2.6 SNMPv2

2.6.1 Antecedentes

Las limitaciones de SNMP llevan a la definición en 1992 de dos nuevos protocolos: S-SNMP (SNMP seguro) que añade seguridad al protocolo SNMP y SMP (Simple Management Protocol) un nuevo protocolo sin características de seguridad pero que potencia las características de funcionalidad y rendimiento de SNMP.

En 1993 se unifican estas dos aproximaciones --> **SNMPv2.**

Ahora se plantea un dilema que divide a los grandes fabricantes: migrar a SNMPv2 o directamente a la familia OSI.

SNMPv2 se diferencia de su predecesor en varios apartados adicionales que podemos englobar en los siguientes campos:

- Características adicionales de la SMI.
- Nuevos protocolos de operación.
- Posibilidad de comunicación gestor-gestor.
- Características adicionales en seguridad.

2.6.2 Características adicionales de la SMI

Se añaden las nuevas sintaxis:

- Integer32: para complemento a 2 en 32 bits.
- UInteger32: números naturales hasta 32 bits.
- BitString: numeración de bits en octetos.
- Counter32: contador más amplio.
- NSAPAddress: direcciones OSI.
- Counter64: si el de 32 bits se llena en menos de una hora.

Se incluye una claúsula adicional UNITS para indicar la unidad de medida asociada con un objeto de medida.

Se pueden definir 4 tipos de acceso:

- Not-accesible.
- Read-only
- Read-write
- Read-create
- Accesible-for-notify

Diferencia las tablas sobre las que se permite al gestor la creación de nuevas filas de las que no permiten dicha operación.

Se define la claúsula AUGMENTS que posibilita la extensión de filas de forma conceptual.

Nuevas características de **semántica:** Macros para añadir información semántica a una MIB. No afecta a la interoperabilidad.

Varios tipos de **estructuras:**

- *Textual Conventions*: Formalizan la semántica de tipos de datos.
- *Module Identity*: Macro con información administrativa sobre la MIB, revisión, persona de contacto, etc...
- *Object Groups*: Macro para formalizar la agrupación de objetos en grupos.
- *Notification Type*: Define información a enviar cuando se produce un evento excepcional.
- *Module-Compliance*: Lo mínimo que un agente debe implementar de esa MIB.
- *Agent-Capabilities*: Lo que realmente implementa un agente.

2.6.3 Nuevos protocolos de operación

Las PDUs de SNMPv2 son encapsuladas en un mensaje. En la cabecera de dicho mensaje se determina cuales serán las políticas de autentificación y autorización.

Se establecen tres tipos de acceso:

- Gestor-Agente Request-Response.

- Gestor-Gestor Request-Response.
- Agente-Gestor unconfirmed.

Se definen dos nuevas operaciones que son: GetBulkRequest e InformRequest.

GetBulkRequest

Permite recuperación de tablas de una manera más eficiente (minimiza el número de intercambios). Actúa de forma parecida a GetNextRequest, pero ahora es posible indicar varios sucesores a uno dado. Pueden especificarse variables de recuperación simple (identico a GetNextRequest) y variables de recuperación múltiple.

Incluye dos nuevos parámetros en su PDU:

- *non-repeaters*: Número de variables de la lista de la que se realizará recuperación simple.
- *max-repetitions*: Número de veces para recuperaciones múltiples.

Ejemplo:

```
 % GetBulkRequest[non-repeaters=1, max-repetitions=2]
 (sysUpTime,ipNetToMediaPhysAddress,ipNetToMediaType)

 % Response ((sysUpTime.0 = “123446“),
 (ipNetToMediaPhysAddress.1.6.2.3.4 =“000010543210“),
 (ipNetToMediaType.1.6.2.3.4 = “dynamic“),
 (ipNetToMediaPhysAd-
dress.1.10.0.0.51=“000010012345“),
 (ipNetToMediaType.1.10.0.0.51 = “static“))
```

InformRequest

Permite establecer comunicaciones entre gestores. Estas comunicaciones se configuran mediante la M2M MIB.

Dos tipos de comunicaciones: por la ocurrencia de algún evento o a petición de algún gestor.

2.6.4 MIB Gestor-Gestor

Consiste en un conjunto de objetos que describen el comportamiento de una entidad SNMPv2 que actúa de gestor. Mediante esta MIB se establecen las comunicaciones entre gestores SNMPv2.

Contiene 2 grupos: alarm y event.

```
OBJECT-TYPE MACRO ::= BEGIN

TYPE NOTATION ::= "SYNTAX" Syntax
                  UnitsPart
                  "MAX-ACCESS" Access
                  "STATUS" Status
                  "DESCRIPTION" Text
                  ReferPart
                  IndexPart
                  DefValPart

VALUE NOTATION ::= value (VALUE ObjectName)

Syntax ::= type(ObjectSyntax) | "BITS" "{" Kibbles "}"

Kibbles ::= Kibble | Kibbles "," Kibble

Kibble ::= identifier "(" nonNegativeNumber ")"

UnitsPart ::= "UNITS" Text | empty

Access ::= "not-accessible" | "accessible-for-notify" | "read-only" | "read-write" | "read-create"

Status ::= "current" | "deprecated" | "obsolete"

ReferPart ::= "REFERENCE" Text | empty

IndexPart ::= "INDEX" "{" IndexTypes "}" | "AUGMENTS" "{" Entry "}" | empty

IndexTypes ::= IndexType | IndexTypes "," IndexType

IndexType ::= "IMPLIED" Index | Index

Index ::= value (indexobject ObjectName)  --use the SYNTAX value of the
                                          --correspondent OBJECT-TYPE invocation

Entry ::= value (entryobject ObjectName)  --use the INDEX value of the
                                          --correspondent OBJECT-TYPE invocation

DefValPart ::= "DEFVAL" "{" value (Defval ObjectSyntax) "}" | empty

--uses the NVT ASCII character set
Text ::= """" string """"

END
```

FIGURE 11.1 SNMPv2 macro for object definition

```
objectName ::= OBJECT IDENTIFIER

ObjectSyntax ::= CHOICE { simple            SimpleSyntax,
                          application-wide  ApplicationSyntax }

SimpleSyntax ::= CHOICE {
                   integer-value    INTEGER {-2147483648..2147483647},  includes Integer32
                   string-value     OCTET STRING (SIZE (0..65535) ),
                   objectID-value   OBJECT IDENTIFIER }

--indistinguishable from INTEGER, but never needs more
--than 32 bits for a two's complement representation
Integer32 ::= [UNIVERSAL 2] IMPLICIT INTEGER (-2147483648..2147483647)

ApplicationSyntax ::= CHOICE { ipAddress-value          IpAddress,
                               counter-value            Counter32,
                               timeticks-value          TimeTicks,
                               arbitrary-value          Opaque,
                               big-counter-value        Counter64,
                               unsigned-integer-value   Unsigned32} --includes Gauge32

IpAddress ::= [APPLICATION 0] IMPLICIT OCTET STRING (SIZE (4) )

Counter32 ::= [APPLICATION 1] IMPLICIT INTEGER (0..4294967295)      --this wraps

Gauge32 ::= [APPLICATION 2] IMPLICIT INTEGER (0..4294967295)        --this doesn't wrap

--unsigned 32-bit quantity indistinguishable from Gauge32
Unsigned32 ::= [APPLICATION 2] IMPLICIT INTEGER (0..4294967295)

--hundredths of seconds since an epoch
TimeTicks ::= [APPLICATION 3] IMPLICIT INTEGER (0..4294967295)

Opaque ::= [APPLICATION 4] IMPLICIT OCTET STRING --for backward-compatibility only

--for counters that wrap in less than one hour with only 32 bits
Counter64 ::= [APPLICATION 6] IMPLICIT INTEGER (0..18446744073709551615)
```

FIGURE 11.2 Definitions associated with SNMPv2 macro for object definition

```
petTable OBJECT-TYPE
   SYNTAX    SEQUENCE OF PetEntry
   MAX-ACCESS  not-accessible
   STATUS      current
   DESCRIPTION
      "The (conceptual) table listing the characteristics of all pets living at this agent."
   ::= { A }

petEntry OBJECT-TYPE
   SYNTAX      PetEntry
   MAX-ACCESS  not-accessible
   STATUS      current
   DESCRIPTION
      "An entry (conceptual row) in the petTable. The Table is indexed by type of animal. Within
      each animal type, individual pets are indexed by a unique numerical sequence number"
   INDEX       { petType, petIndex }
   ::= { petTable 1 }

petEntry ::= SEQUENCE {
             petType              OCTET STRING,
             petIndex             INTEGER,
             petCharacteristic1   INTEGER,
             petCharacteristic2   INTEGER }

petType OBJECT-TYPE
   SYNTAX      OCTET STRING
   MAX-ACCESS  not-accessible
   STATUS      current
   DESCRIPTION
      "An auxiliary variable used to identify instances of the columnar objects in the petTable."
   ::= { petEntry 1 }

petIndex OBJECT-TYPE
   SYNTAX      INTEGER
   MAX-ACCESS  not-accessible
   STATUS      current
   DESCRIPTION
      "An auxiliary variable used to identify instances of the columnar objects in the petTable."
   ::= { petEntry 2 }

petCharacteristic1 OBJECT-TYPE
   SYNTAX      INTEGER
   MAX-ACCESS  read-only
   STATUS      current
   ::= { petEntry 3 }

petCharacteristic2 OBJECT-TYPE
   SYNTAX      INTEGER
   MAX-ACCESS  read-only
   STATUS      current
   ::= { petEntry 4 }
```

FIGURE 11.4 An example of a table for which row creation and deletion are not permitted

```
moreTable OBJECT-TYPE
   SYNTAX   SEQUENCE OF MoreEntry
   MAX-ACCESS  not-accessible
   STATUS      current
   DESCRIPTION
      "A table of additional pet objects."
   ::= { B }

moreEntry OBJECT-TYPE
   SYNTAX      MoreEntry
   MAX-ACCESS  not-accessible
   STATUS      current
   DESCRIPTION
      "Additional objects for a petTable entry."
   AUGMENTS    { petEntry }
   ::= { moreTable 1 }

moreEntry ::= SEQUENCE {
            nameOfVet        OCTET STRING,
            dateOfLastVisit  DateAndTime }

nameOfVet OBJECT-TYPE
   SYNTAX      OCTET STRING
   MAX-ACCESS  read-only
   STATUS      current
   ::= { moreEntry 1 }

dateOfLastVisit OBJECT-TYPE
   SYNTAX      DateAndTime
   MAX-ACCESS  read-only
   STATUS      current
   ::= { moreEntry 2 }
```

FIGURE 11.6 An example of a table that augments the table of Figure 11.4

```
evalSlot OBJECT-TYPE
   SYNTAX      INTEGER
   MAX-ACCESS  read-only
   STATUS      current
   DESCRIPTION
      "The index number of the first
      unassigned entry in the evaluation
      table.

      A management station should create
      new entries in the evaluation table
      using this algorithm: first, issue a
      management protocol retrieval
      operation to determine the value of
      evalSlot; and, second, issue a
      management protocol set operation to
      create an instance of the evalStatus
      object setting its value to
      createAndGo (4) or
      createAndWait(5). If this latter
      operation succeeds, then the
      management station may continue
      modifying the instances
      corresponding to the newly created
      conceptual row, without fear of
      collision with other management
      stations."
  ::= { eval 1 }

evalTable OBJECT-TYPE
   SYNTAX   SEQUENCE OF EvalEntry
   MAX-ACCESS  not-accessible
   STATUS      current
   DESCRIPTIION
      "The (conceptual) evaluation table."
   ::= { eval 2 }

evalEntry OBJECT-TYPE
   SYNTAX      EvalEntry
   MAX-ACCESS  not-accessible
   STATUS      current
   DESCRIPTION
      "An entry (conceptual row) in the
      evaluation table."
   INDEX       { evalIndex }
   ::= { evalTable 1 }

EvalEntry ::= SEQUENCE {
              evalIndex   Integer32,
              evalString  DisplayString,
              evalValue   Integer32,
              evalStatus  RowStatus }

evalIndex OBJECT-TYPE
   SYNTAX      Integer32
   MAX-ACCESS  not-accessible
   STATUS      current
   DESCRIPTION
      "The auxiliary variable used to
      identify instances of the columnar
      objects in the evaluation table."
   ::= { evalEntry 1 }

evalString OBJECT-TYPE
   SYNTAX      DisplayString
   MAX-ACCESS  read-create
   STATUS      current
   DESCRIPTION
      "The string to evaluate."
   ::= { evalEntry 2 }

evalValue OBJECT-TYPE
   SYNTAX      Integer32
   MAX-ACCESS  read-only
   STATUS      current
   DESCRIPTION
      "The value when evalString was last
      executed."
   DEFVAL { 0 }
   ::= { evalEntry 3 }

evalStatus OBJECT-TYPE
   SYNTAX      RowStatus
   MAX-ACCESS  read-create
   STATUS      current
   DESCRIPTION
      "The status column used for creating,
      modifying, and deleting instances of
      the columnar objects in the evaluation
      table."
   DEFVAL { active }
   ::= { evalEntry 4 }
```

FIGURE 11.7 An example of a table for which row creation and deletion are permitted

```
NOTIFICATION-TYPE MACRO ::= BEGIN

TYPE NOTATION ::= ObjectsPart
                  "STATUS" Status
                  "DESCRIPTION" Text
                  ReferPart

VALUE NOTATION ::= value (VALUE NotificationName)

ObjectsPart ::= "OBJECTS" "{" Objects "}" | empty

Objects ::= Object | Objects "," Object

Object ::= value (Name ObjectName)

Status ::= "current" | "deprecated" | "obsolete"

ReferPart ::= "REFERENCE" Text | empty

Text ::= """" string """"

END
```

FIGURE 11.9 NOTIFICATION-TYPE macro

```
linkUp NOTIFICATION-TYPE
    OBJECTS { ifIndex, ifAdminStatus, ifOperStatus }
    STATUS current
    DESCRIPTION
        "A linkUp trap signifies that the SNMPv2 entity, acting in an
        agent role, has detected that the ifOperStatus object for one of
        its communication links has transitioned out of the down state."
    ::= { snmpTraps 4 }
```

```
OBJECT-GROUP MACRO ::= BEGIN

TYPE NOTATION ::= ObjectsPart
                  "STATUS" Status
                  "DESCRIPTION" Text
                  ReferPart

VALUE NOTATION ::= value (VALUE OBJECT IDENTIFIER)

ObjectsPart ::= "OBJECTS" "{" Objects "}"

Objects ::= Object | Objects "," Object

Object ::= value (Name ObjectName)

Status ::= "current" | "deprecated" | "obsolete"

ReferPart ::= "REFERENCE" Text | Empty

Text ::= """" string """"

END
```

FIGURE 13.4 OBJECT-GROUP macro

```
snmpGroup OBJECT-GROUP
    OBJECTS { snmpInPkts,
              snmpInBadVersions,
              snmpInASNParseErrs,
              snmpBadOperations,
              snmpSilentDrops,
              snmpProxyDrops,
              snmpEnableAuthenTraps }
    STATUS current
    DESCRIPTION
    "A collection of objects providing basic instrumentation and
    control of an SNMPv2 entity."
    ::= { snmpMIBGroups 8 }
```

```
NOTIFICATION-GROUP MACRO ::= BEGIN

TYPE NOTATION ::= NotificationsPart
                  "STATUS" Status
                  "DESCRIPTION" Text
                  ReferPart

VALUE NOTATION ::= value(VALUE OBJECT IDENTIFIER)

NotificationsPart ::= "NOTIFICATIONS" "{" Notifications "}"

Notifications ::= Notification | Notifications "," Notification

Notification ::= value(Name NotificationName)

Status ::= "current" | "deprecated" | "obsolete"

ReferPart ::= "REFERENCE" Text | empty

Text ::= """" string """"

END
```

FIGURE 13.5 NOTIFICATION-GROUP macro

```
snmpBasicNotificationsGroup NOTIFICATION-GROUP
    NOTIFICATIONS { coldStart, authenticationFailure }
    STATUS    current
    DESCRIPTION
      "The two notifications which an SNMPv2 entity is required to
      implement."
    ::= { snmpMIBGroups 7 }
```

```
MODULE-IDENTITY MACRO ::= BEGIN

TYPE NOTATION ::= "LAST-UPDATED" value(Update UTCTime)
                  "ORGANIZATION" Text
                  "CONTACT-INFO" Text
                  "DESCRIPTION" Text
                  RevisionPart

VALUE NOTATION ::= value(VALUE OBJECT IDENTIFIER)

RevisionPart ::= Revisions | empty

Revisions ::= Revision | Revisions Revision

Revision ::= "REVISION" value(Update UTCTime)
             "DESCRIPTION" Text
  --uses the NVT ASCII character set

Text ::= """" string """"

END

OBJECT-IDENTITY MACRO ::= BEGIN

TYPE NOTATION ::= "STATUS" Status
                  "DESCRIPTION" Text
                  ReferPart

VALUE NOTATION ::= value(VALUE OBJECT IDENTIFIER)

Status ::= "current" | "deprecated" | "obsolete"

ReferPart ::= "REFERENCE" Text | empty

Text ::= """" string """"

END
```

FIGURE 11.10 SNMPv2 MODULE-IDENTITY and OBJECT-IDENTITY macros

```
snmpMIB MODULE-IDENTITY
  LAST-UPDATED "9511090000Z"
  ORGANIZATION "IETF SNMPv2 Working Group"
  CONTACT-INFO
      "     Marshall T. Rose
      Postal: Dover Beach Consulting, Inc.
              420 Whisman Court
              Mountain View, CA 94043-2186
              US
      Tel: +1 415 968 1052
      E-mail: mrose@dbc.mtview.ca.us"
DESCRIPTION
    "The MIB module for SNMPv2 entities."
REVISION "9304010000Z"
DESCRIPTION
    "The initial revision of this MIB module was published as
    RFC 1450."
::= { snmpModules 1 }
```

```
MODULE-COMPLIANCE MACRO ::= BEGIN

TYPE NOTATION ::= "STATUS" Status
                  "DESCRIPTION" Text
                  ReferPart
                  ModulePart

VALUE NOTATION ::= value (VALUE OBJECT IDENTIFIER)

Status ::= "current" | "deprecated" | "obsolete"

ReferPart ::= "REFERENCE" Text | Empty

ModulePart ::= Modules | empty

Modules ::= Module | Modules Module

Module ::= "MODULE" ModuleName      --name of module
           MandatoryPart
           CompliancePart

ModuleName ::= modulereference ModuleIdentifier | empty

ModuleIdentifier ::= value (moduleID OBJECT IDENTIFIER) | empty

MandatoryPart ::= "MANDATORY-GROUPS" "{" Groups "}" | empty

Groups ::= Group | Groups "," Group

Group ::= value (group OBJECT IDENTIFIER)

CompliancePart ::= Compliances | empty

Compliances ::= Compliance | Compliances Compliance

Compliance ::= ComplianceGroup | Object

ComplianceGroup ::= "GROUP" value (Name OBJECT IDENTIFIER)
                    "DESCRIPTION" Text

Object ::= "OBJECT" value (Name ObjectName)
           SyntaxPart
           WriteSyntaxPart
           AccessPart
           "DESCRIPTION" Text

--must be a refinement for object's SYNTAX clause
syntaxPart::= "SYNTAX" type (SYNTAX) | empty

--must be a refinement for object's SYNTAX clause
WriteSyntaxPart ::= "WRITE-SYNTAX" type (WriteSYNTAX) | empty

AccessPart ::= "MIN-ACCESS" Access | empty

Access ::= "not-accessible" | "accessible-for-notify" | "read-only" | "read-write" | read-create"

Text ::= """" string """"

END
```

FIGURE 13.6 MODULE-COMPLIANCE macro

```
snmpBasicCompliance MODULE-COMPLIANCE
  STATUS  current
  DESCRIPTION
   "The compliance statement for SNMPv2 entities which implement the
  SNMPv2 MIB."
  MODULE
   MANDATORY-GROUPS { snmpGroup, snmpSetGroup, systemGroup,
                      snmpBasicNotificationsGroup }
  GROUP  snmpCommunityGroup
  DESCRIPTION
    "This group is mandatory for SNMPv2 entities which support
    community-based authentication."
::= { snmpMIBCompliances 2 }
```

```
AGENT-CAPABILITIES MACRO ::= BEGIN

TYPE NOTATION ::= "PRODUCT-RELEASE" Text
                  "STATUS" Status
                  "DESCRIPTION" Text
                  ReferPart
                  ModulePart

VALUE NOTATION ::= value (VALUE OBJECT IDENTIFIER)

Status ::= "current" | "obsolete"

ReferPart ::= "REFERENCE" Text | Empty

ModulePart ::= Modules | empty

Modules ::= Module | Modules Module

Module ::= "SUPPORTS" ModuleName
           "INCLUDES" "{" Groups "}"
           VariationPart

ModuleName ::= identifier ModuleIdentifier

ModuleIdentifier ::= value (moduleID OBJECT IDENTIFIER) | empty

Groups ::= Group | Groups "," Group

Group ::= value (Name OBJECT IDENTIFIER)

VariationPart ::= Variations | empty

Variations ::= Variation | Variations Variation

Variation ::= ObjectVariation | NotificationVariation

NotificationVariation ::= "VARIATION" value (Name NotificationName)
                          AccessPart
                          "DESCRIPTION" Text

ObjectVariation ::= "VARIATIONS" value (Name Objectname)
                    SyntaxPart
                    WriteSyntaxPart
                    AccessPart
                    CreationPart
                    DefValPart
                    "DESCRIPTION" value (description Text)

SyntaxPart ::= "SYNTAX" type (SYNTAX) | empty

WriteSyntaxPart ::= "WRITE-SYNTAX" type (WriteSYNTAX) | empty

AccessPart ::= "ACCESS" Access | empty

Access ::= "not-implemented" | "accessible-for-notify" | "read-only |
           "read-write" | "read-create" | "write-only"

CreationPart ::= "CREATION-REQUIRES" "{" Cells "}" empty

Cells ::= Cell | Cells "," Cell

Cell ::= value (Cell ObjectName)

DefValPart ::= "DEFVAL" "{" value (Defval ObjectSyntax) "}" | empty

Text ::= """" string """"

END
```

FIGURE 13.7 AGENT-CAPABILITIES macro

```
example-agent AGENT-CAPABILITIES
  PRODUCT-RELEASE          "ACME agent release 1.1 for 4BSD"
  STATUS                   current
  DESCRIPTION              "ACME agent for 4BSD"

  SUPPORTS                 SNMPv2-MIB
    INCLUDES               { systemGroup, snmpGroup, snmpSetGroup,
                           snmpBasicNotificationsGroup }
      VARIATION            coldStart
      DESCRIPTION          "A coldStart trap is generated on all reboots."

  SUPPORTS                 IF-MIB
    INCLUDES               { ifGeneralGroup, ifPacketGroup }

      VARIATION            ifAdminStatus
        SYNTAX             INTEGER { up(1), down(2) }
        DESCRIPTION        "Unable to set test mode on 4BSD"

      VARIATION            ifOperStatus
        SYNTAX             INTEGER { up(1), down(2) }
        DESCRIPTION        "Information limited on 4BSD"

  SUPPORTS                 IP-MIB
    INCLUDES               { ipGroup, icmpGroup }

    VARIATION              ipDefaultTTL
      SYNTAX               INTEGER (255..255)
      DESCRIPTION          "Hard-wired on 4BSD"

    VARIATION              ipInAddrErrors
      ACCESS               not-implemented
      DESCRIPTION          "Information not available on 4BSD"

    VARIATION              ipNetToMediaEntry
      CREATION-REQUIRES    { ipNetToMediaPhysAddress }
      DESCRIPTION          "Address mappings on 4BSD require both protocol
                           and media addresses"

  SUPPORTS                 TCP-MIB
    INCLUDES               { tcpGroup }
    VARIATION              tcpConnState
      ACCESS               read-only
      DESCRIPTION          "Unable to set this on 4BSD"

  SUPPORTS                 UDP-MIB
    INCLUDES               { udpGroup }

  SUPPORTS                 EVAL-MIB
    INCLUDES               { functionsGroup, expressionsGroup }
    VARIATION              exprEntry
      CREATION-REQUIRES    { evalString }
      DESCRIPTION          "Conceptual row creation supported"

  ::= { acmeAgents 1 }
```

FIGURE 13.8 Example of AGENT-CAPABILITIES statement

2.7 Seguridad

2.7.1 Introducción

SNMP no proporciona mecanismos de seguridad, es decir, no es capaz de identificar y autentificar la fuente de una mensaje de gestión para prevenir posibles alteraciones.

Un intruso puede observar un mensaje y leer su nombre de comunidad para posteriormente enviar mensajes modificando parámetros de configuración de un dispositivo. Por ello, muchos fabricantes no implementan el comando SetRequest.

Existen cuatro requerimientos básicos de un sistema seguro:

- Confidencialidad: Acceso a información no autorizada.
- Autenticidad: Identificación correcta del origen de un mensaje.
- Integridad: Modificación de datos no autorizados.
- Disponibilidad: Interrupción del correcto funcionamiento de los dispositivos.

GESTIÓN INTERNET

Existen, a su vez, cuatro tipos de amenazas:

- Interrupción: Se interrumpe o destruye la disponibilidad de un recurso.
- Intercepción: Una entidad no autorizada accede a un recurso.
- Modificación: Se modifican los parámetros o datos de un recurso.
- Enmascaramiento: Una entidad se hace pasar por otra.

Estas amenazas la podemos clasificar, a su vez, en pasivas (observación del contenido de un mensaje y análisis del tráfico) y activas (enmascaramiento, replicación, modificación de mensajes y denegación de servicios). Las pasivas son más difíciles de detectar pero menos malignas.

En general, en un entorno SNMP, podemos hablar de los siguientes peligros:

- Enmascaramiento: Una entidad realiza acciones asumiendo la entidad de otra autorizada.
- Modificación de información: Una entidad altera el contenido de un mensaje enviado por otra autorizada.
- Modificación de la secuencia de mensajes: Al tratarse de una comunicación no orientada a conexión, una tercera entidad puede alterar el orden de los mensajes para realizar operaciones no permitidas.
- Descubrimiento: Detección de eventos a partir de la escucha de mensajes gestor-agente.

2.7.2 SNMP seguro (S-SNMP)

Para paliar estos problemas se ha desarrollado el **S-SNMP (SNMP seguro)** que resuelve estos problemas mediante la utilización de un algoritmo de seguridad denominado MD-5.

GESTIÓN OSI

S-SNMP proporciona servicios de seguridad para:

- Integridad de datos: Se asegura que un mensaje es enviado y recibido sin duplicación, inserción, modifición, resecuenciamiento o replicación. El algoritmo MD-5 añade un resumen de 128 bits del mensaje y lo incorpora en su cabecera para asegurar que no se modifica. Se incorpora también una señal de tiempo para asegurar el secuenciamiento.
- Autentificación del origen de los datos: Corroboración de la fuente emisora de un mensaje. Para calcular el resumen anterior se utiliza una semilla secreta que conocen a priori emisor y receptor.
- Confidencialidad de datos: Asegura que la información no pueda ser observada por entidades no autorizadas. Se realiza una encriptación de parte del mensaje utilizando el algoritmo DES (Data Encription Standard).

S-SNMP está basado en un nuevo modelo administrativo que reemplaza el concepto de comunidad de SNMP. Se incluye un identificador que especifica quien es el destino y fuente durante un intercambio. Esta información es incorporada en la cabecera del mensaje.

Se incorpora el concepto de ***Party*** como entidad administrativa que regula el acceso a una entidad. Un SNMP *party* regula cómo llegar a una entidad, que algoritmo de autentificación utiliza y que algoritmo de privacidad usa.

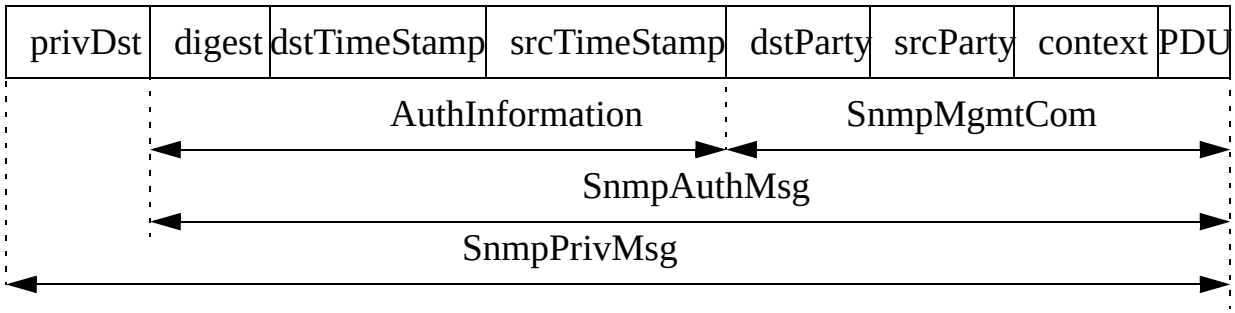
2.7.3 Seguridad con SNMPv2

El mayor cambio con respecto a la primera versión de SNMP se produce en el campo de la seguridad. Permite (opcionalmente) dotar de privacidad y autenticidad a las primitivas de SNMPv2.

Incorpora el concepto de *party* heredado de S-SNMP.

Se incorpora a la cabecera del mensaje información del contexto (vista MIB) sobre el que actuará el mensaje.

Formato del mensaje en SNMPv2:



La **generación de un mensaje** consta de los siguientes pasos:

1.- Se construye el valor del SnmpMgmtCom donde:

- srcParty: identifica la parte origen.
- dstParty: identifica al destino.
- context: indica la vista MIB sobre la que se actuará.
- PDU: representa la operación de gestión deseada.

2.- Se construye el valor de SnmpAuthMsg donde:

- authSrcTimestamp: indica el clock del origen.
- authDstTimestamp: indica el clock del destino.
- authDigest: resumen generado por MD-5 en la parte origen.

3.- Se encripta el SnmpAuthMsg.

4.- Se coloca en el campo privDst de SnmpPrivMsg el identificador de la parte destino.

La **recepción del mensaje** consta, a su vez, de los siguientes pasos:

1.- Si el privDst no es válido, se rechaza el mensaje.

2.- Se desencripta el privData para obtener el SnmpAuthMsg.

3.- Se rechaza el mensaje si no casa el privDst con dstParty o se desconoce el srcParty .

4.- Se rechaza el mensaje si no cuadran los Timestamp dentro del margen.

5.- Se extrae el valor del authDigest y se compara con el nuevo cálculo.

6.- Se consulta el contexto para ver si la operación solicitada es posible.