GESTIÓN DE REDES

PARTE II

Arquitectura de Gestión de Internet

GESTIÓN INTERNET

2.1 Introducción

El desarrollo de **SNMP** ha estado ligado al de **TCP/IP**.

TCP/IP nace con la ARPANET desarrollada por el DoD. Sus estándares están publicados en RFCs. Al principio no tenía protocolos de gestión de red, pero posteriormente nació el ICMP que permitía enviar mensajes de control entre máquinas con IP. --> Comunicación echo/echo-reply (e.g. PING).

Posteriormente las necesidades de gestión de red se incrementaron y la tendencia fue a tres aproximaciones (Marzo de 1987):

- **HEMS** *(High-level Entity-Management System)*: basado en el *Host-Monitoring Protocol (HMP)*.
- **SNMP** *(Simple Network Management Protocol)*: basado en el *Simple Gateway-Monitoring Protocol (SGMP).*
- **CMOT** (CMIP over TCP/IP).

En la revisión de Febrero de 1988 se llegó a la conclusión de que a corto plazo se utilizaría SNMP mientras que a largo plazo se necesitarían aproximaciones CMOT.

Se propuso la estandarización de la información de gestión cuya estructuración puede ser utilizada por las

dos aproximaciones.

SNMP es fácil de implementar y estuvo rápidamente disponible en los equipos.

Al convertirse TCP/IP en el estándar de facto en redes de ordenadores, SNMP se ha convertido en otro estándar de facto, pero con muchas limitaciones.

En Agosto de 1988 se publican las primeras recomendaciones: SNMP, SMI y MIB.

Son revisadas en 1991 las recomendaciones SNMP y MIB, dando lugar, esta última, a la recomendación MIB-II.

A partir de esta fecha comienza el desarrollo de MIBs particulares por parte de los fabricantes.

En Mayo de 1993 aparece SNMPv2 que pretende suplir las deficiencias de SNMP en cuanto a seguridad y funcionalidad se refiere.

El marco de trabajo de SNMP está basado en tres documentos:

- **Structure of Management Information (SMI)**.- RFC 1155.
- **Management Information Base (MIB).**- RFC 1213.

- **Simple Network Management Protocol (SNMP)**.- RFC 1157.

INTERNET => RED DE DATOS => GESTIÓN INTERNET => SNMP

RED TELECOMUNICACIONES => GESTIÓN OSI => CMIP

2.2 Información de Gestión en SNMP

2.2.1 ASN.1 *(Abstract Syntax Notation One)*

Un protocolo como SNMP nos permite llegar al proceso SNMP agente, pero el problema está en llegar a los **recursos que controla el agente**.

Para ello se establece una **MIB** que contiene una **representación estandarizada del objeto gestionado**.

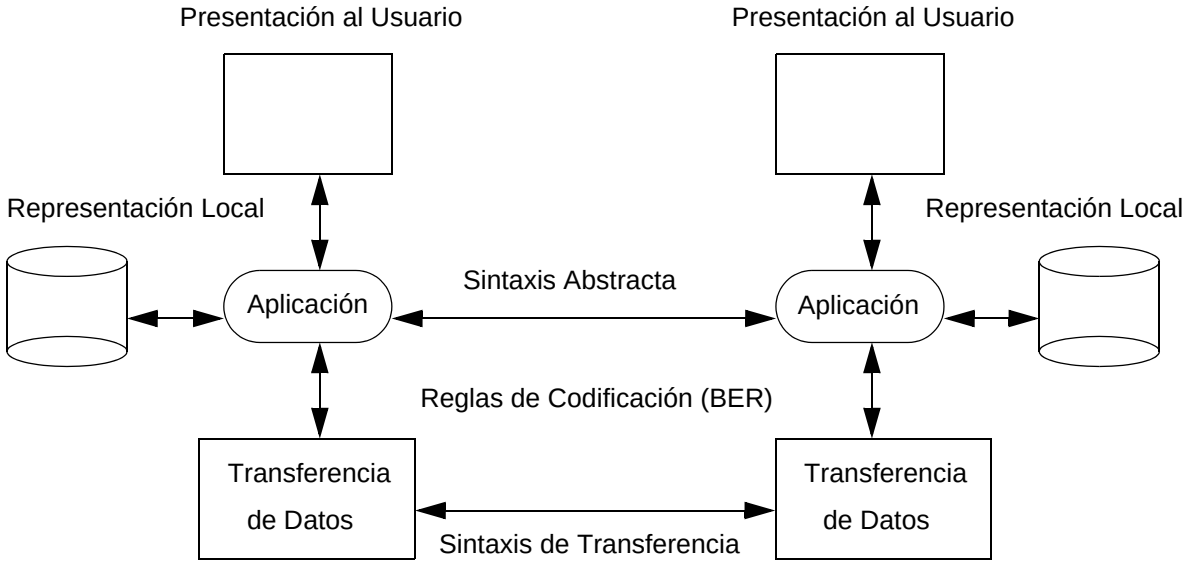
Para definir la sintaxis de estos objetos se utiliza la **Abstract Syntax Notation One (ASN.1).**

Esta sintaxis es necesaria para utilizar una **representación de datos común para el intercambio entre sistemas** y, dentro de un sistema, intercambio de

datos entre aplicaciones que utilizan cada una su representación particular de datos.

Es necesario resaltar los siguientes conceptos:

- ***Sintaxis abstracta,*** define una estructura de datos independiente de la técnica de codificación usada.
- ***Tipos de datos*****,** conjuntos de valores nombrados.
- ***Codificación*****,** secuencia de octetos que representan un valor de un dato.
- ***Sintáxis de transferencia*****,** representación de los datos mediante patrones de bits para su transmisión entre entidades de presentación.
- ***Reglas de codificación*****,** mapeo de una sintaxis en otra.
-



2.2.2 Definición de módulos.

Las estructuras de datos se representan mediante módulos:

```
<module_reference> DEFINITIONS ::=
BEGIN
  EXPORTS
  IMPORTS
  AsignementList
END
```

2.2.3 Convenciones léxicas

- Cuando se introducen varios espacios en blanco, sólo se considera uno.
- Los comentarios deben empezar por --
- Los identificadores, nombre de tipos y módulos están compuestos por letras (mayúsculas y minúsculas), dígitos y guiones.
- Un identificador debe comenzar por una letra minúscula.
- Un tipo o nombre de módulo por una letra mayúscula.
- Las palabras clave o nativas de la gramática, todas en mayúsculas.

2.2.4 Tipos de datos

Se distinguen los siguientes cuatro tipos de datos:

- Tipos simples: Tipos atómicos, sin componentes.
- Tipos estructurados: Tipos compuestos por componentes.
- Etiquetados: Tipos devivados de otros tipos.
- Otros: Esta categoría incluye los tipos CHOICE y ANY.

Todos los tipos de dato ASN.1, a excepción de CHOICE y ANY, tienen asociados una etiqueta. Una etiqueta consiste en un nombre de clase y un número entero no negativo.

Existen cuatro clases de etiqueta diferentes:

- UNIVERSAL (00): Tipos de uso común definidos en el estándar.
- APPLICATION (01): Específicos de un contexto de aplicación.
- CONTEXT-SPECIFIC (10): Sirven para evitar ambigüedades en ciertos contextos.
- PRIVATE (11): Definidos por el usuario.
-

Un tipo de dato es únivocamente identificado por su etiqueta.

Tipos simples: Identifican directamente un conjunto de valores.

- `INTEGER`: Números cardinales.
- `OCTET STRING`: 0 o más octetos. Cada octeto un valor entre 0 y 255.
- `OBJECT IDENTIFIER`: Identificación de objetos.
- `NULL`: Tipo nulo. No se usa en el marco de gestión.

Tipos estructurados: Tipos compuestos.

- `SEQUENCE:` Para hacer listas
- `SEQUENCE OF`: Para hacer tablas.

Notación de los tipos estructurados:

```
SequenceType ::= SEQUENCE {Element-
 TypeList} | SEQUENCE {}
ElementTypeList ::= ElementType | Ele-
 mentTypeList, ElementType
ElementType ::=
    NamedType          |
    NamedType OPTIONAL|
    NamedType DEFAULT Value|
    COMPONENTS OF Type

SequenceOfType ::= SEQUENCE OF Type |
 SEQUENCE
```

SET y SET OF son equivalentes pero sin que el orden tenga relevancia.

Tipos CHOICE y ANY

- Son tipos de datos sin etiqueta.
- Son tipos de datos especiales que permiten una asignación "*run-time*" de tipos según el valor de que se trate.
- CHOICE es un lista de tipos alternativos que pueden asignarse a un valor. Es la unión de los conjuntos de valores de todos los tipos listados.

```
ChoiceType ::= CHOICE {Alternative-
TypeList}
AlternativeTypeList ::= NamedType |
      AlternetiveTypeList, NamedType
```

- ANY decribe un valor arbitrario de un tipo arbitrario.

```
AnyType ::= ANY
```

Tipos etiquetados:

- Permiten definir un nuevo tipo etiquetando uno existente. El nuevo tipo es isomorfo al existente pero distinto, es decir, puede ser distinguido en los esquemas de codificación.

- Se utilizan para distinguir tipos dentro de una aplicación. Bien, para definir tipos de nombres diferentes que son esencialmente el mismo nombre. O bien, para solucionar ambigüedades en tipos estructurados.
- Se construyen con la etiqueta y un número entero.
- Existen dos tipos de etiquetado, el implícito y el explícito.
- El etiquetado implícito simplifica la codificación, y el etiquetado explícito es necesario para evitar ambigüedades si la etiqueta del tipo subyaciente es indeterminado (por ejemplo, CHOICE y ANY).

```
Opaque ::=
      [APPLICATION 4]
         IMPLICIT OCTET STRING
```

2.2.5 Subtipado

Refinan la semántica de un tipo ya existente. Representan subconjuntos del conjunto de valores determinado por el tipo padre.

Existen varias formas de definir este subconjunto:

Valor simple:

```
SmallPrime ::= INTEGER( 2 | 3 | 5 | 7 | 11 ).
Semana ::= ENUMERATED { Lunes (1), Martes (2),
                        Miercoles (3),
                        Jueves (4),
                        Viernes (5),
```

```
                                Sabado (6),
                                Domingo (7) }
        First-quarter ::= Semana (Lunes | Martes)
         Second-quarter ::= Semana (Miercoles | Jueves)
```

Subtipo contenido:

```
      First-half ::= Semana( INCLUDES First-quarter |
                      INCLUDES Seconde-quarter )
```

Rango de valores:

```
       PositiveInteger ::= INTEGER (0<..MAX)
       NegativeInteger ::= INTEGER (MINUS-INFITY..<0)
```

Alfabeto permitido:

```
       DigitString ::= IA5String (FROM
                         (“0” | “1” | “2” | “3” ))
```

Restricción del tamaño:

```
       ItlDataNumber ::= DigitString (SIZE (5..10))
```

Tipado interior:

Aplicable sobre los tipos definidos con SEQUENCE y SET. Permiten añadir restricciones y condiciones a un conjunto de valores para determinar el subconjunto.

2.2.6 Macros

Permiten extender la sintaxis de ASN.1, dotando de

información semántica a los tipos.

Debemos distinguir **Notación, Definición** e **Instancias.**

Una función de definición de macros es un supertipo generando una clase de instancias macro que funcionan como el tipo básico ASN.1.

Puede ser vista como una plantilla que se usa para generar un conjunto de tipos y valores.

Notación:

```
<macroname> MACRO ::=
BEGIN
     TYPE NOTATION ::= <new_type_syntax>
     VALUE NOTATION ::= <new_value_syntax>
     <supporting_productions>
END
```

Ventajas de las macros:

- Permite la definición de familias de tipos.
- Pueden incluirse comentarios y semántica asociada a los tipos.
- No se necesario el isomorfismo entre valor y tipo.

2.2.7 Reglas de codificación:

Describen los métodos para codificar los valores de cada

tipo ASN.1 como una cadena de octetos (sintaxis de transferencia).

La estructura de codificación está compuesta de tripletas [type, length, value] que se aplican de forma recursiva.

Sintaxis de Transferencia:

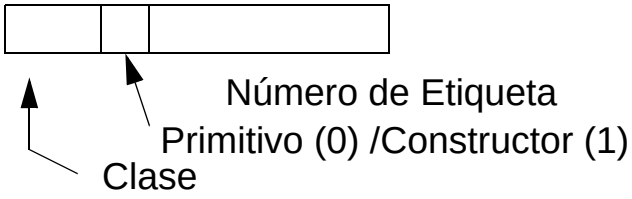
Existen otras, pero la más popular es la conocida como ***Basic Encoding Rules*, BER.**

Cada valor se codifica con una tripleta IDENTIFICACIÓN + LONGITUD + CONTENIDO, donde el contenido puede codificarse como otra tripleta si es un tipo compuesto.

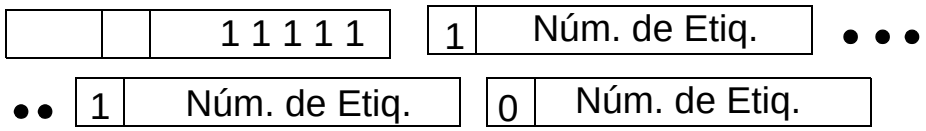
La codificación del contenido depende del tipo.

GESTIÓN INTERNET

IDENTIFICACIÓN

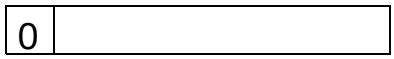


Formato largo (Número>30)

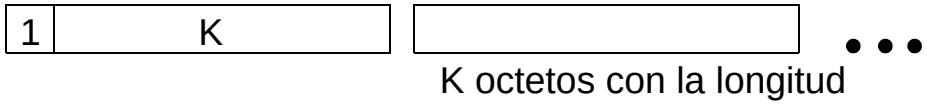


LONGITUD

Formato corto definido

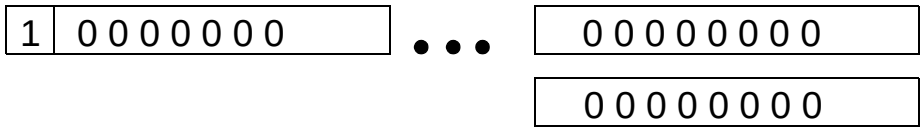


Formato largo definido



K octetos con la longitud

Formato indefinido (Sólo tipos compuestos no strings)



CONTENIDO

Dependiente del valor

TABLE B.2 Universal Class Tag Assignments

Tag	Type Name	Set of Values
		BASIC TYPES
UNIVERSAL 1	BOOLEAN	TRUE or FALSE
UNIVERSAL 2	INTEGER	The positive and negative whole numbers, including zero
UNIVERSAL 3	BIT STRING	A sequence of zero or more bits
UNIVERSAL 4	OCTET STRING	A sequence of zero or more octets
UNIVERSAL 9	REAL	Real numbers
UNIVERSAL 10	ENUMERATED	An explicit list of integer values that an instance of a data type may take
		OBJECT TYPES
UNIVERSAL 6	OBJECT IDENTIFIER	The set of values associated with information objects allocated by this standard
UNIVERSAL 7	Object descriptor	Human-readable text providing a brief description of an information object
		CHARACTER STRING TYPES
UNIVERSAL 18	NumericString	Digits 0 through 9, and the space character
UNIVERSAL 19	PrintableString	Printable characters
UNIVERSAL 20	TeletexString	Character set defined by CCITT Recommendation T.61
UNIVERSAL 21	VideotexString	Set of alphabetic and graphical characters defined by CCITT Recommendations T.100 and T.101
UNIVERSAL 22	IA5String	International alphabet 5 (equivalent to ASCII)
UNIVERSAL 25	GraphicString	Character set defined by ISO 8824
UNIVERSAL 26	VisibleString	Character set defined by ISO 646 (equivalent to ASCII)
UNIVERSAL 27	GeneralString	General character string
		MISCELLANEOUS TYPES
UNIVERSAL 5	NULL	The single value NULL; commonly used where several alternatives are possible but none of them applies
UNIVERSAL 8	EXTERNAL	A type defined in some external document (It need not be one of the valid ASN.1 types.)
UNIVERSAL 23	UTCTime	Consists of the date—specified with a two-digit year, a two-digit month, and a two-digit day—followed by the time—specified in hours, minutes, and optionally seconds—followed by an optional specification of the local time differential from universal time
UNIVERSAL 24	GeneralizedTime	Consists of the date—specified with a four-digit year, a two-digit month, and a two-digit day—followed by the time—specified in hours, minutes, and optionally seconds—followed by an optional specification of the local time differential from universal time
UNIVERSAL 9-15	Reserved	Reserved for addenda to the ASN.1 standard
UNIVERSAL 28-	Reserved	Reserved for addenda to the ASN.1 standard
		STRUCTURED TYPES
UNIVERSAL 16	SEQUENCE and SEQUENCE-OF	Sequence: defined by referencing a fixed, ordered list of types; each value is an ordered list of values, one from each component type Sequence-of: defined by referencing a single existing type; each value is an ordered list of zero or more values of the existing type
UNIVERSAL 17	SET and SET-OF	Set: defined by referencing a fixed, unordered list of types, some of which may be declared optional; each value is an unordered list of values, one from each component type Set-of: defined by referencing a single existing type; each value is an unordered list of zero or more values of the existing type

```
Name:                 John P Smith
Title:                Director
Employee Number:      51
Date of Hire:         17 September 1971
Name of Spouse:       Mary T Smith
Number of Children:   2

Child Information
    Name:             Ralph T Smith
    Date of Birth:    11 November 1957

Child Information
    Name:             Susan B Jones
    Date of Birth:    17 July 1959
```

(a) Informal description of personnel record

```
PersonnelRecord ::= [APPLICATION 0] IMPLICIT SET {
        Name,
        title [0] VisibleString,
        number EmployeeNumber,
        dateOfHire [1] Date,
        nameOfSpouse [2] Name,
        children [3] IMPLICIT SEQUENCE OF ChildInformation DEFAULT {} }

ChildInformation ::= SET {
        Name,
        dateOfBirth [0] Date}

Name ::= [APPLICATION 1] IMPLICIT SEQUENCE {
        givenName VisibleString,
        initial VisibleString,
        familyName VisibleString }

EmployeeNumber ::= [APPLICATION 2] IMPLICIT INTEGER

Date ::= [APPLICATION 3] IMPLICIT VisibleString -- YYYYMMDD
```

(b) ASN.1 description of the record structure

```
                {givenName "John", initial "P", familyName "Smith"},
title           "Director"
number          51
dateOfHire      "19710917"
nameOfSpouse    {givenName "Mary", initial "T", familyName "Smith"},
children
{  {            {givenName "Ralph", initial "T", familyName "Smith"},
   dateOfBirth  "19571111" },
                {givenName "Susan", initial "B", familyName "Jones"},
   dateOfBirth  "19590717" } }
```

(c) ASN.1 description of a record value

FIGURE B.3 Example of use of ASN.1

TABLE B.5 BER Encoding Rules

ASN.1 type	BER encoding rules	Example value	Encoding of example value[1]
BOOLEAN	Primitive. A single octet with a content of zero for FALSE and a nonzero content for TRUE.	TRUE	01 01 FF
INTEGER	Primitive. Two's-complement representation with the minimum number of octets.	–129	02 02 FF 7F
ENUMERATED	Same as the integer value with which it is associated.		
REAL	Primitive. If the value is zero, there are no contents octets. Otherwise, the first contents octet indicates whether the base is 10, 2, 8, or 16, or a special real value (+∞, –∞). If base is 10, first contents octet also specifies one of three ISO 6093 character-encoding schemes for the remaining contents octets. If base is 2, 8, 16, first contents octet also specifies length of exponent, sign of mantissa, and scaling factor to align implied decimal point of mantissa with octet boundary; following the first contents are zero or one additional octets to specify exponent length, followed by octets for the exponent, followed by octets for the mantissa.	0	09 00
BIT STRING	Primitive or constructed. For primitive encoding, the contents consists of an initial octet followed by zero or more subsequent octets. The actual bit string begins with the first bit of the second octet and continues through as many octets as necessary. The first octet indicates how many bits in the last octet are unused. For constructed encoding, the bit string is broken up into substrings; each substring except the last must be a multiple of 8 bits in length. Each substring may be encoded as primitive or constructed, but usually the primitive encoding is used.	"011011100101 110111"	primitive: 03 04 06 6E 5D C0 constructed: 23 09 03 03 00 6E 5D 03 02 06 C0
OCTET STRING	Primitive or constructed. For primitive encoding, the contents octets are identical to the value of the octet string. For constructed encoding, the octet string is broken up into substrings; each substring may be encoded as primitive or constructed, but usually the primitive encoding is used.	01 23 45 67	primitive: 04 04 01 23 45 67 constructed: 24 08 04 02 01 23 04 02 34 56
NULL	Primitive. There are no contents octets.	null	05 00
SEQUENCE	Constructed. The contents octets are the concatenation of the BER encodings of the values of the components of the sequence in order of definition. If the value of a component with the OPTIONAL or DEFAULT qualifier is absent from the sequence, no encoding is included for that component. If the value of a component with the DEFAULT qualifier is	definition: SEQUENCE (INTEGER, INTEGER) value: (3,8)	30 06 02 01 03 02 01 08

	the default value, then encoding of that component may or may not be included.		
SEQUENCE OF	Constructed. The contents octets are the concatenation of the BER encodings of the values of the occurrences in the collection, in the order of occurrence.	definition: SEQUENCE OF (INTEGER) value: (3,8)	30 06 02 01 03 02 01 08
SET	Constructed. The contents octets are the concatenation of the BER encodings of the values of the components of the sequence in any order. If the value of a component with the OPTIONAL or DEFAULT qualifier is absent from the sequence, no encoding is included for that component. If the value of a component with the DEFAULT qualifier is the default value, then encoding of that component may or may not be included.	definition: SET (INTEGER, INTEGER) value: (3,8)	31 06 02 01 03 02 01 08
SET OF	Constructed. The contents octets are the concatenation of the BER encodings of the values of the occurrences in the collection, in any order.	definition: SET OF (INTEGER) value: (3,8)	31 06 02 01 03 02 01 08
CHOICE	Primitive or constructed. The encoding of the CHOICE value is the encoding of the chosen alternative.		
IMPLICIT tag	Primitive or constructed. The encoding of the tag field replaces the underlying tag value. The encoding of the length and contents octets are the same as for the underlying value.	definition: [17] IMPLICIT BOOLEAN value: TRUE	91 01 FF
EXPLICIT tag	Constructed. The contents octets are the complete BER encoding of the underlying value.	definition: [17] BOOLEAN value: TRUE	B1 03 01 01 FF
ANY	Primitive or constructed. The encoding of the ANY value is the encoding of the actual value.		
OBJECT IDENTIFIER	Primitive. The first two components are combined using the formula (X × 40) + Y to form the first subidentifier. Each subsequent component forms the next subidentifier. Each subidentifier is encoded as a non-negative integer using as few 7-bit blocks as possible. The blocks are packed in octets with the first bit of each octet equal to 1 except for the last octet of each subidentifier.	{2 100 3}	06 03 81 34 03
Character String	Primitive or constructed. Encoded as if it had been declared [UNIVERSAL ×] IMPLICIT OCTET STRING.	"Jones"	1A 05 4A 6F 6E 65 73

[1]All encodings are depicted using hexadecimal notation.