

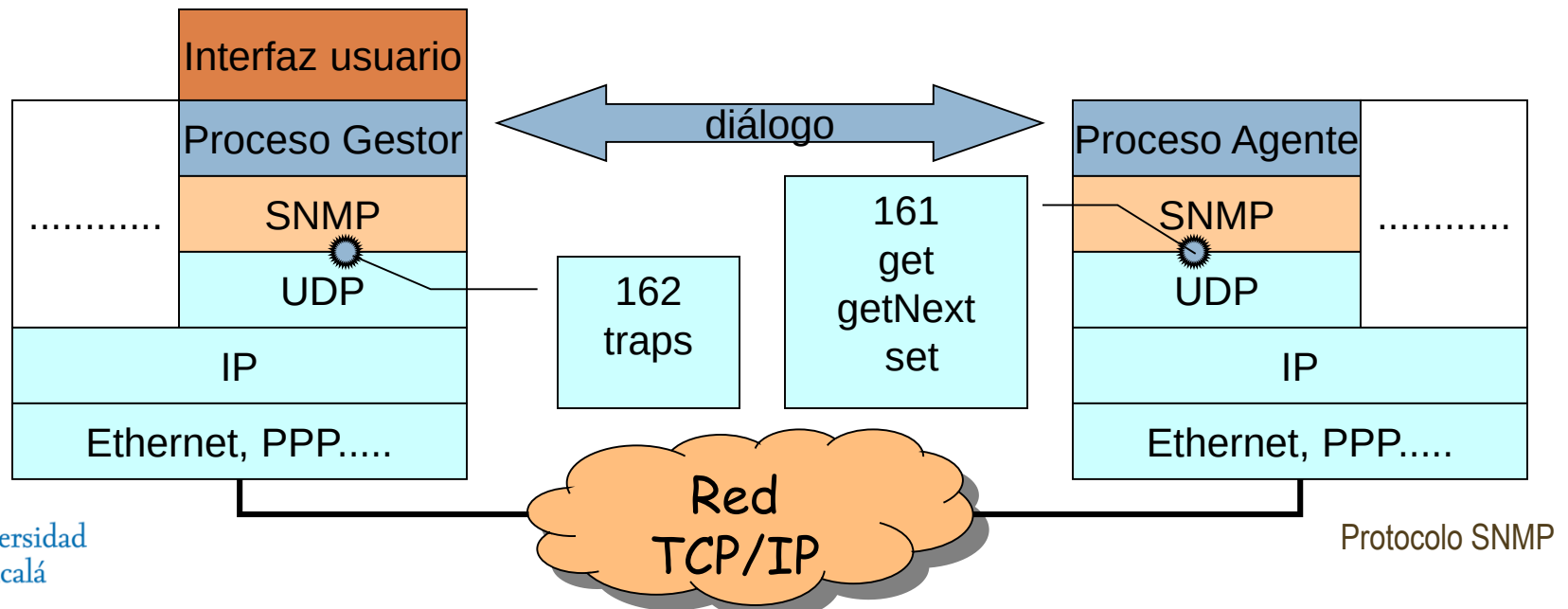
Protocolo SNMP

Gestión de Redes

Arquitectura SNMPv1

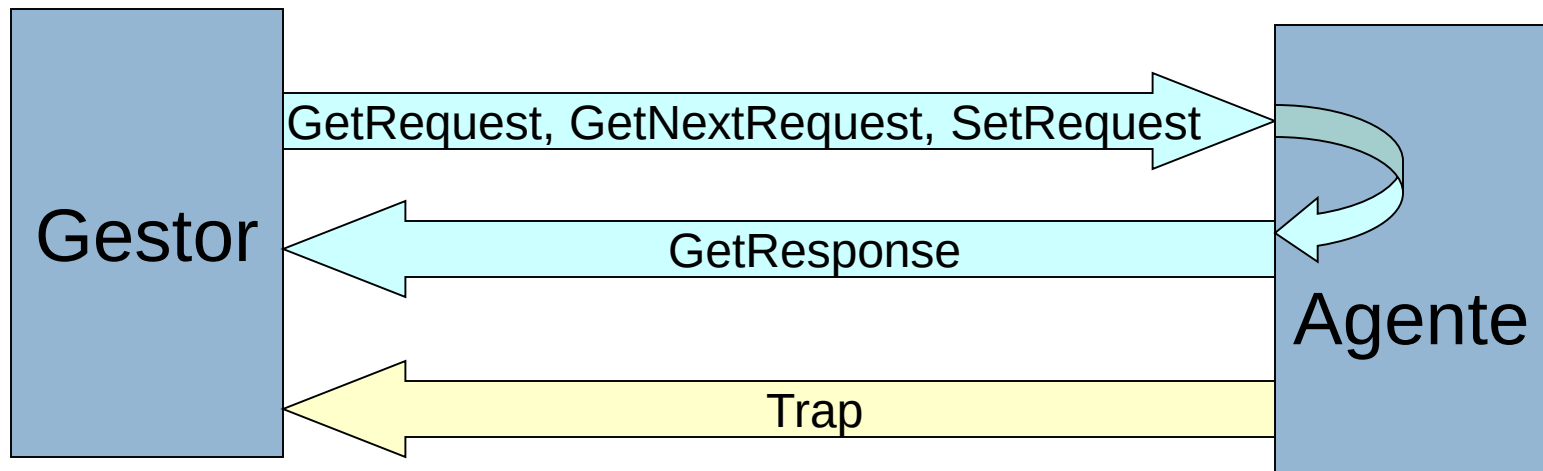
2

- ❑ UDP: servicio no fiable.
- ❑ Sobre UDP se establece un diálogo entre gestor y el agente siguiendo el modelo pregunta/respuesta (excepto en trap).
- ❑ Si se pierde un mensaje SNMP, lo repite hasta un número determinado de veces. Si no se recupera, avisa.



Operaciones

3



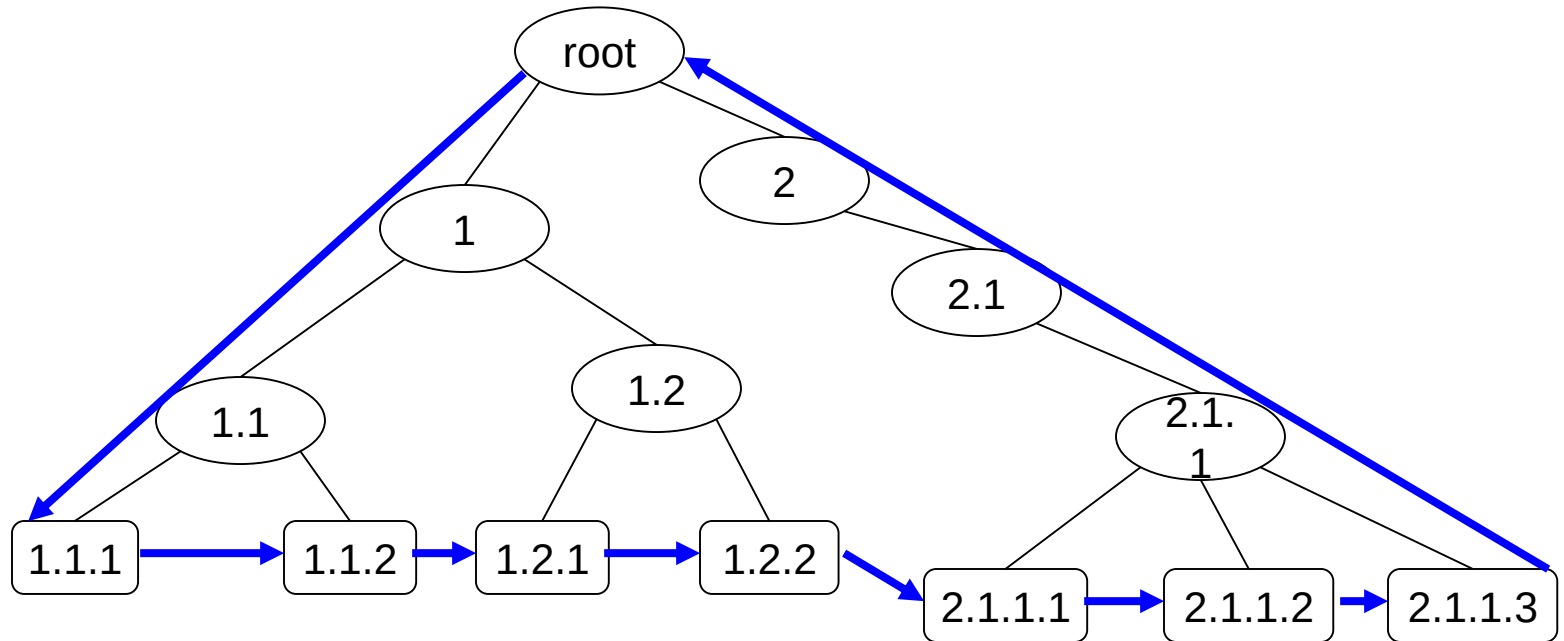
- ❑ **GetRequest:** el gestor realiza una petición de valores específicos de la MIB del agente.
- ❑ **GetNextRequest:** el gestor realiza una petición del objeto siguiente a uno dado en la MIB del agente, siguiendo un *orden lexicográfico*.
- ❑ **GetResponse:** el agente devuelve los valores solicitados por la operaciones anteriores del gestor. Es la respuesta a un SetRequest, GetRequest o GetNextRequest.
- ❑ **SetRequest:** el gestor permite asignar un valor a una variable en el sistema del agente.
- ❑ **Traps:** el agente informa de un suceso inusual predefinido.

- Se establece un orden en los objetos de MIBs que permite un acceso secuencial (GetNext).
- Esto puede ser útil en algunos casos:
 - ▣ Cuando un gestor no conoce la estructura de la vista a la que tiene acceso en un agente.
 - ▣ Acceso a tablas.
- Formalmente se define de la siguiente forma: dadas dos secuencias de números no negativos $X=(x_1,x_2,\dots,x_n)$ e $Y=(y_1,y_2,\dots,y_m)$. X precede a Y si se cumple:
$$[(x_j=y_j \text{ para } 1 \leq j < k) \text{ y } (x_k < y_k \text{ para } k \leq n,m)]$$
$$\text{O}$$
$$[(x_j=y_j \text{ para } 1 \leq j \leq n) \text{ y } (n < m)]$$

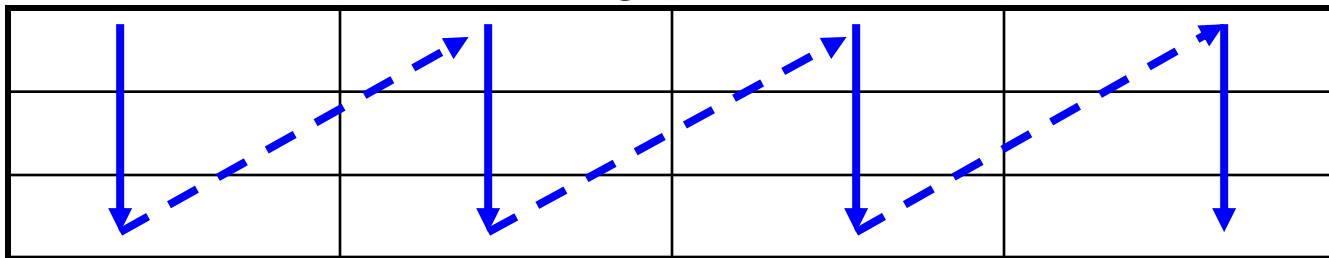
Orden lexicográfico (cont)

6

Orden lexicográfico en un árbol de objetos escalares



Orden lexicográfico en una tabla



Identificación de instancias de tablas

7

- Tablas:
 - ▣ Secuencia de objetos columnares (definidos en la MIB)
 - ▣ Varias instancias por cada objeto columnar.
- Índice de tabla: subconjunto de *objetos columnares*.
- El índice está declarado en la MIB: INDEX.
- El índice debe permitir identificar una instancia de forma única, para ello puede estar formado por uno o más objetos.
 - ▣ **Índice simple.** En *ifTable* el índice es: *ifIndex*.
 - ▣ **Índice compuesto.** En *tcpConnTable* el índice es:
tcpConnLocalAddress.tcpConnLocalPort.tcpConnRemAddress.tcpConnRemPort

Identificación de la instancia:
identificación del *objeto columnar* + valor del índice en la fila dada
(en notación punto)

Ejemplo de identificación en tablas

8

tcpConnTable

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
2	0.0.0.0	99	0.0.0.0	0
5	10.0.0.99	12	9.1.2.3	15
3	10.0.0.99	14	89.1.1.42	84

Ejemplo de identificación en tablas

9

OIDs

X=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 1.3.6.1.2.1.6.13.1.3	tcpConnRemAddress 1.3.6.1.2.1.6.13.1.4	tcpConnRemPort 1.3.6.1.2.1.6.13.1.5
X.1.0.0.0.0.99.0.0 .0.0.0	X.2.0.0.0.0.99.0.0.0.0. 0	X.3.0.0.0.0.99.0.0 .0.0.0	X.4.0.0.0.0.99.0.0.0.0 .0	X.5.0.0.0.0.99.0.0.0. 0.0
X.1.10.0.0.99.12.9 .1.2.3.15	X.2.10.0.0.99.12.9.1.2. 3.15	X.3.10.0.0.99.12. 9.1.2.3.15	X.4.10.0.0.99.12.9.1.2. 3.15	X.5.10.0.0.99.12.9.1. 2.3.15
X.1.10.0.0.99.14.8 9.1.1.42.84	X.2.10.0.0.99.14.89.1.1. 42.84	X.3.10.0.0.99.14. 89.1.1.42.84	X.4.10.0.0.99.14.89.1. 1.42.84	X.5.10.0.0.99.14.89. 1.1.42.84

- Si la sentencia INDEX no está presente, las instancias del objeto pueden identificarse añadiendo un subidentificador, ¿cómo se recomienda calcularlo?
- Se definen las siguientes reglas (RFC1212) según el tipo de valor en la instancia que forma el índice:
 - ▣ **Entero (no negativo)**: valor del entero forma un subidentificador.
 - ▣ **String de longitud fija**: cada octeto del string forma un subidentificador.
 - ▣ **String de longitud variable**: para un string de n octetos, el primer subidentificador es n ; seguido por el valor cada octeto que forma un subidentificador. En total $n+1$ subidentificadores.
 - ▣ **OID**: con n subidentificadores, el primer subidentificador es n ; seguido por el valor de cada subidentificador en orden. En total $n+1$ subidentificadores.
 - ▣ **Dirección IP**: 4 subidentificadores en notación punto.

Instancias/objetos escalares

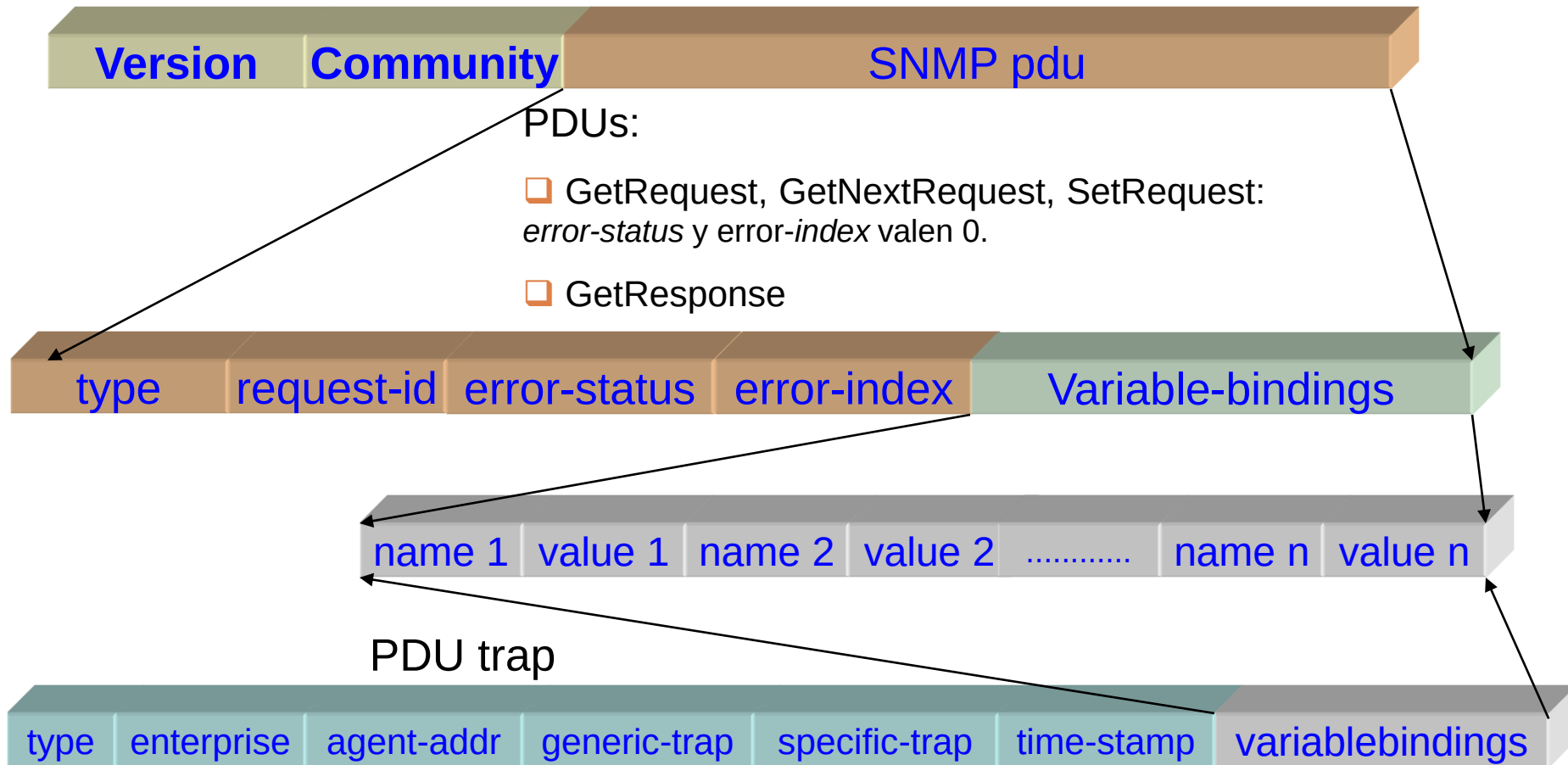
11

- Para los objetos escalares no existe ambigüedad entre objeto e instancia. Cada objeto tiene asociada una única instancia.
- Seguir la convención de los objetos tabulares: distinción entre objeto e instancia.
- Una instancia se nombra con el OID del objeto concatenado con el subidentificador 0.
- Ejemplo:

Nombre del objeto	OID del objeto	OID de la instancia
tcpRtoAlgorithm	1.3.6.1.2.1.6.1	1.3.6.1.2.1.6.1.0
tcpRtoMin	1.3.6.1.2.1.6.2	1.3.6.1.2.1.6.2.0
tcpRtoMax	1.3.6.1.2.1.6.3	1.3.6.1.2.1.6.3.0
tcpMaxConn	1.3.6.1.2.1.6.4	1.3.6.1.2.1.6.4.0

Formato Mensajes

12



Formato de los mensajes

13

- **Versión:** versión del protocolo SNMP.
- **Community:** comunidad que permite la autenticación del gestor ante el agente y el control de acceso.
- **Request-id:** identificador único para cada petición, es el mismo en la respuesta correspondiente.
- **Error-status:** indica si ha ocurrido un error cuando procesa una petición. Los códigos son: noError (0), tooBig (1), noSuchName (2), badValue (3), readOnly (4), genErr (5). Vale 0 en caso de getRequest, getNextRequest y setRequest.
- **Error-index:** cuando se ha producido un error, puede aportar información, indicando qué instancia de la lista de variable-bindings ha producido el error. Vale 0 en caso de no error.

Formato de mensajes: Traps

14

- ❑ **Enterprise:** tipo de sistema que genera el trap, basado en sysObjectID.
- ❑ **Agent-addr:** dirección IP del agente que genera el trap.
- ❑ **Generic-trap:** código de trap genérico.
- ❑ **Specific-trap:** código de trap específico.
- ❑ **Time-stamp:** tiempo desde la última (re)inicialización del agente y la generación del trap (valor de sysUpTime).

- Entidad SNMP para enviar mensaje realiza:
 - ▣ **Generación de PDU:** construye la PDU SNMP utilizando la estructuras ASN.1 definida en RFC 1157.
 - ▣ **Autenticación:** (en la práctica no se suele invocar este proceso)
 - Se pasa a un proceso de autenticación: PDU, puerto fuente, puerto destino, comunidad.
 - El servicio de autenticación realiza tareas como cifrado o generación de autenticación.
 - ▣ **Generación del mensaje:** la entidad del protocolo construye el mensaje: versión, comunidad y salida del proceso anterior (PDU).
 - ▣ **Codificación:** se codifica todo utilizando BER y se pasa al servicio de transporte.

- **Entidad SNMP** una vez recibido un mensaje realiza:
 - ▣ **Chequeo de sintaxis del mensaje:** descartándolo si este falla.
 - ▣ **Verificación de versión:** descarta si no es correcta.
 - ▣ **Autenticación:**
 - La entidad de protocolo pasa al servicio de autenticación: PDU, nombre de comunidad y puertos de origen y destino.
 - Si falla la autenticación se le comunica la entidad de protocolo; ésta genera un trap y descarta el mensaje.
 - Si es auténtico, la entidad de autenticación devuelve a la entidad de protocolo la PDU con el formato ASN.1 de RFC 1157.
 - En la práctica el servicio de autenticación comprueba que el nombre de la comunidad autoriza la recepción del mensaje desde la entidad SNMP fuente.
 - ▣ **Chequea la sintaxis de la PDU** descartando la PDU si falla.
 - ▣ Selecciona la **política de acceso** adecuada según la comunidad.

- **GetRequest (gestor-agente) - GetResponse (agente-gestor)**
- Request-id permite correlar petición con respuesta y detectar duplicados (UDP no fiable).
- Variable-bindings permite multiplexar varias peticiones en un mismo mensaje.
- La respuesta debe ser atómica, es decir, si una petición falla:
 - ▣ No se devuelve el resultado de ninguna.
 - ▣ Error-index apunta al name (variable-bindings) que ha producido el fallo. Ej: si falla el segundo name, error-index = 2.
- Errores en la respuesta (error-status):
 - ▣ noSuchName: OID no referencia a una instancia o no se encuentra en la vista.
 - ▣ tooBig: el tamaño del mensaje de respuesta no cabe en el tamaño máximo de datos UDP impuesto por el sistema.
 - ▣ genErr: cualquier otro motivo.

- ❑ **GetNextRequest (gestor-agente) - GetResponse (agente-gestor)**
- ❑ Se procesa igual que GetRequest.
- ❑ Operación atómica.
- ❑ Devuelve el valor de la siguiente instancia en orden lexicográfico:
 - ❑ OID, no instancia: ejemplo (1.3.6.1.2.1.4.3)
 - Si existe en la vista devuelve la instancia OID.0
 - Si no existe, valor de la siguiente instancia NextOID.0
 - ❑ Instancia OID: ejemplo (1.3.6.1.2.1.4.3.0)
 - Valor de la siguiente instancia: NextOID.0
 - ❑ OID de tabla o Entry de tabla: ejemplo (ifTable o ifTable.1)
 - Valor de la primera instancia de la tabla.
 - ❑ OID columnar: ejemplo (ifTable.1.1)
 - Valor de la primera instancia de la columna de la tabla.
 - ❑ Instancia OID columnar: ejemplo (ifTable.1.1.1)
 - Valor de la siguiente instancia en la vista (puede ser de la misma columna, otra columna o fuera de la tabla).
- ❑ Si no existe la siguiente instancia devuelve NoSuchName.

- **SetRequest (gestor-agente) - GetResponse (agente-gestor)**
- Variable-bindings:
 - ▣ En Set contiene name y value de todas las instancias cuyo valor se quiere modificar.
 - ▣ En response:
 - Contiene name y value tras la modificación.
- Si se produce error en alguna de las peticiones del mensaje, no se modifica ninguna y no se devuelve ningún valor.
- Códigos de error:
 - ▣ noSuchName, TooBig, genErr: mismos casos que GetRequest.
 - ▣ badValue: si tiene algún (name,value) inconsistente (tipo o longitud del valor no válidos, fuera de rango...).
 - ▣ En el caso de noSuchName el error puede indicar que el objeto no puede (control de acceso) ser modificado (aunque exista).
 - ▣ Si el objeto es de sólo lectura devuelve noSuchName. No se utiliza readOnly. En SNMPv2 se utiliza un nuevo código denominado notWritable.

- La norma no especifica si crear nuevas filas en una tabla mediante el comando set, entra dentro de la política del agente.

- **Crear fila:** ante esta llamada

*setRequest (ipRouteDest.130.1.3.3= 130.1.3.3, ipRouteMetric. 130.1.3.3= 1,
ipRouteNextHop.130.1.3.3= 130.1.3.5)*

El agente podría **crear una nueva** entrada en la tabla o devolver un mensaje de error.

- **Borrar fila:** para borrar (o invalidar) una fila de una tabla se utiliza el comando (en las tablas en las que esté permitido):

setRequest(ipRouteType.130.1.3.3= invalid)

- **Modificar:** algunas tablas tienen instancias columnares que pueden ser modificadas con SetRequest (definido en las MIBs).

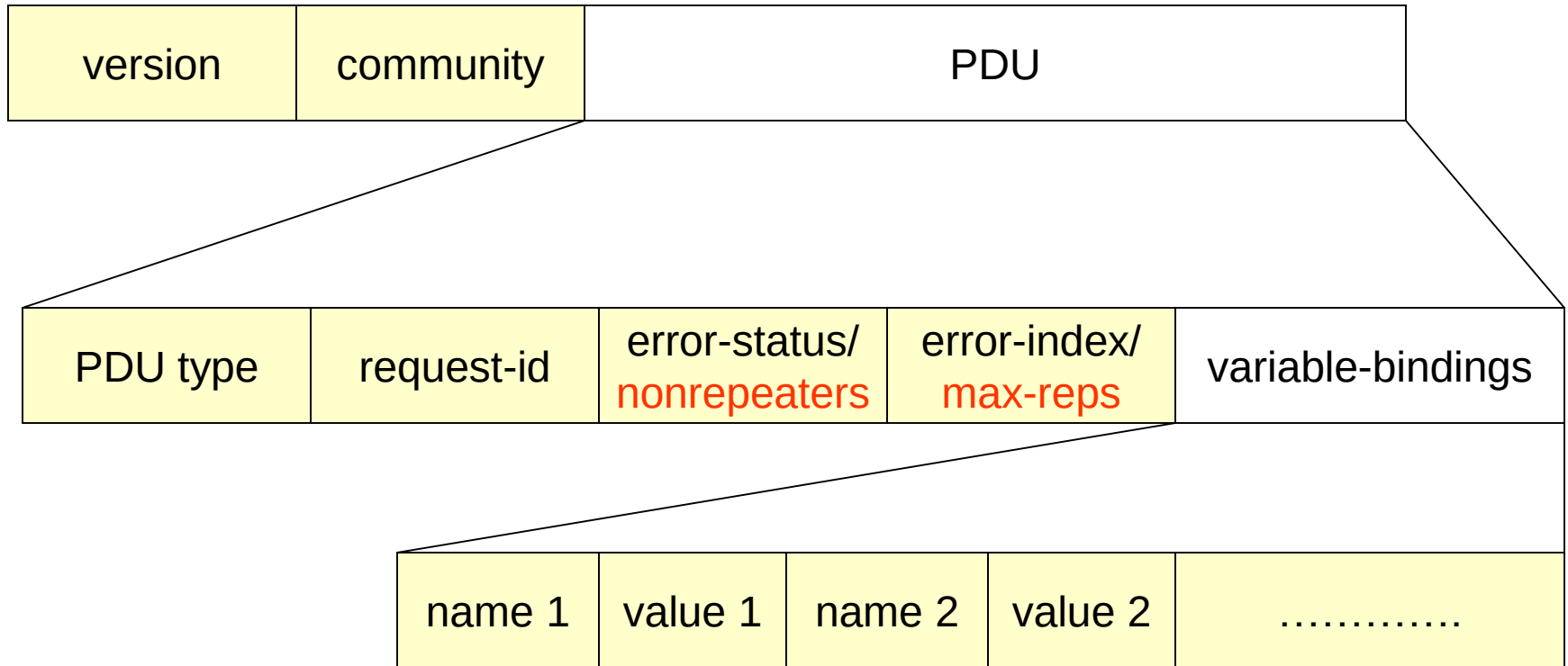
- ❑ **Trap (agente-gestor)**
- ❑ No requiere confirmación: problema si se pierde el mensaje.
- ❑ Motivos de trap:
 - ❑ coldStart (0): reinicio inesperado debido a algún tipo de fallo.
 - ❑ warmStart (1): reinicio en caliente, normalmente programado.
 - ❑ linkDown (2): fallo en uno de los enlaces del sistema. El primer elemento en variablebindings, contiene el valor ifindex del interface.
 - ❑ linkUp (3): uno de los enlaces se ha iniciado. El primer elemento en variablebindings, contiene el valor ifindex del interface.
 - ❑ authenticationFailure (4): se ha recibido un mensaje SNMP con error de autenticación.
 - ❑ egpNeighborLoss (5): se ha perdido la relación con un vecino EGP.
 - ❑ enterpriseSpecific (6): se ha producido algún evento específico de empresa especificado en specific-trap.

- En 1988 se publica la especificación **SNMP**.
 - ▣ Gran aceptación.
 - ▣ Deficiencias funcionales y de seguridad.
- En 1993 se publica una versión mejorada (**SNMPv2**).
 - ▣ Esquema de seguridad complejo (es rechazado).
 - ▣ En 1996 se publica una revisión (RFC1901..1908) que es aceptada (sin seguridad).

- **Estructura:** además de comunicación **agente-gestor** aparece el esquema **gestor-gestor** donde un gestor envía información a otro.
- **Nuevas PDUs:**
 - ▣ **getBulkRequest:** permite solicitar una petición de información múltiple:
 - Objetos con respuesta sencilla del siguiente OID al dado
 - Objetos con respuesta de un nº de OIDs especificado siguientes al dado
 - ▣ **informRequest:** un **gestor** envía un informe a otro con OIDs y sus valores. Tiene el mismo formato que setRequest y se responde con getResponse.
 - ▣ Amplía los códigos de respuesta de error.
- **Nuevos tipos:** Unsigned32, Counter64.

Formato de mensaje SNMPv2

24



Formatos de PDUs SNMPv2 [RFC1905]

25

GetRequest, GetNextRequest, SetRequest, SNMPv2-Trap, InformRequest

PDU type	request-id	0	0	variable-bindings
----------	------------	---	---	-------------------

GetBulkRequest

PDU type	request-id	nonrepeaters	max-repetitions	variable-bindings
----------	------------	--------------	-----------------	-------------------

- **nonrepeaters**: nº de variables en la lista de variable-bindings (las primeras) en que se devuelve el sucesor para cada OID indicado (getnext).
- **max-repetitions**: para el resto de variables en la lista variables-bindings, este parámetro indica el número de variables sucesivas a las especificadas en la lista que son devueltas (getnext).

Response (contestación a todas las PDUs excepto SNMPv2-Trap)

PDU type	request-id	error-status	error-index	variable-bindings
----------	------------	--------------	-------------	-------------------

- En SNMPv2, el formato del trap consiste en una lista de n variable bindings:
 - ▣ 1º : contiene timestamp (sysUpTime.0)
 - ▣ 2º : identifica el trap usando un OID (snmpTrapOID.0) [RFC3418]
 - Ej: coldStart=snmpMIBObjects.snmpTraps.1

SNMP-SMIv2: snmpModules = 1.3.6.1.6.3

RFC3418: snmpMIBObjects = 1.3.6.1.6.3.1.1

snmpTraps OBJECT IDENTIFIER ::= { snmpMIBObjects 5 }

coldStart NOTIFICATION-TYPE

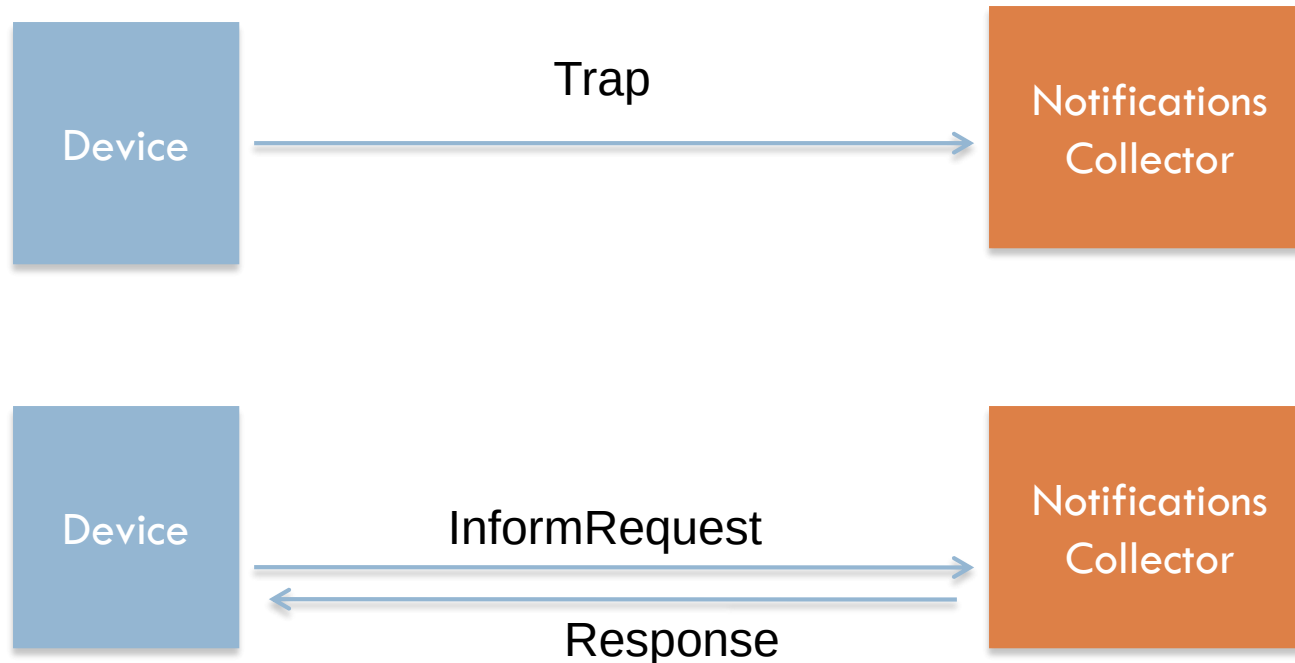
STATUS current

DESCRIPTION "A coldStart trap signifies that..."

::= { snmpTraps 1 }

Notificaciones con respuesta

- Se puede utilizar InformRequest para enviar notificaciones con respuesta



- SNMPv3 (Publicada en 1998).
 - ▣ Se define una arquitectura general de gestión SNMP.
 - Dos capas:
 - Aplicación: gestión de PDUs
 - Motor SNMP (snmpEngine): gestión del mensaje
 - ▣ Nuevos servicios de seguridad:
 - Autenticación: USM (user-based security model)
 - Privacidad: USM
 - Control de acceso: VACM (View-based control module)
 - ▣ USM:
 - No se usa comunidad
 - Los usuarios se autentican con una **password**, a partir de la cual se generan claves de cifrado