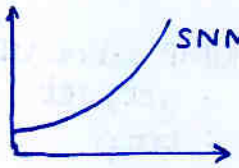
1. Conceptos Básicos
2. Estructura de Información (SMI)
3. MIB_2
4. SNMPv1
5. SNMPv2
6. SNMPv3
7. RMON

1. Conceptos básicos

1.1 Introducción

- SNMP es un protocolo de nivel de aplicación que utiliza UDP (que forma parte de la pila de protocolos TCP/IP)

Siempre ha habido una fuerte relación entre la gestión de red y la pila TCP/IP.

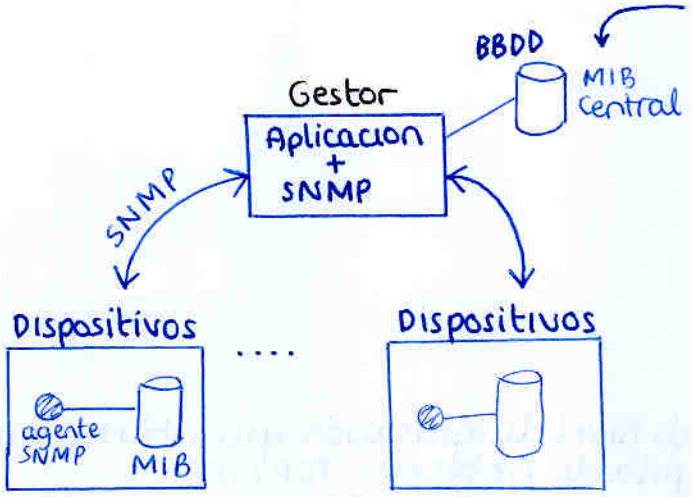
	TCP/IP	Gestión de Red
1970's	DoD (Departamento de Defensa)	No necesidad de sistemas de gestión basta con ICMP (internet control message protocol)
	Arpanet → TCP/IP	
1980's	Internet → - Universidades - Centros Públicos	Se plantea la necesidad como se pensaba q todo sería OSI al final, se propone:
	IETF → RFC's → simple	corto plazo: SNMP (simple network management protocol)
	ISO → OSI → complejo poderoso	medio plazo: HEM (high-entity management)
	(todo el mundo pensabe que todo acabaría siendo OSI)	largo plazo: CMIP (common management information protocol) (el de OSI)
		CMIP over TCP/IP → CMOT
Actual		

RFC 1155 - SMI
RFC 1213 - MIB_2
RFC 1157 - Protocolo SNMPv1

www.rfc-editor.org

1.2 Sistema de gestión de red SNMP

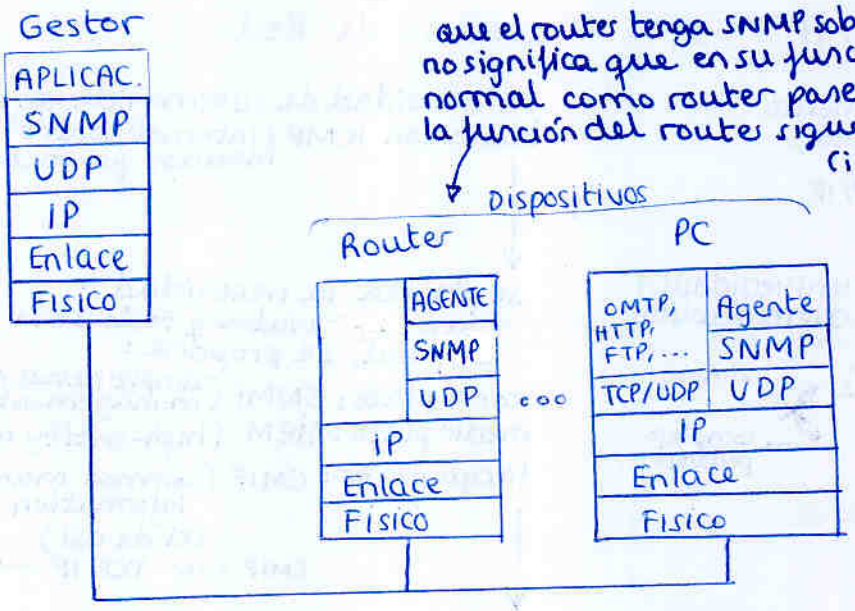
· Arquitectura del sistema



no está estandarizado pero es lógico necesitar base de datos para los datos recogidos así como una MIB central para tener un mapa/listado de variables por las cuales preguntar

nota: la red con la cual hacemos la gestión es la misma que queremos gestionar

· Arquitectura de protocolos



Puertos que utiliza SNMP sobre UDP
Puesto 161 : get, set
Puesto 162 : trap

¿Porqué se utiliza UDP?
Inconvenientes de TCP:
- orientado a la conexión: sobrecarga de tráfico por el establecimiento de la conexión
  - perder tiempo (lento al establecer la conexión)
  - sobrecarga en el gestor por mantener las conexiones

Ventajas de TCP?
- fiable mediante retransmisiones
  pero esto puede ser malo, cuando realmente interesa la gestión de redes por haber congestión, el gestor colaborará a empeorar la situación con las retransmisiones.

- Polling dirigido por TRAP
  - frecuencia de polling sea baja por defecto
  - si recibes un TRAP;
    - aumentas la frecuencia de polling
    - interrogas al dispositivo que ha generado el TRAP y a sus vecinos.

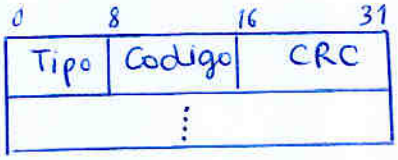
Se puede :· establecemos filtros de TRAP (para evitar o ignorar las alarmas de los vecinos que en realidad serán consecuencia de lo ya sabido)

Recordatorio : ICMP

Ya que IP es no fiable y sin conexión, se usan paquetes ICMP (obligatoriamente)
- router informa a la fuente de los errores con ICMP

Paquete ICMP

se encaminan por la red igual que un paquete IP sin ningún tipo de prioridad

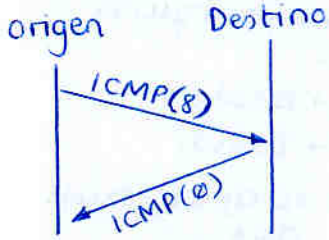


Esto va encapsulado en IP

Tipo 8 : Solicitud de eco
0 : Respuesta de eco
3 : Destino inaccesible
11 : Tiempo excedido para un datagrama (TTL) Time To Live
13 : Solicitud de Timestamp
14 : Respuesta de Timestamp
} Para que el origen calcule el RTT

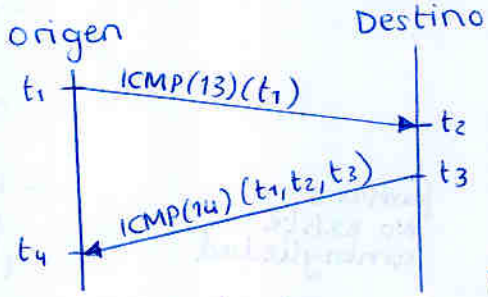
ejemplos

1) Destino Activo



ej : PING

2) cálculo del RTT



la fuente calcula

$RTT = (t_4 - t_1) - (t_3 - t_2)$

además no hace falta que los dos relojes estén sincronizados

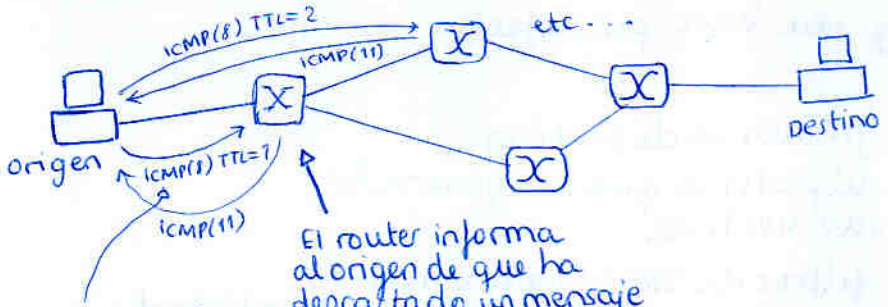
ICMP Tipo 13 y 14

Tipo	Cod	CRC
Identif		Nº secuenc.
Timestamp Origen		
Timestamp Destino		
Timestamp Transm.		

Todos los protocolos que se dedican a trabajar con tiempo usan esta filosofía

Neo Trace

3) Traceroute

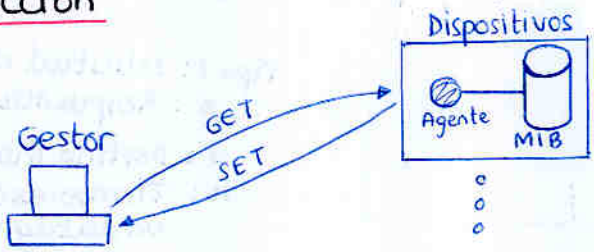


El router informa al origen de que ha descartado un mensaje suyo, por haberse agotado el TTL

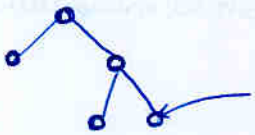
nota: en el nivel IP, la dirección IP destino es la del verdadero destino

2. Estructura de información de gestión (SMI)

2.1 Introducción



Management Information Base → Estructura de Objetos → Representación de recursos (HW, SW)



Definir Objetos

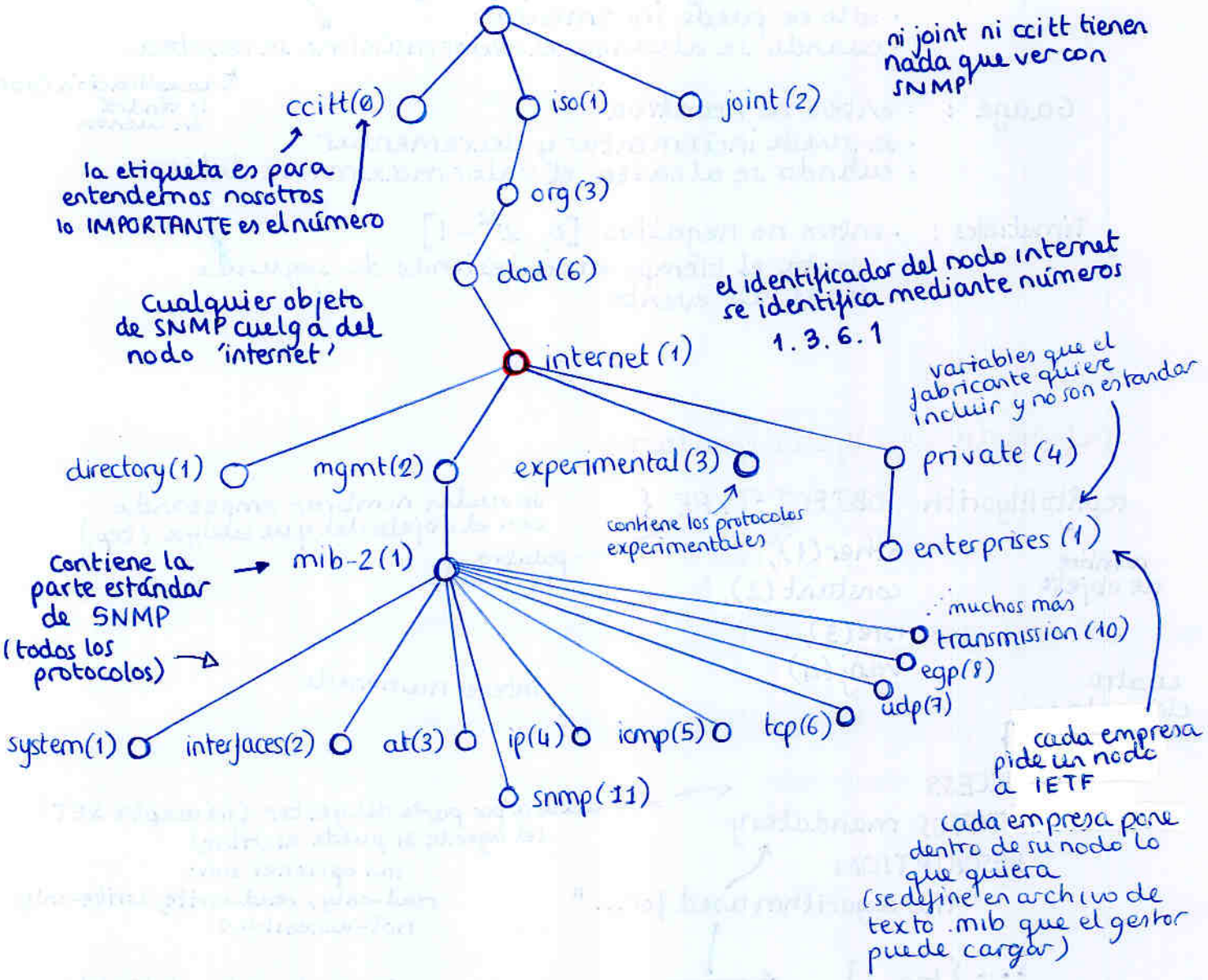
RFC 1155

- estructura jerárquica en árbol
- objetos { Escalar / Tabla
- Lenguaje de especificación formal ASN.1 (Abstract Syntax Notation)

formal ≡ No existe ambigüedad

2.2 SMI

2.2.1 Estructura de la MIB



2.2.2 Sintaxis de objetos

- Tipos de datos → simples = Universales
  → Aplicación = SNMP
- Objeto escalar
- Objeto tabla

Tipos de datos simples (subconjunto de la especificación formal SMI)

- integer : entero de 32 bits
- octet string : (cadena de caracteres, direccion IP, ...)
- null
- object identifier (identificador de objeto : ejemplo 1.3.6.1)
- sequence of (array) } se usan conjuntamente para definir tablas
- sequence (estructura) }

Tipos de datos SNMP

Networkaddress

Ipaddress (octet string (size(4))

Counter:
- entero no negativo $[0, 2^{32}-1]$
- sólo se puede incrementar
- cuando se alcanza el valor máximo se resetea

puede especificarlo menor a $2^{32}-1$

Gauge:
- entero no negativo $[0, 2^{32}-1]$
- se puede incrementar y decrementar
- cuando se alcanza el valor máximo se detiene

↑ la aplicación (gestor) lo tendrá en cuenta

TimeTicks:
- entero no negativo $[0, 2^{32}-1]$
- cuenta el tiempo en centésimas de segundo desde un evento

Definición de objetos escalares

```
tcpRtoAlgorith  OBJECT-TYPE
    SYNTAX INTEGER{
        other(1),
        constant(2),
        rsre(3),
        vanj(4)
    }
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The algorithm used for..."
    ::= {tcp 1}
```

- nombre de objeto
- se suelen nombrar empezando con el objeto del que cuelga (tcp)
- palabra reservada
- el tipo de datos con modificación: es integer numerado (puede valer 1, 2, 3 o 4)
- un valor para cada nombre de algoritmo
- cuatro cláusulas
- sólo lectura por parte del gestor (no acepta SET) (el agente sí puede escribir)
  las opciones son: read-only, read-write, write-only, not-accessible
- donde previamente se ha definido tcp, no como un objeto hoja como estamos viendo, sino como nodo:

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

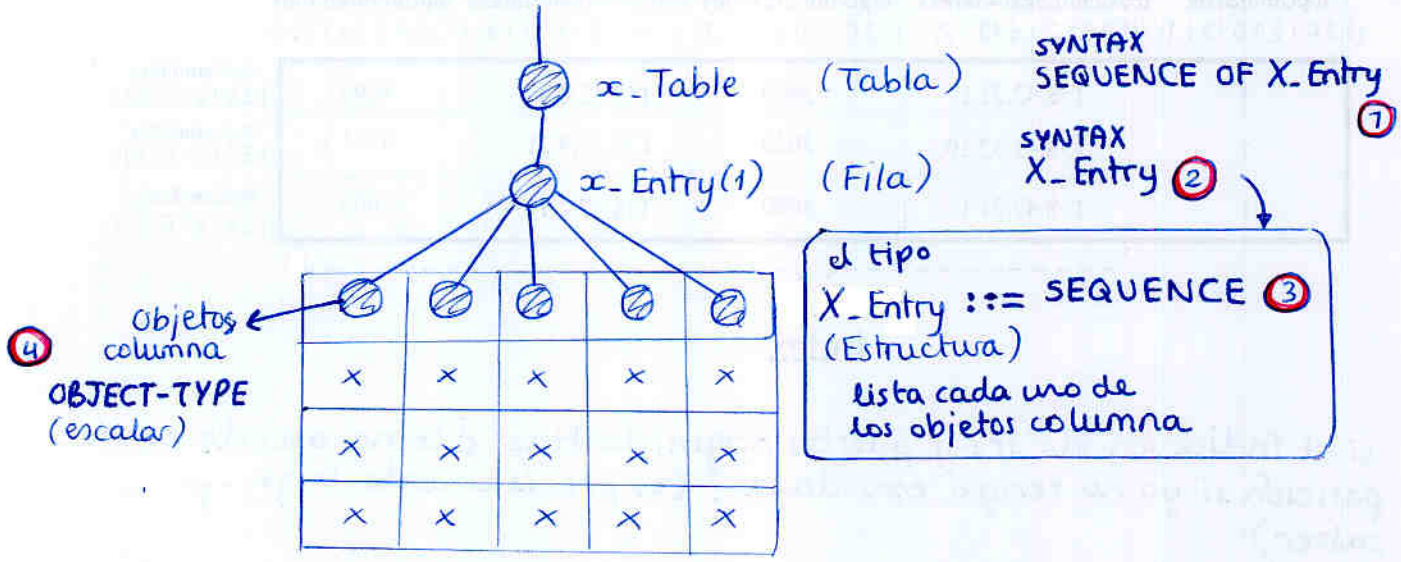
  a su vez, previamente definido mib-2, ... y así hasta iso

  iso OBJECT IDENTIFIER ::= {1}

si un fabricante quiere cumplir con el estándar SNMP-MIB2 debe incluir ésta variable actualizándose con valores reales (no vale llenar con ceros!)

opciones:
- mandatory
- optional
- obsolete ← se recomienda no implementar
- deprecated ← además de opcionales están quedando desfasadas

Definición de tablas



INDEX es una clausula más que debe haber en la definición del objeto FILA que determina los campos necesarios para distinguir de forma unívoca una fila en una tabla (tipicamente una columna aposta para el índice). A veces no hace falta una columna aposta para el índice, ya que se puede identificar sin ambigüedad cada fila con uno o varios campos (columnas)
ej: en la tabla de conexiones TCP se utiliza como índice las IP's y puertos de origen y destino

Definición de Objetos Tabla

```
1  tcpConnTable OBJECT-TYPE
     SYNTAX SEQUENCE OF TcpConnEntry
     ACCESS not-accesible
     STATUS mandatory
     DESCRIPTION
       "A table containing TCP
        connection-specific information."
     ::= { tcp 13 }

2  tcpConnEntry OBJECT-TYPE
     SYNTAX TcpConnEntry
     ACCESS not-accesible
     STATUS mandatory
     DESCRIPTION
       "Information about a particular
        current TCP connection ...."
     INDEX {tcpConnLocalAddress, tcpConnLocalPort,
            tcpConnRemAddress, tcpConnRemPort }
   ::= { tcpConnTable 1 }

3  TcpConnEntry::=
     SEQUENCE {
       tcpConnState          INTEGER,
       tcpConnLocalAddress   IpAddress,
       tcpConnLocalPort      INTEGER (0..65535),
       tcpConnRemAddress     IpAddress,
       tcpConnRemPort        INTEGER (0..65535)
     }

4  tcpConnState OBJECT-TYPE
     SYNTAX INTEGER {
                 closed(1),
                 listen(2),
                 synSent(3),
                 synReceived(4),
                 established(5), ...
     ACCESS   read-write
     STATUS   mandatory
     DESCRIPTION "The state of this TCP
                  connection"
   ::= { tcpConnEntry 1 }
```

tcpConnTable (1.3.6.1.2.1.6.13)

tcpConnState (1.3.6.1.2.1.6.13.1.1)	tcpConnLocalAddress (1.3.6.1.2.1.6.13.1.2)	tcpConnLocalPort (1.3.6.1.2.1.6.13.1.3)	tcpConnRemAddress (1.3.6.1.2.1.6.13.1.4)	tcpConnRemPort (1.3.6.1.2.1.6.13.1.5)	
3	158.42.21.1	2000	158.42.0.1	4090	tcpConnEntry (1.3.6.1.2.1.6.13.1)
7	158.42.32.19	3020	128.55.4.31	3221	tcpConnEntry (1.3.6.1.2.1.6.13.1)
1	158.42.21.1	3080	128.57.7.7	5600	tcpConnEntry (1.3.6.1.2.1.6.13.1)

index

si el índice son las IP's y puertos origen/destino ¿Como accedo a una posición si yo no tengo esos datos? (es precisamente lo que quiero saber)

Se 'ideó' el mensaje GetNext que ya veremos.

Notas:

- Rangos y Tamaños

```
OCTET STRING
OCTET STRING (SIZE(0..10))   ← tamaño entre 0 y 10
OCTET STRING (SIZE(5))       ← tamaño fijo

INTEGER
INTEGER (1..10)
INTEGER { other(1), opcion2(2), ... }   ← integer numerado
```

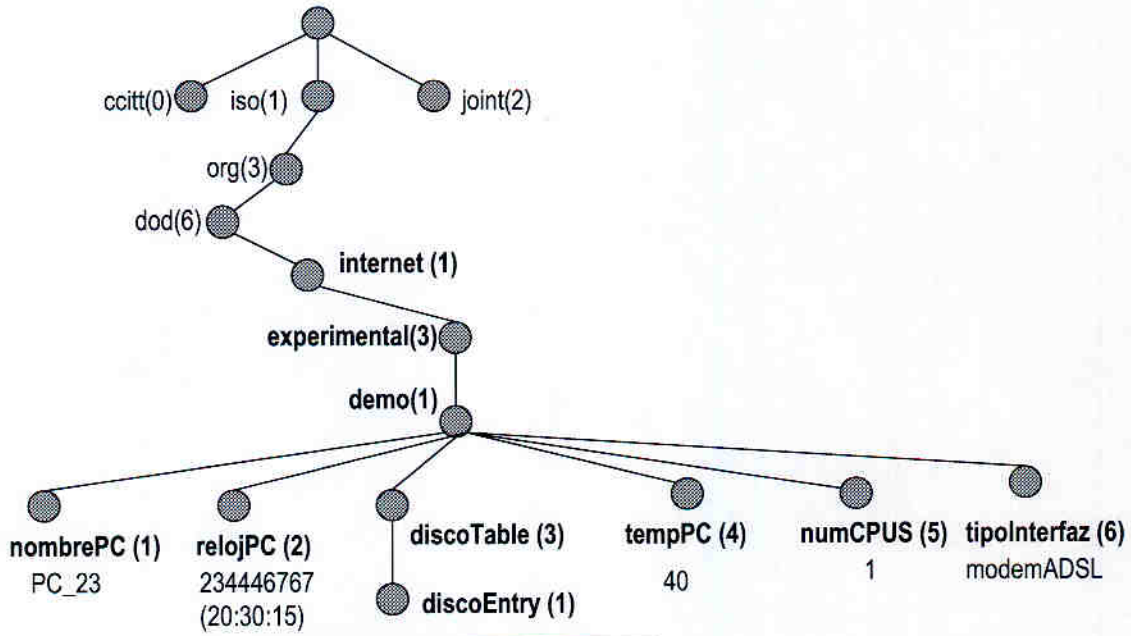
- Recursos en Internet
  - Definición de MIB estándar RFC 1213
  - Comprobación de módulos .mib y links a todas las aplicaciones de gestión de redes que quieras:

    www.simpleweb.org

Asignatura Gestión de Redes
Actividad SMI

Actividad

- Escribir utilizando la sintaxis SMI los siguientes objetos (aparecen en la siguiente figura).
  - Objeto *nombrePC* (parámetro de configuración obligatorio).
  - Objeto *relojPC* (parámetro de consulta opcional).
  - Objetos que forman el objeto *discoTable*. (index, unidad, capacidad)
  - Objeto *tempPC* (indica la temperatura de la CPU)
  - Objeto *numCPUS* (número de CPU's del PC)
  - Objeto *tipoInterfaz* (las diferentes opciones son, *modemRTB*, *adaptadorRDSI*, *modemADSL*, *ethernet*, *fastEthernet*, *atm*)



index	unidad	capacidad
1	C	1000
2	D	5000

II-2.3

Actividad SMI : Solución

```
ACTIVIDADSMI-MIB DEFINITIONS ::= BEGIN

    IMPORTS
        experimental, TimeTicks, Gauge
          FROM RFC1155-SMI

        OBJECT-TYPE
          FROM RFC-1212

    demo OBJECT IDENTIFIER ::= { experimental 1 }

    nombrePC OBJECT-TYPE
        SYNTAX OCTET STRING (SIZE (0...255))
        ACCESS read-write
        STATUS mandatory
        DESCRIPTION "Nombre del PC"
        ::= { demo 1 }

    relojPC OBJECT-TYPE
        SYNTAX TimeTicks
        ACCESS read-only
        STATUS optional
        DESCRIPTION "Reloj del PC (en centésimas de segundo)"
        ::= { demo 2 }

    discoTable OBJECT-TYPE
        SYNTAX SEQUENCE OF DiscoEntry
        ACCESS not-accessible ←
        STATUS mandatory
        DESCRIPTION "Tabla con los discos duros"
        ::= { demo 3 }

    discoEntry OBJECT-TYPE
        SYNTAX DiscoEntry
        ACCESS not-accessible ←
        STATUS mandatory
        DESCRIPTION "Información de un disco duro"
        INDEX { index }
        ::= { discoTable 1 }
```

hay que definir los objetos en el orden en que cuelgan

```
DiscoEntry ::=
    SEQUENCE {
        index       INTEGER (0..10),
        unidad      OCTET STRING (SIZE(1)),
        capacidad   INTEGER
    }
```

no hace falta indicarlo aun

Definimos el tipo DiscoEntry

```
index OBJECT-TYPE
    SYNTAX    INTEGER (0..10)
    ACCESS    read-only
    STATUS    mandatory
    DESCRIPTION "Indice"
    ::= { discoEntry 1 }
```

Es obligado rango (para saber la memoria que se debe reservar para la tabla)

```
unidad OBJECT-TYPE
    SYNTAX OCTET STRING (SIZE(1))
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION "  Unidad"
    ::= { discoEntry 2 }

capacidad OBJECT-TYPE
    SYNTAX integer
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION "Capacidad"
    ::= { discoEntry 3 }

tempPC OBJECT-TYPE
    SYNTAX Gauge
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION "Temperatura del PC"
    ::= {demo 4}

numCPUS OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION "Nº de CPUs"
    ::= {demo 5}

tipoInterfaz OBJECT-TYPE
    SYNTAX INTEGER { other(1), modemRTB(2), adaptadorRDSI(3),
                     modemADSL(4), ethernet(5), fastEthernet(6), atm(7) }
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION "Tipo de Interfaz"
    ::= {demo 6}

END
```

3. • MIB-2

Asignatura: Gestión de Redes

MIB-2: RFC 1213
- estándar
- protocolos TCP/IP

Criterios usados para definirla
- todos los objetos son actuales
- no hay duplicados
- todos los objetos son útiles

MIB-2

Grupos de la *MIB-2*

ya se recomienda no utilizarlo

MIB-2

Grupo ***system***

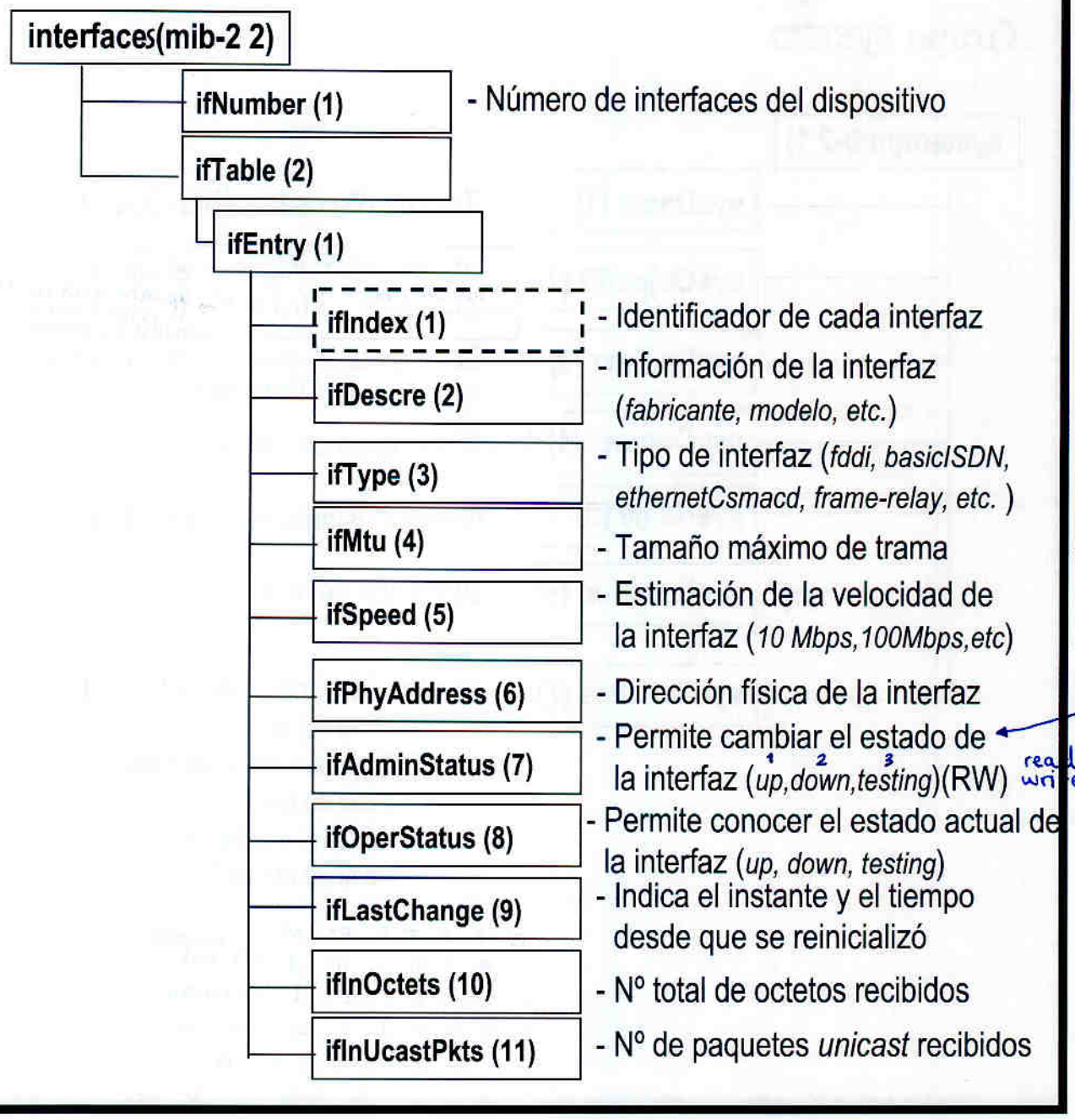
system(mib-2 1)

- **sysDescr (1)** - Tipo de dispositivo (*Hub 3COM, etc.*)
- **sysObjectID (2)** - Identificador asignado al objeto (1.3.6.1.4.1) tipo SYNTAX OBJECT IDENTIFIER → apunta a un nodo de privado/empresas con info del modelo
- **sysUpTime (3)** - Tiempo activo desde la reinicialización (*Timeticks*)
- **sysContact (4)** - Persona de contacto
- **sysName (5)** - Nombre asignado al dispositivo
- **sysLocation (6)** - Ubicación del dispositivo
- **sysServices (7)** - Niveles OSI implementados (7 bits)
  - 1: Nivel físico
  - 2: Nivel de enlace de datos
  - 3: Nivel de red
  - 4: Extremo a extremo
  - 7: Nivel de aplicación

A	P	S	T	R	ED	NF	
0	0	0	0	0	0	1	— [Repetidor Hub (1)
0	0	0	0	0	1	1	— [switch (3)
0	0	0	0	1	x	x	— [Router (4-7)
1	0	0	1	0	0	0	— [PC (72)

MIB-2

Grupo *interface*



Tema 4. MIB-2

Grupo *interface*

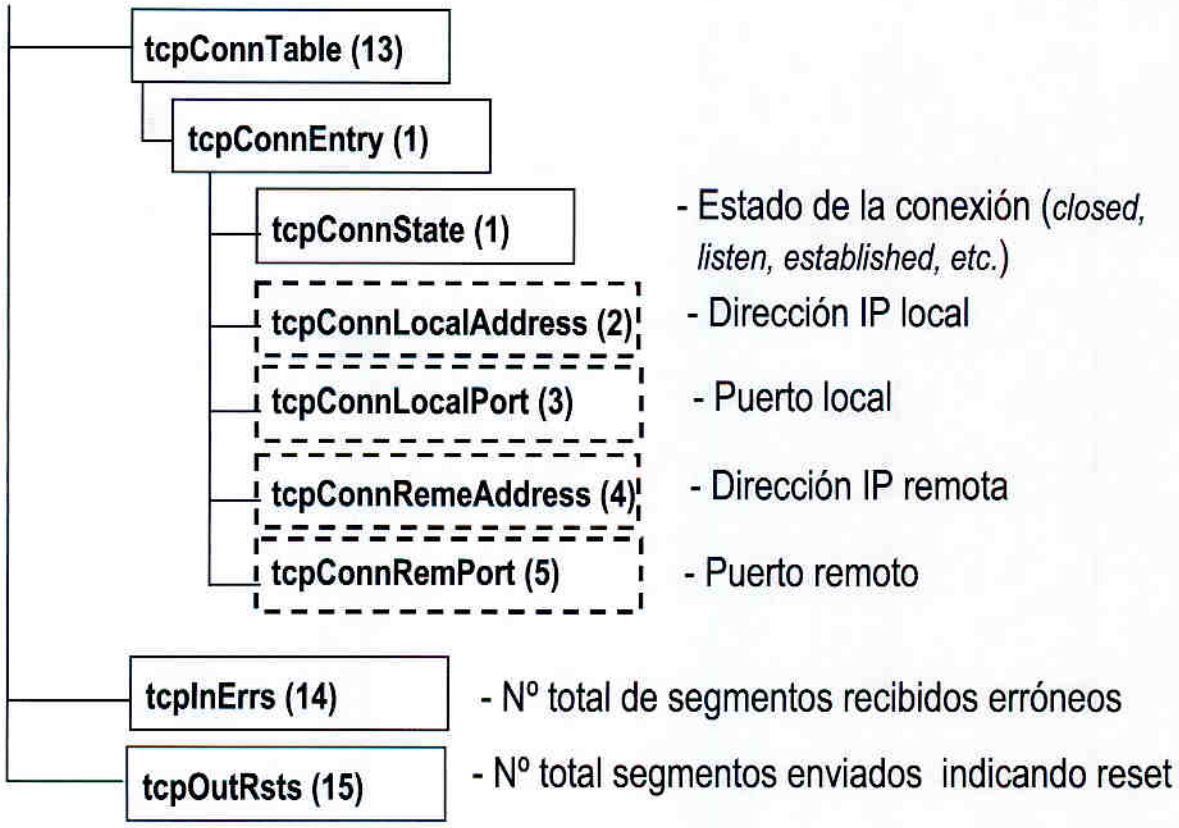
ifInNUcastPkts (12)	- Nº de paquetes *no-unicast* recibidos
ifInDiscards(13)	- Nº de paquetes recibidos descartados
ifInErrors (14)	- Nº de paquetes recibidos erróneos
ifInUnknownProtos(15)	- Nº de paquetes recibidos con protocolo desconocido
ifOutOctets (16)	- Nº total de octetos transmitidos
ifOutUcastPkts (17)	- Nº de paquetes *unicast* transmitidos
ifOutNUcastPkts (18)	- Nº de paquetes *no-unicast* transmitidos
ifOutDiscards(19)	- Nº de paquetes descartados
ifOutErrors (20)	- Nº de paquetes erróneos no transmitidos
ifOutQLen (21)	- Tamaño del buffer de transmisión
ifSpecific (22)	- Referencia a un objeto de la MIB relacionada con la interfaz física utilizada

Grupo *tcp*

tcp (mib-2 6)

tcpRtoAlgorithm (1)	- Algoritmo de retransmisión (*timeout*)
tcpRtoMin (2)	- Valor mínimo del temporizador
tcpRtoMax (3)	- Valor máximo del temporizador
tcpMaxConn (4)	- Nº máximo de conexiones TCP
tcpActiveOpens (5)	- Nº de conexiones activas soportadas
tcpPassiveOpens (6)	- Nº de conexiones pasivas soportadas
tcpAttemptFails (7)	- Nº de establecimientos erróneos
tcpEstabResets (8)	- Nº de resets que se han producido
tcpCurrEstab (9)	- Nº de conexiones en estado *establecido* o *esperando cierre*
tcpInSegs (10)	- Nº de segmentos recibidos
tcpOutSegs (11)	- Nº de segmentos transmitidos
tcpRetransSegs (12)	- Nº de segmentos retransmitidos

Grupo *tcp*


tcpConnTable (13)
tcpConnEntry (1)
tcpConnState (1)
- Estado de la conexión (closed, listen, established, etc.)
tcpConnLocalAddress (2)
- Dirección IP local
tcpConnLocalPort (3)
- Puerto local
tcpConnRemeAddress (4)
- Dirección IP remota
tcpConnRemPort (5)
- Puerto remoto
tcpInErrs (14)
- Nº total de segmentos recibidos erróneos
tcpOutRsts (15)
- Nº total segmentos enviados indicando reset

```
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 }

        -- historical (some say hysterical)
        -- cmot         OBJECT IDENTIFIER ::= { mib-2 9 }

        transmission OBJECT IDENTIFIER ::= { mib-2 10 }

        snmp            OBJECT IDENTIFIER ::= { mib-2 11 }
```

```
-- the System group

-- Implementation of the System group is mandatory for all
-- systems.  If an agent is not configured to have a value
-- for any of these variables, a string of length 0 is
-- returned.

sysDescr OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "A textual description of the entity.  This value
            should include the full name and version
            identification of the system's hardware type,
            software operating-system, and networking
            software.  It is mandatory that this only contain
            printable ASCII characters."
    ::= { system 1 }

sysObjectID OBJECT-TYPE
    SYNTAX  OBJECT IDENTIFIER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The vendor's authoritative identification of
            the
            network management subsystem contained in the
            entity.  This value is allocated within the SMI
            enterprises subtree (1.3.6.1.4.1) and provides
            an
            easy and unambiguous means for determining `what
            kind of box' is being managed.  For example, if
            vendor `Flintstones, Inc.' was assigned the
            subtree 1.3.6.1.4.1.4242, it could assign the
            identifier 1.3.6.1.4.1.4242.1.1 to its `Fred
            Router'."
    ::= { system 2 }

sysUpTime OBJECT-TYPE
    SYNTAX  TimeTicks
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The time (in hundredths of a second) since the
            network management portion of the system was
            last
            re-initialized."
    ::= { system 3 }

sysContact OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "The textual identification of the contact
            person
            for this managed node, together with information
            on how to contact this person."
```

```
    ::= { system 4 }

sysName OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "An administratively-assigned name for this
            managed node.  By convention, this is the node's
            fully-qualified domain name."
    ::= { system 5 }

sysLocation OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "The physical location of this node (e.g.,
            `telephone closet, 3rd floor')."
    ::= { system 6 }

sysServices OBJECT-TYPE
    SYNTAX  INTEGER (0..127)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "A value which indicates the set of services
            that this entity primarily offers.

            The value is a sum.  This sum initially takes
            the value zero, Then, for each layer, L, in the
            range 1 through 7, that this node performs
            transactions
            for, 2 raised to (L - 1) is added to the sum.
            For
            example, a node which performs primarily routing
            functions would have a value of 4 (2^(3-1)).  In
            contrast, a node which is a host offering
            application services would have a value of 72
            (2^(4-1) + 2^(7-1)).  Note that in the context
            of
            the Internet suite of protocols, values should
            be
            calculated accordingly:

                 layer  functionality
                     1  physical (e.g., repeaters)
                     2  datalink/subnetwork (e.g., bridges)
                     3  internet (e.g., IP gateways)
                     4  end-to-end  (e.g., IP hosts)
                     7  applications (e.g., mail relays)

            For systems including OSI protocols, layers 5
            and
            6 may also be counted."
    ::= { system 7 }
```

```
-- the TCP group

        -- Implementation of the TCP group is mandatory for all
        -- systems that implement the TCP.

        -- Note that instances of object types that represent
        -- information about a particular TCP connection are
        -- transient; they persist only as long as the connection
        -- in question.

        tcpRtoAlgorithm OBJECT-TYPE
            SYNTAX  INTEGER {
                        other(1),     -- none of the following

                        constant(2), -- a constant rto
                        rsre(3),      -- MIL-STD-1778, Appendix B
                        vanj(4)       -- Van Jacobson's algorithm
                    }
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
                    "The algorithm used to determine the timeout
                    value
                    used for retransmitting unacknowledged octets."
            ::= { tcp 1 }

        tcpRtoMin OBJECT-TYPE
            SYNTAX  INTEGER
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
                    "The minimum value permitted by a TCP
                    implementation for the retransmission timeout,
                    measured in milliseconds.  More refined
                    semantics
                    for objects of this type depend upon the
                    algorithm
                    used to determine the retransmission timeout.
                    In
                    particular, when the timeout algorithm is
                    rsre(3),
                    an object of this type has the semantics of the
                    LBOUND quantity described in RFC 793."
            ::= { tcp 2 }


        tcpRtoMax OBJECT-TYPE
            SYNTAX  INTEGER
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
                    "The maximum value permitted by a TCP
                    implementation for the retransmission timeout,
                    measured in milliseconds.  More refined
                    semantics
                    for objects of this type depend upon the
                    algorithm
                    used to determine the retransmission timeout.
                      In
                    particular, when the timeout algorithm is
```

```
            rsre(3),
            an object of this type has the semantics of the
            UBOUND quantity described in RFC 793."
    ::= { tcp 3 }

tcpMaxConn OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The limit on the total number of TCP
            connections
            the entity can support.  In entities where the
            maximum number of connections is dynamic, this
            object should contain the value -1."
    ::= { tcp 4 }

tcpActiveOpens OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of times TCP connections have made a
            direct transition to the SYN-SENT state from the
            CLOSED state."
    ::= { tcp 5 }

tcpPassiveOpens OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of times TCP connections have made a
            direct transition to the SYN-RCVD state from the
            LISTEN state."
    ::= { tcp 6 }


tcpAttemptFails OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of times TCP connections have made a
            direct transition to the CLOSED state from
              either
            the SYN-SENT state or the SYN-RCVD state, plus
              the
            number of times TCP connections have made a
              direct
            transition to the LISTEN state from the SYN-RCVD
            state."
    ::= { tcp 7 }

tcpEstabResets OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of times TCP connections have made a
            direct transition to the CLOSED state from
```

```
            either
            the ESTABLISHED state or the CLOSE-WAIT state."
    ::= { tcp 8 }

tcpCurrEstab OBJECT-TYPE
    SYNTAX  Gauge
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of TCP connections for which the
            current state is either ESTABLISHED or CLOSE-
            WAIT."
    ::= { tcp 9 }

tcpInSegs OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of segments received,
              including
            those received in error.  This count includes
            segments received on currently established
            connections."
    ::= { tcp 10 }

tcpOutSegs OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of segments sent, including
            those on current connections but excluding those
            containing only retransmitted octets."
    ::= { tcp 11 }

tcpRetransSegs OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of segments retransmitted -
              that
            is, the number of TCP segments transmitted
            containing one or more previously transmitted
            octets."
    ::= { tcp 12 }
```

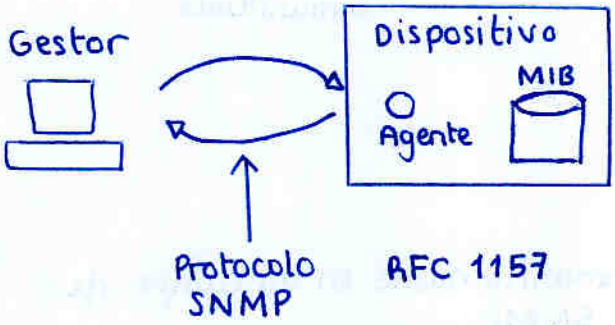
```
-- the TCP Connection table

-- The TCP connection table contains information about this
-- entity's existing TCP connections.

tcpConnTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF TcpConnEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "A table containing TCP connection-specific
            information."
    ::= { tcp 13 }

tcpConnEntry OBJECT-TYPE
    SYNTAX  TcpConnEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "Information about a particular current TCP
            connection.  An object of this type is
             transient,
            in that it ceases to exist when (or soon after)
            the connection makes the transition to the
             CLOSED
            state."
    INDEX   { tcpConnLocalAddress,
              tcpConnLocalPort,
              tcpConnRemAddress,
              tcpConnRemPort }
    ::= { tcpConnTable 1 }


TcpConnEntry ::=
    SEQUENCE {
        tcpConnState
            INTEGER,
        tcpConnLocalAddress
            IpAddress,
        tcpConnLocalPort
            INTEGER (0..65535),
        tcpConnRemAddress
            IpAddress,
        tcpConnRemPort
            INTEGER (0..65535)
    }
```

```
tcpConnState OBJECT-TYPE
    SYNTAX  INTEGER {
                closed(1),
                listen(2),
                synSent(3),
                synReceived(4),
                established(5),
                finWait1(6),
                finWait2(7),
                closeWait(8),
                lastAck(9),
                closing(10),
                timeWait(11),
                deleteTCB(12)
            }
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "The state of this TCP connection.

            The only value which may be set by a management
            station is deleteTCB(12).  Accordingly, it is
            appropriate for an agent to return a `badValue'
            response if a management station attempts to set
            this object to any other value.

            If a management station sets this object to the
            value deleteTCB(12), then this has the effect of
            deleting the TCB (as defined in RFC 793) of the
            corresponding connection on the managed node,
            resulting in immediate termination of the
            connection.

            As an implementation-specific option, a RST
            segment may be sent from the managed node to the
            other TCP endpoint (note however that RST
              segments
            are not sent reliably)."
    ::= { tcpConnEntry 1 }

tcpConnLocalAddress OBJECT-TYPE
    SYNTAX  IpAddress
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The local IP address for this TCP connection.
              In
            the case of a connection in the listen state
              which
            is willing to accept connections for any IP
            interface associated with the node, the value
            0.0.0.0 is used."
    ::= { tcpConnEntry 2 }

tcpConnLocalPort OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The local port number for this TCP connection."
    ::= { tcpConnEntry 3 }
```

```
tcpConnRemAddress OBJECT-TYPE
    SYNTAX  IpAddress
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The remote IP address for this TCP connection."
    ::= { tcpConnEntry 4 }

tcpConnRemPort OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The remote port number for this TCP
             connection."
    ::= { tcpConnEntry 5 }


-- additional TCP objects

tcpInErrs OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of segments received in error
            (e.g., bad TCP checksums)."
    ::= { tcp 14 }

tcpOutRsts OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of TCP segments sent containing the
            RST flag."
    ::= { tcp 15 }

.
.
.

END
```

4. SNMPv1

4.1 Introducción



- nivel de aplicación
- funciona sobre UDP (ya lo discutimos)

3 operaciones: get : leer de la MIB } "Polling"
set : escribir
trap : alarma } "Event Report"

Con SNMP sólo podemos/queremos acceder a los objetos HOJA (que es donde está la información)

4.2 Política de acceso

Mecanismo de seguridad: el objetivo es proteger al dispositivo gestionado y a la MIB asociada del acceso desde gestores no autorizados y limitar el acceso sobre determinados objetos de la MIB.

Proteger de acceso desde gestores no autorizados → se implementa mediante comunidades

Acceso a determinados objetos " privilegios → se implementa mediante perfil de acceso

Concepto COMUNIDAD

Estamos ante un sistema distribuido (varios gestores y varios dispositivos)

- Crear comunidades en el Agente (herramientas de configuración del dispositivo) (ej: router vía web) (la configuración de comunidades no se incluye en SNMP)

- Establecemos relación

Agente ⟷ Gestor

"community name" ≡ password

El gestor debe incluir el community name en un campo que tienen todos los mensajes SNMP

El campo no va cifrado; por tanto puedo fácilmente conocer el community name para poder usar mi gestor: grandísimo fallo de seguridad

PERFIL DE ACCESO

- usando herramientas de configuración del dispositivo
- se asocia a cada "community name" un perfil de acceso

Vista MIB: escojo los objetos que quiero que sean visibles en la MIB

Modo de acceso: sólo lectura (RO) o lectura/escritura (RW)

ejemplo: por defecto existen dos comunidades

community name	Vista MIB	Modo acceso
public	Toda MIB	RO
private	Toda MIB	RW

Perfil de acceso

obviamente la escritura sólo se puede en los objetos definidos como read-write

4.3 Identificación de instancias

En SNMP no es lo mismo el OBJETO ej: tcpRtoAlgorith

que la INSTANCIA asociada (valor que contiene) ej: 4

Aunque en los objetos escalares no tiene mucho sentido, es necesario para las tablas.

Para referirnos a la INSTANCIA, hay que añadir .0 en un escalar

ejemplo tcpRtoAlgorithm
Identificador de objeto 1.3.6.1.2.1.6.1
Identificador de instancia 1.3.6.1.2.1.6.1.0

¿Qué ocurre con las tablas?

el objeto columna ya era conocido al definirlo, y luego la fila la identifico con el índice de esa fila (valor de las columnas)

ej:

tcpConnTable 1.3.6.1.2.1.6.13

↳ objeto fila 1.3.6.1.2.1.6.13.1

tcpConnState 1.3.6.1.2.1.6.13.1.1	tcpConnLocalAddress 1.3.6.1.2.1.6.13.1.2	tcpConnLocalPort ...1.3	tcpConnRemAddress ...1.4	tcpConnRemPort ...1.4
3	158.42.21.1	2000	158.42.0.1	4090
7	158.42.32.19	3020	128.55.4.31	3221

4 columnas del índex

Ésta instancia se define como:

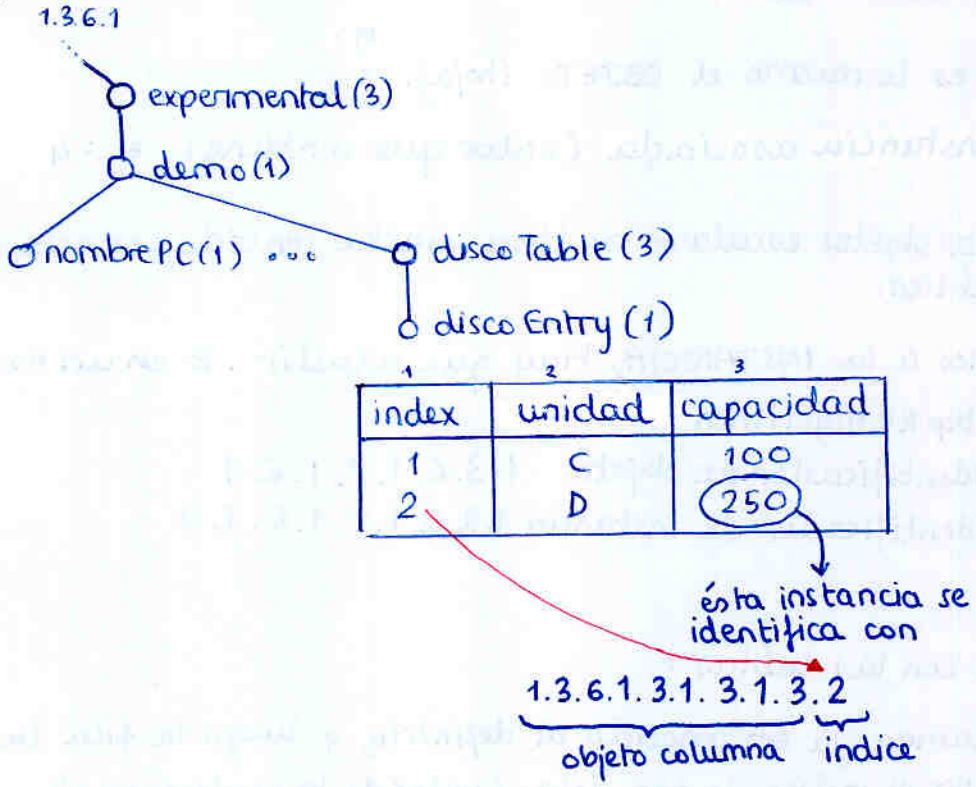
1.3.6.1.2.1.6.13.1.1. 158.42.21.1 . 2000 . 158.42.0.1 . 4090

objeto columna que queremos

valor de las columnas índice de la fila que queremos

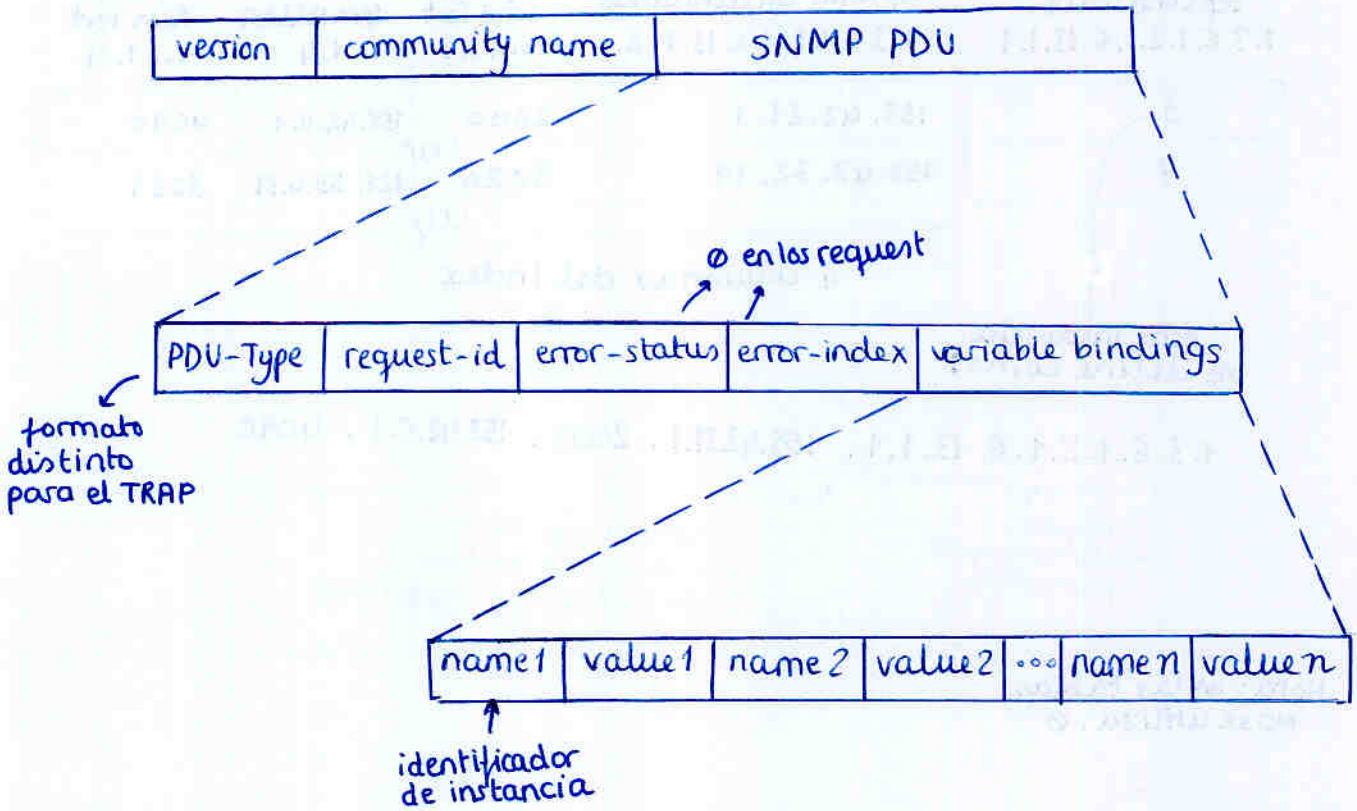
Nota: en las tablas NO se utiliza .0

ejemplo:



4.4 Mensajes SNMP

Puertos 161, 162



GetRequest PDU

Gestor Agente

identificador distinto para cada petición.
La respuesta GetResponse tendrá este mismo request-id

identificadores de todas las instancias que queremos (en el campo valor tienen NULL)

GetResponse PDU

Idéntico salvo por err-status y err-index y porque las variables ya llevan valor (a no ser que haya error)

- noError
- noSuchName → posición de la variable que ha dado error en el campo variable bindings
- tooBig
- genErr → posición de la variable que ha dado error en el campo variable bindings

nota: si varias dan error se devuelve la primera

Es una operación "atómica" (indivisible) si ha habido error no se devuelve NINGUNA variable

atómica → o todas o ninguna

ejemplo: usamos ésta nomenclatura

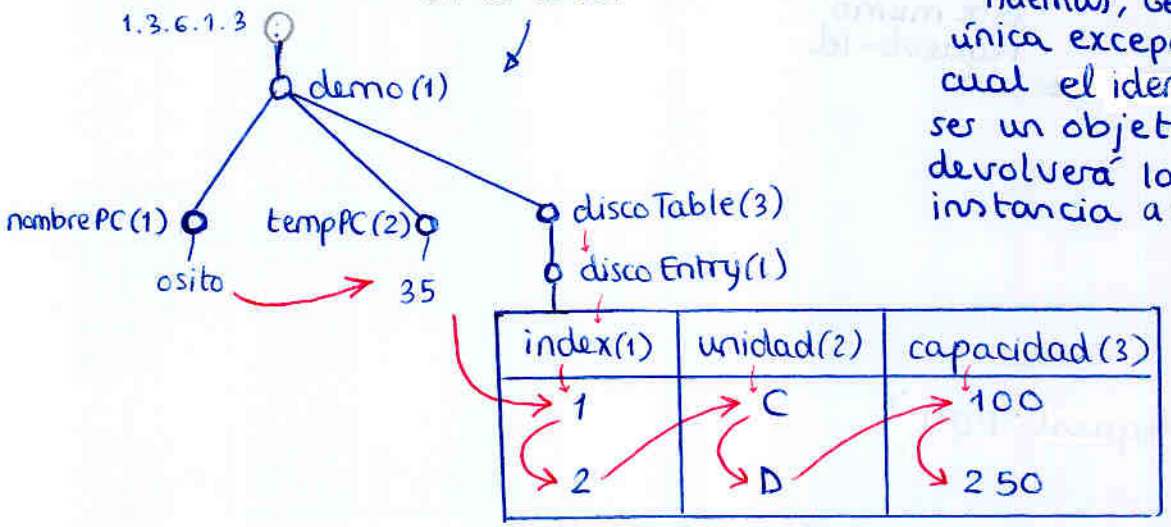
GetRequest (tcpInSegs. 0)
GetResponse (tcpInSegs. 0 = 2000)

GetNextRequest PDU

PDU-Type	request-id	0	0	variable bindings

Devuelve el valor de la SIGUIENTE INSTANCIA válida a la que pides

(SIGUIENTE: buscando recursivamente en el árbol; INSTANCIA: no objeto)



Además, GetNext es la única excepción en la cual el identif puede ser un objeto; y se te devolverá la siguiente instancia a ese objeto

ejemplo: GetNext con request-id 1.3.6.1.3.1.1.0 devuelve 35

GetNext con request-id 1.3.6.1.3.1.2.0 devuelve 1

y como hemos dicho, request-id puede ser un objeto

GetNext con request-id 1.3.6.1.3.1.3 devuelve 1

GetNext con request-id 1.3.6.1.3.1.3.1 devuelve 1

GetNext con request-id 1.3.6.1.3.1.3.1.1 devuelve 1

GetNext con request-id 1.3.6.1.3.1.3.1.2 devuelve C

1) recuperación de datos escalares

Con GetRequest: GetRequest (tcpInSegs.0, tcpOutSegs.0)
GetResponse ((tcpInSegs.0 = 2000), (tcpOutSegs.0 = 3500))

si escribimos mal un identif: GetRequest (tcpInsegs.2.3 !?, tcpOutSegs.0)
GetResponse (error-status = noSuchName)

Con GetNextRequest: (id de objeto (no instancia))
GetNextRequest (tcpInSegs, tcpOutSegs)
GetResponse ((tcpInSegs.0 = 2000), (tcpOutSegs.0 = 3500))

si escribimos mal un identif; él igualmente busca la siguiente instancia válida

GetNextRequest (tcpInsegs.2 ?!, tcpOutSegs)
GetResponse ((tcpOutsegs.0 = 3500), (tcpOutSegs.0 = 3500))

2) Recuperación de objetos desconocidos

No es necesario que los identificadores de instancias especificados sean válidas: pueden ser el id. de un objeto o incluso un id. inexistente, el igualmente buscará la SIGUIENTE INSTANCIA VÁLIDA

GetNextRequest (tcp)
GetResponse ((tcpRtoAlgorithm. 0= vanj))

3) Acceso a las tablas

ifTable

ifEntry

ifIndex	ifType	ifSpeed
1	ethernet-csmacd	10
2	fddi	100

GetNextRequest (ifIndex, ifType, ifSpeed)
GetResponse ((ifIndex.1 = 1), (ifType.1 = etherne-csmacd), (ifSpeed.1= 100))

usar la columna(s) que sean índice

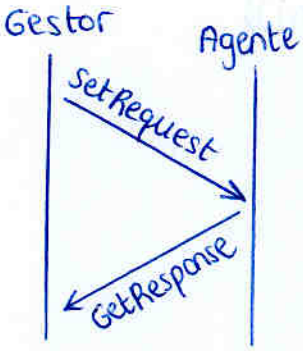
GetNextRequest (ifIndex.1, ifType.1, ifSpeed.1)
GetResponse ((ifIndex.2=2), (ifType.2 = fddi), (ifSpeed.2 = 100))

GetNextRequest (ifIndex.2, ifType.2, ifSpeed.2)
GetResponse ((ifType.1=ethernet-csmacd), (ifSpeed.1= 10), (ifPhyAddress.1=...))

el gestor puede saber que se han acabado las filas de la tabla

4.3 setRequest PDU

PDU-Type	request-id	∅	∅	variable bindings



- Permite escribir en la MIB
  (debo tener derecho a escribir y la variable debe ser de lectura/escritura)

Todo es igual al GetRequest salvo que variable bindings especifica una lista de instancias y los valores que se deben escribir en la MIB

El mensaje de respuesta es de tipo GetResponse (el mismo que respondía a GetRequest); y devuelve la lista de variables con los valores correspondientes

SetRequest (sysName.0 = HubLabTel)
GetResponse (sysName.0 = HubLabTel)

En éste caso, la respuesta GetResponse incluye un tipo de error adicional: badValue

- campo error-status de GetResponse cuando contesta a un setRequest

noError
noSuchName
tooBig
genErr
→ badValue

en la RFC hay una errata
readOnly está especificado pero luego dijeron que si era de sólo lectura devolviesen noSuchName
por tanto el código de error readOnly existe pero no se utiliza

- Añadir una fila a una Tabla

Si yo estoy modificando una Tabla, debería previamente bloquearla para que no entre otro a mitad; pero ya lo veremos mientras tanto, se puede hacer la chapucilla:

ipRouteTable

ipRouteDest	ipRouteMetric1	ipRouteNextHop
9.1.2.3	3	99.0.0.3
10.0.0.51	5	158.1.1.32

- añadir una fila a una tabla:

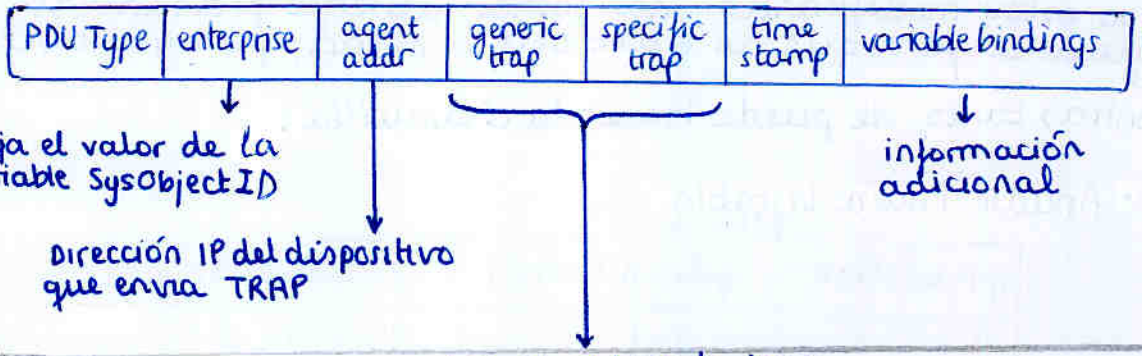
```
SetRequest ((ipRouteDest.11.3.2.12 = 11.3.2.12),
            (ipRouteMetric1.11.3.2.12 = 9),
            (ipRouteNextHop.11.3.2.12 = 91.0.0.5))

GetResponse ((ipRouteDest.11.3.2.12 = 11.3.2.12),
             (ipRouteMetric1.11.3.2.12 = 9),
             (ipRouteNextHop.11.3.2.12 = 91.0.0.5))
```

- borrar una fila de una tabla

en algunas tablas existe un campo que le indica al agente que debe borrar la fila

```
SetRequest ((ipRouteType.11.3.2.12 = invalid))
GetResponse ((ipRouteType.11.3.2.12 = invalid))
```

4.4 Trap PDU



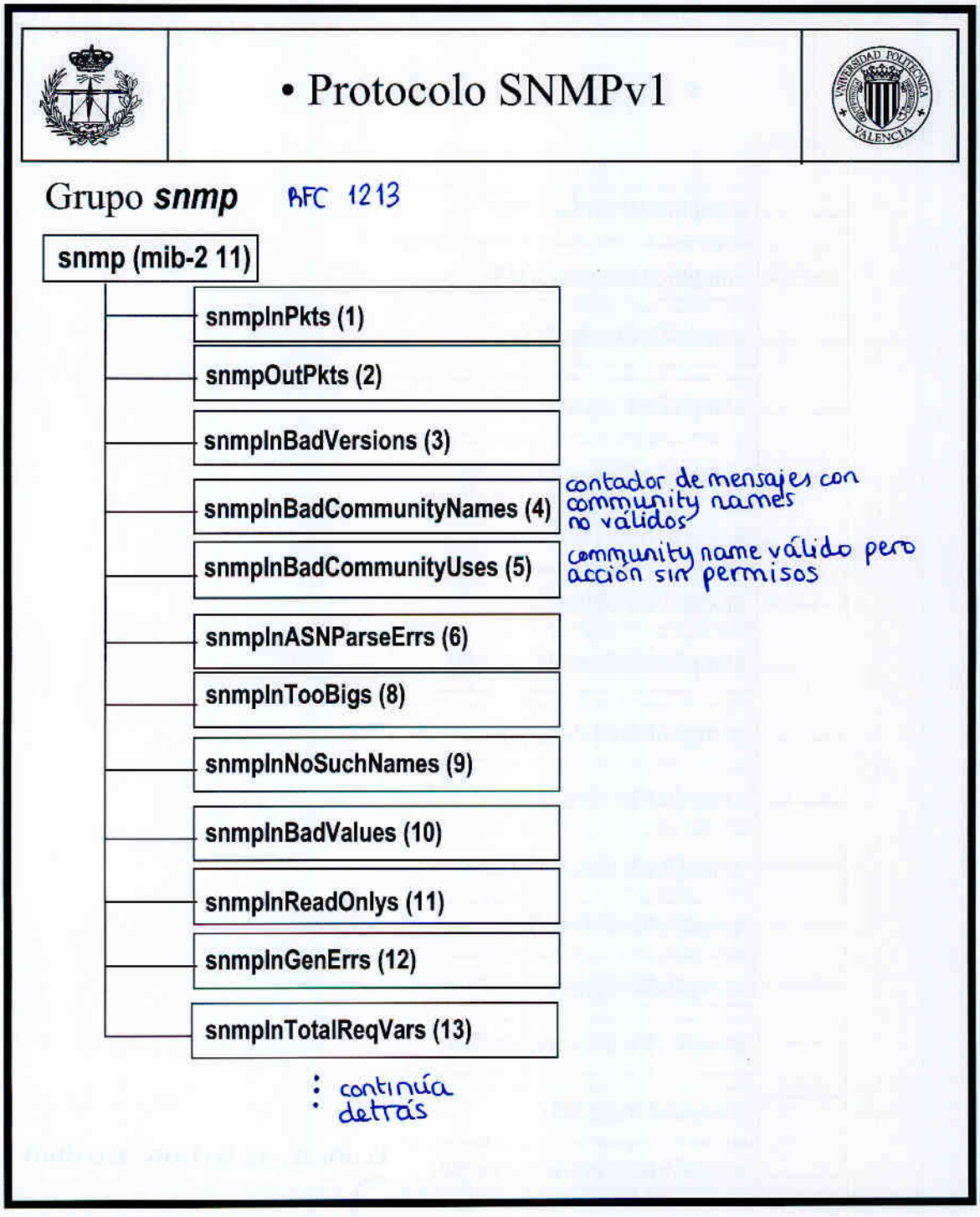
contienen el tipo de trap

generic trap : condiciones estándar ante las cuales todos los dispositivos envían TRAP

- coldStart : fallo grave ha provocado reinicialización del sistema
- warmStart : reinicialización intencionada del sistema
- linkDown : fallo en una de las interfaces del dispositivo (variable bindings : numero de interfaz ifIndex)
- linkUp : activación de una de las interfaces (variable bindings : numero de interfaz ifIndex)
- authenticationFailure : el agente ha recibido un mensaje con community name erróneo (se puede configurar con variable snmpEnableAuthenTraps)
- enterpriseSpecific : el evento corresponde con un evento configurado por el fabricante, indicado en el campo specific trap

La lista de IP's a las cuales se mandan los TRAPS se pueden configurar en el dispositivo (la configuración no se incluye en el estándar, cada fabricante lo implementa.
ej típico: configurar router vía web)

4.5 Grupo SNMP



Gestión de Redes

Depende de donde lo estés ejecutando, habrán variables a cero
ejemplo: OutGetNext = 0 en un router

• Protocolo SNMPv1

- snmpIntotalSetVars (14)
- snmpInGetRequests (15)
- snmpInGetNexts (16)
- snmpInSetRequests (17)
- snmpInGetResponses (18)
- snmpInTraps (19)
- snmpOutTooBigs (20)
- snmpOutNoSuchNames (21)
- snmpOutBadValues (22)
- snmpOutGenErrs (24)
- snmpOutGetRequests (25)
- snmpOutGetNexts (26)
- snmpOutSetRequests (27)
- snmpOutGetResponses (28)
- snmpOutTraps (29)
- snmpEnableAuthenTraps (30)

la única de lectura/escritura

configura si se envían TRAP bajo la condición authenticationFailure o no.
enabled(1)
disabled(2)

4.6 Limitaciones SNMPv1

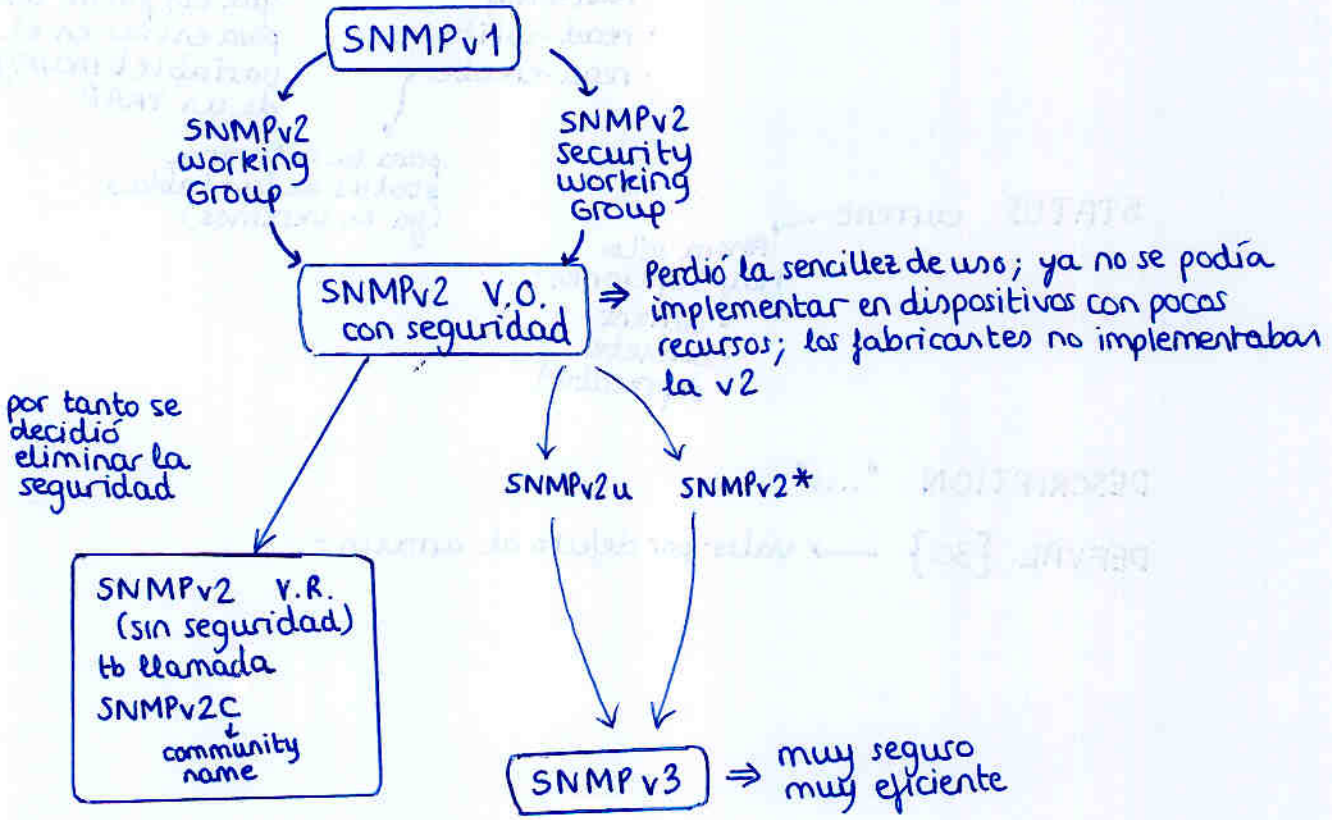
- muy baja eficiencia de acceso a tablas → (SNMPv2)
- Seguridad → (SNMPv3)
- orientado a dispositivos
  - Puedo obtener información de UN dispositivo
  - Me gustaría obtener información global de segmentos de red
- Alarmas: sólo tengo la posibilidad de los 5 trap estándar más los que haya especificado el fabricante

} lo resuelve RMON

5. SNMPv2

5.1 Introducción

No fue fácil llegar a SNMPv2, hubieron varios 'intentos'



5.2 Definición de los objetos

Cambia la forma de definir objetos

SNMPv1 → SMI
SNMPv2 → SMIv2

- **Objetos escalares**

nombre OBJECT-TYPE

SYNTAX INTEGER → Nuevas opciones para tener más o menos bits (Unsigned32, Counter64, Gauge32)

UNITS "seconds" → descriptivo, no obligatorio

MAX-ACCESS read-create → El "MAX" enfatiza que esto está por encima de los permisos
Añade nuevas opciones y quita las innecesarias: dejando:

- not-accesible
- accesible-for-notify → no accesible (ni en lectura) salvo por el propio agente, que las puede usar para enviar en el variablebindings de un TRAP
- read-only
- read-write
- read-create → para la columna status de las tablas (ya lo veremos)

STATUS current → Ahora solo hay 3 opciones:
- current
- obsolete
- deprecated

DESCRIPTION "...."

DEFVAL {30} → valor por defecto al arrancar

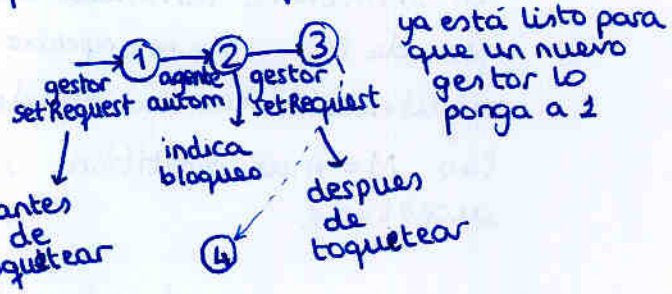
- **Objetos Tabla**

La definición de tablas es EXACTAMENTE igual, pero podemos añadir/borrar/modificar filas.

Para ello se usa la columna nombre Status

SYNTAX RowStatus ⟶ integer numerado con 'estados' para implementar semáforos
MAX-ACCESS read-create

ejemplo de tabla donde interesa añadir y quitar filas:
PingTable
una fila para cada IP a la cual queremos que el dispositivo vaya haciendo pings



5.3 Protocolo

version	community name	SNMP PDU

GetRequest
GetNextRequest
SetRequest
GetResponse
Trap
} EXACTAMENTE igual a SNMPv1 excepto lógicamente el campo versión

Hay dos nuevos mensajes ⟶ { GetBulkRequest, InformRequest }

GetBulkRequest

GetBulkRequest PDU

PDUType	request-id	non-repeaters	max-repetitions	variable bindings

Reduce el nº de mensajes necesarios para leer grandes cantidades de variables: funcionamiento muy similar al GetNextRequest, ya que busca la SIGUIENTE instancia válida, pero ahora podemos pedirle que de las N= non-repeaters primeras variables nos dé la siguiente instancia válida, y del resto de variables nos de las M= max-repetitions siguientes instancias válidas sucesivas.

non-repeaters: indica el nº de variables (las primeras del campo variable bindings) de las que solicitamos sólo la siguiente instancia válida

max-repetitions: para el resto de variables del campo variable bindings, indica el número de sucesivas instancias válidas que queremos

ejemplo:

ipRouteTable

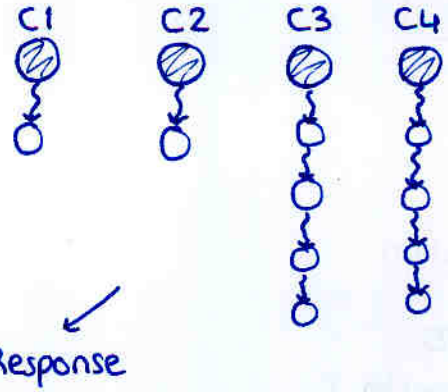
ipRouteDest	ipRouteMetric1	ipRoutNextHop
9.1.2.3	3	99.0.0.3
10.0.0.51	5	158.1.1.32

GetBulkRequest [non-repeaters = 1, max-repetitions = 2]
(sysUpTime, ipRouteDest, ipRouteMetric)

Response ((sysUpTime.0 = "123"),
(ipRouteDest.9.1.2.3 = 9.1.2.3),
(ipRouteMetric.9.1.2.3 = 3),
(ipRouteDest.10.0.0.51 = 10.0.0.51),
(ipRouteMetric.9.1.2.3 = 5))

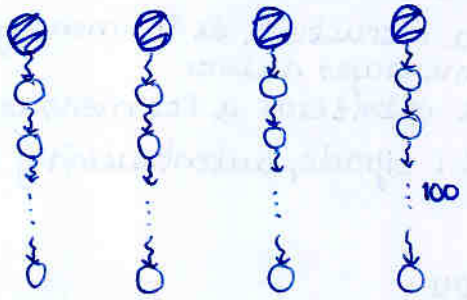
ejemplo 2 :

GetBulkRequest
[non-repeaters = 2,
max-repetitions = 4]
(C1, C2, C3, C4)



ejemplo 3 :

non-repeaters = 0
max-repetitions = 100

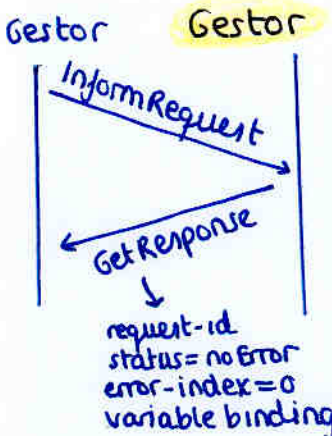


nota : si te acabas la columna le da igual; sigue buscando sucesivamente la siguiente instancia válida

Inform Request

InformRequest PDU

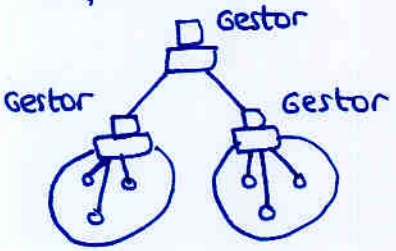
PDUType	request-id	0	0	variable-bindings



Es como un TRAP confirmado entre gestores
↓
i.e. requiere respuesta GetResponse

variable bindings lleva info de gestión

Permite construir una jerarquía de gestores, ya que éstos pueden comunicarse entre ellos con InformRequest



6. SNMPv3

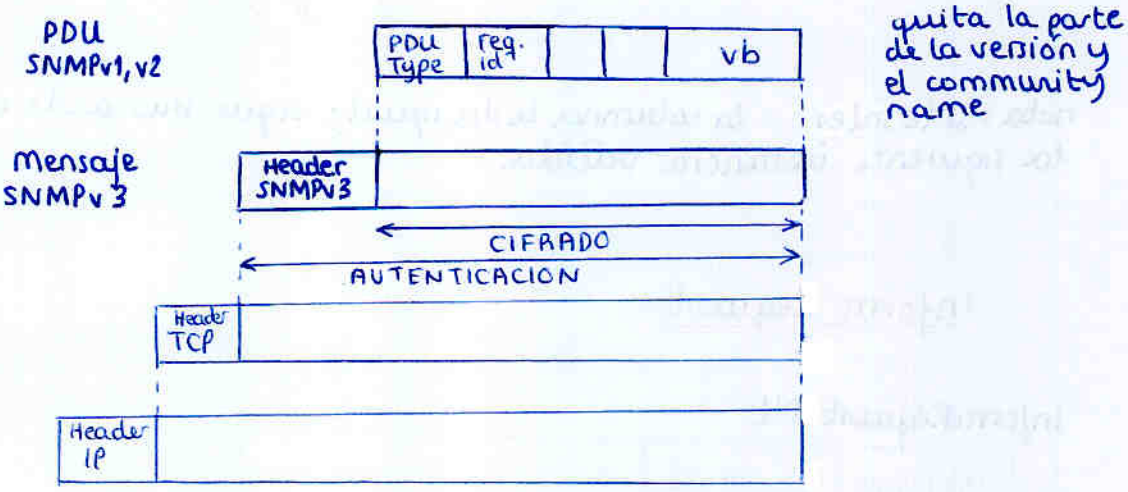
Active SNMP
www.cscare.com

6.1 Introducción

RFC's 2271-2275

¿Qué novedad aporta?

- Arquitectura modular → una guía para implementar gestores y agentes
- Mantiene la estructura de los mensajes SNMPv1, v2 (PDU)
  No añade mensajes nuevos
  Sólo añade cabeceras a los mensajes (campos control)
- Seguridad: cifrado, autenticación y temporización



La nomenclatura es muy estricta:

tener clarísimo lo que es: [PDU SNMPv1, v2
Mensaje SNMPv3

6.2 Arquitectura

Entidad SNMP ≡ agente o gestor

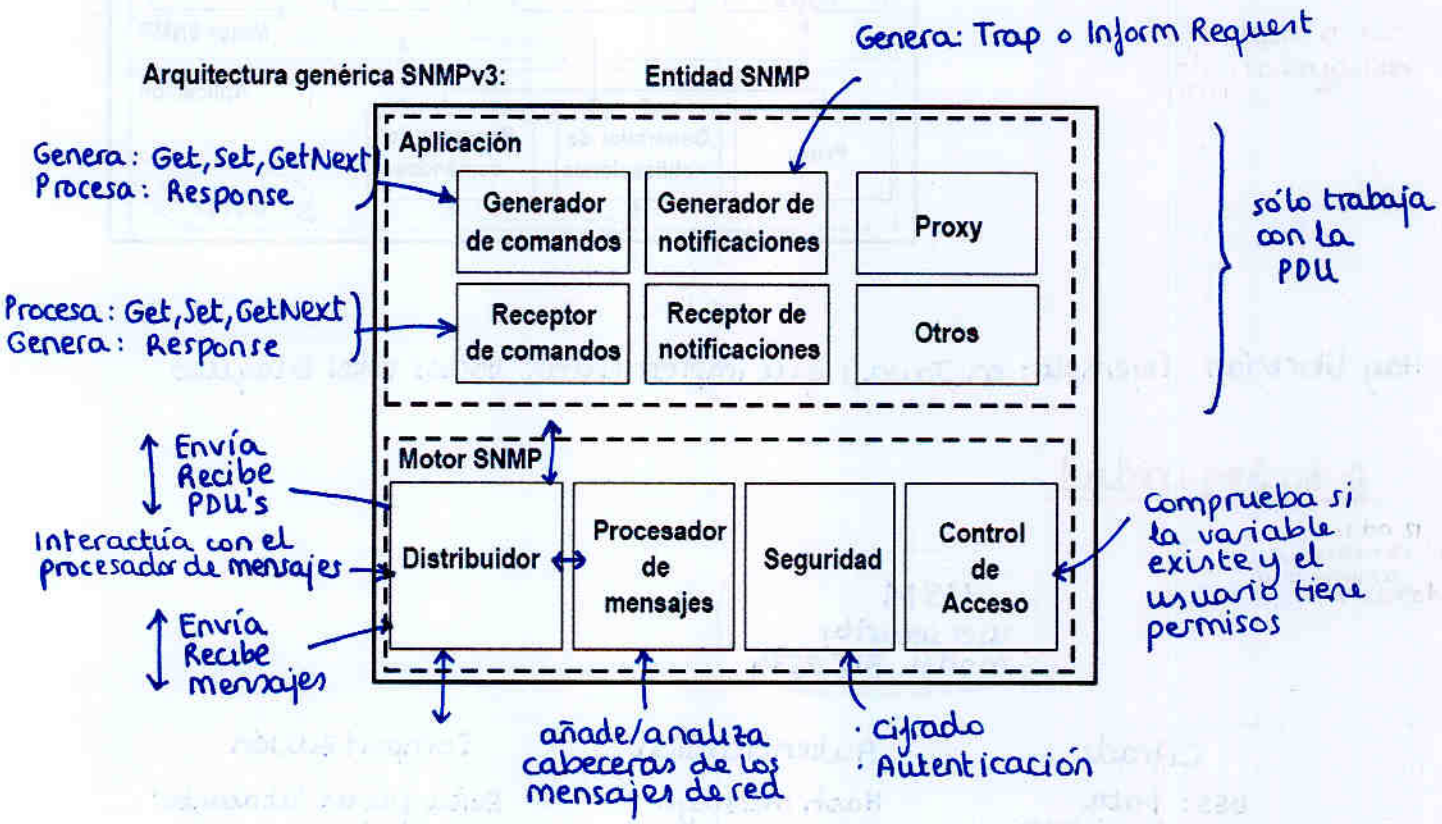
se propone arquitectura modular de la entidad SNMP

(+) optimización en implementación (sólo tener los módulos que necesites)

(+) mejoras de módulos independientes

- Comunicación entre módulos (mensajes = primitivas)

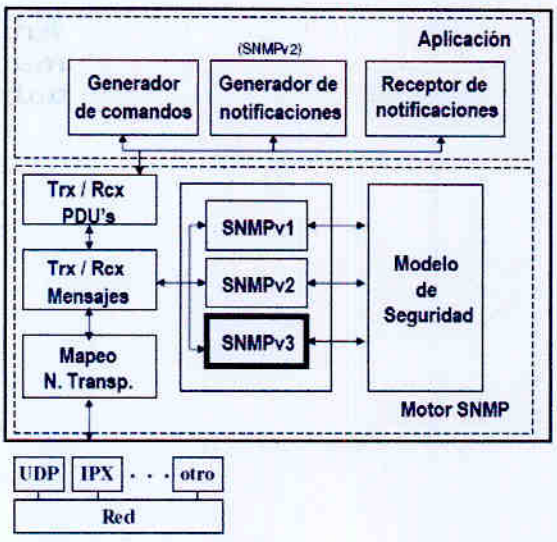
Arquitectura genérica SNMPv3: Entidad SNMP



ejemplo: gestor SNMP

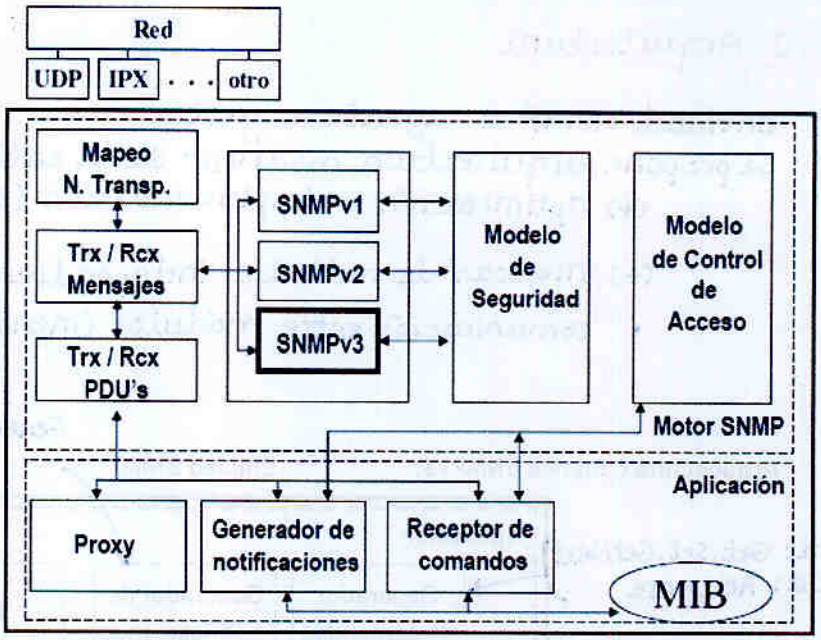
· El generador de notificaciones se usa para el InformRequest entre gestores (no para TRAPS)

Gestor SNMPv3:



ejemplo: agente SNMP

- Al módulo de control de acceso, después de haber descifrado y autenticado, se le pasa un parámetro "UserName" que es un campo que viaja en las cabeceras SNMPv3



Hay librerías (ejemplo: en Java) que implementan todos éstos bloques

6.3 Seguridad

USM
user security model RFC 2274

Cifrado

DES: Data Encription Standard

Autenticación

Hash message authentication code
HMAC-MD5-96
HMAC-SHA-96

Verifica si el mensaje es auténtico

Temporización

Evita que un 'atacante' a pesar de no saber descifrar el mensaje, se lo guarde y lo pueda usar cuando quiera

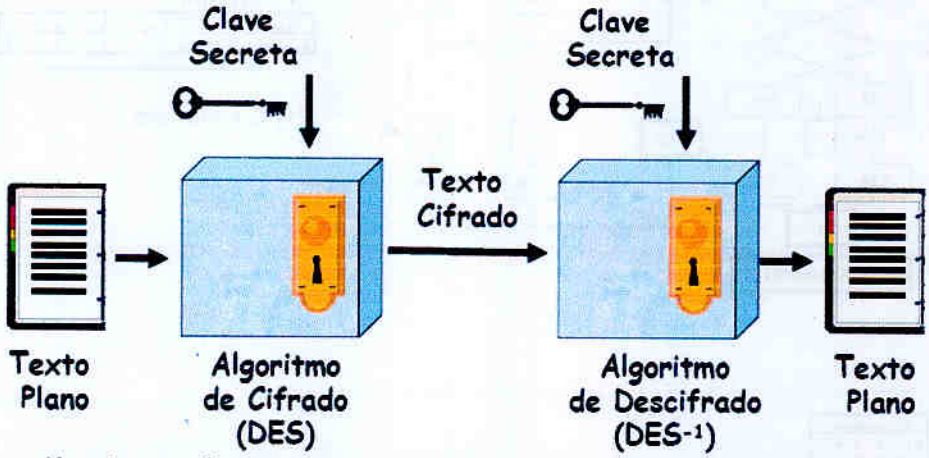
Se soluciona con un protocolo de sincronización (time stamps)

Podemos configurar un tiempo máximo

DES (Data Encryption Standard)

- Desarrollado por IBM (estandar del gobierno de EEUU en 1976)
- Cifrado simétrico y de una sola clave
  ↳ hizo algunos cambios (menos bits)(64 a 56) (cambio en cajas S)
- Hoy en dia ya se sabe descifrar (se sospecha que el gobierno siempre ha sabido, ya que los cambios que introdujo protegen contra un tipo de ataque conocido hace poco con el cual se ha logrado descifrar), pero cuesta mucho.

Simétrico = mismo algoritmo para encriptar y para desencriptar



- **Algoritmo robusto**: incluso conociendo el texto plano y el texto cifrado resultante es difícil obtener la clave o descifrar otros textos.
- **Clave secreta**: tanto el emisor como el receptor deben obtener copias de la clave de forma segura y mantenerla de forma segura.

Tipos de ataque:

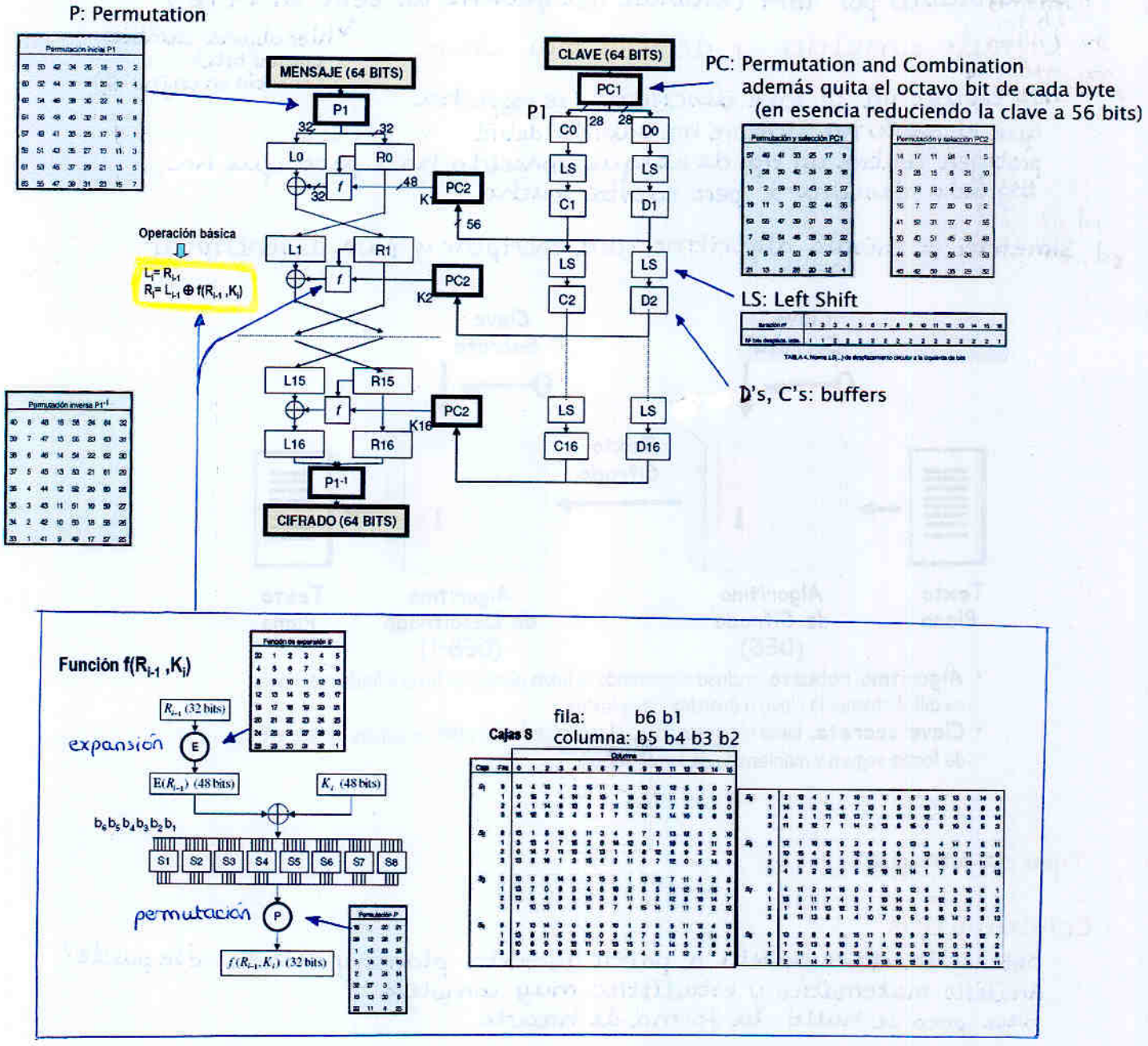
- Criptoanálisis:
  Obtener la clave secreta a partir del texto plano y cifrado: ¿se puede?
  Análisis matemático y estadístico muy complejo
  Hace poco se halló la forma de hacerlo

- Fuerza bruta:
  Teniendo el texto plano y cifrado, ir probando todas las posibles claves

Tamaño Clave	Tiempo PC (1 encript/µs)	Tiempo Sist. Paralelo (10^6 encript/µs)
32	35,8 min	2,15 ms
64	1142 años	10,01 horas
128	$5,4 \times 10^{24}$ años	$5,4 \times 10^{18}$ años

DES = 56 bits

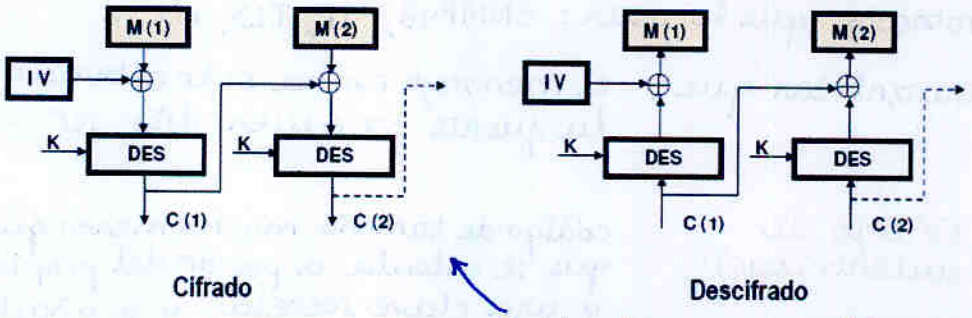
Cifrado DES



Para descifrar se hace EXACTAMENTE igual pero de abajo arriba
i.e. necesito calcular todas las subclaves K1, K2, ... K16 antes de empezar nada.

Si el mensaje son más de 64 bits (parece lógico)

Cipher Block Chaining (CBC)

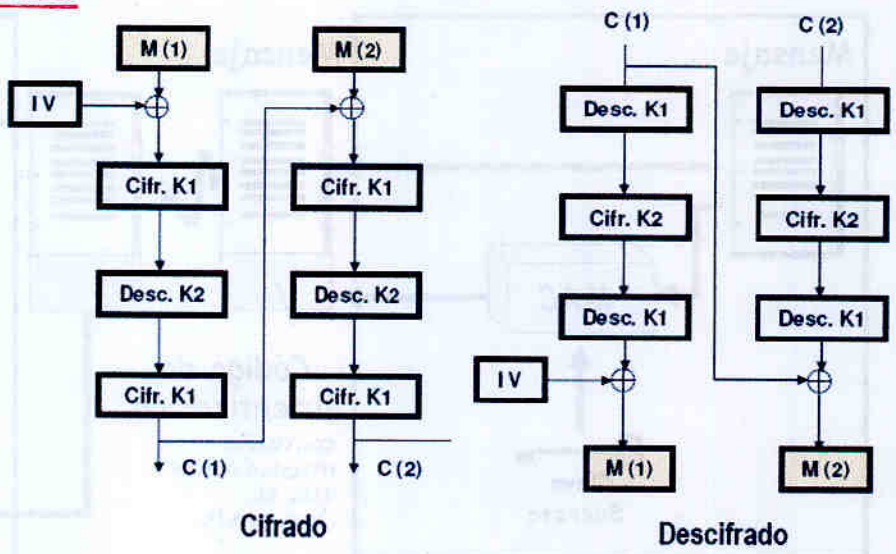


Dividir el mensaje en bloques, y en lugar de cifrar cada uno por separado, primero se le suma el bloque anterior cifrado.

Si se desea aún mayor seguridad:

Triple DES:

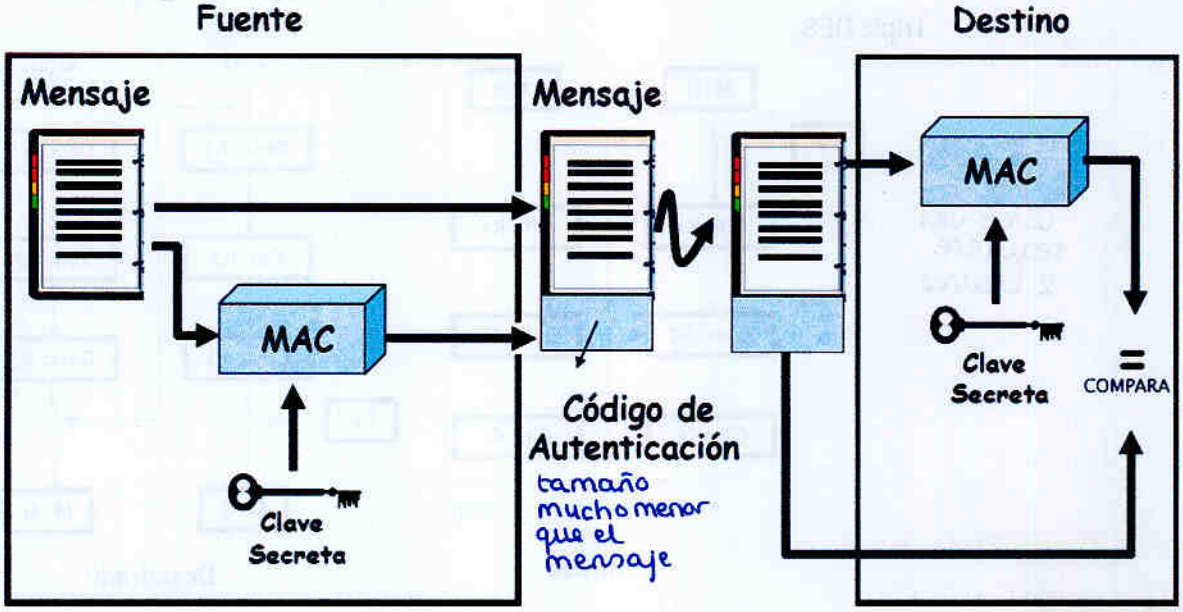
Como ves requiere 2 claves



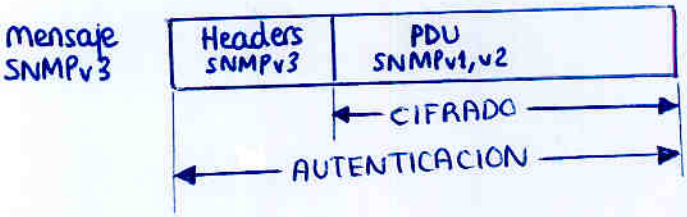
Autenticación : HMAC (Hash Message Authentication Code)

- Protocolos que lo usan: SNMPv3, IP, TLS, etc ...
- Garantizar que: el mensaje no ha sido alterado
  la fuente es quien dice ser

Código de autenticación: código de tamaño mucho menor que el mensaje que se calcula a partir del propio mensaje y una clave secreta, y se añade en un campo del mensaje



Nota: la autenticación se calcula sobre TODO el mensaje SNMPv3
Para ello se considera siempre (tanto en tx como rx) que el campo "código de autenticación" son todo ceros, antes de pasar el mensaje al bloque MAC



Tema 2

7 • Monitorización de Red Remota (RMON)

Asignatura: Gestión de Redes

1. Introducción

RMON ⇔ Extensión de SNMP

– *RMON (Remote Network Management)*
(Monitorización de Red Remota)

– Conceptos válidos de SNMP: SMI, MIB-2 y Protocolo SNMP

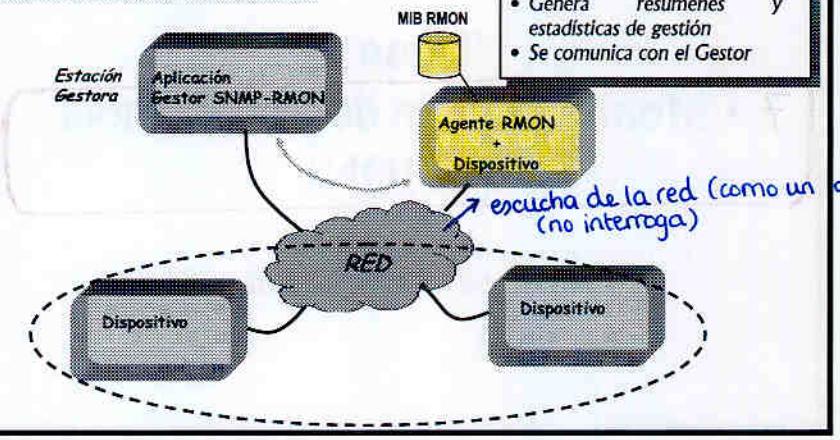
RFC's:

RFC 1757: **"Remote Network Monitoring Management Information Base"**

RFC 2021: **"Remote Network Monitoring Management Information Base Version 2 using SMIv2"**

Tema 2. RMON

RMON y Agente Monitor:



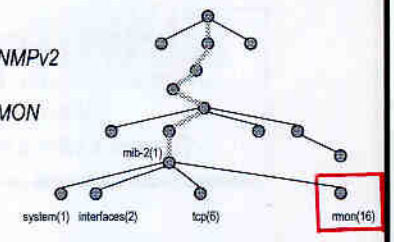
escucha de la red (como un analiz. de protocolos) (no interroga)

Tema 2. RMON

Extensiones

Limitaciones SNMPv1:

- **Eficiencia** en el acceso a la información: *SNMPv2*
- **Seguridad**: *SNMPv3*
- **Monitorización de redes de área local**: *RMON*



nuevo nodo en la MIB. Todos sus objetos son tablas

RMON incorpora dos diferencias:

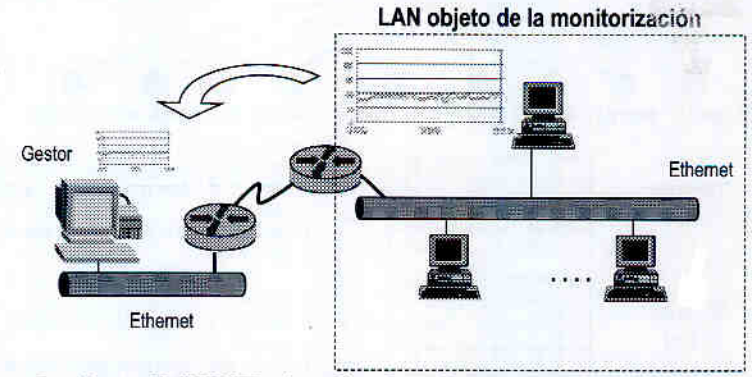
1. **Extensión de MIB-2**: se define una extensión de la MIB-2 denominada MIB RMON
2. **Agente (sonda, monitor) RMON**: es capaz de leer/escribir sobre la MIB RMON e implementa funciones típicas de un monitor.

Diferencias entre agente SNMP y agente RMON (bastante más caro)

RMON

Tema 2. RMON

Ejemplo: Obtener información de una LAN?

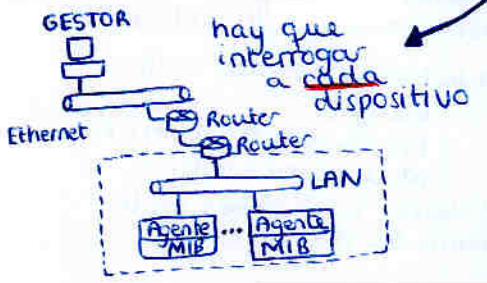


Opciones? SNMP, Analizador de protocolos, RMON

yo quiero tener la información en mi gestor

Agente RMON (router, switch, PC, ...)
Mismas ventajas que el analizador de protocolos (realmente el agente RMON ES un analiz. de protocolos)
(+) Monitorización remota
(+) Comunicación Gestor ↔ Agente RMON usando SNMP

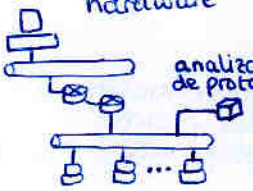
SNMP



hay que interrogar a cada dispositivo

sniffer o hardware

Analizador de Protocolos



(+) Capturas todo el tráfico
(+) Filtro de captura
(+) Almacenar info
(+) Estadística

(-) Requiere desplazarse (monitoriz no remota)
(El analiz es muy caro)
(-) No existe comunicación estándar entre gestor y analiz.

Tema 2. RMON

2. Gestión de tablas RMON

Gestión de tablas RMON

- Configuración de tablas
- MIB RMON

  Tabla de control y Tabla de datos

Tema 2. RMON

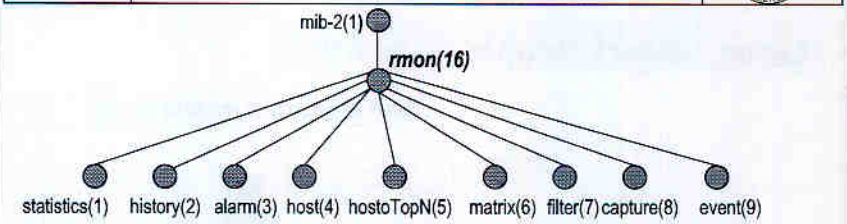


Tabla de Control

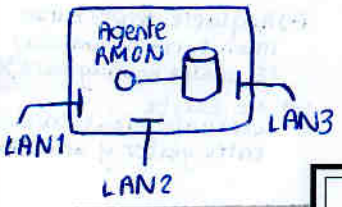
- Lectura/Escritura
- Columnas: origen de los datos, frecuencia de captura, nº max de muestras, variable, umbrales, etc.
- Gestor: configura el agente RMON añadiendo filas a la tabla

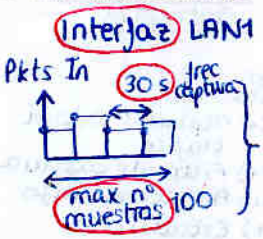
Tabla de Datos

- Lectura
- Columnas: parámetros capturados (octetos, paquetes, errores, etc.)
- Agente: crea nuevas filas según la configuración de la Tabla de Control
- Gestor: lee los datos almacenados

Tengo que configurar cada grupo accediendo a la tabla de control

ejemplo: quiero que me almacene los paquetes entrantes por interfaz LAN1





estoy almacenando los últimos 50 minutos

En la tabla de control escribo:
- Interfaz
- Frecuencia de captura
- Max nº muestras

Los datos se escriben en la tabla de datos

Tema 2. RMON

índice	parámetros de control	_owner	_status
1		monitor	valid
2		admin	valid
3		admin	valid

Tabla de Control

éstas dos últimas columnas SIEMPRE están

índice	índice2	datos
1	1	
1	2	
1	3	
2	1	
2	2	
2	3	
3	1	
3	2	

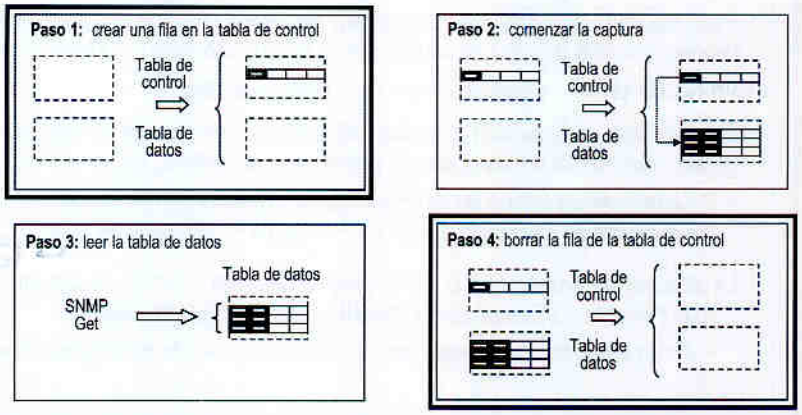
Tabla de Datos

identificador datos.2.3

el índice2 se va incrementando (incluso cuando sobreescribimos por haber llegado al max nº muestras)

Tema 2. RMON

- Ciclo de vida de las tablas de control y de datos



automaticamente se borran las filas de la tabla de datos

Gestión de Redes

Tema 2. RMON

⇨ Múltiples gestores

- Los agentes RMON (igual que los agentes SNMP) pueden ser gestionados desde diferentes Gestores
- Los gestores RMON (a diferencia de SNMP) pueden añadir y modificar filas en las Tablas de Control
  - ☒ **Identificar quién ha creado cada fila de la Tabla de Control** → para que un administrador no le borre sin querer configuracion a otro. ↓ solución: columna owner
  - ☒ **Problemas de concurrencia** (usar semaforos)

Gestión de Redes

Tema 2. RMON

☒ **Identificar quién ha creado cada fila de la Tabla de Control**

- Asociada con cada Tabla de Control hay un objeto columna que identifica al dueño de una fila concreta de la tabla y de la función asociada **(__Owner)**

Columna de tipo: **OwnerString ::= DisplayString**

- El propietario se especifica indicando: dirección IP, nombre de la estación de gestión, nombre del administrador de red, número de teléfono, etc.
  - ✓ *Cuando se adquiere un agente RMON, en general, ya estará configurado con un conjunto de funciones* ⇨ (etiqueta: **monitor**)
- La etiqueta de propiedad NO sirve como contraseña o como mecanismo de control de acceso **(Comunidad + Perfil = Política de Acceso).**
  - ✓ *Es un sistema para "respetar" la configuración de otros gestores*

Tema 2. RMON

☒ **Problemas de concurrencia**

- Asociada con cada Tabla de Control hay un objeto columna que identifica el estado en que se encuentra la fila **(__Status)**

```
EntryStatus ::= INTEGER { valid (1)
                          createRequest (2)
                          underCreation (3)
                          invalid (4)     }
```

- Para evitar que dos o más peticiones desde estaciones gestoras creen filas iguales se crea una máquina de estados.

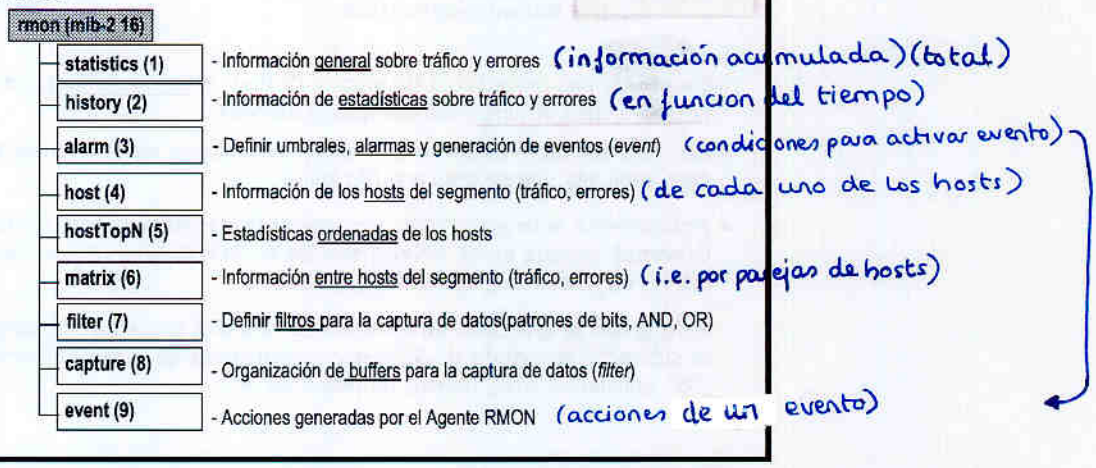
createRequest (2) -> underCreation (3) -> valid (1)

gestor escribe un 2 cuando quiera configurar fila

gestor escribe un 1 al acabar

Tema 2. RMON

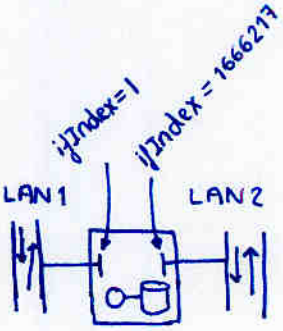
3. MIB RMON



Tema 2. RMON

- Cada grupo almacena datos y estadísticas derivadas de los datos capturados por el monitor.
- Un monitor puede tener más de un interfaz físico.
- Los datos almacenados en cada grupo representan datos recogidos de una o más de las subredes a las que está conectado el monitor.
- Todos los grupos son opcionales pero existen dependencias:
  - alarm ⇨ event
  - hostTopN ⇨ host
  - capture ⇨ filter
- Se pueden agrupar:
  - Estadísitcas: ⇨ statistics, history, host, hostTopN, matrix, tokenRingcapture
  - Alarmas ⇨ alarm, filter, capture, event

STATISTICS



Tema 2. RMON

⇨ statistics

Objetivo: Contiene las estadísticas básicas para cada subred monitorizada.

- Consta de una única tabla (Control + Datos), etherStatsTable, con una fila para cada interfaz monitorizada (subred).
- Las estadísticas vienen en forma de contadores que se inicializan a cero cada vez que se crea una fila válida.
- Actualmente este grupo sólo contiene objetos definidos para interfaz Ethernet, aunque otras extensiones de la MIB aceptan otros tipos de LAN (Fast Ethernet, Token-Ring, FDDI, ...).
- Este grupo proporciona información de utilidad acerca de la carga de la subred y el estado de la misma: condiciones de error (errores de CRC, colisiones, etc), tamaño de paquetes, etc.

un parámetro de configuración es el nº de interfaz ifIndex

Gestión de Redes

Tema 2. RMON

statistics (rmon 1)

es el índice de la tabla

- etherStatsTable (1)
  - etherStatsEntry (1)
    - etherStatsIndex (1) (**)
    - etherStatsDataSource (2)
    - etherStatsDropEvents (3)
    - etherStatsOctets (4)
    - etherStatsPkts (5)
    - etherStatsBroadcastPkts (6)
    - etherStatsMulticastPkts (7)
    - etherStatsCRCAlignErrors (8)
    - etherStatsUndersizePkts (9)
    - etherStatsOversizePkts (10)
    - etherStatsFragments (11)
    - etherStatsJabbers (12)
    - etherStatsCollisions (13)
    - etherStatsPkts64Octets (14)
    - etherStatsPkts65to127Octets (15)
    - etherStatsPkts128to255Octets (16)
    - etherStatsPkts256to511Octets (17)
    - etherStatsPkts512to1023Octets (18)
    - etherStatsPkts1024to1518Octets (19)
    - etherStatsOwner (20)
    - etherStatsStatus (21) (*)

hay que escribir el ifIndex de la interfaz

<64
>1518
<64 & CRC erróneo
>1518 & CRC erróneo

Gestión de Redes

La tabla de control y de datos están juntas en una única tabla.

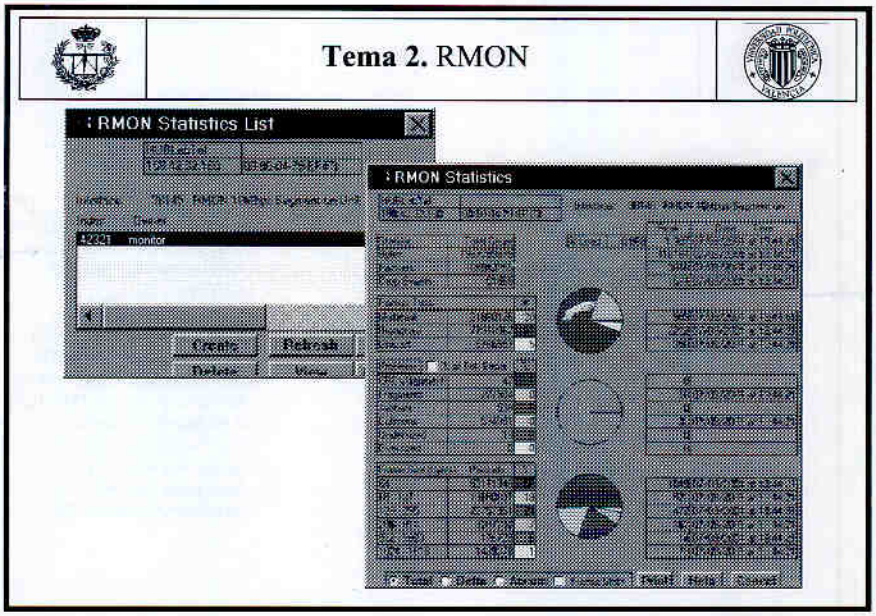
Las columnas de datos son contadores

Las columnas de control son las 2 primeras y las 2 últimas

_Index	DataSource	... datos ...	Owner	Status
1	1		admin	1
2	1666217		admin	1

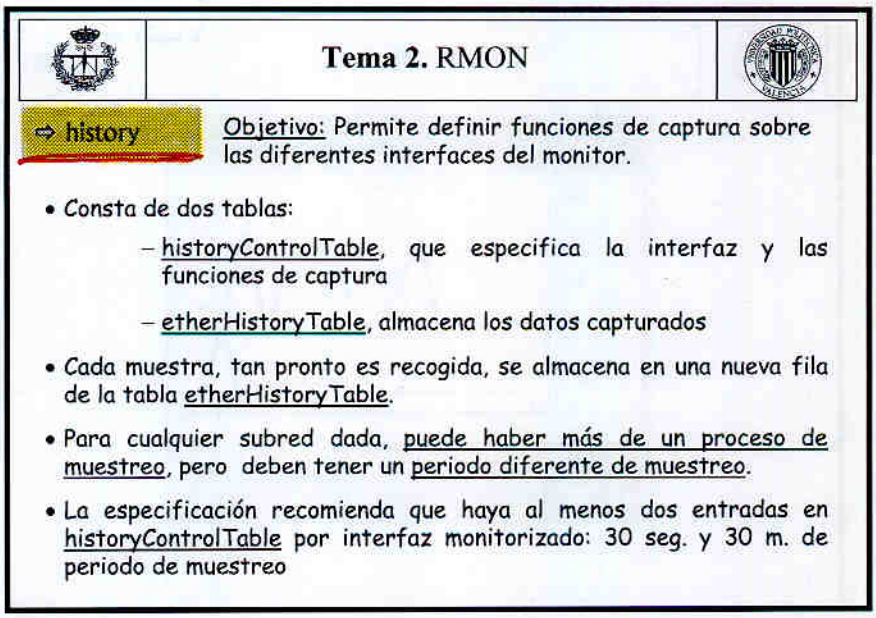
la tabla de datos no añade nuevas filas, sólo nuevas columnas.

una fila para cada interfaz monitorizada



Gestión de Redes

HISTORY

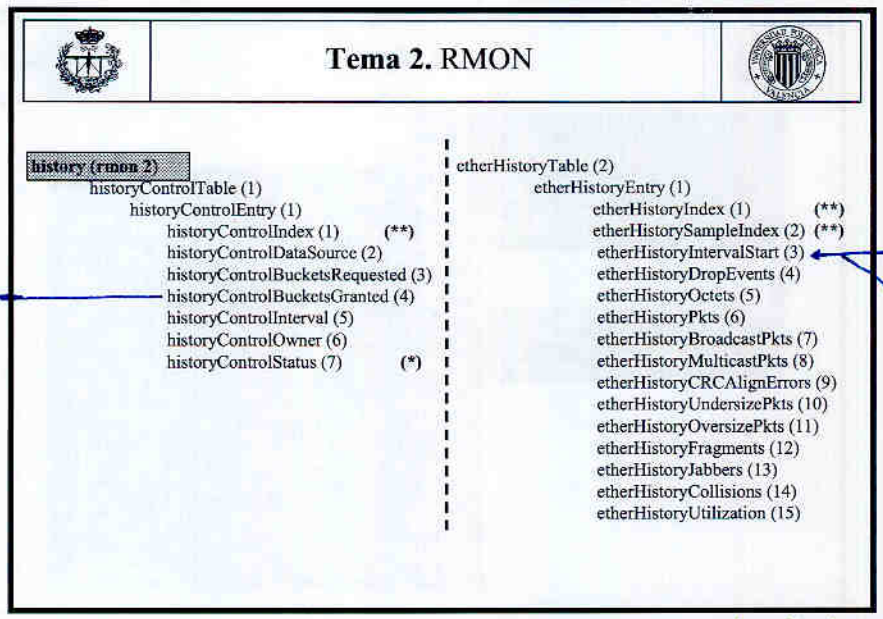


Gestión de Redes

Parámetros de control

- Interfaz a escuchar
- Intervalo de monitorización (tiempo durante el cual cuento)
- Maximo numero de muestras (buckets requested)

ej: 120 muestras de cada 30s → tenemos las muestras de la ultima hora
96 muestras de cada 30 min → tenemos las muestras de los últimos 2 dias



sólo lectura: indica las muestras q. realmente ha podido almacenar (por cuestiones de espacio)

sysUpTime de cuando empezó la captura de esa muestra

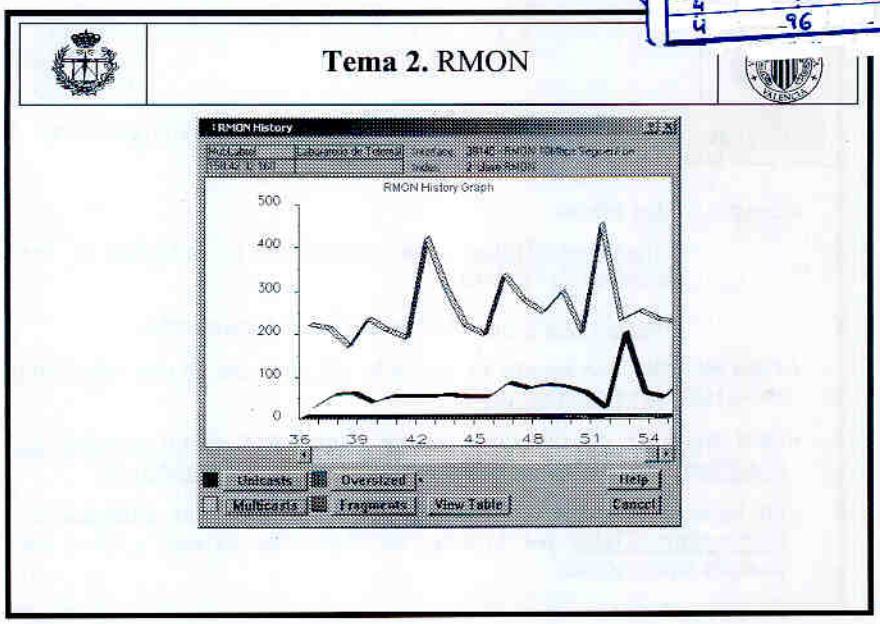
ejemplo:

history Control Table

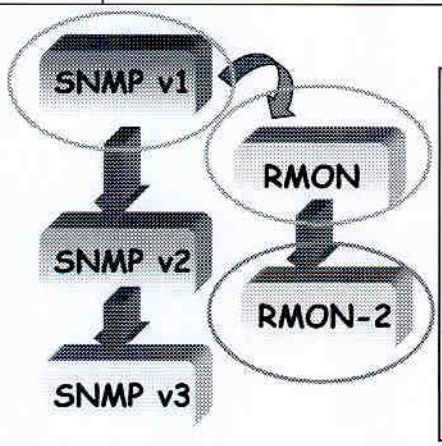
Index	DataSource	BktsReq	BktsGrntd	Interval	Owner	Status
1	1	120	120	30s	admin	1
2	1	96	96	30min	admin	1
3	1666217	120	120	30s	admin	1
4	1666219	96	96	30min	admin	1

etherHistoryTable

Index	SampleIndex	to	.....
1	1		
1	⋮		120
1	120		
2	1		
2	⋮		96
2	96		
3	1		
3	⋮		120
3	120		
4	1		
4	⋮		96
4	96		



Tema 2. RMON



RMON-2:

- *Objetivo: subir en la pila del protocolos y proporcionar estadísticas de tráfico a nivel de red y de aplicación.*

Ventajas:

- *RMON2 no es un substituto de RMON1*
- *Ambas MIBs son necesarias (monitorizar LAN y Aplicaciones)*

más grupos en los cuales se almacena información en función de la dirección IP o del protocolo del nivel de aplicación