

# Codificación SNMP

Gestión de Redes

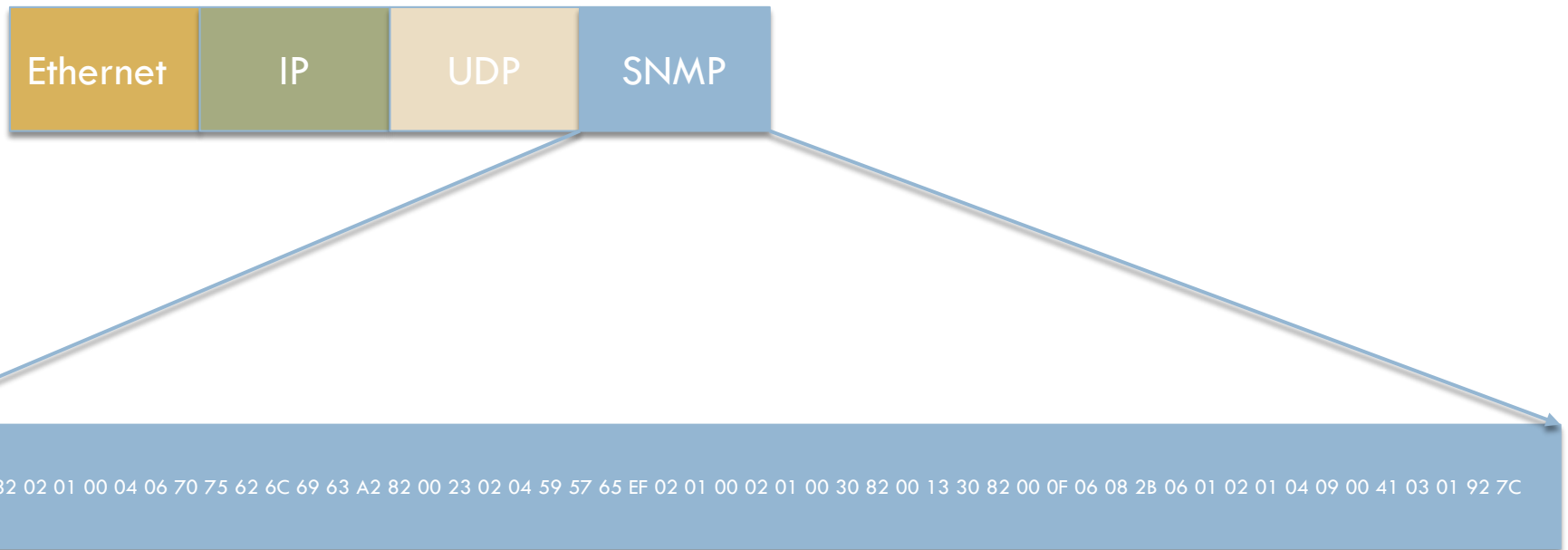
# ¿Qué es esto?

2

30 82 00 32 02 01 00 04 06 70 75 62 6C 69  
63 A2 82 00 23 02 04 59 57 65 EF 02 01 00  
02 01 00 30 82 00 13 30 82 00 0F 06 08 2B  
06 01 02 01 04 09 00 41 03 01 92 7C

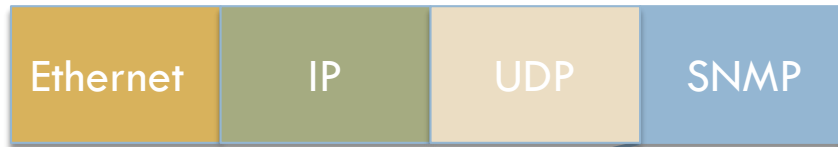
# ¿Qué es esto?

3

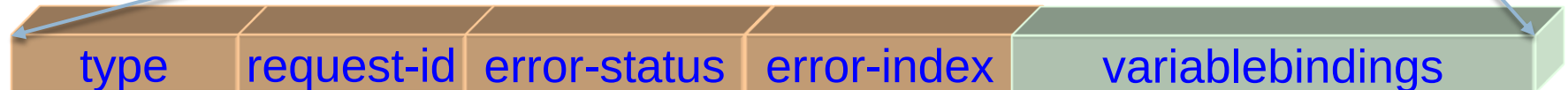


# ¿Qué es esto?

4



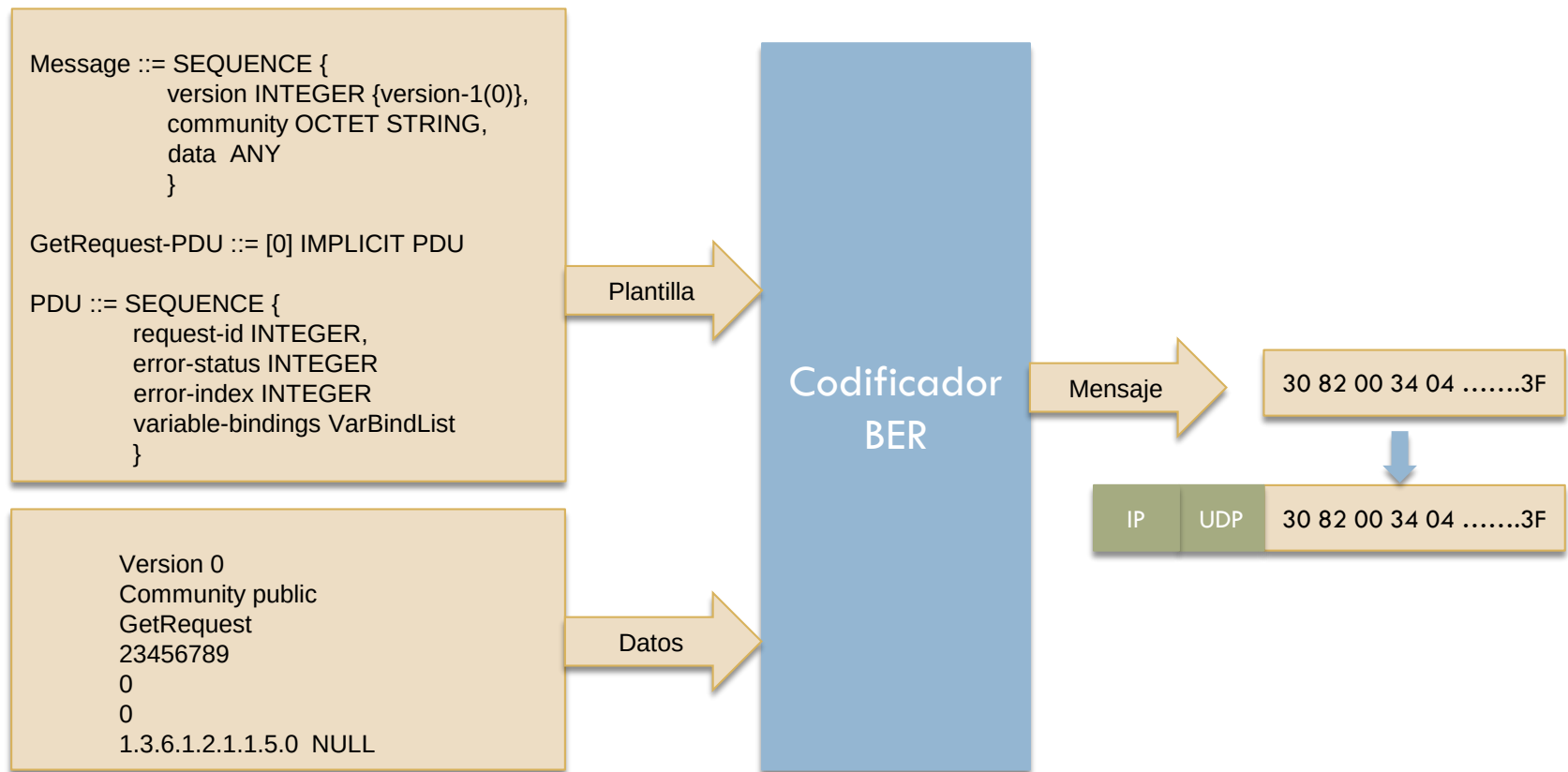
30 82 00 32 02 01 00 04 06 70 75 62 6C 69 63 A2 82 00 23 02 04 59 57 65 EF 02 01 00 02 01 00 30 82 00 13 30 82 00 0F 06 08 2B 06 01 02 01 04 09 00 41 03 01 92 7C



# Codificación del mensaje

5

- El formato está declarado en ASN.1 (RFC1157).
- El mensaje se codifica con BER antes de enviarlo.



# Reglas de codificación BER

6

- BER (Basic Encoding Rules) define cómo codificar los valores definidos en ASN.1 para ser transmitidos.
- La codificación de cada campo consta de tres o cuatro campos:
  - ▣ Tipo: etiqueta que identifica el valor.
  - ▣ Longitud: del valor en octetos.
  - ▣ Valor (OPCIONAL): codificado según el tipo.
  - ▣ Marca de fin de valor (OPCIONAL): si la longitud es indefinida.

# Notación de representación

7

- Valores expresados en hexadecimal, cada octeto se separa por un espacio en blanco.

```
TT LL comentario  
VV VV VV VV comentario
```

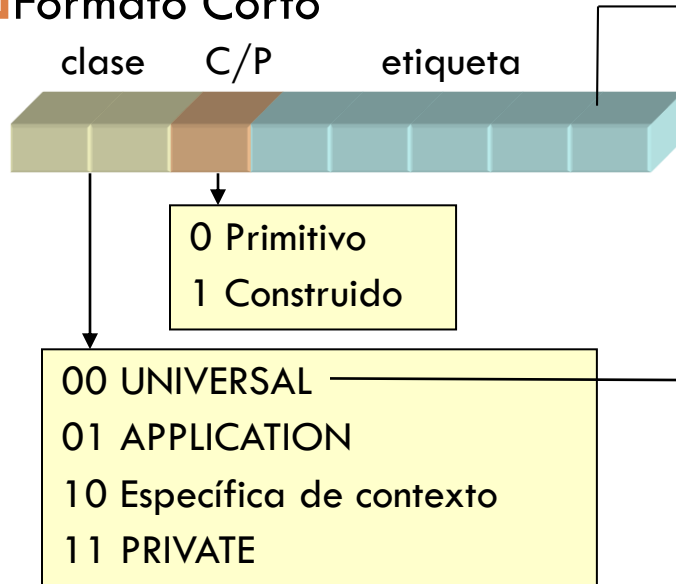
- Se pueden dar codificaciones de tipos contruidos (estructuras o etiquetas explicitas).

```
TT LL tipo construido, después se codifica otro tipo  
TT LL tipo primitivo, después viene el valor  
VV VV VV VV  
TT LL tipo primitivo, después viene el valor  
VV VV
```

# Codificación de tipo (TT)

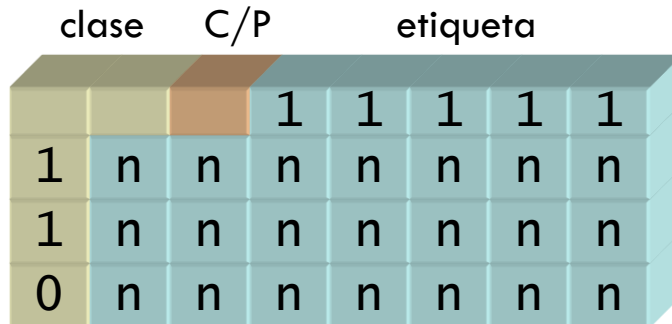
8

## Formato Corto



| Etiquetas Universales |    |       |
|-----------------------|----|-------|
| BOOLEAN               | 1  | 00001 |
| INTEGER               | 2  | 00010 |
| BIT STRING            | 3  | 00011 |
| OCTET STRING          | 4  | 00100 |
| NULL                  | 5  | 00101 |
| OBJECT IDENTIFIER     | 6  | 00110 |
| ObjectDescriptor      | 7  | 00111 |
| EXTERNAL              | 8  | 01000 |
| REAL                  | 9  | 01001 |
| ENUMERATED            | 10 | 01010 |
| SEQUENCE [OF]         | 16 | 10000 |
| SET [OF]              | 17 | 10001 |
| NumericString         | 18 | 10010 |
| PrintableString       | 19 | 10011 |
| TeletextString        | 20 | 10100 |
| VideoString           | 21 | 10101 |
| IA5String             | 22 | 10110 |
| UTCTime               | 23 | 10111 |
| GeneralizedTime       | 24 | 11000 |
| GraphicString         | 25 | 11001 |
| visibleString         | 26 | 11010 |
| GeneralString         | 27 | 11011 |

## Formato Largo: etiquetas > 30

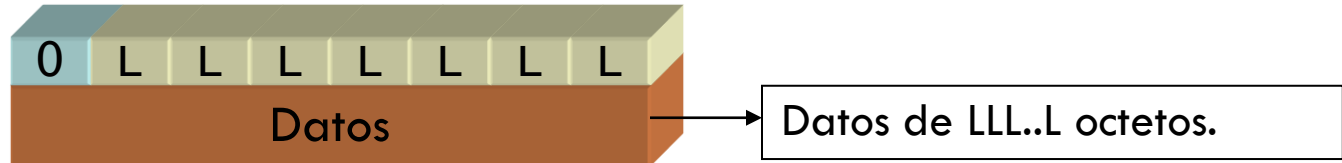


Etiqueta = nnnn.....nnnn

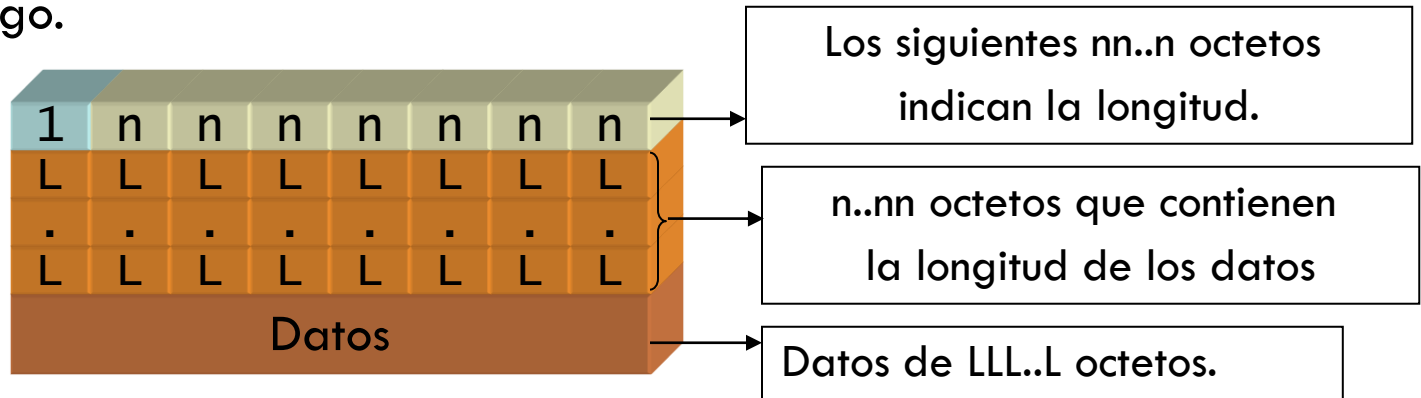
# Codificación de longitud (LL)

9

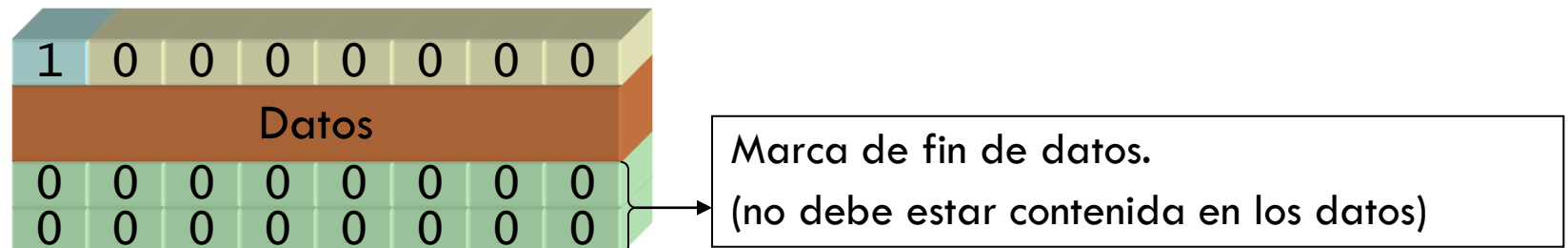
## Formato Corto.



## Formato Largo.



## Formato Indefinido.



# Codificación de valor (VV)

10

## □ La codificación del valor depende del tipo.

A continuación se muestran ejemplos concretos de codificación de valores, según el tipo:

- ▣ BOOLEAN
- ▣ INTEGER
- ▣ REAL
- ▣ BIT STRING
- ▣ OCTET STRING
- ▣ NULL
- ▣ OBJECT IDENTIFIER

# Codificación de BOOLEAN

11

- Se codifica en un octeto:
  - ▣ FALSE: 0
  - ▣ TRUE: cualquier otro valor
- Ejemplo: codificación del valor FALSE

|          |   |   |   |   |   |   |   |   |
|----------|---|---|---|---|---|---|---|---|
| Tipo     | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Longitud | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Valor    | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

01 01 [UNIVERSAL 1], LONGITUD 1  
00 valor FALSE

# Codificación de INTEGER

12

- Se codifica en complemento a dos utilizando tantos octetos como sea necesario.
- Ejemplo: codificación del valor  $-31$  (11100001).

|          |   |   |   |   |   |   |   |   |
|----------|---|---|---|---|---|---|---|---|
| Tipo     | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Longitud | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Valor    | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |

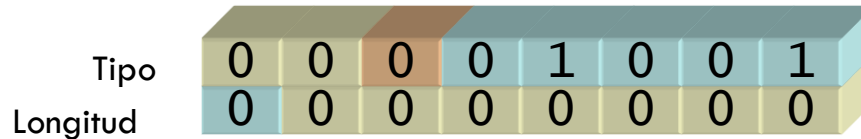
02 01 [UNIVERSAL 2], LONGITUD 1  
E1 valor -31

- ENUMERATED se codifica como un entero, aunque utiliza su propia etiqueta [UNIVERSAL 19].

# Codificación de REAL

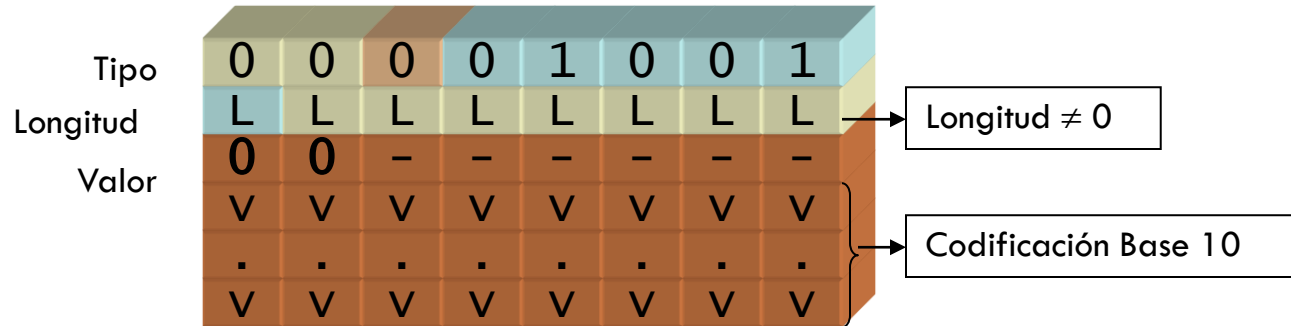
13

- Si el **valor** es **0** no se codifica, el campo de longitud vale 0.

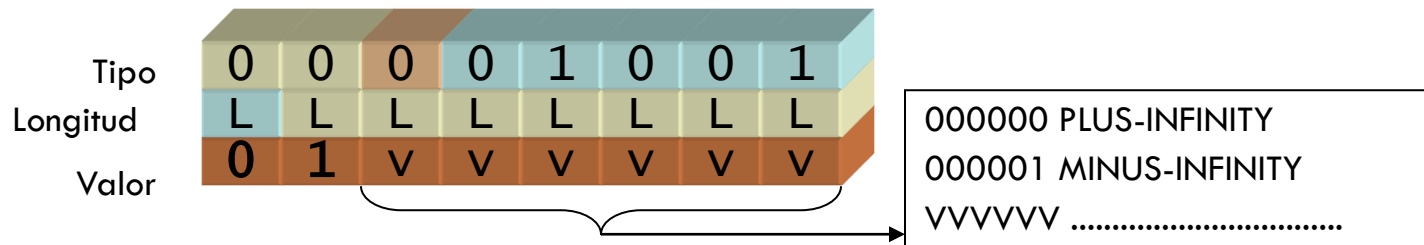


- Para el resto de casos el primer byte de valor indica el esquema de codificación:

- Comienza con 00: codificación en **base 10 según ISO 6093**.



- Comienza con 01: codificación de **valores especiales**.



# Codificación de REAL (cont)

14

Comienza con 1: **codificación en base 2, 8, ó 16.**

|          |   |   |   |   |   |   |   |   |
|----------|---|---|---|---|---|---|---|---|
| Tipo     | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| Longitud | L | L | L | L | L | L | L | L |
| Valor    | 1 | S | B | B | F | F | Q | Q |
|          | E | E | E | E | E | E | E | E |
|          | • | • | • | • | • | • | • | • |
|          | E | E | E | E | E | E | E | E |
|          | N | N | N | N | N | N | N | N |
|          | • | • | • | • | • | • | • | • |
|          | N | N | N | N | N | N | N | N |

Longitud  $\neq 0$

El valor se forma con la expresión:

$$S * N * 2^F * B^E$$

Donde  $N * 2^F$  es la mantisa.

S: signo de la mantisa, 0 positivo y 1 negativo.

N: número para calcular la mantisa.

F: factor para calcular la mantisa, 00 = 0, 01 = 1, 10 = 2 y 11 = 3.

B: base, 00 = 2, 01 = 8 y 10 = 16.

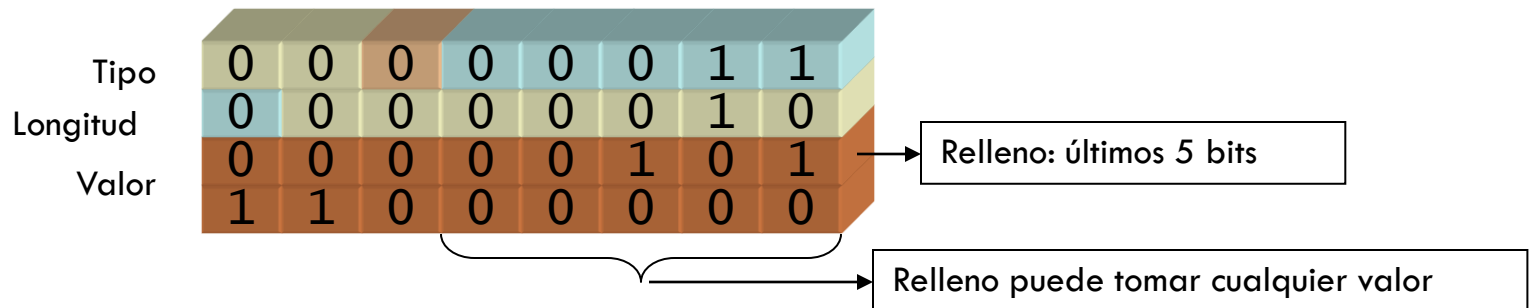
Q: tamaño del campo E en octetos, 00 = 1, 01 = 2, 10 = 3 y 11 = en el siguiente octeto.

E: exponente codificado como entero en complemento a 2.

# Codificación de BIT STRING

15

- ❑ La longitud debe ser múltiplo de 8 bits, por lo tanto incluye un campo que indica los bits que son de relleno.
- ❑ Ejemplo: codificación del valor 110.



03 02 [UNIVERSAL 3], LONGITUD 2  
05 c0 valor 110

# Codificación de OCTET STRING y derivados

16

- ❑ Se codifican los octetos tal como son definidos.
- ❑ Ejemplo: codificación del valor 25 3F.

|          |   |   |   |   |   |   |   |   |
|----------|---|---|---|---|---|---|---|---|
| Tipo     | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Longitud | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Valor    | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
|          | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

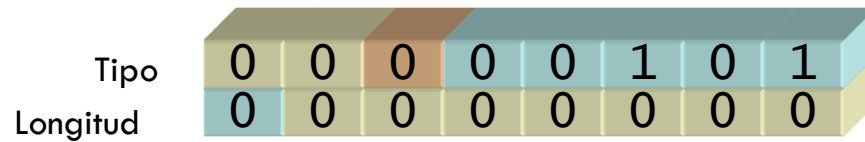
04 02 [UNIVERSAL 4], LONGITUD 2  
25 3F valor 25 3F

Los **tipos predefinidos** derivados de OCTET STRING (IA5String, NumericString, PrintableString...) se codifican de la misma forma, lo único que cambia es el **valor de la etiqueta**.

# Codificación de NULL

17

- ❑ No tiene valor, por lo tanto se codifica con longitud 0.



05 00 [UNIVERSAL 5], LONGITUD 0

# Codificación de OBJECT IDENTIFIER

18

- ❑ Secuencia de enteros:  $N_1...N_k$  que se codifican:
  - ❑  $N_1$  y  $N_2$  se codifican como el entero:  $40*N_1+N_2$
  - ❑ El resto  $N_3.....N_k$  se codifica cada entero por separado.
- ❑ La codificación de cada entero se realiza:
  - ❑ Se codifica en octetos por orden de mayor a menor peso.
  - ❑ El bit de mayor peso de cada octeto indica:
    - ❑ 0 = último octeto del entero.
    - ❑ 1 = vienen más enteros.

- ❑ Ejemplo: 1.3.6.1.4.1.1243.1.0

$40*1+3=43$ : 00101011

6: 00000110

1: 00000001

4: 00000100

1: 00000001

1243: 10001001 01011011

1 : 00000001

0 : 00000000

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

06 09

2B 06 01 04 01 89 5B 01 00

# Codificación de Estructuras

19

- ❑ SEQUENCE, SEQUENCE OF: se codifica cada uno de los campos dentro del tipo construido de etiqueta 16.
- ❑ SET, SET OF: igual con etiqueta 17 (sin orden determinado).
- ❑ CHOICE: no tiene asignada etiqueta ya que se codifica sólo el tipo seleccionado.
- ❑ Ejemplo:

## Sintaxis:

```
DatosPersonales ::= SEQUENCE {  
    nombre IA5String,  
    dni      NumericString  
}
```

## Valores:

```
DatosPersonales ::= {  
    nombre "Narciso Yepes",  
    dni "19135467"  
}
```

```
30 19 [UNIVERSAL 16], construido, longitud 25  
  16 0D [UNIVERSAL 22], primitivo, longitud 13  
    4E 61 72 63 69 73 6F 20 59 65 70 65 73  
  12 08 [UNIVERSAL 18], primitivo, longitud 8  
    31 39 31 33 35 34 36 37
```

# Ejemplo 1

20

## Sintaxis:

```
DatosPersonales ::= SEQUENCE {  
    nombre IA5String,  
    domicilio IA5String,  
    edad     INTEGER  
}
```

## Valores:

```
DatosPersonales ::= {  
    nombre "Antonio Sanz",  
    dni "C/ Santa Cruz"  
    Edad 25  
}
```

```
30 20 [UNIVERSAL 16], construido, longitud 32  
    16 0C [UNIVERSAL 22], primitivo, longitud 12  
        "Antonio Sanz"  
    16 0D [UNIVERSAL 22], primitivo, longitud 13  
        "C/ Santa Cruz"  
    02 01 [UNIVERSAL 2], primitivo, longitud 1  
        19
```

# Ejemplo 2

21

Le ponemos etiquetas explicitas al ejercicio anterior

```
DatosPersonales ::= [APPLICATION 1] SEQUENCE {  
    nombre [0] IA5String,  
    domicilio [1] IA5String,  
    edad [2] INTEGER  
}
```

```
61 28 [APPLICATION 1], construido, longitud 40  
  30 26 [UNIVERSAL 16], construido, longitud 38  
    A0 0E [0], construido, longitud 14  
      16 0C [UNIVERSAL 22], primitivo, longitud 12  
        "Antonio Sanz"  
    A1 0F [1], construido, longitud 15  
      16 0D [UNIVERSAL 22], primitivo, longitud 13  
        "C/ Santa Cruz"  
    A2 03 [2], construido, longitud 3  
      02 01 [UNIVERSAL 2], primitivo, longitud 1  
        19
```

# Ejemplo 3

22

Añadimos IMPLICIT en el ejercicio anterior.

```
DatosPersonales ::= [APPLICATION 1] IMPLICIT SEQUENCE {  
  nombre [0] IMPLICIT IA5String,  
  domicilio [1] IMPLICIT IA5String,  
  edad [2] IMPLICIT INTEGER  
}
```

```
61 20 [APPLICATION 1], construido, longitud 32  
  80 0C [0], primitivo, longitud 12  
    "Antonio Sanz"  
  81 0D [1], primitivo, longitud 13  
    "C/ Santa Cruz"  
  82 01 [2], primitivo, longitud 1  
    19
```

# Declaración del formato del mensaje SNMP

23

- El formato está declarado en ASN.1 (RFC1157).
- El mensaje se codifica con BER antes de enviarlo.

```
RFC1157-SNMP DEFINITIONS ::= BEGIN
```

```
IMPORTS
```

```
    ObjectName, ObjectSyntax, NetworkAddress, IpAddress, TimeTicks
```

```
FROM RFC1155-SMI;
```

```
Message ::= SEQUENCE {  
    version INTEGER {version-1(0)}, -- Version-1 for this RFC  
    community OCTET STRING,        -- Community name  
    data ANY -- E.g., PDUs if trivial authentication is being used  
}
```

# Declaración del formato del mensaje SNMP

24

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

```
GetRequest-PDU ::= [0] IMPLICIT PDU  
GetNextRequest-PDU ::= [1] IMPLICIT PDU  
GetResponse-PDU ::= [2] IMPLICIT PDU  
SetRequest-PDU ::= [3] IMPLICIT PDU
```

```
PDU ::= SEQUENCE {  
    request-id INTEGER,  
    error-status INTEGER {noError(0), tooBig(1), noSuchName(2),  
                        badValue(3), readOnly(4), genErr(5)},  
    error-index INTEGER,          -- Sometimes ignored  
    variable-bindings VarBindList -- Values are sometimes ignored  
}
```

# Declaración del formato del mensaje SNMP

25

```
Trap-PDU ::= [4] IMPLICIT SEQUENCE {  
    enterprise OBJECT IDENTIFIER,      -- Type of object generating  
                                       -- trap, see sysObjectID in [5]  
    agent-addr NetworkAddress,         -- Address of object generating trap  
    generic-trap INTEGER {             -- Generic trap type  
        coldStart(0), warmStart(1), linkDown(2), linkUp(3),  
        authenticationFailure(4), egpNeighborLoss(5), enterpriseSpecific(6)  
    },  
  
    specific-trap INTEGER,              -- Specific code, present even if generic-trap  
                                       -- is not enterpriseSpecific  
    time-stamp TimeTicks,               -- Time elapsed between the last(re)initialization  
                                       -- of the network entity and the generation of  
                                       -- the trap  
    variable-bindings VarBindList      -- "interesting" information  
}
```

# Declaración del formato del mensaje SNMP

26

```
VarBind ::= SEQUENCE {  
    name ObjectName,  
    value ObjectSyntax  
}  
VarBindList ::= SEQUENCE OF VarBind  
END
```