SNMP
Simple Network Management Protocol

Taller de Gestión de Redes

Facultad de Ingeniería – Tecnólogo en Telecomunicaciones

Un poco de historia...

Herramientas específicas para problemas específicos:

- Basadas en ICMP:
  - ping : verificación de conectividad en capa 3
  - traceroute : trazado de ruta en capa 3
  - netstat/nslookup : chequeos varios de red en hosts
- Una herramienta para cada problema
  - Difícil integración
  - No hay apoyo al diagnóstico de problemas
  - A pesar de todo... ¡Siguen siendo muy útiles y muy usadas!

Un poco de historia…

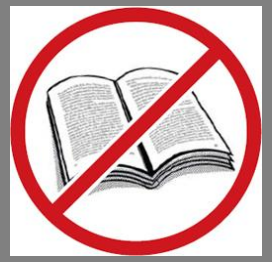
- Muy pocas herramientas para gestión de rendimiento.
  - Medición de RTT con el PING.
  - Sin herramientas para medición de tráfico.
- Consecuencias:
  - Surgimiento de soluciones propietarias, muchas veces incompletas y casi siempre incompatibles entre si.

Un poco de historia…

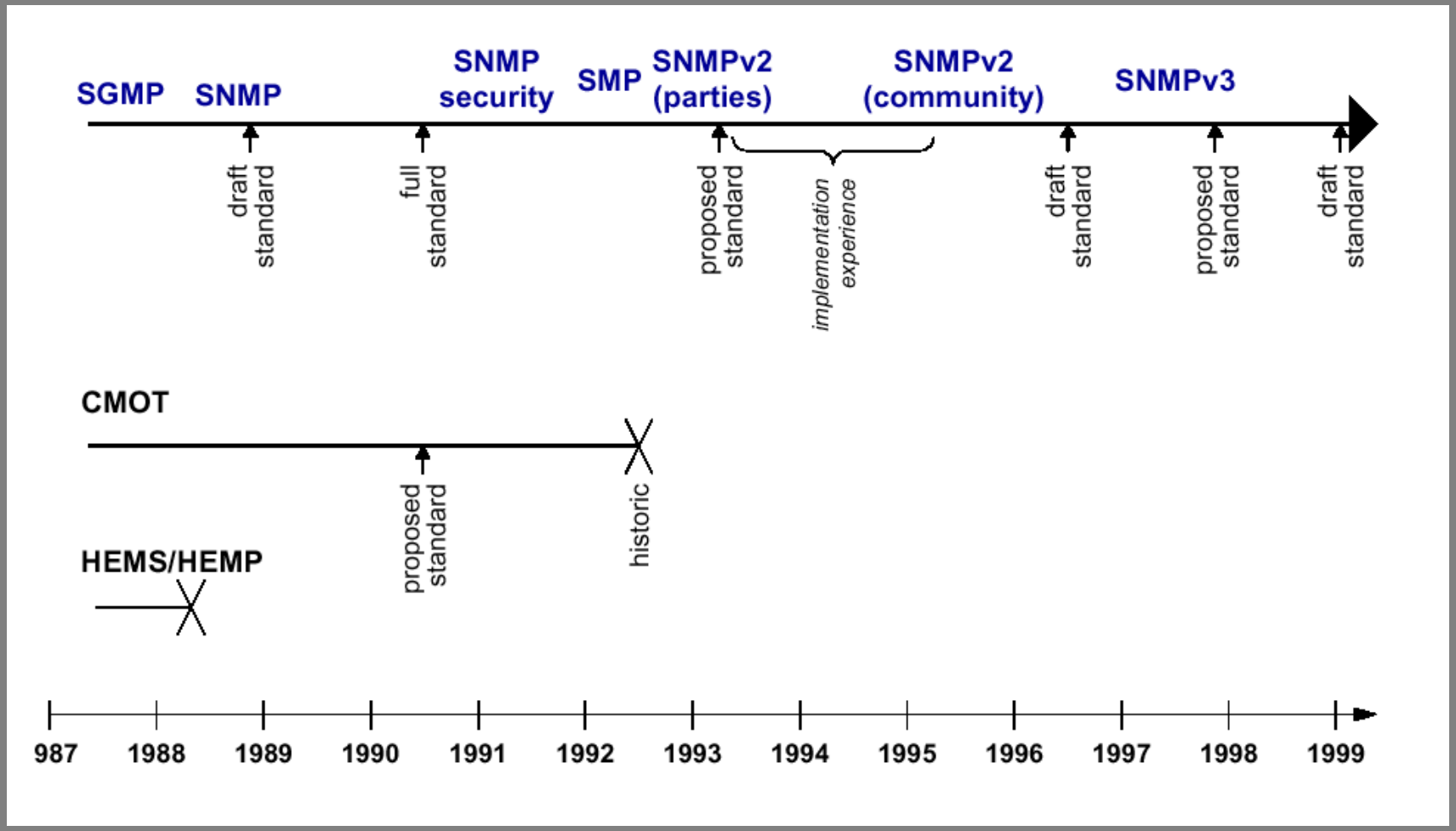
- Surgimiento de una necesidad:
  - Estandarizar:
    - Un protocolo de transporte para la información de gestión.
      - *SNMP*
    - Una sintaxis o un idioma común para expresar la información de gestión.
      - *ASN.1*
    - Un modelo de información de gestión
      - *SMI*
      - *MIBs*

SNMP - Introducción

- Versión 1: aprox. 1987-1989
  - Integración de la gestión
  - Introducción del concepto agente/gestor
  - Protocolo “puente” hasta la llegada de la gestión OSI
    - Razón de la mayoría de las debilidades
    - Hecho nunca concretado, SNMP subsiste hasta la actualidad como sinónimo de gestión de redes.



SNMP - Introducción



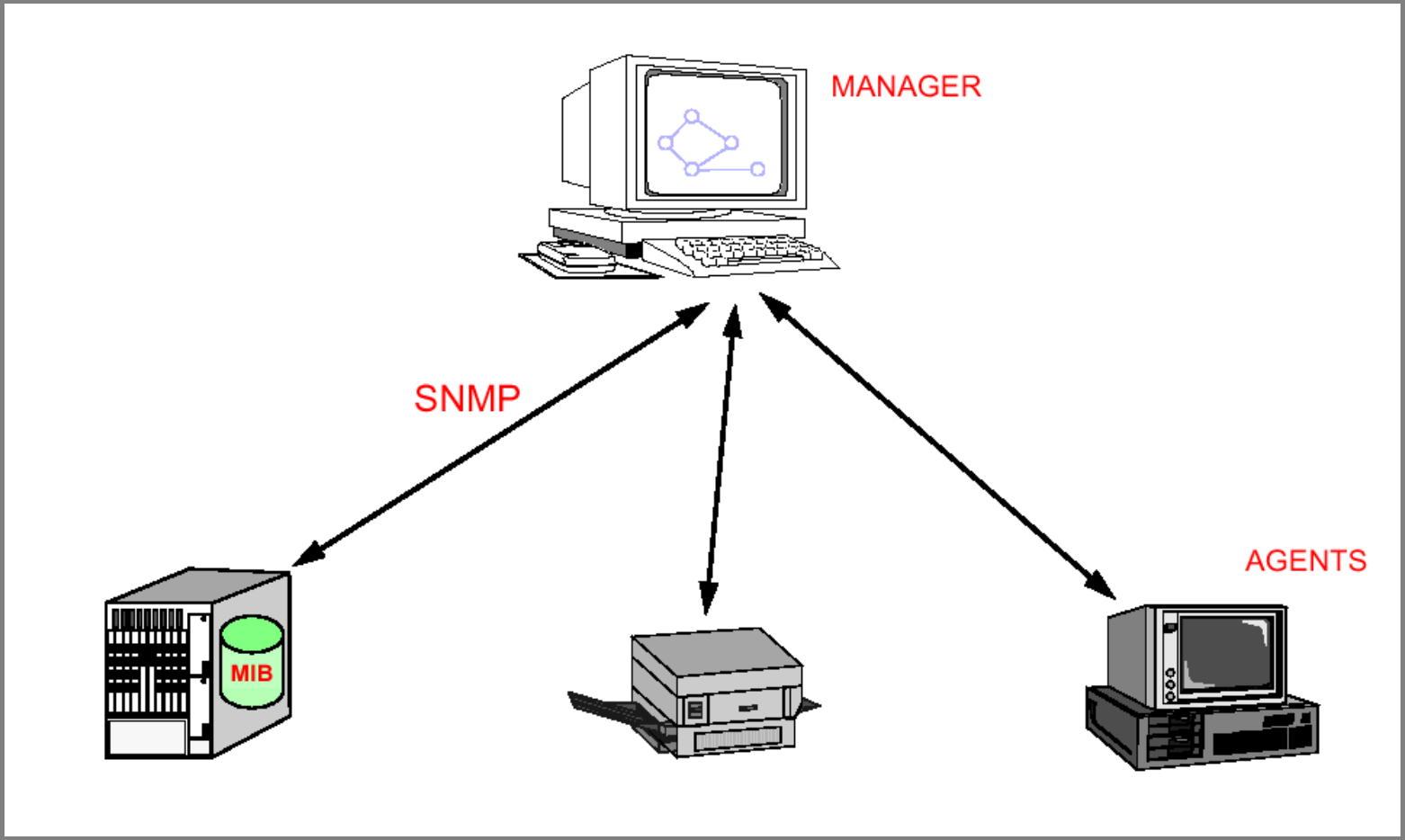
SNMP - Introducción

- Objetivos del diseño de SNMP :
  - Universalidad: implementable de los equipos más pequeños a los más grandes
  - El agregado de SNMP a sistemas debería ser sencillo y accesible:
    - Implementable en pocas líneas de código
    - Funcionalidades limitadas
  - El agregado de extensiones debería ser posible (nuevas MIBs)
  - Gestión debería ser lo más robusta posible

SNMP - Introducción

- El modelo propuesto por IETF para la gestión introduce:
  - Estación de Gestión (*gestor*)
  - Agente de Gestión (*agente*)
  - Base de Información de Gestión (MIB)
  - Protocolo de Gestión de Red

SNMP - Introducción



SNMP - Introducción

- Estación de Gestión :
  - Una interfaz a través de la cual el administrador de la red pueda monitorizar y cambiar el estado de la red
  - Aplicaciones para recuperación de fallas, configuración de sistemas, análisis de la información obtenida de los agentes.
  - Trasladar los requerimientos del administrador en requerimientos a los agentes y elementos de monitorización.
  - Mantener una base de datos con la información obtenida de todos los agentes presentes en la red.

SNMP - Introducción

- MIB : *Management Information Base*
  - Todo *recurso* de red pasible de ser gestionado debe ser representado a través de un *objeto*
  - Un objeto es una variable que contiene la información del recurso (diferentes tipos según el recurso, etc)
  - El conjunto de todas las variables conocidas por un agente es la MIB de este agente.
  - El Gestor deberá implementar su funcionalidad accediendo a los objetos presentes en la MIB de cada agente a gestionar.
    - Lo que se puede hacer con cada agente depende de la MIB implementada por este.

SNMP – Introducción

- Protocolo de Gestión de Red :
  - La recomendación del IETF para esta funcionalidad es el protocolo *SNMP*.
  - Se propone que cualquier protocolo de GR debe implementar como mínimo las siguientes funcionalidades básicas:
    - **`Get`**: permite que el gestor acceda a los valores de las variables de la MIB.
    - **`Set`**: permite que el gestor configure valores en algunas de las variables de la MIB.
    - **`Trap`**: permite que el agente notifique al gestor de eventos significativos (presencia de fallas, condiciones anormales, etc).

SNMP - Introducción

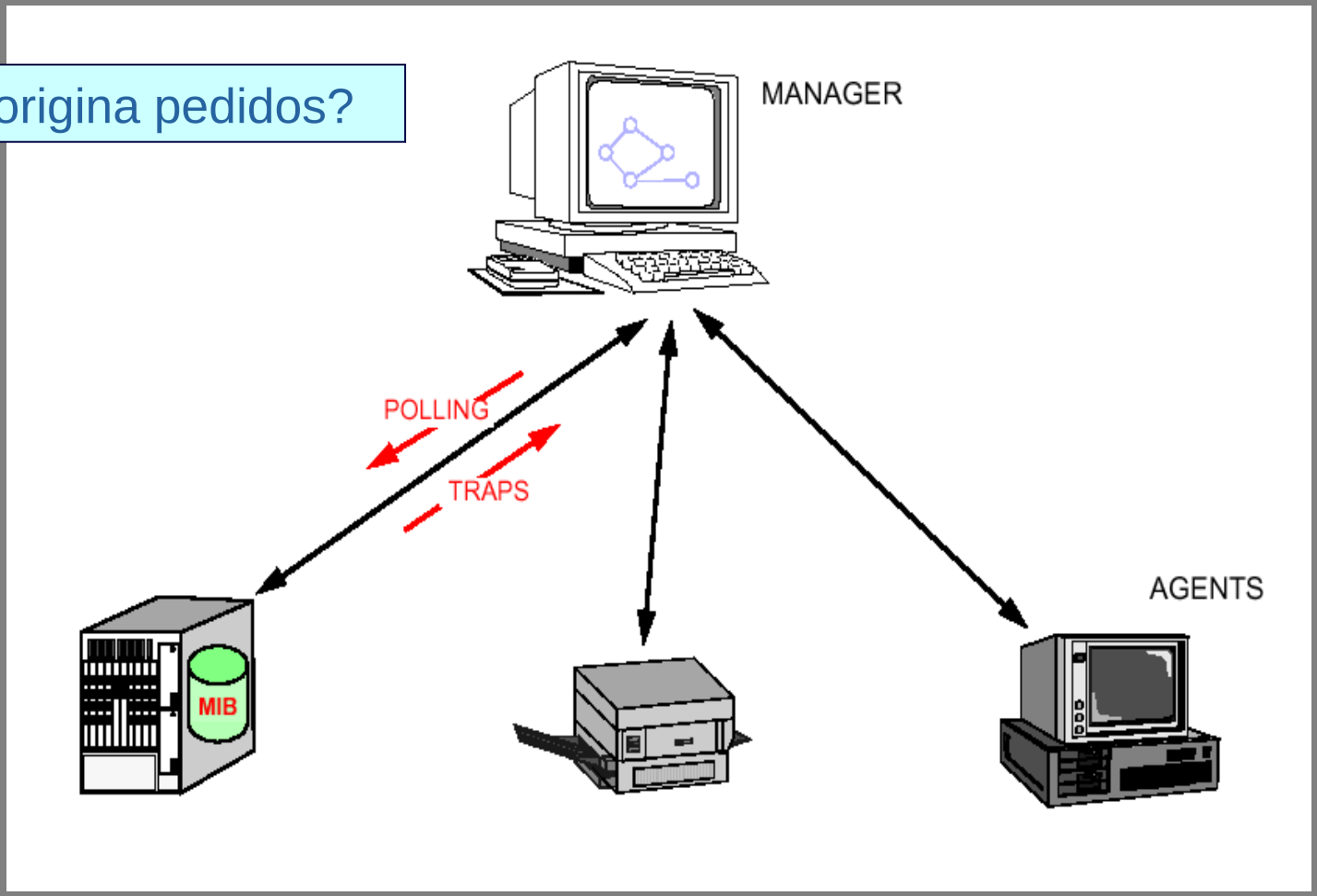
- Poll:
  - Es un pedido de información generado por la estación de gestión dirigido a un agente SNMP.
  - Generalmente realizado de manera periódica por la estación.
  - Utilizado para colección de información de variables como tráfico, uso de memoria, consumo de CPU, etc.

SNMP - Introducción

- Trap:
  - Es un paquete de información generado por un agente SNMP dirigido a la estación de gestión.
  - Generado en respuesta a situaciones excepcionales (similar a una interrupción)
  - Utilizado por los agentes para informar de situaciones como cambios de estado de enlaces, reboots, etc.

SNMP - Introducción

¿Quién origina pedidos?



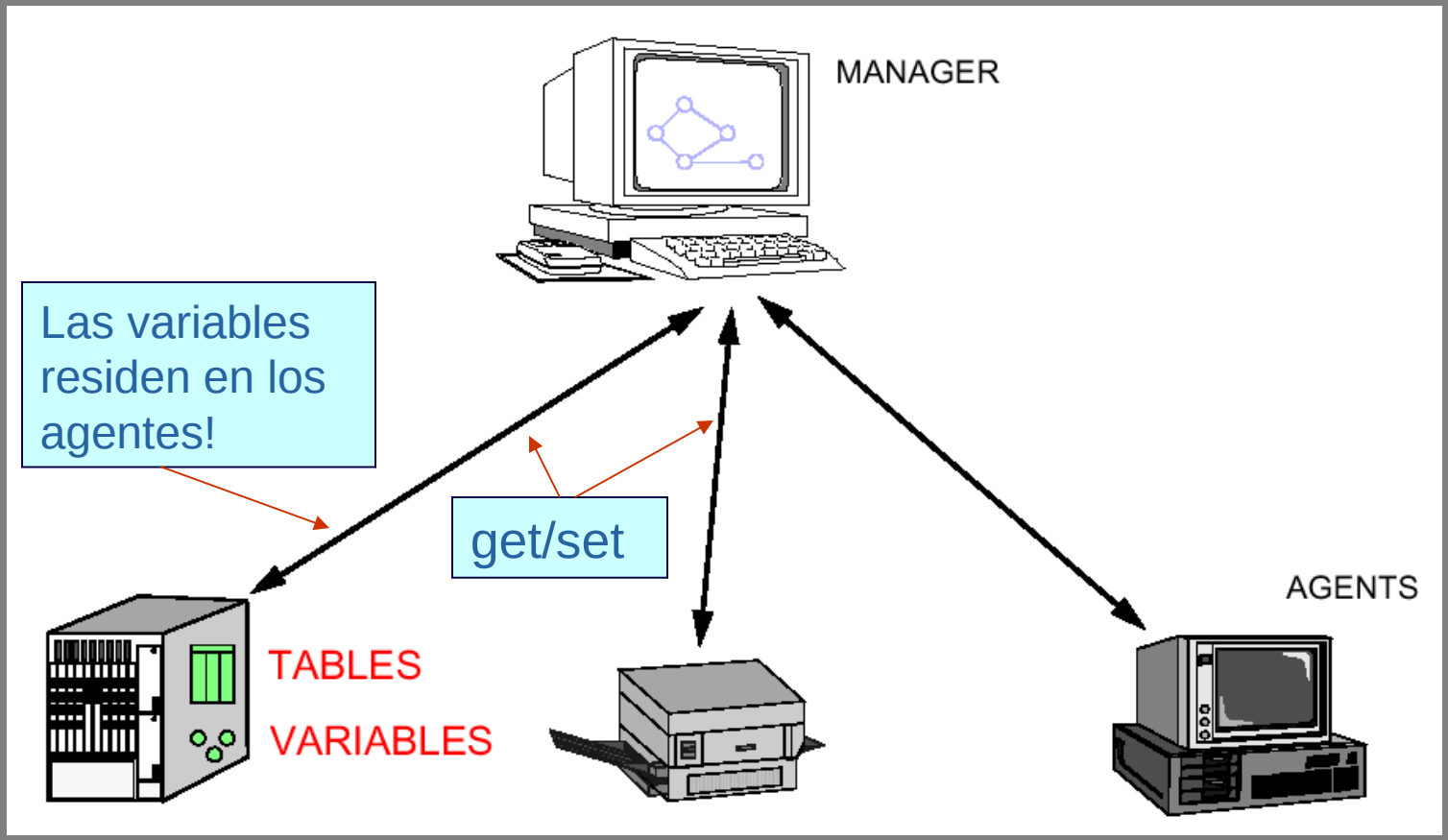
SNMP

- Elección de protocolo de transporte:
  - Directamente sobre IP (como ICMP)
    - Se puede lograr el menor overhead posible
    - A cambio de muy poca flexibilidad y de reinventar la rueda!
  - Sobre TCP :
    - Mucha mayor complejidad
      - Manejo de conexiones
      - Números de secuencia, control de flujo, etc.
  - Sobre UDP :
    - Mejor compromiso flexibilidad/eficiencia
  - La elección es UDP!

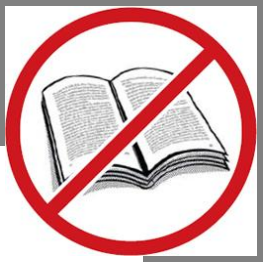
SNMP

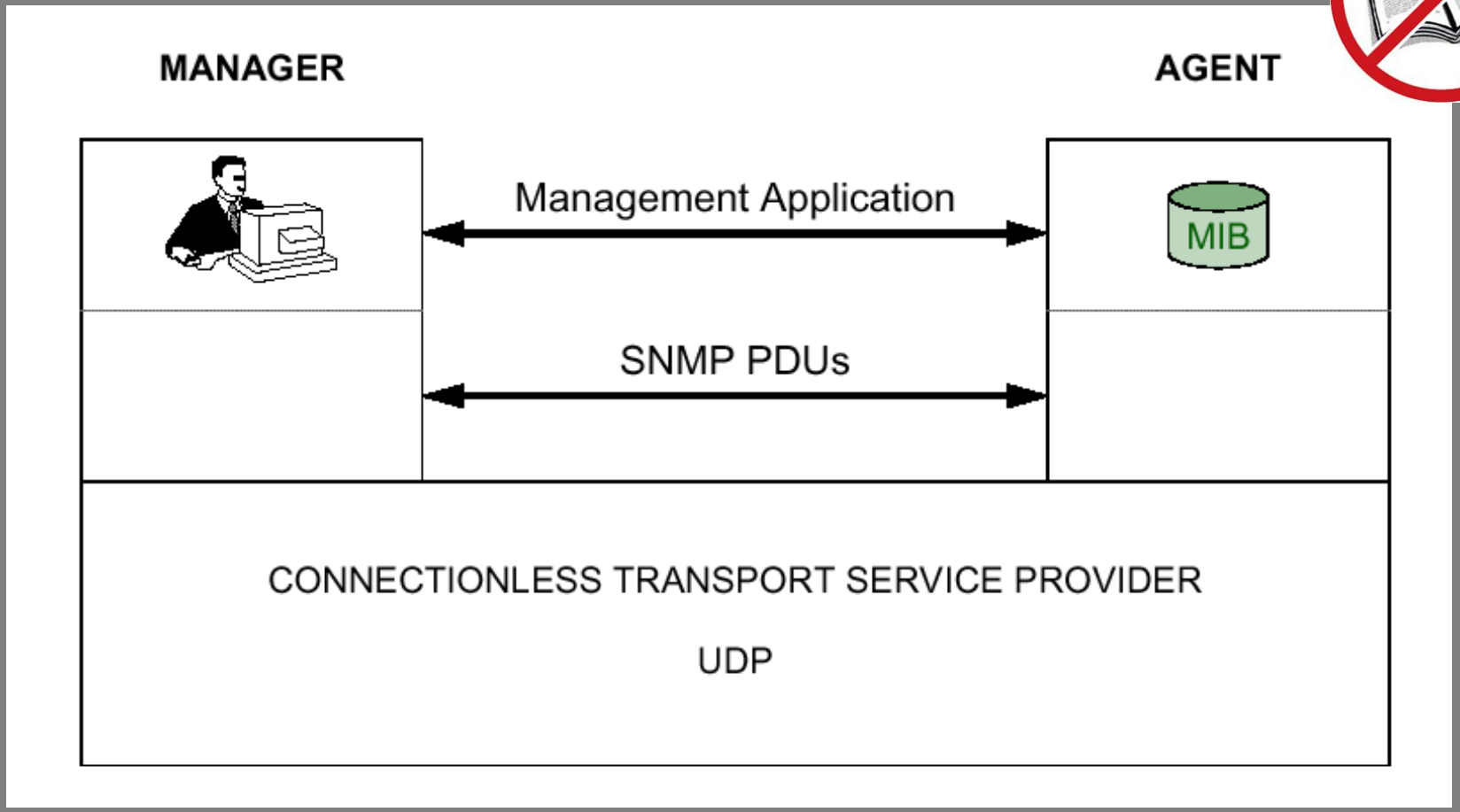
- Tareas del Agente:
  - El agente SNMP debe implementar entonces también tanto UDP como IP.
  - El agente además debe encargarse de controlar su MIB, de medir todos los parámetros soportados y de atender a los pedidos del gestor.
  - El gestor y el agente deben estar de acuerdo en la definición de la MIB!

SNMP


MANAGER
Las variables residen en los agentes!
get/set
AGENTS
TABLES
VARIABLES

SNMP




MANAGER
AGENT
Management Application
MIB
SNMP PDUs
CONNECTIONLESS TRANSPORT SERVICE PROVIDER
UDP

SNMP

- Trap-directed polling:
  - Cuando la cantidad de agentes en la red crece y la cantidad de objetos a monitorizar, también, la cantidad de información a manejar también.
  - No es práctico recoger con alta frecuencia toda la información de la MIB.

SNMP

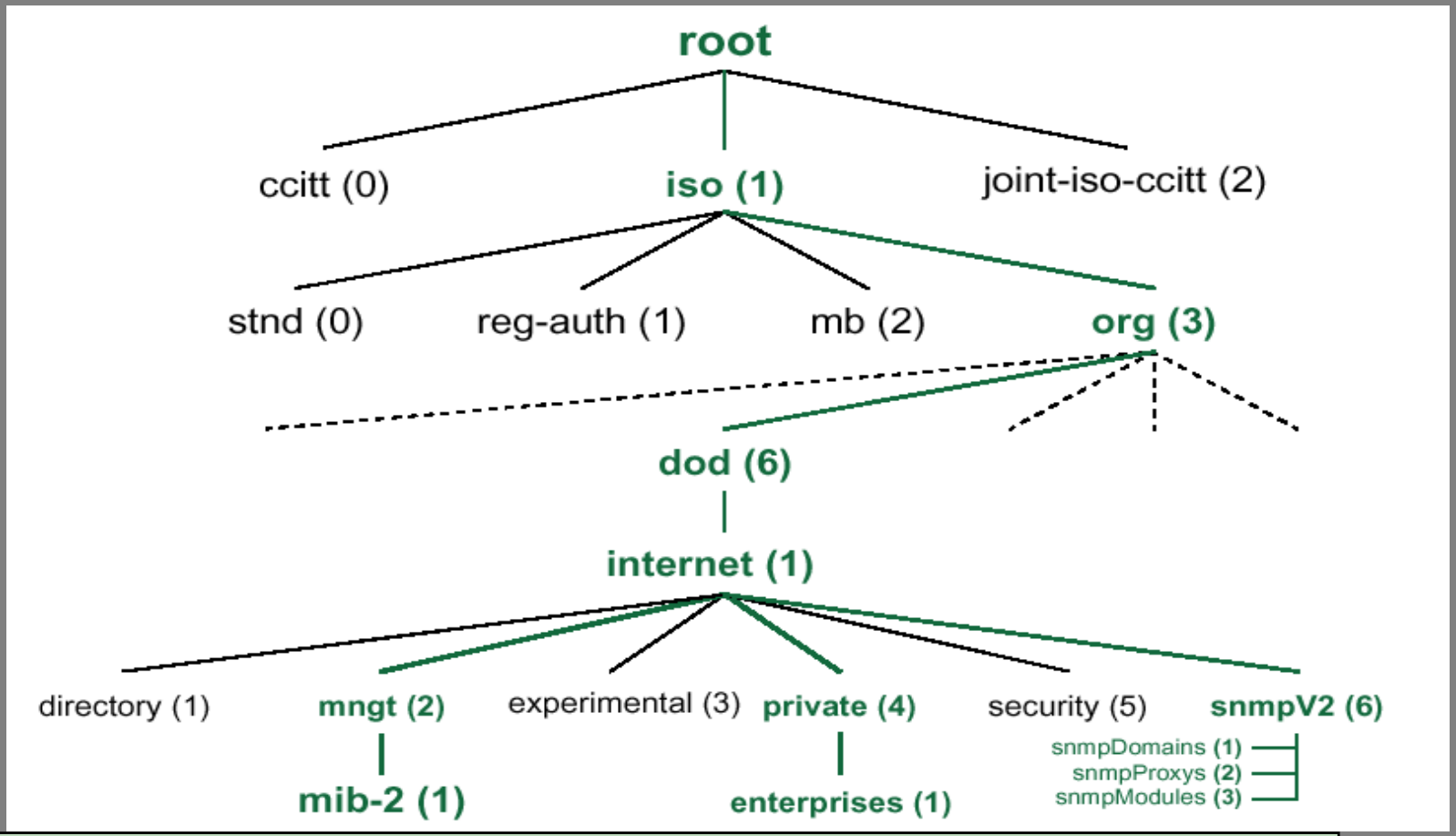
- Trap-directed polling:
  - Con baja frecuencia (cada varias horas o una vez por día) el gestor pregunta a los agentes que conoce (*polls*) por la información que se desea monitorizar de la MIB.
  - Los agentes notifican al gestor (a través de *traps*) de situaciones significativas.
  - Una vez notificado, el gestor puede optar por pollear al agente que envió el *trap* y eventualmente a otros agentes relacionados a los efectos de obtener información que permita diagnosticar el problema.
  - Ventaja: ahorro de recursos de ancho de banda, procesamiento en los agentes, espacio de disco en el gestor, etc.

SNMP

- MIB: Management Information Base
  - El o los objetos utilizados para representar un recurso deben ser los mismos en cada sistema.
  - Debe manejar un esquema común de representación para asegurar la interoperabilidad.
    - Objetivo de la especificación SMI

SNMP

- Estructura de Árbol de la MIB



SE VERÉ EN EL CURSO SOLO LO DE COLOR VERDE - ESTRUCTURA DEL ÁRBOL - INTERNET

SNMP

- Hay un único árbol de variables en el cual se enmarca todas las posibles variables.
- El árbol refleja una división administrativa en sus ramas más bajas y una división funcional en sus ramas más altas.
  - División según organismos (iso, ccitt, etc)
  - División según protocolos, etc.
- La definición de objetos utiliza un subconjunto de ASN.1 (Abstract Syntax Notation One)
  - Estándar ISO
  - Notación para definir tipos de datos y estructuras independiente del protocolo y de la implementación.

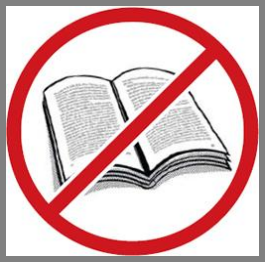
SNMP

- Toda variable esta definida por :
  - Una denominación numérica
    - 1.3.6.1....
      - Refleja el “camino en el árbol” para llegar a ella
  - Una denominación textual
    - Iso.org.dod.internet....
      - También refleja el camino en el árbol, pero de una manera más humanamente comprensible.
  - Tipo de datos
    - Entero/string
    - Escalar/tabla

SNMP

- MIB – Tipos de datos
  - Básicos (Universales de ASN.1)
    - **`Integer`**
    - **`Octectstring`**
    - **`Null`**
    - **`Object identifier`**
    - **`Sequence`**

SNMP



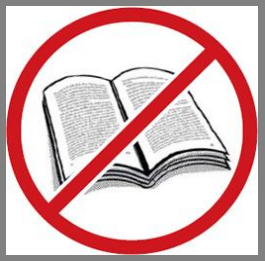
- MIB – Tipos de datos
  - Tipos definidos por IETF para la aplicación SNMP
    - **`ipaddress:`** valor de 32 bits que representa una dirección IP
    - **`counter:`** entero de 32 bits que puede ser incrementado pero NO decrementado
    - **`gauge:`** entero de 32 bits que puede ser tanto incrementado como decrementado
    - **`timeticks:`** contador que especifica un numero de centésimas de segundo desde un cero predefinido.
    - **`opaque:`** tipo reservado para el pasaje de datos arbitrarios.

SNMP

- Primitivas:
  - Operaciones que todo agente debe soportar.
  - Utilizadas por los gestores para obtener información de los agentes, o para configurar valores en variables donde esto es posible.
  - En SNMP solo pueden transmitirse escalares (a diferencia de lo propuesto por la gestión OSI)

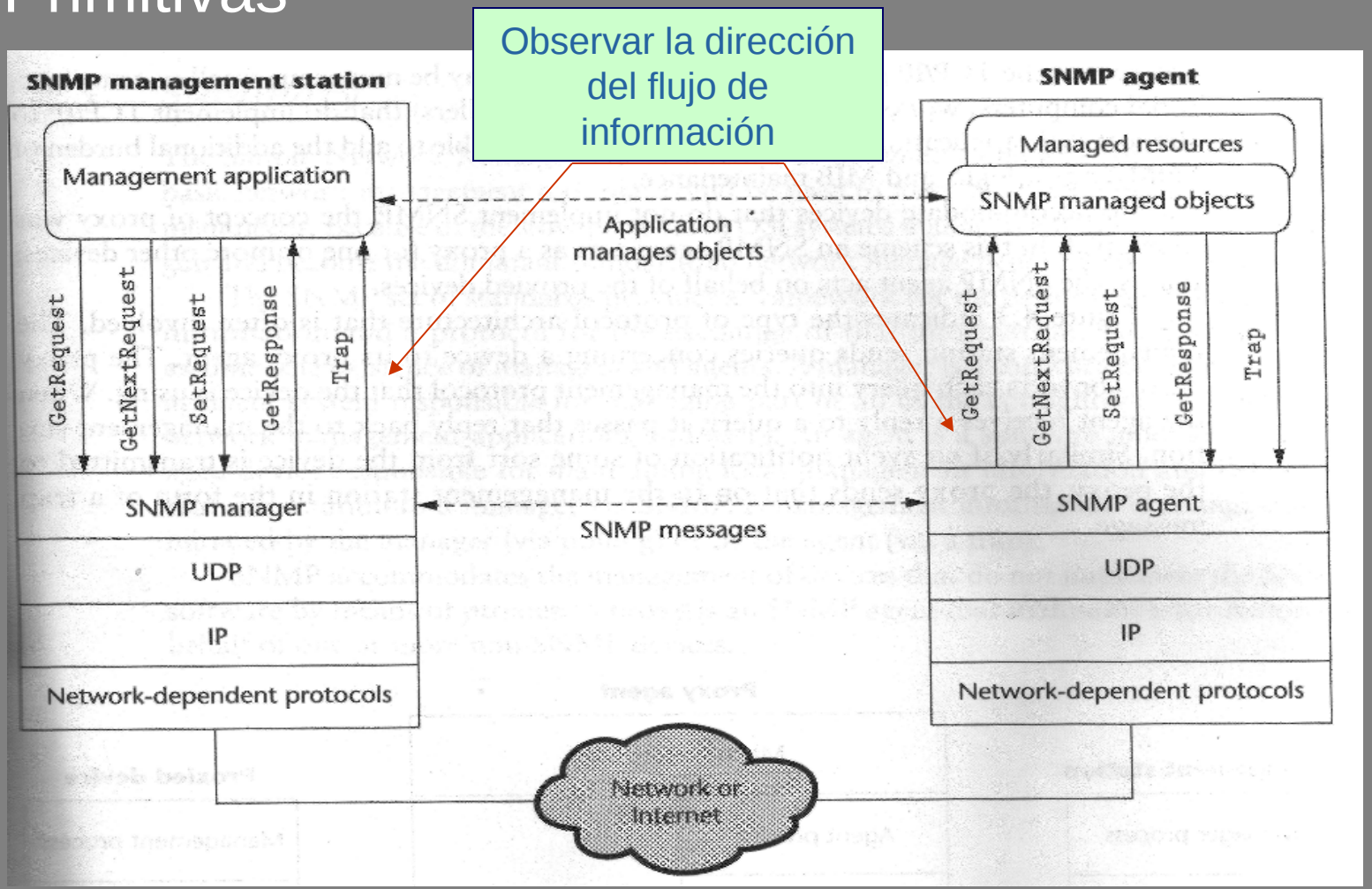
SNMP

- Primitivas
  - **`getRequest`**
    - Obtener el valor de una variable
  - **`getNextRequest`**
    - Obtener el valor de la “siguiente” (de acuerdo al árbol) variable
  - **`getResponse`**
    - Paquete de respuesta a un get/getNext
  - **`setRequest`**
    - Cambiar el valor de una variable
  - **`Trap`**
    - Envío de información no solicitada por el gestor por parte del agente.



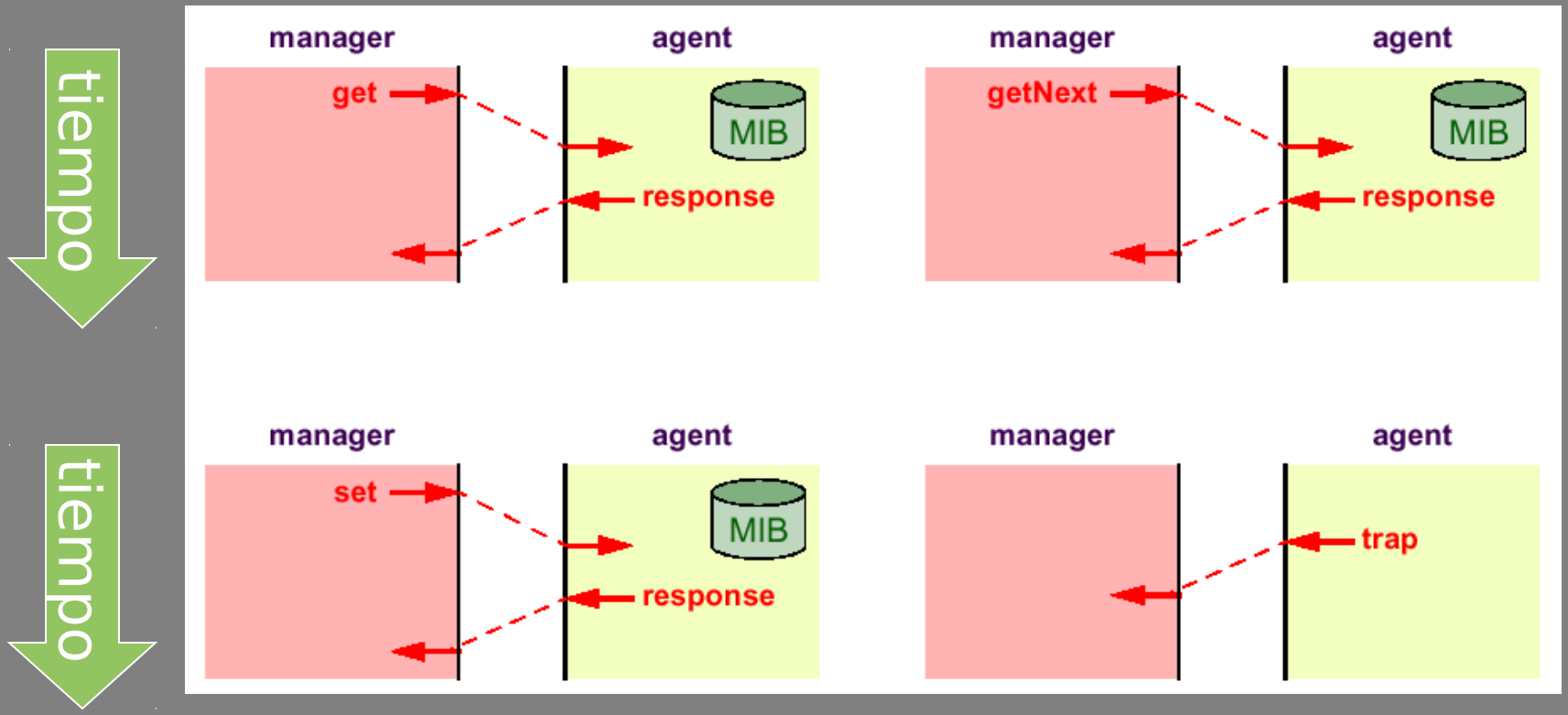
SNMP

- Primitivas



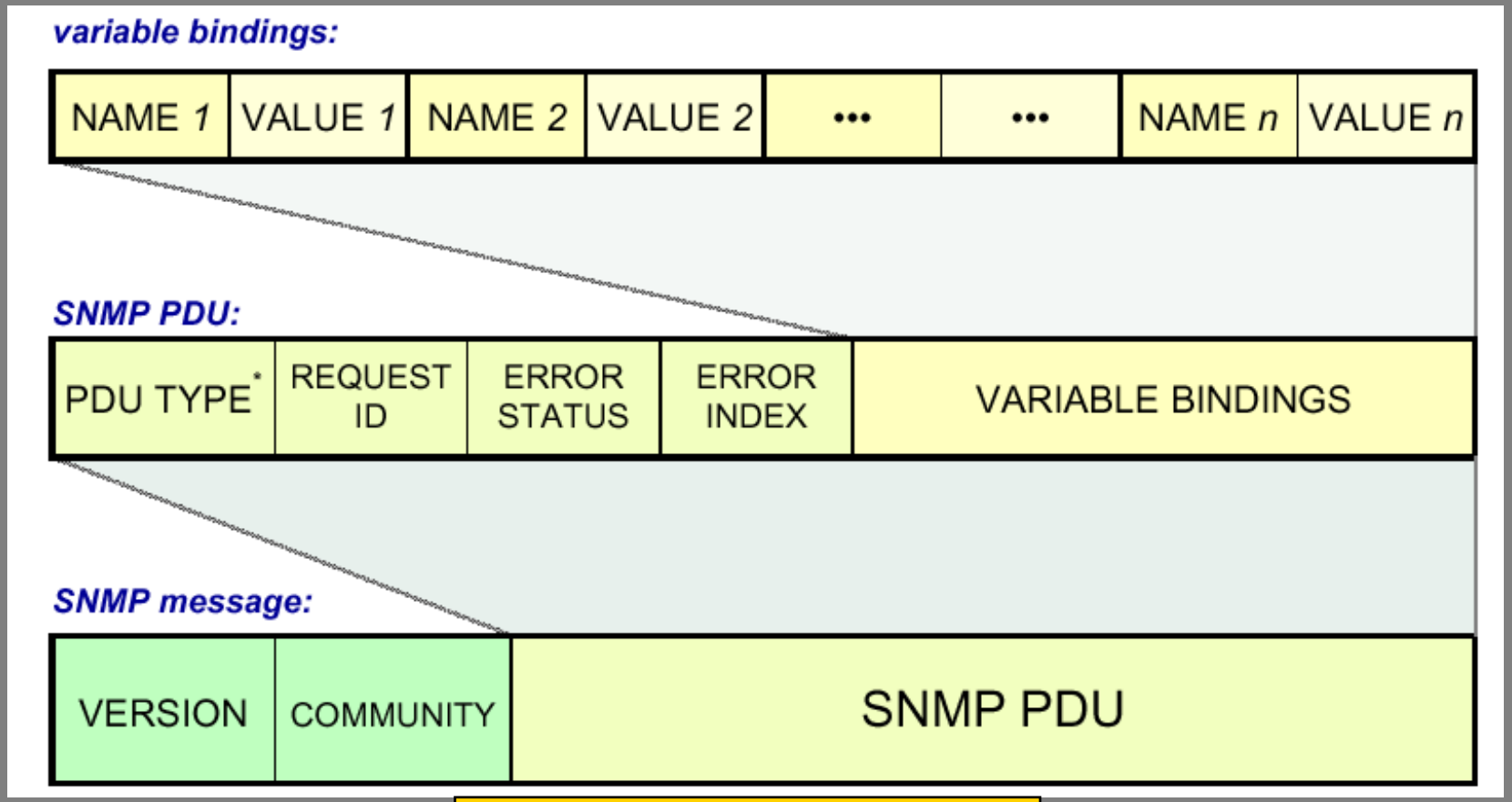
SNMP

- Primitivas

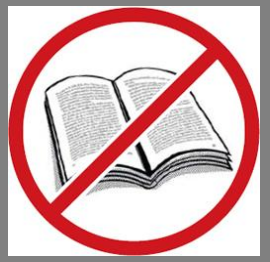


SNMP

- Formato de las PDUs: formato genérico:



SE DEBE LEER DE ABAJO HACIA ARRIBA



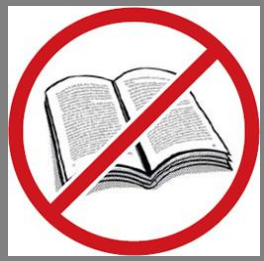
SNMP

- Primitivas:
  - **`Get`**, **`GetNext`**, **`GetResponse`** y **`Set`** se manejan con el mismo formato de PDU.
  - `ErrorStatus` y `ErrorIndex` van a cero en los requests (pero no en las responses!)

PDU type	request-id	0	0	variablebindings

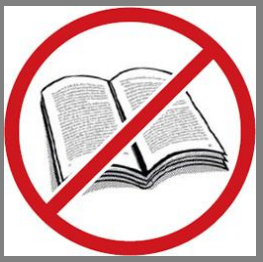
GetRequest PDU, GetNextRequest PDU, and SetRequest PDU

SNMP

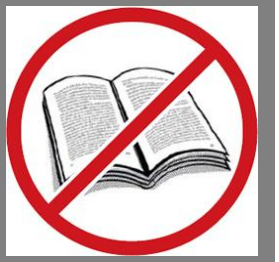


- **`Get`**:
  - Los campos de error van a cero
  - Los valores de las variables (*`variablebindings`*) deben ponerse a `null`.
  - **`RequestID`** se utiliza para identificar diferentes pedidos.
    - Las RFCs no especifican más nada que esto acerca de este campo!
      - P.ej. No tienen por qué ser únicos, o consecutivos.
      - No se puede utilizar este campo para detectar duplicados o pedir retransmisiones!

SNMP

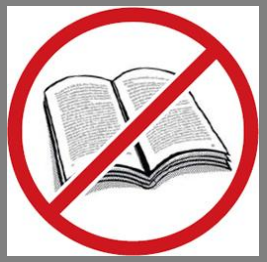


- **`GetNextRequest`**:
  - Igual al **`GetRequest`** en casi todos los aspectos.
  - Los valores devueltos corresponden al de los objetos “*siguientes*” a los pedidos en el árbol.
  - Utilizando esta primitiva podemos recorrer el árbol y obtener objetos no-escalares (tablas)



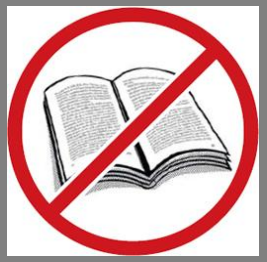
SNMP

- **`GetResponse`**:
  - Igual formato de PDU para Get y GetNext
  - Se asocia con el Request a través del `RequestID`.
  - Los campos de valor no están a `null` sino al valor que corresponde
  - Pueden existir condiciones de error:
    - `noSuchName`: el objeto no existe o no es una hoja del árbol.
    - `tooBig`: la respuesta no cabe dentro del tamaño máximo de PDU.
      - No hay que olvidarse que tenemos una cota máxima de tamaño de respuesta dada por el medio de transmisión (MTU)
    - `genErr`: cualquier otra condición.



SNMP

- **`SetRequest`**:
  - Igual formato de PDU a los anteriores
  - Utilizado para configurar valores en el agente (para aquellos que esto es posible)
  - Los valores a configurar se pasan en el campo `variablebindings`.
  - Condiciones de error (además de las anteriores)
    - `badValue`: el dato proporcionado para configurar es incompatible con la definición que tiene el agente para el mismo (en su MIB).

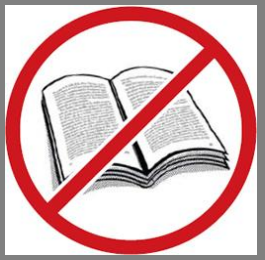


SNMP

- **`Trap`**:
  - Única PDU originada por el agente
  - Campos:
    - `enterprise`: identificador de entidad que envia el trap (tomado del grupo System de la MIB-II).
    - `agent-addr`: direccion IP del originador
    - `generic-trap`: identificador de tipo de trap (genericos)
    - `specific-trap`: identificacion mas especifica del tipo de trap
    - `time-stamp`:

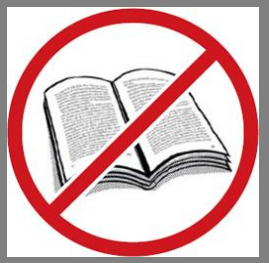
PDU type	enterprise	agent-addr	generic-trap	specific-trap	time-stamp	variablebindings

Trap PDU



SNMP

- Traps Genéricos:
  - `coldStart:`
    - Reboot “duro”
  - `warmStart:`
    - Reboot “blando”
  - `linkDown:`
    - Caída de un enlace
  - `linkUp:`
    - Levantada de un enlace
  - `authenticationFailure:`
    - Falla de autenticación (comunidad)
  - `egpNeighborLoss:`
    - No se usa más!!!
  - `enterpriseSpecific:`
    - Específico del fabricante



SNMP

- Traps:
  - `specific-trap`: lo utiliza cualquier fabricante que quiera dar información mas especifica sobre una falla
  - `variablebindings`: nos permite asociar variables con valores en el momento de ocurrir el trap.

SNMP

- Consideraciones de Seguridad:
  - Uno de los grandes problemas con SNMPv1 es la seguridad.
  - Tema que se dejó siempre “para otro momento”, ya que en el momento del desarrollo de SNMPv1, se suponía que el protocolo era una solución temporal.
  - A su vez, la Internet era otro mundo en 1989, era una red casi completamente académica, donde muchas cosas quedaban libradas a la confianza y había una ética de trabajo.
  - Un simple control “solo-lectura” o “lectura-escritura” era más que suficiente.

SNMP

- Consideraciones de Seguridad:
  - Aspectos a tener en cuenta :
    - Autenticación:
      - restricción de acceso de los agentes solamente a una o algunas estaciones de gestión
      - las restricciones pueden ser
        - no-acceso
        - solo-lectura
        - lectura-escritura
    - Políticas de acceso:
      - manejo de restricciones diferenciales de acceso según diferentes estaciones de gestión.
      - restricciones de acceso diferenciales según ramas de la MIB soportada por el agente.

SNMP

- Comunidades:
  - La RFC 1157 define solamente un mecanismo precario de autenticación, el concepto de ***comunidad***.
  - Una comunidad representa un conjunto de entidades SNMP que comparten y aplican los mismos criterios de autenticación y acceso.
  - Cada comunidad recibe un nombre único que es compartido por todos los sus miembros.

SNMP

- Comunidades:
  - Los gestores deben incluir el nombre de comunidad en todas las PDUs.
  - Los agentes deben compararla antes de enviar las respuestas.
    - recordar el trap `authenticationFailure`!
  - Los gestores también comparan la comunidad de las respuestas con la propia.

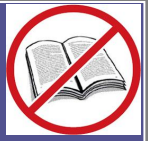
SNMP

- Políticas de Acceso:
  - Pueden ser de dos tipos:
    - MIB Views:
      - restringir el acceso a un subconjunto de los objetos de la MIB.
      - Puede ser una rama o un conjunto de ramas.
    - Modo de acceso:
      - solo-lectura y lectura-escritura
        - diferenciable por gestor y por variable

SNMP

- Ejemplo de objeto MIB:

```
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 }
```



SNMP

- Ejemplo de objeto MIB:

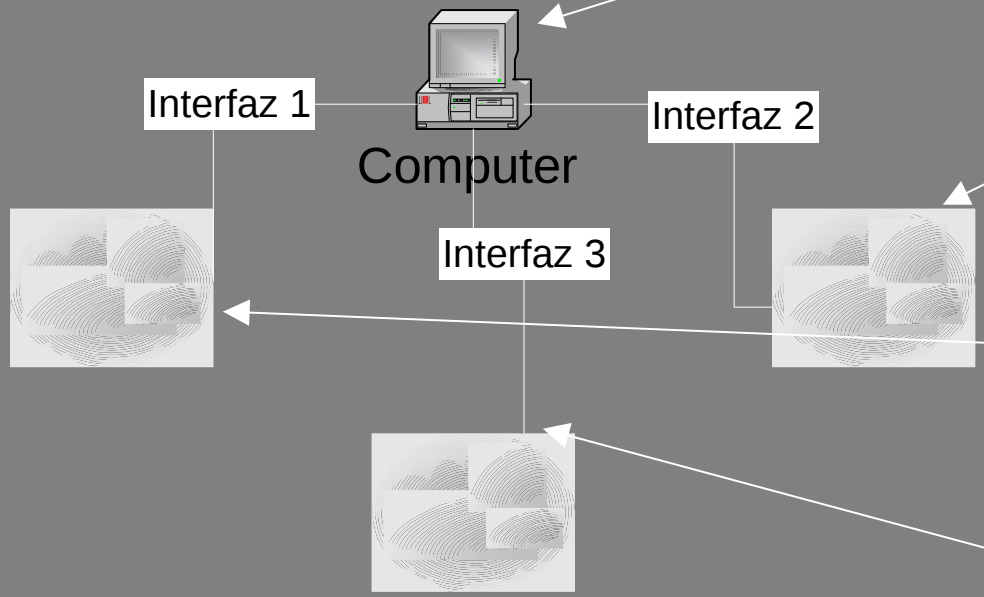
```
sysUpTime OBJECT-TYPE
            SYNTAX  TimeTicks
            ACCESS  read-only
            STATUS  mandatory
            DESCRIPTION
               "The time (in hundredths of a second) since the
               network management portion of the system was last
               re-initialized."
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

SNMP

- Objetos e Instancias:
  - La MIB define un conjunto de *objetos*.
  - Cada objeto de la MIB puede tener una o más *instancias*.
  - Una variable es *simple* cuando tiene una única instancia.
    - En este caso se la referencia agregando un cero (0) a su identificador de objeto (OID).
      - 1.3.6.1.1.----.0

SNMP

- Objetos e Instancias:



Una variable:
ifOutOctects

Instancia:
ifOutOctects.2

Instancia:
ifOutOctects.1

Instancia:
ifOutOctects.3

SNMP

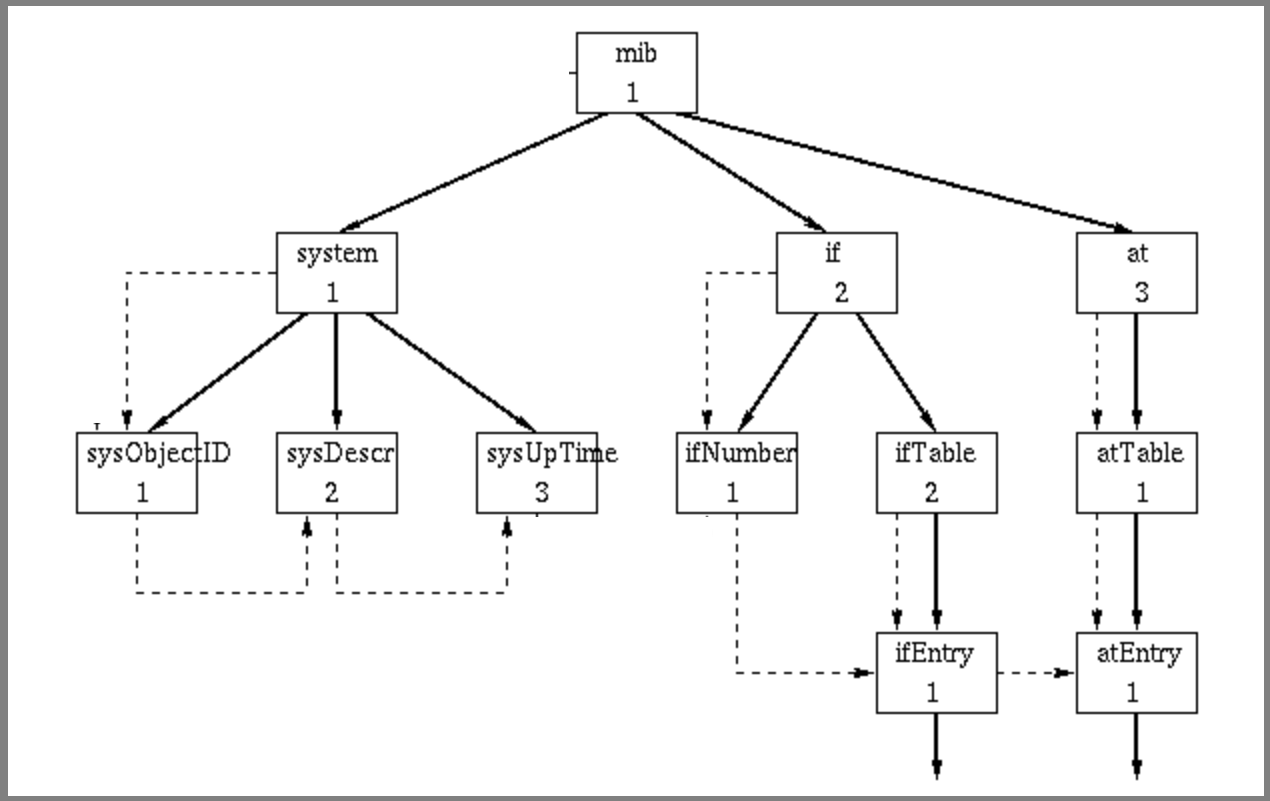
- Orden Lexicográfico
  - La propia estructura en árbol de la MIB le confiere lo que se denomina *orden lexicográfico*.
  - Recordemos que el camino a través del árbol para un objeto es su OID (**O**bject **ID**entifier).

SNMP

- Ordenamiento lexicográfico:
  - Sean dos secuencias S y T que denotan dos OIDs:
    - S = {s_0 s_1 s_2... s_N}
    - T = {t_0 t_1 t_2... t_M}
  - **Regla "A"**: Definimos que S "**es igual a**" T si y solo si:
    - N=M y $s_i = t_i$ para todo i<=M
  - **Regla "B"**:Definimos que S "**es menor que**" si se cumple alguna de las siguientes condiciones:
    - N<M y $s_i = t_i$ para todo i<=N
    - Existe *i* tal que $s_i < t_i$ y $s_j = t_j$ para todo j<i
      - i<=min(N,M)

SNMP

- Orden Lexicográfico:



SNMP

- Orden Lexicográfico:
  - Reglas Fáciles (“*rules of thumb*”):
    - Objetos que están al mismo nivel del árbol y son hijos del mismo padre (son “hermanos”):
      - El ordenamiento queda dado por el último tag del OID.
    - Objetos que están al mismo nivel del árbol pero son hijos de distintos padres (son “primos”):
      - El ordenamiento queda dado por el ordenamiento de los padres, sin importar el tag de ellos mismos.

SNMP

- Cómo recorrer una tabla:
  - Por esto es particularmente importante el ordenamiento lexicográfico!
  - Aprovechamos el ordenamiento natural de las MIBs y la primitiva `GetNext`:
  - La primitiva `GetNext` tratará de devolvernos el valor del siguiente objeto según el ordenamiento lexicográfico.

SNMP

- Recorrer una tabla, ejemplo :
  - La tabla de enrutamiento de un router IP es accesible vía SNMP.
  - Consideremos la siguiente tabla de enrutamiento:

ipRouteDest	ipRouteNextHop	ipRouteMetric1
10.0.0.99	89.1.1.42	5
9.1.2.3	99.0.0.3	3
10.0.0.51	89.1.1.42	5

SNMP

- Recorrer tabla:
  - GetNextRequest (ipRouteDest, ipRouteNextHop, ipRouteMetric1)
  - GetResponse ((ipRouteDest.9.1.2.3 = "9.1.2.3"), (ipRouteNextHop.9.1.2.3 = "99.0.0.3"), (ipRouteMetric1.9.1.2.3 = 3))
  - GetNextRequest (ipRouteDest.9.1.2.3, ipRouteNextHop.9.1.2.3, ipRouteMetric1.9.1.2.3)
  - GetResponse ((ipRouteDest.10.0.0.51 = "10.0.0.51"), (ipRouteNextHop.10.0.0.51 = "89.1.1.42"), (ipRouteMetric1.10.0.0.51 = 5))

SNMP

- Recorrer Tabla:
  - GetNextRequest (ipRouteDest.10.0.0.51, ipRouteNextHop.10.0.0.51, ipRouteMetric1.10.0.0.51)
  - GetResponse ((ipRouteDest.10.0.0.99 = "10.0.0.99"), (ipRouteNextHop.10.0.0.99 = "89.1.1.42"), (ipRouteMetric1.10.0.0.99 = 5))
- Para aquellas variables que sabemos son tablas, debemos aplicar este algoritmo.

SNMP

- Como sabemos que son tablas ?
  - Porque así se definen en la MIB!
  - Ejemplo: Tabla de Enrutamiento

```
-- the IP routing table

     -- The IP routing table contains an entry for each route
     -- presently known to this entity.

     ipRouteTable OBJECT-TYPE
         SYNTAX  SEQUENCE OF IpRouteEntry
         ACCESS  not-accessible
         STATUS  mandatory
         DESCRIPTION
                 "This entity's IP Routing table."
         ::= { ip 21 }
```

SNMP

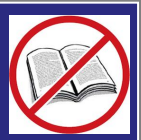
- Definición de tablas en la MIB

```
ipRouteEntry OBJECT-TYPE
              SYNTAX  IpRouteEntry
              ACCESS  not-accessible
              STATUS  mandatory
              DESCRIPTION
                 "A route to a particular destination."
              INDEX   { ipRouteDest }
              ::= { ipRouteTable 1 }
```

SNMP

- Definición de tablas en la MIB (cont.)

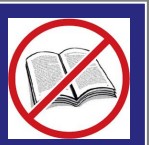
```
IpRouteEntry ::=
    SEQUENCE {
        ipRouteDest
            IpAddress,
        ipRouteIfIndex
            INTEGER,
        ipRouteMetric1
            INTEGER,
        ipRouteNextHop
            IpAddress
    }
```



SNMP

- Definición de tablas en la MIB (cont.)

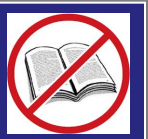
```
ipRouteDest OBJECT-TYPE
              SYNTAX  IpAddress
              ACCESS  read-write
              STATUS  mandatory
              DESCRIPTION
                 "The destination IP address of this route.  An
                 entry with a value of 0.0.0.0 is considered a
                 default route.  Multiple routes to a single
                 destination can appear in the table, but access to
                 such multiple entries is dependent on the table-
                 access mechanisms defined by the network
                 management protocol in use."
              ::= { ipRouteEntry 1 }
```



SNMP

- Definición de tablas en la MIB (cont.)

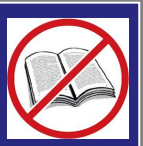
```
ipRouteIfIndex OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
            "The index value which uniquely identifies the
            local interface through which the next hop of this
            route should be reached.  The interface identified
            by a particular value of this index is the same
            interface as identified by the same value of
            ifIndex."
    ::= { ipRouteEntry 2 }
```



SNMP

- Definición de tablas en la MIB (cont.)

```
ipRouteMetric1 OBJECT-TYPE
               SYNTAX  INTEGER
               ACCESS  read-write
               STATUS  mandatory
               DESCRIPTION
                  "The primary routing metric for this route.  The
                  semantics of this metric are determined by the
                  routing-protocol specified in the route's
                  ipRouteProto value.  If this metric is not used,
                  its value should be set to -1."
              ::= { ipRouteEntry 3 }
```



SNMP

- Definición de tablas en la MIB (cont.)

```
ipRouteNextHop OBJECT-TYPE
               SYNTAX  IpAddress
               ACCESS  read-write
               STATUS  mandatory
               DESCRIPTION
                  "The IP address of the next hop of this route.
                  (In the case of a route bound to an interface
                  which is realized via a broadcast media, the value
                  of this field is the agent's IP address on that
                  interface.)"
               ::= { ipRouteEntry 7 }
```