Universitat Oberta de Catalunya
Enginyeria tècnica d'informàtica de sistemes
Xarxes de computadors
Treball de final de carrera

Gestió de xarxes Internet basada en SNMP

Autor
Pere Rodríguez Rodríguez

Tutor
Maria Isabel March Hermo

14 de gener de 2006

A la Carme, la meva parella i mare del nostre fill.

Sense el seu suport m'hagués estat impossible cursar aquests estudis.

RESUM

Les xarxes de dades s'han esdevingut essencials per a l'emergent societat de la informació. Xarxes cada cop més complexes, formades per una gran diversitat de components actius i passius de diferents fabricants, constitueixen la base de la societat de la informació.

El desenvolupament de les xarxes ha requerit d'eines per facilitar, i fins i tot automatitzar, la gestió i monitoratge de la xarxa; eines fàcils d'implementar, d'instal·lar i d'utilitzar, i que a més a més, consumeixin pocs recursos.

Aquest escenari, o necessitat, propicia l'aparició del protocol *Simple Network Management Protocol* (SNMP). Introduït al 1988 per a gestionar xarxes IP, SNMP ràpidament s'esdevé l'eina més extensament utilitzada per a la gestió de xarxes, essent implementada per tots els grans fabricants. L'èxit d'SNMP es sustenta en la seva senzillesa, facilitat d'ús i implementació, consum de poc ample de banda i capacitats d'escalabilitat.

SNMP està basat en tres conceptes: gestors, agents i la *Managed Information Base* (MIB). A qualsevol configuració, com a mínim un dels nodes de la xarxa té instal·lat el programari de gestió; altres dispositius de la xarxa com servidors, encaminadors, commutadors, concentradors, impressores, ... actuen com agents, equipats amb els programaris escaients.

Els agents tenen el rol d'inspeccionar els recursos del node que controlen i proporcionar informació sobre aquests. Així, cada agent disposa d'una MIB, la qual es composa de diferents objectes que representen els recursos a gestionar. El rol del gestor consisteix a interrogar l'agent pels diferents objectes de la seva MIB, per exemple, un gestor pot demanar a un encaminador el nombre de paquets entrants en una interfície determinada.

Un dels grans problemes d'SNMP, en les seves primeres versions, és la manca de seguretat. La transferència de dades es fa sense cap xifrat i l'autenticació és limita a una simple paraula de pas que es transmet per la xarxa sense cap xifrat.

Al 1993 s'introdueix la segona versió d'SNMP consolidant-se al 1996. SNMPv2 aporta noves funcionalitats i en millora d'altres de la 1a. versió, no obstant el problema de la seguretat continua existint. Fruit d'aquesta manca de seguretat, moltes implementacions d'SNMPv1 i SNMPv2 és limiten a operacions de lectura.

Al 1998 apareix SNMPv3 per cobrir els problemes de seguretat de les versions anteriors. SNMPv3 aporta xifrat i autenticació basats en clau simètrica, a més a més, es defineix una estructura modular per facilitar l'escalabilitat del protocol.

Índex de contingut

Índex d'il·lustracions

Índex de taules

1 INTRODUCCIÓ

1.1 Justificació

La gestió de xarxes és imprescindible per al correcte funcionament de qualsevol xarxa. SNMP ha esdevingut el protocol per excel·lència adoptat per tots els fabricants per a monitoritzar i controlar els seus aparells.

Sota aquesta premissa, és imprescindible que qualsevol administrador de sistemes sàpiga utilitzar correctament SNMP. D'altra banda, als estudis d'ETIS no s'imparteix aquesta matèria, fet que propicia l'estudi i implementació d'aquest protocol en el present treball com a complement dels estudis cursats.

1.2 Objectius

- Assolir els coneixements necessaris d'SNMP per tal de poder utilitzar el protocol en la gestió xarxes Internet.
- Analitzar l'estat del mercat en quant a llibreries disponibles per a SNMP. Es prioritzarà l'anàlisi de llibreries amb llicència de lliure distribució.
- Implementar una llibreria SNMP per consolidar tots els conceptes d'SNMP.

1.3 Metodologia

El treball es divideix en dos grans parts, una primera eminentment teòrica en la que s'estudia SNMP, i una segona íntegrament pràctica en la que s'implementa el protocol. Per a cada part es defineix una metodologia ben diferenciada.

La primera part es basa en la consulta de bibliografia en format paper i digital, se n'extrauen els conceptes més importants i es plasmen en aquest document. Dir que tot i l'extensió del document, aquest té un caire minimalista.

La segona part es centra en la implementació del protocol. Per a fer-ho s'estudien algunes llibreries i en funció d'aquestes i els coneixements adquirits a la 1a. part, s'analitza, és dissenya i implementa una llibreria SNMP. La nova llibreria es batejada amb el nom de NetSNMP.NET.

1.4 Planificació

El fet de dividir el treball en dos grans parts clarament condiciona la planificació, alhora que facilita el lliurament de les PPAC 2 i 3, donat es presenta la 1a. part com a PAC2 i la segona com a PAC3.

Part	Mòdul	Inici
Estudi d'SNMP	Introducció a la gestió de xarxes	13/10/2005
	SNMP	17/10/2005
	SNMPv2	25/10/2005
	SNMPv3	31/10/2005
	Lliurament PAC2	14/11/2005
Llibreria NetSNMP.NET	Cerca i estudi de llibreries d'SNMP	15/11/2005
	Anàlisi i disseny de NetSNMP.NET	21/11/2005
	Implementació de NetSNMP.NET	5/12/2005
	Test i exemples	19/12/2005
	Lliurament PAC3	23/12/2005
Lliurament memòria	Últims retocs memòria final	2/1/2006
	Lliurament memòria final	4/1/2006

1.5 Resultats esperats

Donada la divisió del projecte, s'espera assolir els següents dos aspectes:

1. El present document ha de constituir una bona base per a iniciar-se en el protocol SNMP. Aquesta memòria ha de ser suficient per a que el lector comprengui el funcionament del protocol en les seves tres versions, i a de donar peu a aprofundir en alguns aspectes d'SNMP que aquí queden incomplets.
2. Amb la implementació de la llibreria SNMP, anomenada NetSNMP.NET, es pretén donar al programador una eina robusta i eficient, alhora que fàcil d'utilitzar, per atacar agents SNMP.

1.6 Estructura i continguts

Un cop més, la divisió del projecte en dos àares, condiciona la seva estructuració. La primera part, clarament teòrica, es centra en l'estudi d'SNMP i la segona, exclusivament pràctica, tracta l'anàlisi, disseny i implementació d'una llibreria SNMP.

El contingut per capítols és el següent:

1. *Introducció*. El present capítol. Es dóna una visió de conjunt del que és i com ha estat el TFC.
2. *Introducció a la gestió de xarxes*. El seu títol és suficient per a definir el capítol. Es conceptualitza l'arquitectura de sistemes de gestió de xarxes i el monitoratge i control de xarxa.
3. *SNMP*. S'ofereix el primer contacte amb SNMP. S'explica el funcionament del protocol, la MIB i la seguretat.
4. *SNMPv2*. La segona versió d'SNMP canvia pocs aspectes de la 1a., així aquest capítol presenta les novetats d'SNMPv2 i en desenvolupa exclusivament les noves funcions.
5. *SNMPv3*. La darrera versió d'SNMP constitueix una revolució en quant a seguretat i estructuració modular. En aquest capítol es defineix l'estructura d'SNMPv3, i es presenta el nou model de seguretat basat en l'usuari (USM) i el nou control d'accés basat en vistes (VACM).
6. *Anàlisi i disseny*. Aquest és el primer capítol de la segona part. Es dóna una petita situació del mercat de llibreries SNMP, s'identifiquen els requeriments de la llibreria a implementar i finalment es presenta breument el model de negoci de la llibreria.
7. *Implementació*. Es desenvolupa l'entorn d'implementació de NetSNMP.NET i es defineix l'estructuració del projecte.
8. *Conclusions i línies futures de treball*. Per acabar, en aquest apartat es sintetitzen algunes idees sobre SNMP.

Annexos a Estudi d'SNMP. Es dóna una breu referència a les MIB de cada versió d'SNMP.

Annexos a NetSNMP.NET. D'una banda és donen indicacions per a instal·lar el programari necessari per a executar NetSNMP.NET i de l'altra s'ofereix un breu manual d'ús de NetSNMP.NET amb alguns exemples.

La documentació de la llibreria NetSNMP.NET és poc extensa per tal de no incrementar en excés el nombre de planes de la memòria. Per completar-la cal consultar l'annex a NetSNMP.NET, documentació en format Monodoc i els fonts.

1a. part

Estudi d'SNMP

2 INTRODUCCIÓ A LA GESTIÓ DE XARXES

La informatització de sistemes, cada cop més grans i complexes, amb aplicacions distribuïdes i amb un gran nombre d'elements que interactuen entre si, a derivat en que les xarxes hagin adquirit una major rellevància dins el sistema, fins al punt de convertir-se en un component crític d'aquest.

L'expansió de les xarxes, composades per un gran nombre d'elements, ha fet necessari disposar de sistemes de gestió de xarxa (SGX) per controlar possibles avaries, degradacions de rendiment i també per controlar de forma centralitzada el conjunt de la xarxa.

2.1 Arquitectura dels Sistemes de Gestió de Xarxa

Un SGX es defineix com un conjunt d'eines integrades per al control i el monitoratge de la xarxa.

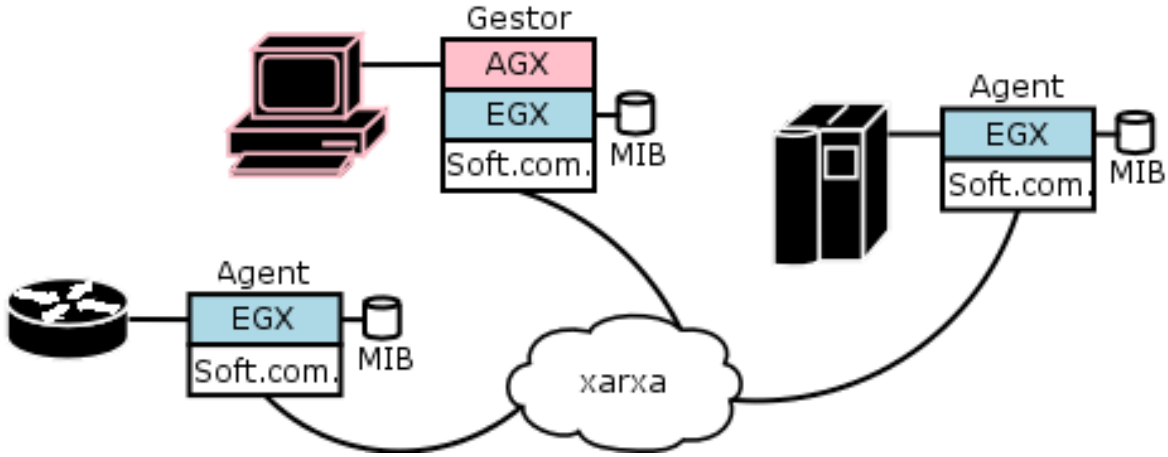
Els **components** que composen un SGX són els següents:

- *Entitat de Gestió de Xarxa (EGX):* Es troba present en cada node de la xarxa. Les seves funcions són recollir i emmagatzemar estadístiques locals de l'activitat de la xarxa, respondre a comandes del Centre de Gestió de Xarxa (CGX) i, si s'escau, enviar missatges al CGX en resposta de determinats events que es puguin donar al node (per exemple informar d'una avaria).
- *Aplicació de Gestió de Xarxa (AGX):* Constitueix una interfície d'usuari per a la gestió de la xarxa. L'aplicació permet mostrar la informació dels diferents EGX així com interrogar-los.
- *Agent:* Es refereix a cada node que forma part del sistema gestionat i que inclou un EGX.
- *Gestor o Centre de Gestió de Xarxa (CGX):* Node de la xarxa que actua com a gestor, a més a més del seu EGX conté el programari AGX.
- *Base d'informació gestionada (MIB[1]):* Defineix la informació relacionada amb el sistema de gestió. La MIB d'un agent conté informació del node, mentre que d'un gestor conté la de tots els agents que gestiona.

1 *Management Information Base*

En quant al **tipus d'informació** gestionada diferenciem entre:

- *Estàtica:* És la informació que caracteritza la configuració de la xarxa. En ella es descriuen els elements que composen la xarxa així com la seva configuració. Podríem parlar d'un mapa de xarxa, que contindria el conjunt dels diferents elements de la xarxa amb una identificació mínima (IP, nom, ...) més la configuració de cada element ubicada en el mateix dispositiu (per exemple la configuració d'un router).
- *Dinàmica:* Events que es produeixen a la xarxa, tals com canvis d'estat en el serveis oferts per un host o la simple transmissió d'un paquet.
- *Estadística:* Es la informació derivada de la informació dinàmica, com per exemple la mitjana de paquets transmesos per unitat de temps.



Il·lustració 2.1: Sistema de gestió de xarxa

Un altra element a destacar en l'arquitectura són els **proxies**. En una situació idíl·lica tots els EGX serien compatibles entre ells permetent així la comunicació entre el gestor i tots els seus agents. Això no sempre és possible perquè l'existència de components antics, sistemes molt petits, ... fa que el gestor no es pugui comunicar amb tots els agents. Per a solucionar aquest problema apareix la figura del proxy, que actua com una interfície entre el SGX i els elements que no són compatibles amb ell.

En la gestió de xarxa distingirem cinc **àrees** principals **de gestió**: rendiment, avaries, *accounting*[2], configuració i seguretat. D'altra banda, les tasques de gestió de xarxa es poden classificar com a monitoratge o com a control. Així doncs, tot i que cap de les àrees definides es pot encasellar perfectament com a monitoratge o com a control, sí que distingirem les tres primeres (rendiment, avaries i *accounting*) com a tasques de monitoratge i les dos restants (configuració i seguretat) com a control.

2 És refereix a la utilització que se'n fa dels recursos de la xarxa per part dels usuaris. S'utilitza el terme anglès perquè no s'ha trobat un mot català que clarament reflecteixi aquest concepte.

2.2 Monitoratge de la xarxa

El monitoratge de la xarxa es basa en l'observació i anàlisi de l'estat de la mateixa amb la finalitat de conèixer el comportament del sistema. Les Àrees que comporta el monitoratge són rendiment, avaries i *accounting*.

La informació recol·lectada pels agents és la base per al monitoratge de la xarxa. Aquesta informació pot arribar fins als gestors utilitzant dos tècniques: petició (*polling*) i comunicació d'events (*event reporting*).

La petició es basa en la interrogació per part del gestor a l'agent. El gestor interroga els agents per tal d'obtenir informació dels nodes, els agents responen amb la informació emmagatzemada a la seva MIB.

D'altra banda, a la a comunicació d'events, és l'agent qui pren la iniciativa de comunicar un event al gestor. Aquests events normalment es refereixen a estats del node que cal assabentar al gestor, per exemple, si el node és un servidor HTTP, un possible event a comunicar seria que el servei HTTP no està disponible.

2.2.1 Rendiment

En tres paraules: "qualitat de funcionament". L'objectiu de l'anàlisi de rendiment es assegurar que la capacitat i prestacions de la xarxa correspon a les necessitats dels usuaris. El monitoratge de rendiment portarà a fer les modificacions necessàries per a millorar el rendiment, estant aquestes tasques dins l'àmbit del control de la xarxa (fixem-nos que ja s'ha indicat que és difícil separar monitoratge i control).

El monitoratge es basarà en l'estudi de diversos indicadors. D'aquests indicadors en tenim orientats a servei i orientats a eficiència.

2.2.1.1 Indicadors orientats a servei

- *Disponibilitat:* Es mesura com el percentatge de temps que un element de està disponible per al usuari. Respon a l'expressió $D=\frac{TF}{TF+TR}$ on TF és el temps de funcionament entre avaries i TR és el temps de reparació. La disponibilitat per a elements en sèrie és D^2 i en paral·lel $2D-D^2$.
- *Temps de resposta:* Es defineix com el temps que el sistema triga a respondre per a una entrada donada. El temps de resposta està directament relacionat amb la càrrega del sistema.
- *Errors de transmissió:* Percentatge de paquets amb error respecte al total de paquets.

2.2.1.2 Indicadors orientats a eficiència

- *Rendiment:* El rendiment és un indicador orientat a aplicació. Exemples en podrien ser:
  - nombre de transaccions per unitat de temps.
  - nombre de sessions d'usuari durant un cert temps.
- *Utilització:* El percentatge d'utilització de la capacitat teòrica d'un recurs. Segurament l'ús més important d'aquest indicador radica en la cerca de *colls d'ampolla* i àrees de congestió.
  Aquest és un indicador molt important perquè normalment el temps de resposta incrementa de forma exponencial respecte a l'increment d'utilització.

2.2.2 Avaries

L'objectiu in extremis del monitoratge d'avaries és detectar-les el més ràpidament possible, identificar-ne la causa i actuar en conseqüència. Veiem que un cop més, el monitoratge com a tal, desencadena una acció de control.

El monitoratge d'avaries es pot efectuar des del gestor cap a l'agent o pot ser l'agent qui informi al gestor (*event reporting*) de la incidència.

Per tal de detectar incidències a la xarxa, el gestor efectuarà accions de control com tests de connectivitat, de temps de resposta, ..., si algun d'aquests tests falla aleshores es disparará l'alarma d'avaria.

D'altra banda, pot ser l'agent qui detecti l'avaria i alerti el gestor del fet. En aquest cas direm que el gestor dispara un *trap*, altrament dit, comunica al gestor un determinat event d'error.

2.2.3 Accounting

Entenem com *accounting* el seguiment de l'ús de recursos de la xarxa per part d'un usuari o grups d'usuaris.

Les motivacions poden ser entre d'altres:

- Facturació
- Vigilància de l'ús abusiu de privilegis d'accés que poden donar lloc a sobrecàrregues de la xarxa en perjudici de la resta d'usuaris.
- Ús ineficient de la xarxa. El gestor pot ajudar a modificar la forma de treballar per tal de millorar les prestacions.
- Ús indegut de la xarxa, per exemple, control sobre els accessos a internet.

2.3 Control de la xarxa

El control de la xarxa permet alterar els paràmetres de la xarxa tot canviant la configuració dels seus elements. Com s'ha vist, en ocasions les tasques de control seran desencadenades fruit del monitoratge de la xarxa per tal de corregir algun element.

Les àrees de control són la configuració i la seguretat. En aquest cas, serà sempre el gestor qui inicia la comunicació amb un agent per tal de modificar algun paràmetre.

2.3.1 Configuració

Les tasques de configuració principals es poden resumir en els punts següents:

- Inicialització (*start-up*) i desconnexió (*shut-down*) dels components de la xarxa.
- Manteniment i configuració dels components de xarxa.
- Arrencada i aturada de serveis de xarxa.
- Configuració de programari distribuït.

2.3.2 Seguretat

Es defineix com el procés per a controlar l'accés a la informació continguda en els elements de la xarxa, i protecció de la mateixa front fonts hostils o accidentals.

Les tasques de seguretat inclouen:

- Control d'accés més xifratge de la informació enviada per la xarxa.
- Gestió de fitxers de log, que emmagatzemen informació del que passa a la xarxa pel seu posterior anàlisi.
- Generació, distribució i xifratge de claus d'accés.
- Registre dels usuaris que consulten determinada informació i durant quant de temps, així com els intents fallits d'accés a la susdita informació.

3 SNMP

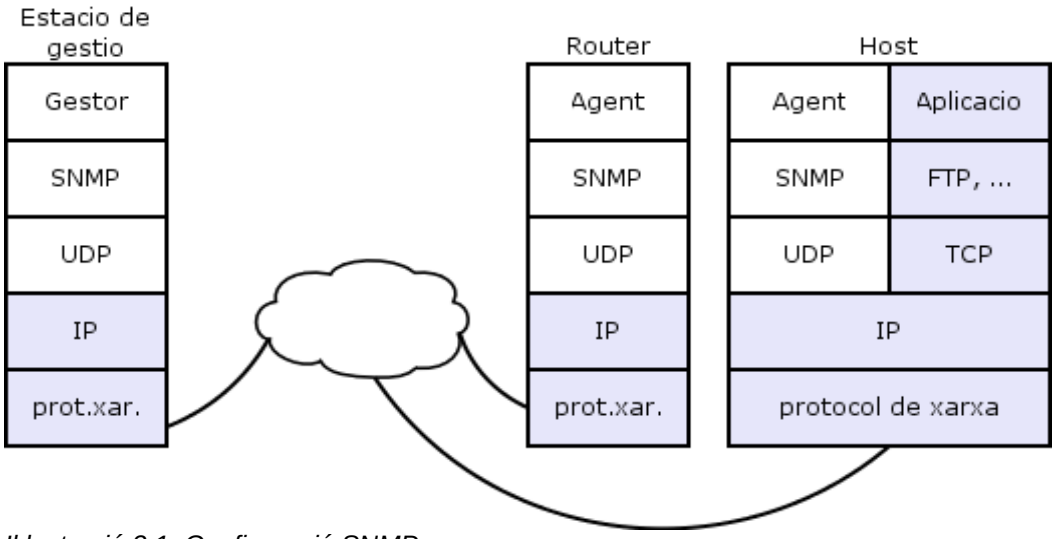
3.1 Conceptes preliminars

SNMP és l'acrònim de *Simple Network Management Protocol*. L'estàndard SNMP defineix un marc per a la gestió de la informació de xarxa i un protocol per a l'intercanvi de la mateixa. Així doncs, el model de gestió de xarxes introduït al punt anterior es pot implementar amb SNMP.

El model de gestió per xarxes TCP/IP es base en els elements següents:

- *Estació de gestió:* Programari de gestió de la xarxa. Gestiona la informació dels diferents nodes de xarxa mitjançant peticions SNMP als seus agents. També pot rebre informació des d'un node mitjançant un *trap* enviat per l'agent.
- *Agent:* Programari que gestiona la informació local d'un node. Es responsable de respondre peticions i disparar *traps*.
- *MIB (Management Information Base):* Definició d'objectes que constitueixen la informació gestionada per l'agent en aquell node.
- *Protocol de xarxa:* Protocol d'intercanvi d'informació, òbviament SNMP.

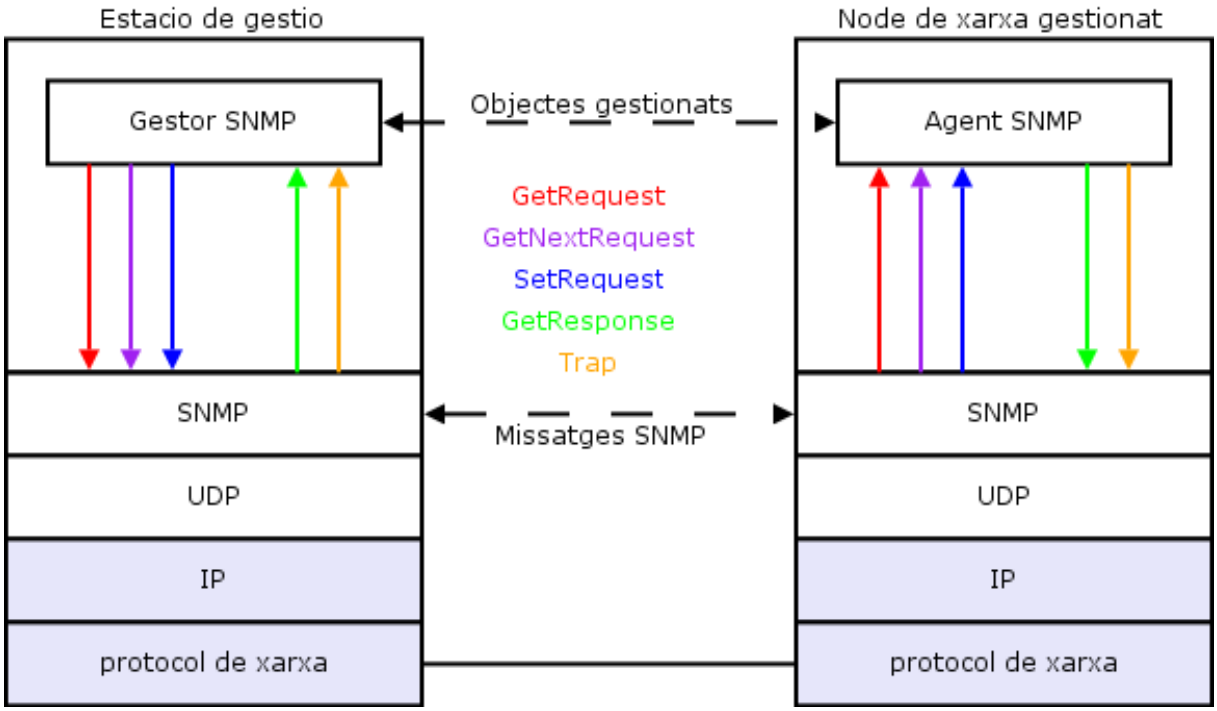
Fixem-nos que d'aquests quatre punts, sols els dos darrers estan subjectes l'especificació SNMP.



Il·lustració 3.1: Configuració SNMP

SNMP és un protocol d'aplicació transportat sobre UDP, per tant, SNMP es un protocol no orientat a la connexió. Això és així per tal de minimitzar el transit de xarxa fruit de la gestió d'aquesta.

El gestor, a més a més d'obtenir informació dels agents, també pot configurar determinats paràmetres de l'element que gestiona el client. D'altra banda, obtenir la informació de tots els agents interrogant periòdicament aquests pot resultar costos, és per això que s'implementen els *traps*. Un agent pot decidir comunicar determinada informació a un gestor mitjançant un missatge *trap*, d'aquesta manera la gestió és molt més lleugera.



Il·lustració 3.2: El rol SNMP

La MIB defineix un conjunt d'objectes a ser gestionats, així, un mateix objecte tindrà la mateixa definició en tots els nodes de xarxa. La informació es transmet com a instàncies d'aquests objectes. Un exemple en seria l'objecte *ifName* que representaria el nom d'una interfície de xarxa. Aquest objecte podria tenir dos instàncies, la *ifName.0* i *ifName.1* amb els noms de les dos interfícies de xarxa del node.

Per últim introduir la figura del *proxy* que actua com a element d'enllaç entre un gestor i un agent que no es poden comunicar directament per incompatibilitat de protocols, ja sigui a nivell d'SNMP o fins i tot de xarxa.

3.2 Estructura de la informació de gestió (SMI)

La estructura de la informació de gestió o SMI (*Structure of Management Information*), especificada a la RFC 1155, determina un marc per a la definició i construcció de la MIB. L'SMI defineix quins tipus de dades es poden utilitzar a la MIB i com els recursos s'anomenen i s'identifiquen dins la MIB. Una de les màximes de l'SMI és la simplicitat i l'escalabilitat de la MIB, per això la MIB sols pot emmagatzemar tipus de dades escalars i taules bi-dimensionals d'escalars.

L'SMI proveeix tècniques per a:

- Definir l'estructura d'una MIB.
- Definir els objectes de la MIB, incloent la sintaxi i els valors de cada objecte.
- Codificar els valors de cada objecte.

L'SMI utilitza l'estàndard ASN.1 (*Abstract Syntax Notation One*) per a escriure el conjunt definicions que conformen una MIB.

3.2.1 Estructura de la MIB

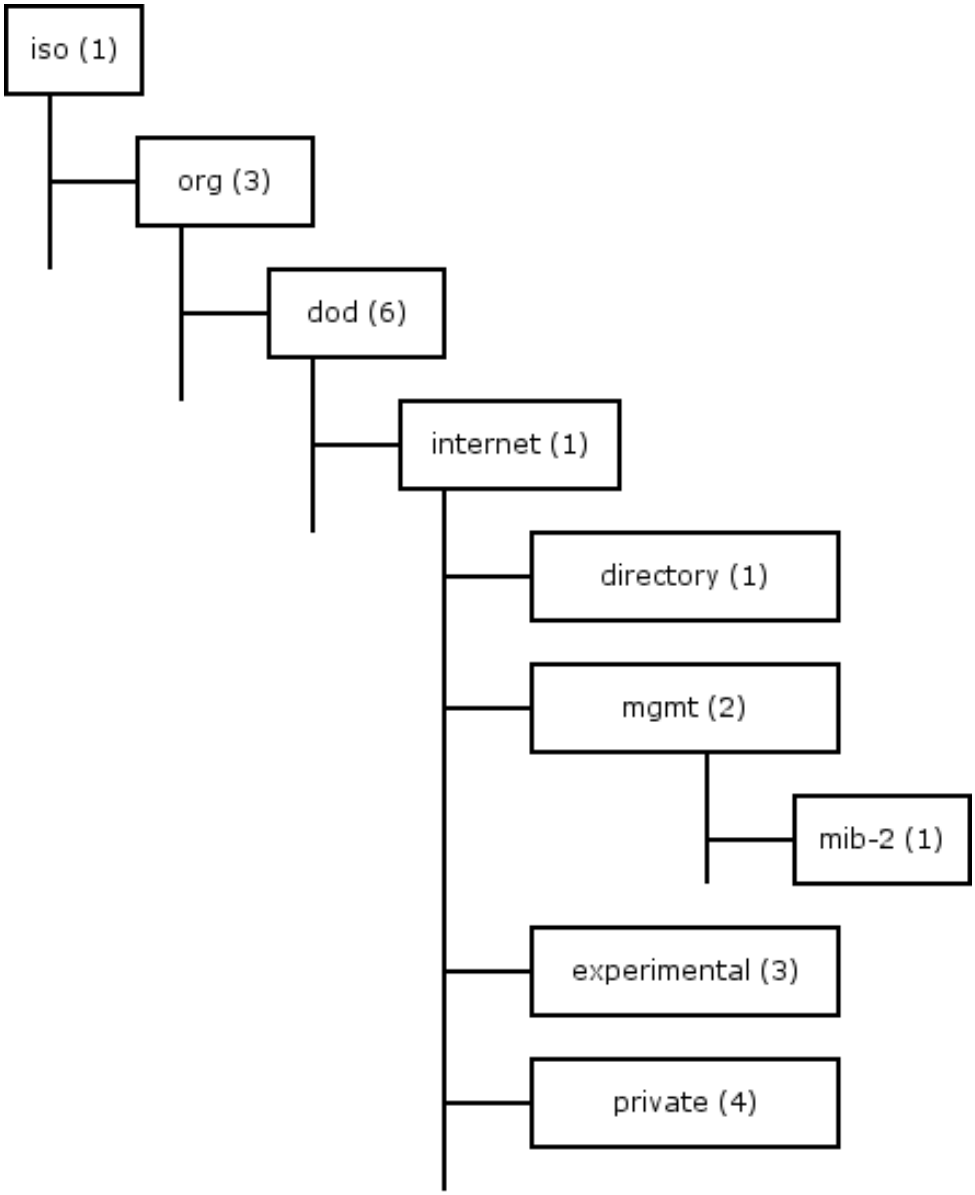
Tots els objectes gestionats en un entorn SNMP estan classificats en una estructura en forma d'arbre. Les fulles de l'arbre ubiquen els objectes i un objecte pot representar un recurs, activitat o qualsevol informació que sigui susceptible de ser gestionada.

A cada objecte de la MIB s'associa un identificador únic (OID) de tipus OBJECT IDENTIFIER especificat a l'estàndard ASN.1. Un OID té valor jeràrquic obtingut a partir del valor dels nodes que hi ha per sobre d'ell i del seu propi valor.

L'arbre d'OID és una especificació d'ISO, a la figura 3.3 se'n pot veure la seva estructura. L'SMI defineix quatre nodes sota el node *internet*:

- *directory (1):* reservat per a ús futur del directori OSI (X.500).
- *mgmt (2):* utilitzat per a objectes definits per l'IAB. Les MIB estàndard que més endavant es veuran s'ubiquen sota aquest node.
- *experimental (3):* per a objectes en fase d'experimentació.
- *private (4):* per a objectes definits per organitzacions de caire privat. Per exemple, Cisco podria definir una MIB (o varies) per als seus aparells sota aquest node.

Com a curiositat, destacar que el node *dod (6)*, sota el qual penja *internet (1)*, pertany al Departament de Defensa dels Estats Units.



Il·lustració 3.3: Arbre d'OID descrit per ISO

Per exemple, l'OID del node *internet (1)* és 1.3.6.1.

Fins al moment s'han estandarditzat dos MIB, la MIB-I i la MIB-II. La segona és una extensió de la primera. Ambdues MIB tenen els mateixos OID i com a conseqüència, en una configuració donada, sols una de les dos MIB pot estar present.

3.2.1.1 Sintaxi d'objecte

Com ja s'ha comentat la sintaxi per a definir objectes respon a l'estàndard ASN.1, no obstant, tant sols una petita part d'aquesta especificació es permet.

La taula següent especifica els tipus de dades permesos:

Tipus	***Classe ASN.1***	***Descripció***
INTEGER	UNIVERSAL 2	Valor enter comprés entre -214748648 i 2147483647 inclusive
OCTET STRING	UNIVERSAL 4	Cadena de fins a 65535 caràcters
NULL	UNIVERSAL 5	Valor nul
OBJECT IDENTIFIER	UNIVERSAL 6	Identificador d'objecte únic. Es format per una seqüència d'enters, anomenats subidentificadors, separats per punts (1.2.3).
SEQUENCE, SEQUENCE-OF	UNIVERSAL 6	Construcció de taules com a seqüències de tipus primaris.
NetworkAddress		Permet escollir el format d'adreça. Actualment sols està definit el format *IpAddress*.
IpAddress	APPLICATION 0	Adreça de 32 bits utilitzant el format especificat a IP.
Counter	APPLICATION 1	Enter no negatiu que pot incrementar però no decrementar. El seu valor màxim és 2^{32}-1. Quan s'assoleix el valor màxim el comptador comença a comptar des de 0.
Gauge	APPLICATION 2	Enter no negatiu que pot incrementar o decrementar. El seu valor màxim és 2^{32}-1. Quan s'assoleix el valor màxim el comptador manté aquest valor fins que no és resetejat.
TimeTicks	APPLICATION 3	Enter no negatiu que compta el temps en centèsimes de segon.
Opaque	APPLICATION 4	Té la capacitat d'ubicar qualsevol tipus de dada, actua com un comodí. Emmagatzema en format *octetstring*.

Taula 3.1: Tipus de dades permesos a SNMP

Cada objecte dins la MIB té una definició formal mitjançant la macro OBJECT-TYPE. Per definir els objectes de la MIB-I s'utilitza la RFC 1155, i per a la MIB-II la RFC 1212 que inclou més informació.

Un exemple de definició d'objecte seria el següent:

```
tcpMaxConn OBJECT-TYPE
          SYNTAX INTEGER
```

```
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
                  “The limit on te total number of TCP connections”
        ::= { tcp 4 }
```

A destacar:

- El valor de *SINTAX*, a més a més del tipus, pot enumerar uns conjunt de valors vàlids, o especificar un rang vàlid.
- *ACCESS* pot ser *read-only*, *read-write*, *write-only* i *not-accessible*.
- El valor de *STATUS* pot ser *mandatory* o *optional*.
- El valor del node dins de l'arbre d'OID és 4, i el seu OID és composa a partir del de l'objecte *tcp* concatenant-hi “.4”.

3.2.1.1.1 Identificació d'instàncies (OIID) d'escalars

La informació es representa com a instàncies d'aquests objectes. Així, si és donés el cas, cada objecte podria tenir més d'una instància. Cada instància d'objecte té un identificador únic anomenat OIID i es composa de l'OID de l'objecte més un nivell més, que comença per 0, per a identificar la instància. Per exemple, l'objecte aquí presentat *tcpMaxConn* per definició sols pot tenir una instància que serà *tcpMaxConn.0*, o també tcp.4.0 o 1.3.6.1.2.1.6.4.0. No obstant, altres objectes com *ifName*, que representa el nom d'una interfície de xarxa, poden tenir més d'una instància amb els OIID *ifName.0*, *ifName.1*, ...

3.2.1.1.2 Definició de taules

L'SMI permet la definició de taules bi-dimensionals per a recollir informació com per exemple connexions TCP d'un host. La definició de taules és un tant complicada i potser difícil d'entendre per això es plantejarà un exemple i sobre ell es faran les explicacions oportunes.

La taula d'exemple que es mostrarà representarà les connexions TCP d'un node. Per a representar-la extrapolarem el model relacional, així doncs la nostra taula tindrà un nom de taula i unes columnes que formaran la fila. Al igual que en el model relacional, algunes d'aquestes columnes composaran la clau primària que identifica la fila. Així doncs, la taula té la forma següent:

> ***tcpConnTable*** *(tcpConnLocalAddress, tcpConnLocalPort, tcpConnRemAddress, tcpConnRemPort, tcpConnState)*[3]

A destacar que el model SNMP no requereix que la clau primària d'una fila sigui única, això significa que podem tenir dos files amb els mateixos índexs. No obstant, a la pràctica, les implementacions si que respecten que la clau primària

3 Els valors subratllats composen la clau primària de la taula.

sigui única per a cada fila.

```
tcpConnTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF TcpConnEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "A table containing TCP connection-specific information."
    ::= { tcp 13 }

tcpConnEntry OBJECT-TYPE
    SYNTAX      TcpConnEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "A conceptual row of the tcpConnTable  ..."
    INDEX   { tcpConnLocalAddress,
              tcpConnLocalPort,
              tcpConnRemAddress,
              tcpConnRemPort }
    ::= { tcpConnTable 1 }

TcpConnEntry ::= SEQUENCE {
        tcpConnState          INTEGER,
        tcpConnLocalAddress   IpAddress,
        tcpConnLocalPort      INTEGER,
        tcpConnRemAddress     IpAddress,
        tcpConnRemPort        INTEGER
    }

tcpConnState OBJECT-TYPE
    SYNTAX      INTEGER {
                    closed(1),
                    listen(2),
                    synSent(3),
                    synReceived(4),
                    established(5),
                    finWait1(6),
                    finWait2(7),
                    closeWait(8),
                    lastAck(9),
                    closing(10),
                    timeWait(11),
                    deleteTCB(12)
                }
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
            "The state of this TCP connection ..."
    ::= { tcpConnEntry 1 }
```

```
tcpConnLocalAddress OBJECT-TYPE
    SYNTAX      IpAddress
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "The local IP address for this TCP connection ..."
    ::= { tcpConnEntry 2 }

    ...
    ...
```

Els passos seguits per a la definició de la taula han estat:

1. **Definició de la taula** (*tcpConnTable*). És defineix la taula de nom *tcpConTable* amb la macro OBJECT-TYPE. Fixem-nos que el valor de SYNTAX és de tipus SEQUENCE OF X, tot indicant que és una seqüència de files de tipus X.
2. **Definició de la fila** *(tcpConnEntry)*. En aquest cas cal observar que el valor de SYNTAX és el tipus de fila. Destacar que apareix un element nou, INDEX, que defineix els camps de la fila composen la clau primària.
3. **Definició del tipus de fila** *(TcpConnEntry)*. El tipus de fila es defineix amb la clau SEQUENCE i senzillament enumera el nom i tipus de cada camp. Fixem-nos que la fila no és un objecte i per tant no s'utilitza la macro OBJECT-TYPE.
4. **Definició de camps de fila** *(tcpConnState, tcpConnLocalAddress, ...)*. Per acabar definir els camps de fila amb la macro OBJECT-TYPE.

Tot seguit es mostra un exemple de la taula.

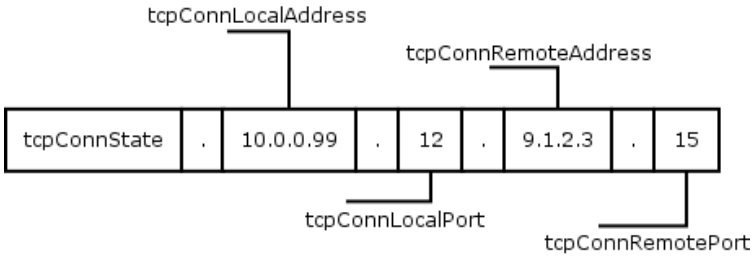
tcpConnTable (1.3.6.1.2.1.6.13)				
tcpConnState (1.3.6.1.2.1.6.13.1.1)[4]	tcpConnLocalAddress (1.3.6.1.2.1.6.13.1.2)	tcpConnLocalPort (1.3.6.1.2.1.6.13.1.3)	tcpConnRemAddress (1.3.6.1.2.1.6.13.1.4)	tcpConnRemPort (1.3.6.1.2.1.6.13.1.5)
5	10.0.0.99	12	9.1.2.3	15
2	0.0.0.0	99	0.0.0.0	0
3	10.0.0.99	14	89.1.1.42	84

Taula 3.2: Instància de la taula tcpConnTable

4 L'OID de les columnes es forma a partir del de tcpConnEntry que és 1.3.6.1.2.1.6.13.1.

Bé, ja tenim la taula definida i representada, però, i com llegim els valors?, perquè ja s'ha comentat que SMI sols permet escalars i una representació de taules amb escalars. La resposta està en que cada cel·la té un OIID únic.

Així doncs, com es forma l'OIID de cada cel·la?, amb el OID de l'objecte que representa més el valors de cada columna que formen la clau primària. Per exemple, l'OIID de la primera cel·la (*tcpConnState=5*) és:



Il·lustració 3.4: OID d'una entrada de la taula tcpConnTable

Cal aclarir que no sempre s'agafa el valor exacte de la instància index. En funció del tipus es segueixen les normes següents:

- *INTEGER:* El valor enter de la instància.
- *Cadenes de longitud fixa:* Cada caràcter de la cadena es codifica amb el codi del caràcter i constitueix un subidentificador. Per exemple, la cadena "ex. cad." es codificaria com "101.120.46.32.99.97.100.46".
- *Cadenes de longitud variable:* Igual que les cadenes de longitud fixa afegint-hi al principi un subidentificador que indica la longitud de la cadena. Per exemple, la cadena "ex." es codificaria com "3.101.120.46".
- *OBJECT-IDENTIFIER:* En aquest cas s'agafa l'OID i al davant se li afegeix un subidentificador que indica la longitud de l'OID. Per exemple, "1.2.3" es codificaria com "3.1.2.3".
- *IpAddress:* El valor de la instància.

Ara ens podríem preguntar: i com sabrem quin és el valor de les columnes que formen la clau primària, si per obtenir el seu valor es requereix el seu valor?. La resposta és simple: no es pot.

El problema s'ha plantejat així amb el propòsit de saber com es forma l'OIID de cada fila. El funcionament no obstant és diferent, no és el gestor que interroga a l'agent pel valor d'una cel·la, perquè de fet ni tant sols sap si aquella cel·la existeix. El gestor el que fa és demanar instàncies d'un objecte (*tcpConnState, tcpConnLocalAddress, ...*), més endavant és veurà com, i amb els OIID obtinguts és capaç de composar la taula donat que cada OIID conté, a més a més de la definició de l'objecte, la seva posició a la taula.

3.3 Seguretat SNMP

La gestió de seguretat d'SNMP es troba ubicada en el nodes a ser gestionats, és a dir, és l'agent qui disposa de la configuració de seguretat. Així doncs, cada agent té configurat quins gestors el poden atacar, quina informació de la seva MIB està disponible per a cada gestor i quin grau d'accés en té (lectura, escriptura).

La gestió de seguretat contempla tres aspectes:

- *Servei d'autenticació*. L'estació gestionada controla quins gestors poden atacar-lo.
- *Política d'accés.* Determina quina informació de la MIB està disponible per a un determinat gestor, també especifica l'accés a la informació (lectura, escriptura).
- *Servei proxy.* Un agent pot actuar com a proxy d'altres estacions gestionades, així doncs, s'implementa configuració de seguretat per als sistemes que pot gestionar el proxy.

3.3.1 Les comunitats

El concepte de comunitat en un entorn SNMP es defineix com la relació entre un agent SNMP i un conjunt d'estacions de gestió SNMP, que defineix unes característiques d'autenticació i control d'accés.

L'agent estableix una comunitat per a cada combinació desitjada d'autenticació i control d'accés, i a cada comunitat se li assigna un nom únic dins l'àmbit de l'agent (*community name*). Les estacions de gestió hauran de proveir el nom de comunitat per a totes les operacions realitzades sobre l'agent.

3.3.2 Servei d'autenticació

El servei d'autenticació es defineix a l'RFC 1157. La metodologia emprada per a l'autenticació és molt trivial. Cada missatge SNMP inclou el nom de comunitat que actua com a clau d'accés i s'assumeix que el missatge és vàlid si l'agent coneix la comunitat.

S'observa la simplicitat del servei d'autenticació, que és limita a una simple paraula clau (el nom de comunitat) i que a més a més viatja per la xarxa sense cap tipus de xifratge. Aquest fet ha portat a que molts administradors de xarxa hagin decidit proveir sols d'operacions de lectura als seus sistemes SNMP.

3.3.3 Política d'accés

Ja s'ha vist que mitjançant la definició de comunitats un agent pot limitar l'accés a unes determinades estacions de gestió, però no s'ha especifica a quines dades i com poden accedir-hi.

Un agent proveeix diferents accessos a les seves dades també mitjançant la definició de comunitats, així, un agent pot definir més d'una comunitat per a una mateixa estació de gestió. Hi ha dos aspectes sobre els quals un agent defineix la política d'accés d'una comunitat:

- *Vistes sobre la MIB*. Subconjunt d'objectes de la MIB.
- *Mode d'accés SNMP*. READ-ONLY o READ-WRITE.

La combinació d'una vista de la MIB i un mode d'accés es denomina perfil de comunitat SNMP (*SNMP community profile*). La política d'accés SNMP (*SNMP access policy*) es basa en l'assignació d'un *SNMP community profile* a cada comunitat.

Per últim determinar com interactua el grau d'accés definit a la MIB per a un objecte en la seva clàusula ACCESS i el mode d'accés definit al *SNMP community profile*.

MIB ACCESS	***Mode d'accés SNMP***	
	READ-ONLY	READ-WRITE
read-only	read-only	
read-write	read-only	read-write
write-only	read-only (depenent amb la implementació)	read-write (depenent amb la implementació)
not accessible	not accessible	

Taula 3.3: Relació entre MIB ACCESS i el mode d'accés SNMP

3.4 Especificació del protocol SNMP

El protocol SNMP s'especifica a l'RFC 1157 de maig de 1990. L'especificació del protocol SNMP, al igual que tot el que envolta SNMP, pretén ser el més simple possible, és per això que les úniques operacions suportades per SNMP són la inspecció i l'alteració de variables.

Específicament, es poden realitzar tres operacions sobre objectes escalars:

- *Get:* permet a una estació de gestió recuperar el valor d'un objecte escalar d'una estació gestionada.
- *Set:* possibilita que una estació de gestió alteri un valor d'un objecte escalar d'una estació gestionada.
- *Trap:* una estació gestionada envia un valor d'un objecte escalar cap a una estació de gestió sense que aquesta el sol·liciti.

No es possible canviar l'estructura d'una MIB agregant o esborrant instàncies d'objectes. Tampoc és possible enviar comandes a una estació gestionada per a que aquesta realitzi una determinada acció. A més a més, sols es proporciona accés a objectes fulla dins de l'arbre d'identificació d'objectes. Es a dir, no és possible accedir a una taula sencera o a una fila mitjançant una operació atòmica. Aquestes restriccions simplifiquen notablement la implementació d'SNMP, però també limiten la capacitat dels sistemes de gestió.

3.4.1 Missatges SNMP

En SNMP la informació s'intercanvia en forma de missatges. Cada missatge inclou un camp *Version* amb el número de versió (0 per SNMPv1), un camp *Community* amb el nom de la comunitat, i un dels cinc tipus d'unitats de dades de protocol (PDU[5]). La figura següent mostra aquesta estructura.

Missatge SNMP

Version (0)	Community	SNMP PDU

PDUs GetRequest, GetNextRequest i SetRequest

PDU type	request-id	0	0	variablebindings

PDU GetResponse

PDU type	request-id	error-status	error-index	variablebindings

PDU Trap

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

5 Normalment per PDU s'identifica el bloc complet d'informació transferida. En el cas d'SNMP això no és així, donat que la PDU identifica sols la informació pròpia de la funció sol·licitada.

variablebindings

name 1	value 1	name 2	value 2	...	name n	value n

Il·lustració 3.5: Format d'un missatge SNMP

Destacar el el format de les PDUs de *GetRequest*, *GetNextRequest* i *SetRequest* és el mateix que el de la PDU de *GetResponse* amb els camps *error-status* i *error-index* sempre a 0. Un cop més, això és així per a reduir el nombre de formats de PDU que ha de gestionar una entitat SNMP i fer d'aquesta manera la gestió més simple.

3.4.1.1 Variable Bindings

Es possible agrupar varies operacions del mateix tipus en un únic missatge. Així, un gestor pot sol·licitar, per exemple, tots els objectes d'un grup en un sol missatge. Aquesta tècnica redueix considerablement la càrrega de xarxa.

3.4.1.2 GetRequest

La PDU GetRequest és emesa per una entitat de gestió amb el propòsit d'obtenir el valor d'una o algunes instàncies d'objectes. Els camps de la PDU són:

- *PDU type:* indica el tipus de PDU, en aquest cas GetRequest.
- *request-id:* identificació de PDU a nivell d'estació de gestió. El *request-id* permet a l'aplicació de gestió relacionar les respostes rebudes amb les comandes fetes.
- *variablebindings:* correspon a la llista d'instàncies sol·licitades. El camp *value* s'assigna a *null*.

L'entitat receptora respon amb una PDU *GetResponse* que conté el mateix *request-id*. L'operació *GetRequest* és atòmica, això vol dir que o es reben tots els valors demanats o no se'n rep cap, per tant, si no es pot retornar el valor d'una instància sol·licitada no se'n retornarà cap.

3.4.1.3 GetNextRequest

La PDU *GetNextRequest* té el mateix patró d'intercanvi i el mateix format que la PDU *GetRequest*. La diferència està en que, l'agent que respon la comanda (*GetRespons*), en lloc de retornar el valor de les instàncies especificades el que retorna és la instància següent i el seu valor. S'observa que en aquest case, a *variablebindings*, també es pot especificar un objecte al que *GetNextRequest* respondrà amb la primeria instància d'aquest.

En aquest cas l'operació també és atòmica, no obstant el risc d'obtenir un error com a resposta és molt més baix, donat que si és demana una instància d'un

objecte i aquesta no existeix senzillament és retornarà la següent dins l'ordre lexicogràfic de la MIB.

Aquesta operació és útil per explorar totes les instàncies d'una branca de la MIB, o fins i tot de tota la MIB. A més a més, és fonamental per a explorar taules. Per a explorar una taula s'utilitzarà *GetNextRequest* per a obtenir els índexs de cada fila,i amb aquests es podrà recuperar la resta de valors de la taula amb *GetRequest*.

3.4.1.4 SetRequest

L'operació *SetRequest* és emesa per una estació de gestió cap un agent amb el propòsit de modificar el valor d'alguna de les instàncies de la MIB de l'agent. El camp *variablebindings* conté l'OIID de les instàncies a modificar i el valor que és vol assignar. Si tot va bé, l'entitat receptora respon amb un *GetResponse* que conté el mateix *request-id* i el mateix *variablebindings*. En aquest cas l'operació també és atòmica, així o s'actualitzen tots el valors o no se n'actualitza cap.

A destacar el tractament de les files d'una taula. Quan es vol inserir una fila en una taula caldrà enviar un *SetRequest* amb un *variablebindings* que contingui com a mínim tots els valors de la clau primària de la fila. No obstant, l'especificació SNMP reserva a la implementació la decisió d'acceptar o rebutjar la inserció d'una fila en una taula.

3.4.1.4.1 Esborrat de files d'una taula

L'operació *SetRequest* permet esborrar les files d'una taula assignant el valors de les columnes que composen la clau primària al valor *invalid*. Recordar que algunes taules de la MIB-II tenen columnes per a especificar l'operació d'esborrat.

3.4.1.4.2 Execució d'accions

SNMP no proporciona mecanismes per a enviar comandes a l'agent amb propòsits d'endegar alguna acció, no obstant, és possible activar alguna variable vigilada per algun altre procés al costat de l'agent i que sàpiga que fer en funció del valor de la susdita variable.

3.4.1.5 Trap

La PDU *Trap* és emesa des de l'agent cap a un o varis gestors sense que cap gestor hagi sol·licitat la tramesa. S'utilitza per a notificar de forma asíncrona a l'estació de gestió el desencadenament d'algun event significatiu. Els camps de la PDU són:

- *PDU type:* indica el tipus de PDU, concretament *Trap*.

- *enterprise:* identifica el subsistema de gestió de xarxa que genera el trap (el valor s'agafa de *sysObjectID* del grup *System*).
- *agent-addr:* l'adreça IP de l'entitat que genera el trap.
- *generic-trap:* un dels tipus de traps predefinits.
- *specific-trap:* un codi que indica de forma especifica la natura del trap.
- *time-stamp:* el temps transcorregut des de que l'entitat que genera el trap es va inicialitzar fins que es genera el trap.
- *variablebindings:* informació addicional relacionada amb el trap.

3.4.2 Suport a nivell de transport

SNMP requereix l'ús d'un servei de transport per a repartir els missatges. El protocol no fa suposicions sobre la confiabilitat o no del servei de transport, ni si el mateix és orientat a connexió o no.

La majoria de les implementacions d'SNMP estan dins l'arquitectura TCP/IP i utilitzen el transport UDP. Dos núemeros de port han estat assignats per a l'ús d'SNMP. Els **agents escolten** comandes *GetRequest*, *GetNextRequest* i *SetRequest* **pel port 161**, mentre que les **estacions de gestió escolten** els missatges *Trap* **pel port 162**.

Donat que el protocol UDP no garantitza que un datagrama arribi al seu destí, SNMP no pot garantitzar la distribució de missatges. Això provoca que l'aplicació que l'està utilitzant ha de prendre les mesures oportunes en el cas produir-se una pèrdua.

En el cas de comandes *GetRequest*, *GetNextRequest* i *SetRequest* l'aplicació esperarà un *GetRespons* i en no produir-se pot prendre accions al respecte.

En el cas dels Traps no hi una forma simple de detectar una pèrdua degut a que no hi ha cap tipus de retorn per aquest tipus de missatge.

3.5 Avantatges i inconvenients d'SNMP

Com a avantatges tenim:

- És un protocol madur, estàndard de facto acceptat per l'industria.
- Està disponible en una gran quantitat de productes.
- És fàcil d'implementar i requereix pocs recursos del sistema.

Com a inconvenients tenim:

- No és adequat per a gestionar xarxes grans degut a que el gran volum de paquets requerit pel el *polling*, pot repercutir en el rendiment de la xarxa.
- No és gens fàcil la recuperació de grans quantitats de dades, com per exemple una taula d'enrutament.
- L'ús de missatges crítics mitjançant un Trap no garanteix que el gestor rebi el missatge donat l'ús del transport UDP.
- Proporciona un servei d'autenticació massa trivial que fa que el seu ús es limiti al monitoratge.
- No permet enviar comandes que desencadenin una acció a l'estació agent.
- No suporta comunicació entre dos aplicacions de gestió. Això impedeix efectuar una gestió jerarquitzada.
- En la definició de taules no obliga a que els índexs identifiquin de forma única una sola fila.

S'observa que els inconvenients són força més que els avantatges, no obstant cal dir que no hi ha una alternativa estandarditzada que pugui substituir SNMP, la qual cosa repercuteix en que la gran majoria de fabricants implementen SNMP en els seus productes, el qual constitueix un gran avantatge en quant a l'abast del protocol sobre els elements d'una xarxa.

D'altra banda també cal avançar que algun d'aquests inconvenients o fins i tot defectes són corregits en posteriors versions d'SNMP.

4 SNMPv2

Per tal de suplir les mancances de la 1a. versió d'SNMP es publicà el nou estàndard SNMPv2. L'especificació detallada es troba en les RFC 1901 a 1908. Les principals millores dutes a terme respecte a SNMPv1 són les següents:

- Nova estructura de la informació de gestió (SMIv2).
- Capacitat de diàleg entre dos estacions de gestió per tal de jerarquitzar la gestió de la xarxa.
- Noves operacions definides al protocol.
- Nova base d'informació de gestió (MIB SNMPv2).

En aquest capítol es reflectiran les principals diferències envers la versió 1, així doncs, en ocasions, per a obtenir una documentació més completa d'aquesta versió caldrà completar amb la documentació de la primera versió.

4.1 SMI SNMPv2

La nova estructura d'informació de gestió (SMI SNMPv2 o SMIv2) és basa en l'SMI d'SNMP, i sobre aquesta primera versió treballa els punts següents:

- Definició d'objectes. Es detalla al punt 4.1.1.
- Gestió de taules. Es detalla al punt 4.1.1.1.
- Definicions de notificació. L'RFC 1902 defineix la macro NOTIFICATION-TYPE que s'utilitza per a definir la informació que s'enviarà en cas de succeir un determinat event (trap).
- Mòduls d'informació. Un mòdul d'informació especifica un grup de definicions relacionades.

4.1.1 Sintaxi d'objecte

Al igual que en SNMPv1, per a la definició d'objectes s'utilitza la macro OBJECT-TYPE definida en ASN.1. Les principals aportacions de la nova versió són les següents:

- S'inclou la clàusula opcional *UNITS*, la qual permet especificar les unitats de mesura de l'objecte.
- S'inclou la clàusula *MAX-ACCESS*. Aquesta clàusula és té el mateix paper que la clàusula *ACCES* d'SNMPv1, no obstant els valors permesos canvien. Els valors possibles són *not-accessible, accessible-for-notify, read-only, read-write, read-create*.
- Els valors de la clàusula *STATUS* passen a ser *current, obsolete, deprecated*.
- Es permeten les enumeracions.
- Es defineixen nous tipus de dades. Tot seguit és mostra una taula amb els nous tipus de dades.

Tipus	***Descripció***
Unsigned32	Enter comprés entre 0 i 2^{32}-1.
Counter32	Similar al tipus *Counter* d'SMIv1 amb la diferència que no té valor inicial. Per la seva natura no sols té sentit mesurar l'increment entre dos valors.
Counter64	Ídem al tipus *Counter* però amb 64 bits.
Gauge32	Similar al tipus *Gauge* d'SMIv1. La diferència radia en que es pot definir un valor màxim inferior a 2^{32}-1.

Taula 4.1: Tipus de dades afegits a SMI SNMPv2

4.1.1.1 Gestió de taules

SNMPv2 millora les especificacions donades per a les taules a l'RFC 1212 i a l'RFC 1757 (RMON) tot facilitant la creació i l'esborrat de files així com també l'accés a les mateixes. Les noves especificacions contemplen la definició de dos tipus de taules: les que permeten a un gestor la creació i esborrat de noves files i les que no.

El procés de definició de taules és el mateix que per a SNMPv1, però cal tenir present els següents aspectes nous:

- Les columnes que formen la clau primària de la taula, és a dir els camps de la clàusula *INDEX*, han d'identificar de forma unívoca cada fila.

- La clàusula *INDEX* permet l'atribut opcional *IMPLIED*. Aquest atribut s'aplica a l'últim camp que composa l'índex i té com a propòsit simplificar l'OIID de la cel·la. L'atribut actua en funció del tipus de camp de la següent manera:
  - *Cadena de longitud variable:* No és necessari posar el subidentificador de longitud de cadena.
  - *OBJECT-IDENTIFIER:* No és necessari posar el subidentificador de longitud.
  - Sobre la resta de tipus aquest atribut no té cap efecte.
- Apareix la clàusula *AUGMENTS* com alternativa a la clàusula *INDEX*. Aquesta clàusula s'utilitza per a crear noves taules en base a d'altres amb el propòsit d'afegir columnes a la taula. En la definició de la fila s'utilitza *AUGMENTS*, tot especificant la fila base, amb el que la clau primària és la mateixa que la de la taula base.
- Els objectes que formen part de la clau primària no són accessibles.

4.1.1.1.1 Creació de files

Per a que una taula permeti la creació i eliminació de files ha de tenir obligatòriament un objecte columna de tipus *RowStatus* i el valor de *MAX-ACCESS* ha de ser *read-create*. Els possibles valors de la columna d'estat són:

- *active:* La fila es troba disponible per a ser treballada per un gestor.
- *notInService:* La fila existeix però no es troba disponible per a que l'agent la pugui utilitzar.
- *notReady:* La fila existeix però algunes de les seves columnes no tenen encara informació. No està disponible per a que l'agent l'utilitzi.
- *createAndGo:* Valor que un gestor envia per a la columna d'estat quan vol crear una fila amb el mètode *createAndGo*. Remarcar que la columna d'estat no agafa aquest valor.
- *createAndWait:* Valor que un gestor envia per a la columna d'estat quan vol crear una fila amb el mètode *createAndWait*. Remarcar que la columna d'estat no agafa aquest valor.
- *destroy:* Valor que envia un gestor per a la columna d'estat quan vol esborrar la fila.

Com ja es deu poder intuir per a la creació d'una fila és disposa de dos mètodes *createAndWait* i *createAndGo*. Tot seguit es detalla cada mètode.

4.1.1.1.1.1 createAndWait

Els passos que és segueixen són els següents:

1. El gestor assigna la columna d'estat al valor *createAndWait*. L'OIID de la columna d'estat estarà composat per l'OID de l'esmentada columna més la part de clau primària, per tant, implícitament, també s'està donant els valors de les columnes índex.

   1.1. En resposta l'agent crea la nova fila i assigna els valors per defecte per a cada columna. Si totes les columnes tenen valor la columna d'estat s'inicialitza a *notInService*, altrament *notReady*.

2. El gestor llegeix tots els camps de la nova fila.

   2.1. En resposta l'agent retorna el valor de totes les columnes de la fila. Les columnes que encara no tinguin valor prendran el valor *noSuchInstance*. Si la taula té columnes que l'agent no pot gestionar, aleshores aquestes columnes tindran el valor *noSuchObject*.

3. El gestor dona valor, com a mínim, a les columnes no inicialitzades.

   3.1. Si totes les files ja tenen valor, aleshores la columna d'estat passa al valor *notInService*.

4. El gestor posa la columna d'estat al valor *active*.

   4.1. L'agent activa la fila tot posant el valor de la columna estat a *active*. A partir d'aquest moment la fila ja està plenament disponible per al gestor.

4.1.1.1.1.2 createAndGo

Aquest mètode és més simple que l'anterior, però té dos limitacions: els objectes de la fila s'han de poder enviar en una sola PDU i el gestor no coneix els valors per defecte dels objectes. Els passos que es segueixen són:

1. El gestor, en una sola operació d'escriptura, assigna totes les columnes de la fila. A la fila d'estat se li dóna el valor *createAndGo*.

   1.1. En resposta l'agent crea la fila. Donat que totes les columnes tenen valor la fila automàticament s'activa, és a dir, la columna estat passa a tenir el valor *active*.

4.1.1.1.2 Esborrat de files

Per a esborrar una fila el gestor simplement ha d'assignar la columna d'estat al valor *destroy*. En resposta, l'agent esborrarà la fila sol·licitada.

4.2 Especificació del protocol SNMPv2

L'especificació del protocol SNMPv2 no aporta grans diferències respecte a la versió 1. Les característiques a destacar del nou protocol són:

- Es permet la comunicació entre gestors. SNMPv1 sols permetia comunicacions del tipus Gestor-Agent i Agent-Gestor, i SNMPv2 afegeix el tipus Gestor-Gestor. Això permet jerarquitzar la gestió de la xarxa.
- Es donen algunes diferències en les PDUs dels missatges. Al punt següent és detalla amb més precisió.

4.2.1 Missatges SNMPv2

L'intercanvi de dades a SNMPv2 es produeix a través de missatges al igual que a SNMPv1. Les diferències aportades per SNMPv2 són:

- El camp versió dels missatges agafa el valor 1.
- Les operacions d'obtenció de dades *GetRequest* i *GetNextRequest* deixen de ser atòmiques. Així, si un objecte no es troba es retornarà *notSuchObject* i si el nom no és correcte o no és visible *notSuchInstance.*
- *La PDU GetResponse passa anomenar-se Response*, tot mantenint el seu format idèntic al de la versió SNMPv1.
- S'afegeixen dos noves operacions: *GetBulkRequest* i *InformRequest*. Aquestes noves PDUs es veuran per separat més endavant.
- Desapareix la PDU trap i apareix l'operació *SNMPv2-Trap*. Més endavant es detalla l'operació.

Tot seguit és mostrà una imatge amb les PDUs d'SNMPv2.

Missatge SNMPv2

Version (1)	Community	SNMPv2 PDU

PDUs GetRequest, GetNextRequest, SetRequest, SNMPv2-Trap i InformRequest

PDU type	request-id	0	0	variablebindings

PDU Response

PDU type	request-id	error-status	error-index	variablebindings

PDU GetBulkRequest

PDU type	request-id	non-repeaters	max-repetitions	variablebindings

variablebindings

name 1	value 1	name 2	value 2	...	name n	value n

Il·lustració 4.1: Format d'un missatge SNMPv2

4.2.1.1 GetBulkRequest

És la millora més important aportada per SNMPv2 a l'especificació de protocol perquè la nova operació permet recuperar un gran nombre d'informació amb un sol missatge, minimitzant així l'intercanvi d'aquests.

L'operació *GetBulkRequest* presenta un funcionament similar a *GetNextRequest* donat que actua sobre la instància següent a un objecte o instància donat. La diferència rau en que *GetBulkRequest* permet retornar un nombre R de successors, mentre que *GetNextRequest* sempre és limita a 1.

El camp *variablebindings* inclou una llista de N+R noms de variable. Per als primers N noms s'opera igual que *GetNextRequest*. Per als R noms restants es retornen múltiples successors lexicogràfics.

En aquesta PDU es troben dos camps que no es troben a la resta de PDUs: *non-repeaters* i *max-repetitions*. El primer especifica el nombre de variables de la llista *variablebindings* per a les que s'ha de retornar un sol successor lexicogràfic. El segon determina la quantitat de successors lexicogràfics que s'han de tornar per a la resta de variables especificades a *variablebindings*.

4.2.1.2 SNMPv2-Trap

Aquesta operació te la mateixa funció que l'operació *trap* d'SNMPv1, però presenta un format diferent. Aquest format és el mateix que altres PDUs com per exemple *GetRequest*, facilitant així la tasca de processament al receptor. El camp *variablebindigns* conté els valors següents:

- *sysUpTime.0 i snmpTrapOID.0*.
- Si la clàusula *OBJECT* és present a la invocació corresponent de la macro *NOTIFICATION-TYPE*, aleshores cada variable i el seu valor es copien al camp *variablebindings*.
- L'agent pot incloure variables addicionals.

Al igual que a SNMPv1, l'agent no rep confirmació de l'arribada del trap.

4.2.1.3 InformRequest

Aquest missatge s'envia entre dos entitats SNMPv2 amb el rol de gestor. La PDU d'aquesta operació és igual a la d'altres com *GetRequest*. El camp *variablebindings* conté els mateixos elements que *SNMPv2-Trap*.

Quan es rep una PDU d'aquest tipus, l'entitat receptora, prèvia comprovació de que la longitud de la resposta no excedeix el valor màxim permès, retorna una PDU de resposta amb el mateix *request-id* i els mateixos camps rebuts al *variablebindings* de la petició *InformRequest*.

4.2.2 Coexistència amb SNMPv1

Per a que gestors SNMPv2 puguin operar amb agents SNMPv1 es requereix introduir un proxy entre ambdues entitats donat que algunes PDUs, com s'ha vist, presenten algunes diferències i d'altres són totalment noves.

4.3 Avantatges i inconvenients

SNMPv2 no soluciona tots els problemes d'SNMPv1, no obstant si que millora aspectes com:

- Millora en la definició i gestió de taules.
- Facilita el transport de volums d'informació grans.
- Permet jerarquitzar la gestió de xarxes a través de la comunicació entre gestors. Això permet distribuir la càrrega de gestió de xarxa tot limitant la despesa de recursos necessaris per a la gestió.

No obstant, encara hi ha presents deficiències com:

- L'ús de missatges crítics mitjançant un Trap no garanteix que el gestor rebi el missatge donat l'ús del transport UDP.
- Proporciona un servei d'autenticació massa trivial que fa que el seu ús es limiti al monitoratge.
- No permet enviar comandes que desencadenin una acció a l'estació agent.

Tot i aquests inconvenients, podem dir el mateix que es va dir per a la primera versió. L'ampli ús d'SNMP per part dels fabricants, juntament amb la seva senzillesa, fan d'SNMP una excel·lent eina per a la gestió de xarxa.

5 SNMPv3

El gran problema d'SNMPv2 encara era la seguretat. Sobre aquest estandard es fan propostes que desencadenen SNMPv2u i SNMPv2*, i finalment, a partir d'ells, al 1998 sorgeix SNMPv3, definit als RFCs 2271-2275.

D'SNMPv3 cal destacar:

- La seva gran aportació és la seguretat a SNMP.
- Defineix una arquitectura modular per a la implmentació d'entitats.
- Les PDUs d'SNMPv2 continuen sent vigents. SNMPv3 no defineix cap nova PDU, el que fa és definir un marc de treball que permet afegir seguretat a les PDUs d'SNMPv2.
- "SNMPv3 és SNMPv2 més seguretat i administració".

SNMPv3 brinda seguretat als seus missatges en els aspectes següents:

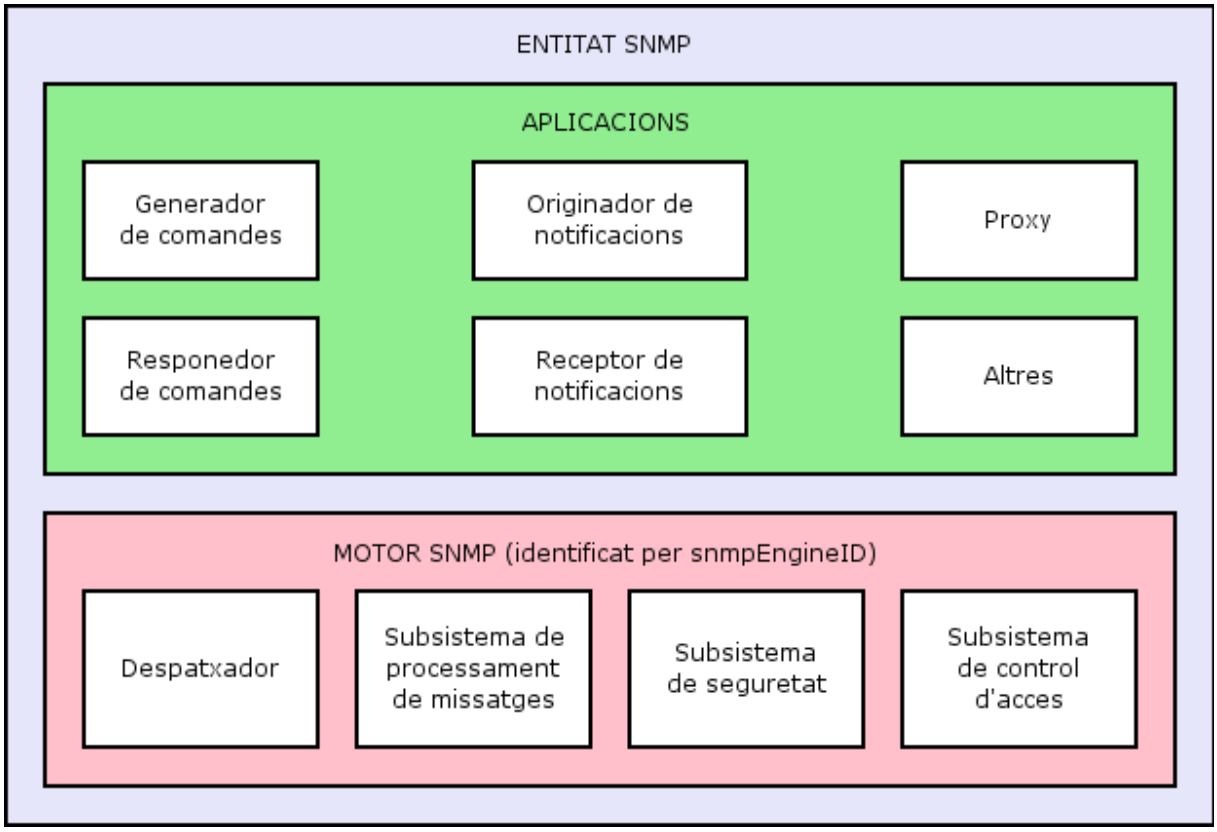
- Modificació de la informació.
- Suplantació d'identitat.
- Modificació del flux de missatges.
- Revelació de continguts.

5.1 Arquitectura SNMPv3

A les versions prèvies d'SNMP es parlava d'entitats SNMP, ja fossin agents, proxies o gestors, sense entrar en com es constituïen aquestes entitats. Cada entitat actuava com una caixa negra que havia de respondre a unes especificacions de comportament donades. SNMPv3 destapa aquestes caixes negres, les entitats, tot definint una arquitectura modular de components per a la implementació d'entitats. Aquesta nova estructura modular permet actualitzar cada mòdul per separat sense haver de modificar l'estàndard al complet.

El rol d'agent o gestor per part d'una entitat és defineix en funció dels mòduls que implementa. Cada entitat SNMP es compon d'un conjunt d'aplicacions i d'un motor SNMP que implementa les funcions d'enviar i rebre missatges, autenticar i xifrar/desxifrar missatges, i controlar l'accés als objectes gestionats.

Tot seguit es mostra una imatge amb els possibles mòduls d'una entitat SNMP.



Il·lustració 5.1: Entitat SNMPv3

Les funcionalitats que ha de dur a terme cada mòdul són:

- *Generador de comandes:* Inicia la generació de PDUs *GetRequest*, *GetNextRequest*, *GetBulkRequest* i/o *SetRequest*, i en processa les PDUs de resposta *Response*.
- *Responedor de comandes:* Rep les PDUs *GetRequest*, *GetNextRequest*, *GetBulkRequest* i/o *SetRequest* adreçades al motor SNMP local, les aten, i genera les PDUs de resposta corresponents (*Response*).
- *Originador de notificacions:* Monitoritza en un sistema l'esdeveniment d'events o condicions particulars i genera les PDUs *SNMPv2-Trap* i/o *InformRequest.* Processa la PDU *Response* en resposta a *InformRequest*.
- *Receptor de notificacions:* Escolta permanentment per l'arribada de notificacions (PDUs *SNMPv2-Trap* i *InformRequest*) i en genera la PDU de resposta (Response) a PDUs *InformRequest*.
- *Proxy:* Reenvia missatges SNMP.
- *Despatxador:* S'encarrega de la tramesa i recepció de missatges i PDUs entre la xarxa i els diferents mòduls de l'entitat.

- *Subsistema de processament de missatges:* Per a missatges sortints prepara les PDUs per a la seva tramesa dins un missatge i per a missatges entrants extreu les PDUs. Permet suportar una o més versions d'SNMP.
- *Subsistema de seguretat:* Proporciona els serveis de seguretat (autenticació i xifrat) al subsistema de processament de missatges. Implementa el "model de seguretat basat en usuaris" (USM). En un futur podria implementar altres models de seguretat.
- *Subsistema de control d'accés:* Proporciona un conjunt de serveis d'autorització per a comprovar els drets d'accés a la MIB d'una entitat. Implementa el "model de control d'accés basat en vistes" (VACM). En un futur podria implementar altres models de control d'accés.

Molt resumidament, s'ha d'entendre que la comunicació entre aplicacions i el motor SNMP es fa mitjançant PDUs SNMPv2. Les aplicacions generen PDUs SNMPv2 i les envien al motor SNMP. El motor SNMP afegeix a aquestes PDUs un envolcall amb forma de missatge d'SNMPv3 (més endavant es veurà) i les envia sobre un datagrama UDP. El motor SNMP també rep missatges, els processa tot verificant-ne l'autenticitat i n'extrau la PDU corresponent que es enviada a les aplicacions.

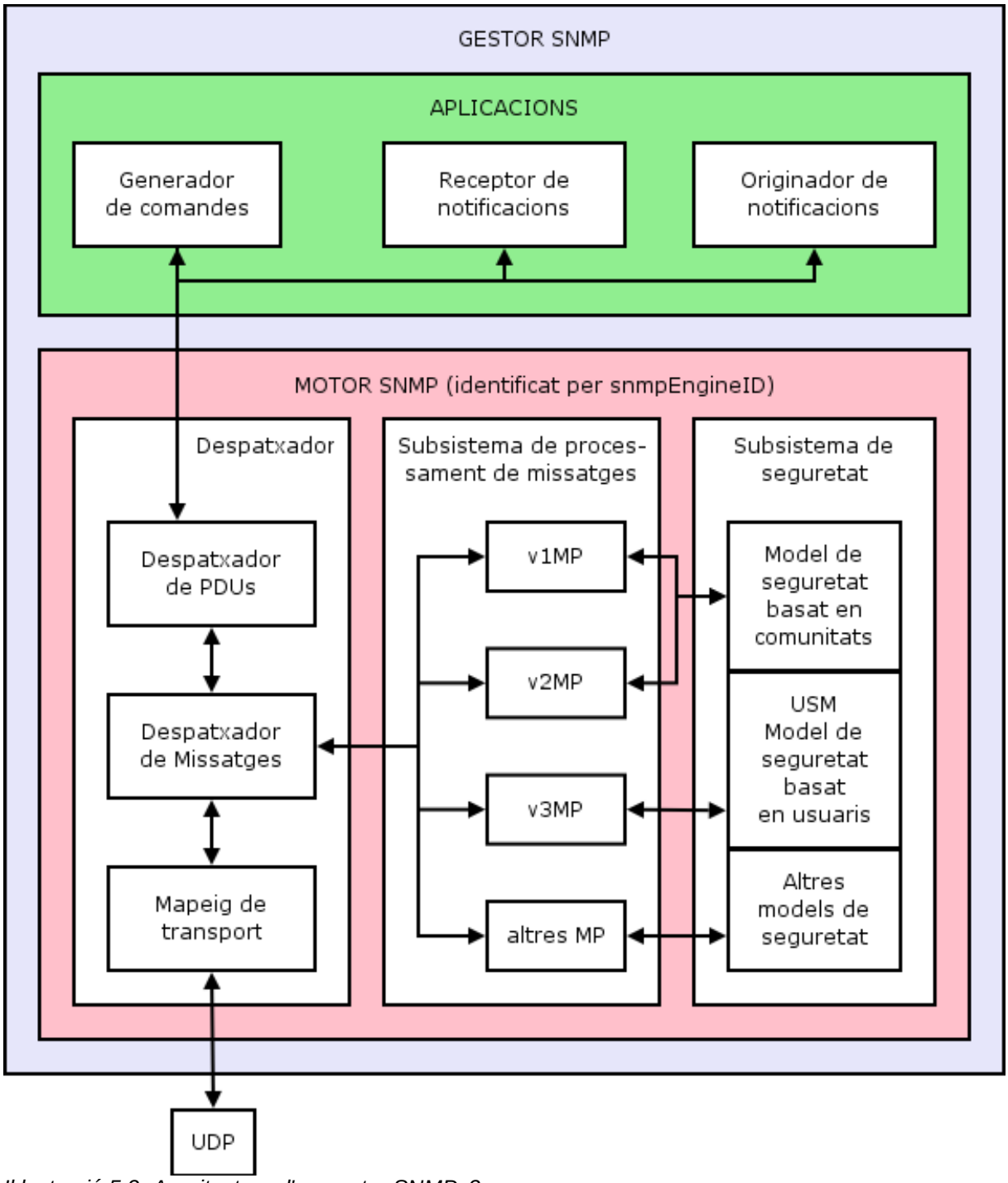
La comunicació entre mòduls s'efectua mitjançant funcions de baix nivell anomenades ***primitives*** que són independents de la implementació. En aquest document no s'entrarà amb el detall de les primitives donat que responen al funcionament intern de les entitats SNMPv3 i el propòsit del document és tenir una visió d'SNMPv3 per tal de poder utilitzar-lo. Així, es presenta simplement la relació de primitives de cada mòdul, per tal de donar una idea d'aquestes.

- *Despatxador:* sendPdu, processResponsePdu, processPdu, returnResponsePdu, registerContextEngineID, unregisterContextEngineID.
- *Subsistema de processament de missatges:* prepareOutgoingMessage, prepareDataElements, prepareResponseMessage.
- *Subsistema de seguretat:* generateRequestMessage, processIncominMessage, generateResponseMessage.
- *Subsistema de control d'accés:* isAccesAllowed.

Per últim, dir que el protocol de transport utilitzat per a trametre els missatges normalment és UDP, però també podria ser IPX o d'altres. És fa aquest petit incís perquè en les imatges que es donaran tot seguit es mostrarà UDP com a únic transport. Això s'ha fet així perquè aquest és el més utilitzat amb diferència.

5.1.1 Arquitectura d'un gestor

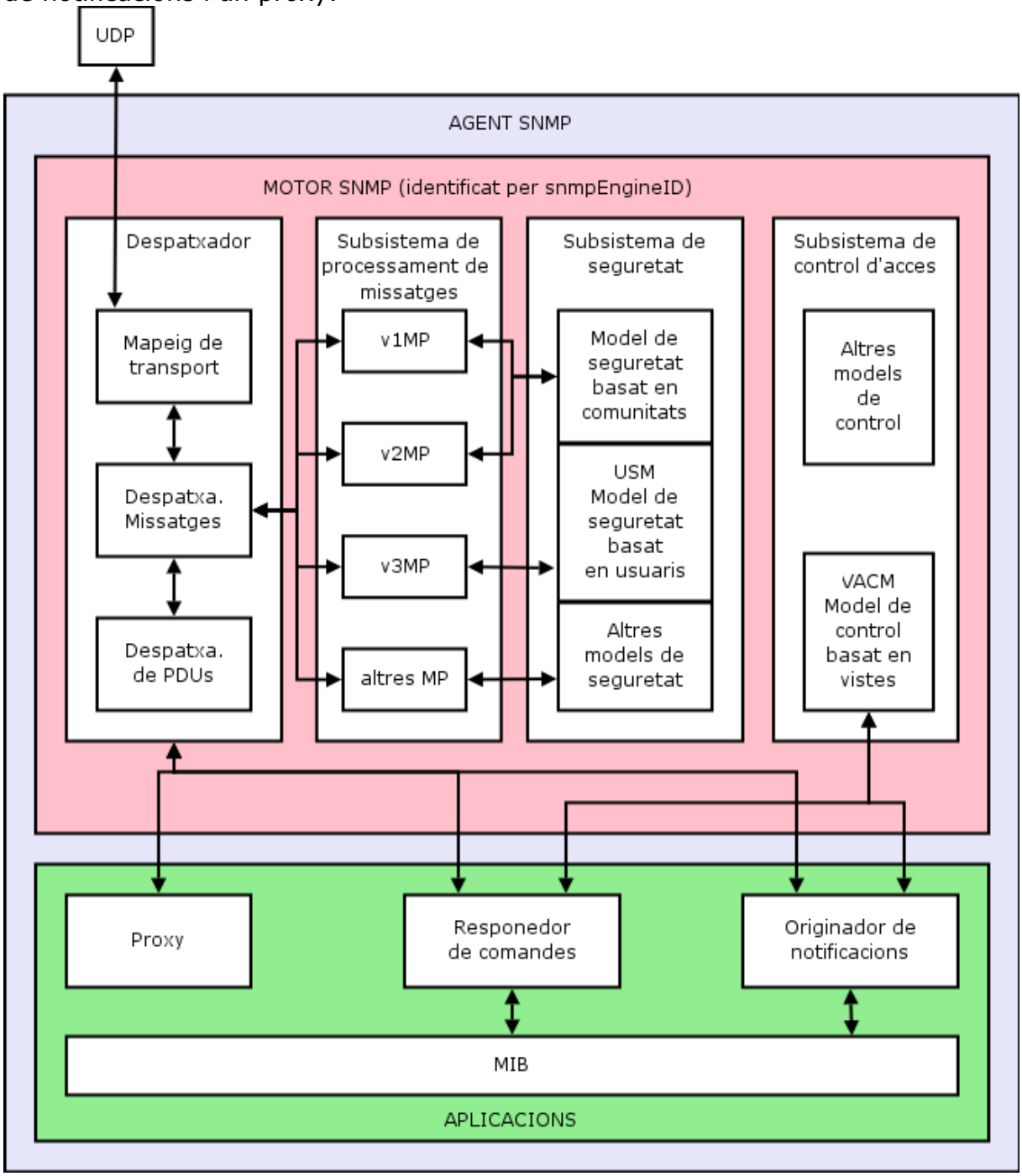
Un gestor SNMP convencional inclou les aplicacions de generació de comandes, generació de notificacions i recepció de notificacions. El motor SNMP inclou els mòduls despatxador, subsistema de processament de missatges i de seguretat.



Il·lustració 5.2: Arquitectura d'un gestor SNMPv3

5.1.2 Arquitectura d'un agent

Un agent SNMPv3 convencional inclou els mòduls despatxador, subsistema de processament de missatges, subsistema de seguretat i subsistema de control d'accés. També inclou una aplicació responedora de comandes, una generadora de notificacions i un proxy.



Il·lustració 5.3: Arquitectura d'una agent SNMPv3

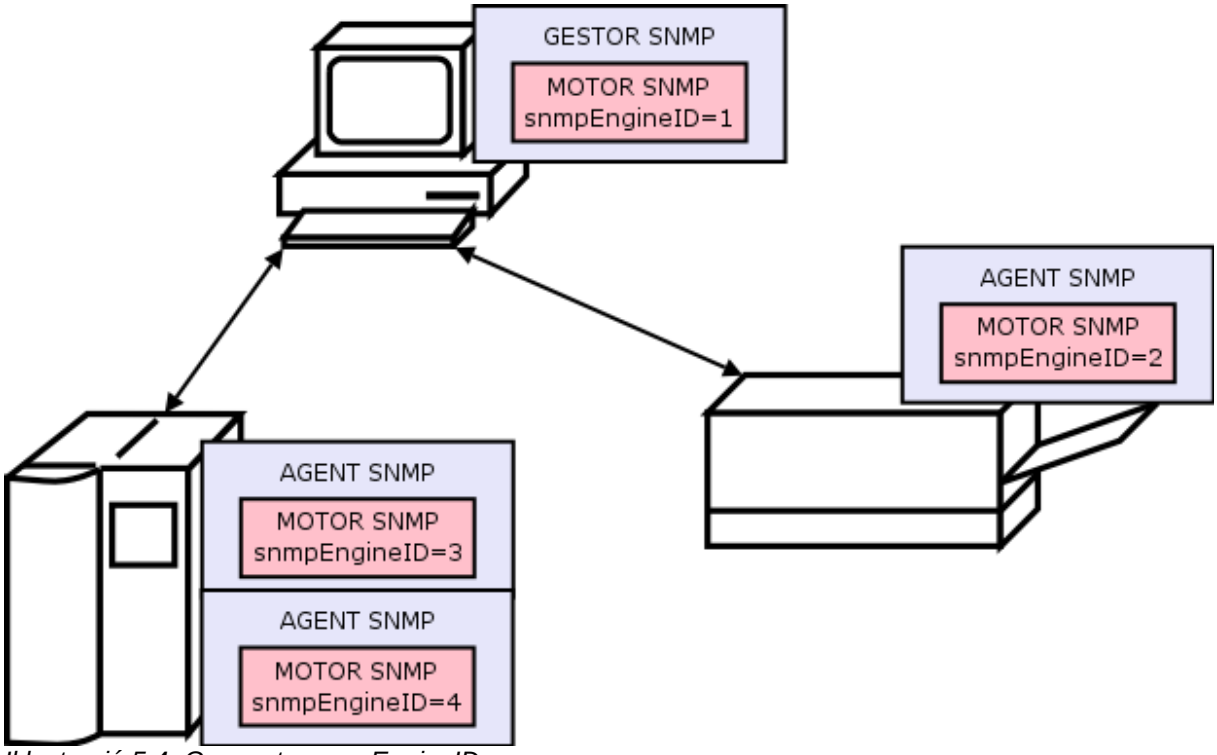
5.2 Especificació del protocol SNMPv3

5.2.1 Terminologia

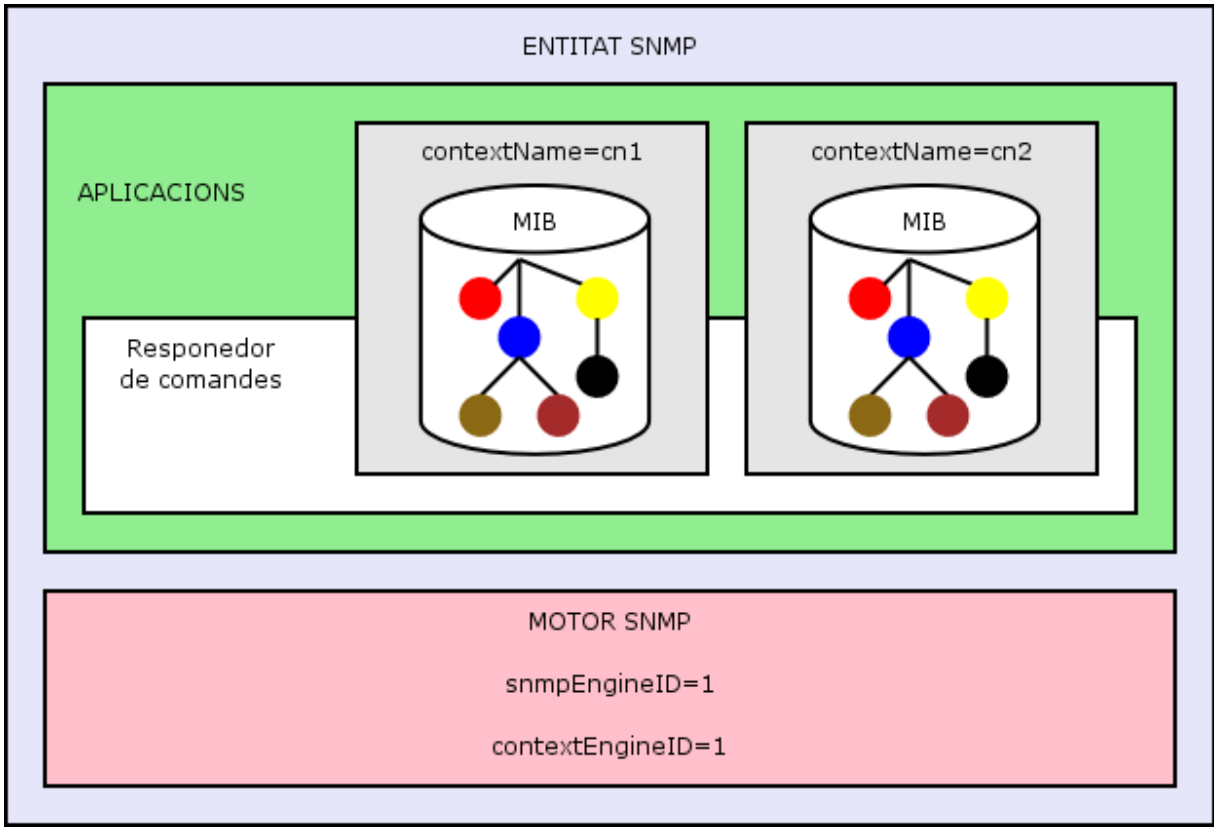
Abans d'endinsar-nos en el protocol SNMPv3 es fa un petit glossari de noms que fan referència a elements de l'arquitectura d'SNMPv3. Aquest conceptes s'utilitzaran més endavant i cal tenir clar que és cadascun d'ells.

- *snmpEngineID:* Identificador únic i no ambigu d'un motor SNMP i de l'entitat a que correspon el susdit motor.
- *contextEngineID:* Identificador únic d'una instància de context d'una entitat SNMP amb un *contextName* concret. Certament la definició no és gens aclaridora. A la pràctica s'implementa com un camp d'un missatge SNMPv3 i correspon al *snmpEngineID*.
- *contextName:* Identifica un context concret dins un motor SNMP.
- *scopedPDU:* Bloc de dades que conté un *contextEngineID*, un *contextName*, i una PDU SNMPv2. Es passa com a paràmetre des de/cap al subsistema de seguretat.
- *snmpMessageProcessingModel:* Identificador únic d'un model de processament de missatges del subsistema de processament de missatges. Els valors possibles són SNMPv1, SNMPv2c i SNMPv3.
- *snmpSecurityModel:* Identificador únic d'un model de seguretat del subsistema de seguretat. Els valors possibles són SNMPv1, SNMPv2c i USM.
- *snmpSecurityLevel:* El nivell de seguretat amb el que un missatge s'envia o amb el que una operació s'ha de processar. Els valors possibles són noAuthnoPriv (no autenticació, no xifrat), authNoPriv (autenticació, no xifrat) i authPriv (autenticació, xifrat).
- *principal:* El concepte de *principal* es podria entendre com l'actor que sol·licita un servei a un altra entitat. En SNMPv3 es considera que el missatges s'envien en nom d'algú, aquest algú és el *principal*. Un principal pot ser un individu actuant en un rol concret, un conjunt d'individus, cadascun dels quals actuaria en un determinat rol, un conjunt d'aplicacions o una combinació dels descrits.
- *securityName:* Un cadena llegible que representa al principal.

Per acabar amb la terminologia, s'ofereixen unes il·lustracions per tal d'aclarir alguns dels conceptes.



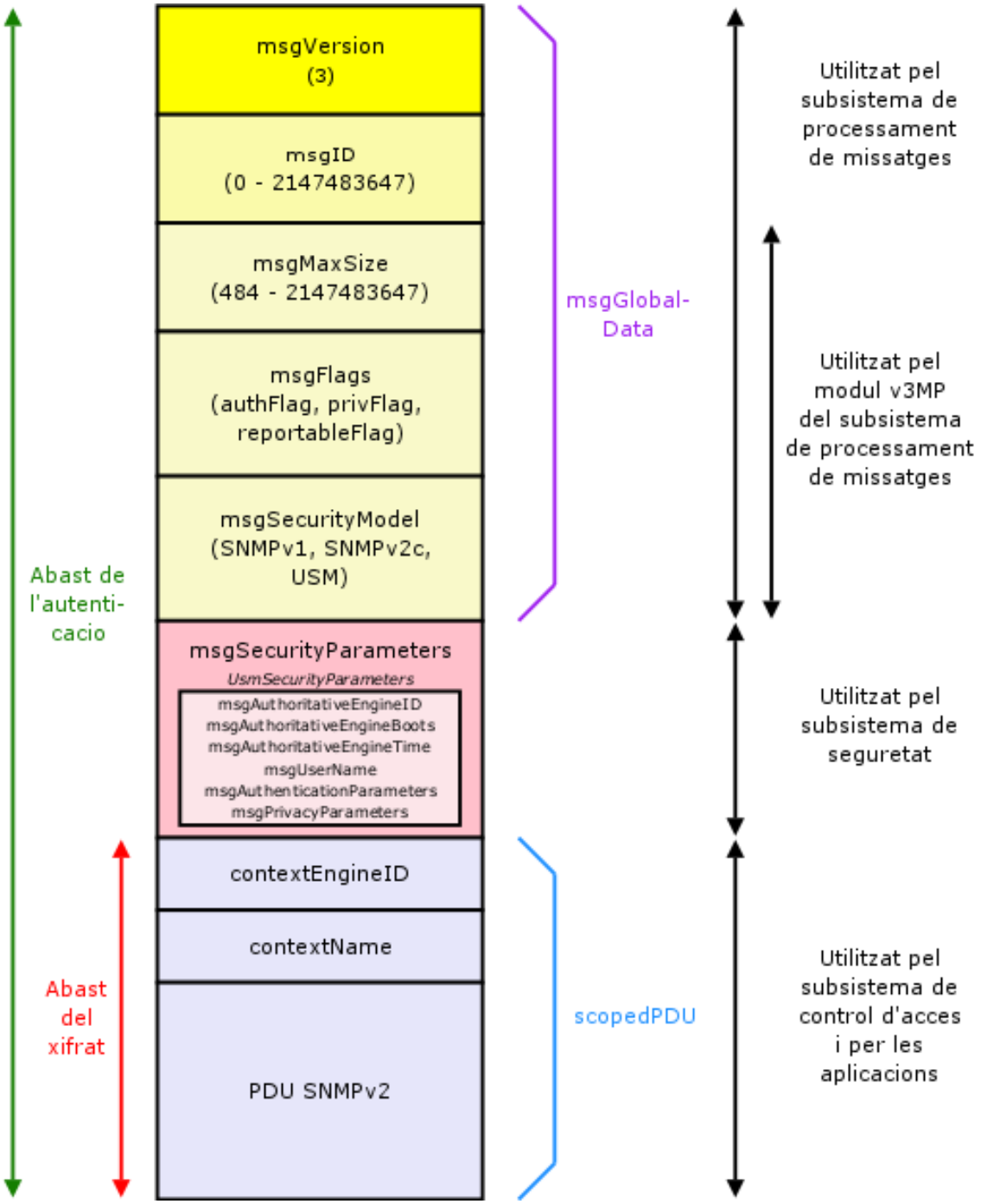
Il·lustració 5.4: Concepte snmpEngineID



Il·lustració 5.5: Conceptes contextName i contextEngineID

5.2.2 Missatges SNMPv3

Al igual que a les versions prèvies d'SNMP, SNMPv3 utilitza missatges per a intercanviar la informació entre les entitats SNMP. Cada missatge inclou un encapçalament de missatge i una PDU. L'estructura del missatge es mostra a l'il·lustració següent.



Il·lustració 5.6: Format d'un missatge SNMPv3

Tot seguit és descriu breument els camps que formen el missatge:

- *msgVersion i msgID:* Igual que a les versions 1 i 2 d'SNMP.
- *msgMaxSize:* Mida màxima de missatge (*bytes*) que suporta l'emisor.
- *msgFlags:* Conté 3 flags als bits menys significatius.
  - *authFlag:* Aplicar autenticació al missatge.
  - *privFlag:* Aplicar xifrat al missatge.
  - *reportableFlag:* Igual a 1 si cal generar una PDU de resposta a la recepció d'aquest missatge.

    Els bits *autFlag* i *privFlag* permeten definir els tres nivells de seguretat: *noAuthNoPriv*, *authNoPriv*, *authPriv.*
- *msgSecurityModel:* Identificador que indica el model de seguretat emprat per al missatge. Els possibles valors són 1 per SNMPv1, 2 per SNMPv2 i 3 per USM.
- *msgSecurityParameters:* Paràmetres del subsistema de seguretat. És descriu amb més detall a l'apartat 5.4.1.4.
- *contextEngineID:* Identificador únic de l'entitat SNMP que ha de processar el missatge entrant, altrament dit, *contextEngineID* de l'entitat SNMP destí.
- *contextName: contextName* a l'entitat SNMP destí.
- *PDU:* Una PDU SNMPv2.

L'especificació del protocol SNMPv3 es completa amb el model de seguretat USM descrit al punt següent en el context de seguretat SNMPv3.

5.3 Seguretat SNMPv3

El punt fort d'SNMPv3 és l'aportació de seguretat a SNMP, intentant solventar les deficiències de les versions anteriors. SNMPv3 defineix dos aspectes relacionats amb la seguretat:

- Model de seguretat basat en l'usuari (USM, *User-based Security Model*)
- Model de control d'accés basat en vistes (VACM, *View-based Acces Control Mode*)

5.3.1 Model de seguretat basat en l'usuari (USM)

L'USM es defineix a l'RFC 2274. Aquesta especificació avarca:

- *Autenticació:* Integritat de dades i autenticació de l'origen de les dades.
- *Privacitat:* Protegeix el missatge front la lectura de tercers.
- *Puntualitat:* Protecció contra el retard o reenviament de missatges.
- *Format del missatge:* Defineix l'estructura *UsmSecurityParameters* que suporta les funcions d'autenticació, puntualitat i privacitat.
- *Descobriment:* Defineix procediments mitjançant els qual un motor SNMP obté informació d'un altre motor SNMP.
- *Gestió de claus:* Defineix procediments per a la generació, actualització i ús de les claus.

5.3.1.1 Motor autoritzat

Abans de continuar cal introduir la definició de motor autoritzat. En qualsevol transmissió de missatges, una de les dos entitats es designa com a motor autoritzat i l'altra com a motor no autoritzat. Per a fer-ho es segueixen les regles següents:

- Quan el missatge SNMP conté una PDU que genera una resposta (*GetRequest*, *GetNextRequest*, *GetBulkRequest*, *SetRequest* o *InformRequest*), el receptor del missatge constitueix el motor autoritzat.
- Quan el missatge SNMP conté una PDU que no genera una resposta (*SNMPv2-Trap* o *Response*), el transmissor del missatge constitueix el motor autoritzat.

Fixem-nos que en el cas típic d'un gestor que interroga un agent, l'agent constitueix el motor autoritzat.

5.3.1.2 Funcions criptogràfiques

USM té definides dos funcions criptogràfiques: autenticació i xifratge. Per a suportar aquestes funcions, un motor SNMP requereix dos valors: un clau d'autenticació (*authKey*) i una de xifrat (*privKey*). Els valors d'aquestes claus es mantenen pels usuaris següents:

- *Usuaris locals:* Qualsevol principal en aquest motor SNMP per a qui les operacions de gestió estan autoritzades.
- *Usuaris remots:* Qualsevol principal en un motor SNMP remot per a qui es permet la comunicació.

Fixem-nos que es treballa amb criptografia de clau simètrica. Els valors *privKey* i *authKey* no són accessibles mitjançant SNMP.

Per a l'autenticació USM permet la utilització de dos protocols: HMAC-MD5-96 i HMAC-SHA-96.

En quant al xifrat, USM utilitza el mode *cipher block chaining* (CBC) del *Data Encryption Standart* (DES).

5.3.1.3 Mecanismes de puntualitat (Timeliness)

El conjunt de mecanismes de puntualitat USM inclou:

- *Gestió de rellotges autoritzats:* Tots els motors que poden actuar com a autoritzats han de mantenir dos objectes, *snmpEngineBoots* i *snmpEngineTime*, que indiquen el temps local.
- *Sincronització:* Un motor SNMP no autoritzat ha de mantenir-se aproximadament sincronitzat amb el motors SNMP autoritzats amb que es comunica. Per a fer-ho manté una copia local de tres variables per a cada motor conegut: *snmpEngineBoots*, *snmpEngineTime* i *latestRecivedEngineTime*. Inicialment, un motor no autoritzat pot obtenir els valors *snmpEngineBoots* i *snmpEngineTime* d'un motor autoritzat, mitjançant un procés de descobriment. A partir d'aquí la sincronització és manté a mesura que arriben missatge del motor autoritzat, i en absència d'aquests es fa una estimació.
- *Test de puntualitat:* SNMPv3 determina que un missatge s'ha de rebre dins una finestra de temps raonable, per a evitar els atacs per retard o reenviament de missatges. Fixem-nos que en una transmissió el motor no autoritzat té una estimació del temps del motor autoritzat, així, si la transmissió té l'origen al no autoritzat, el missatge portarà una estimació del temps de l'autoritzat, i a l'inrevés, si l'origen és l'autoritzat, el missatge portarà el temps d'aquest. El receptor podrà comprovar el temps que ha trigat el missatge i en general es considerarà que un missatge no pot trigar més de 150 segons en anar del receptor a l'emissor.

5.3.1.4 Format del missatge (UsmSecurityParameters)

Es defineix l'estructura *UsmSecurityParameters* que és instanciada pel camp *msgSecurityParameters* d'un missatge SNMPv3. Aquesta estructura dona cabuda als paràmetres d'autenticació, privacitat i puntualitat. Els atributs de l'estructura són els següents:

- *msgAuthoritativeEngineId:* El *snmpEngineID* del motor SNMP autoritzat.
- *msgAuthoritativeEngineBoots:* El *snmpEngineBoots* del motor SNMP autoritzat.

- *msgAuthoritativeEngineTime:* El *snmpEngineTime* del motor SNMP autoritzat.
- *msgUserName:* L'usuari (principal) en nom de qui s'està fent l'intercanvi.
- *msgAuthenticationParameters:* És un paràmetre d'autenticació. Mitjançant el mecanisme d'autenticació utilitzat permet validar l'autenticitat del missatge. Si no s'utilitza autenticació és NULL.
- *msgPrivacityParameters:* Paràmetre de xifrat. Si no s'utilitza és NULL.

5.3.1.5 Descobriment

USM requereix de l'ús d'un procés de descobriment per a obtenir informació d'altres motors SNMP, per a poder-se comunicar amb ells. Per a fer-ho, és produeix un intercanvi de missatges, amb origen al motor no autoritzat, per tal d'obtenir la informació requerida del motor autoritzat.

5.3.1.6 Gestió de claus

Per a utilitzar els serveis d'autenticació i privacitat que ofereix SNMPv3, basats en criptografia de clau simètrica, és necessari compartir les claus secretes entre un principal en un motor no autoritzat i el motor autoritzat. La RFC 2274 proporciona mecanismes per a la creació, actualització i gestió d'aquestes claus.

Per a simplificar la gestió de claus als principals, cadascun d'ells ha de mantenir una única clau d'autenticació i una de xifrat. A més a més, aquestes claus és poden derivar d'una password.

Per tal de distribuir les claus d'un principal entre entitats, i no haver d'introduir aquesta clau en cada entitat, SNMPv3 defineix el concepte de clau localitzada. L'objectiu és que un usuari hagi de mantenir una sola clau (o dos, si es requereix autenticació i xifrat), per accedir a tots els agents. El procés mitjançant el qual una clau és transforma en múltiples claus úniques, una per a cada motor SNMP remot, és denomina localització de claus. La clau és mapeja en diferents claus localitzades mitjançant una funció one-way no reversible (funció hash segura).

5.3.2 Model de control d'accés basat en vistes (VACM)

El model de control d'accés *View-Based* (VACM) es defineix a l'RFC 2275. Aquest model té dos característiques a destacar:

- El VACM determina quan s'ha de permetre l'accés a un objecte de la MIB local des d'un principal remot.
- El VACM fa ús d'una MIB que defineix la política de control d'accés per a l'agent, i fa possible la utilització de la configuració remota.

5.3.2.1 Elements del model VACM

A l'RFC 2275 es defineixen cinc elements que formen part del model: grups, nivells de seguretat, contexts, vistes i polítiques d'accés. Tot seguit es descriurà cadascun d'ells.

Per a gestionar la configuració de control d'accés s'utilitza la MIB VACM. Aquesta MIB consta de 4 categories:

- Contexts locals (*vacmContextTable*).
- Grups (*vacmSecurityToGroupTable*).
- Drets d'accés (*vacmAccessTable*).
- MIB views (grup *vacmMIBViews*).

5.3.2.1.1 Grups

Un grup s'identifica mitjançant un nom de grup (*groupName*) i es defineix com un conjunt de zero o més tuples (*securityModel*, *securityName*), que especifiquen qui i com pot accedir als objectes gestionats per l'entitat. El paràmetre *securityModel* es refereix al model de seguretat que pot ser SNMPv1, SNMPv2 o USM. El paràmetre *securityName* es refereix a un principal. Cada combinació *securityModel*, *securityName* sols pot pertànyer a un grup.

5.3.2.1.2 Nivell de seguretat

Els missatges SNMPv3 especifiquen el nivell de seguretat desitjat, així, els drets d'accés per a un grup dependrà del nivell de seguretat del missatge. Per exemple, un agent pot permetre accés de lectura per a missatges sense autenticació, però requerir autenticació per a les comandes d'escriptura.

5.3.2.1.3 Contexts

Un context MIB s'utilitza per a referenciar un conjunt d'instàncies d'objectes de la MIB local. Els contexts són molt útils per a definir polítiques d'accés a col·leccions d'objectes. Els contexts tenen les característiques següents:

- Una entitat SNMP, identificada unívocament per *contextEngineID*, pot disposar de més d'un context.
- Un objecte o instància pot aparèixer en més d'un context.
- Quan una instància d'un objecte pot pertànyer a més d'un context, també cal especificar el *contextName* i *contextEngineID.*

No s'ha d'entendre un context com una agrupació d'objectes, aquest senzillament indica una casuística d'utilització de la MIB local.

5.3.2.1.4 MIB views

La MIB view es defineix com una col·lecció, o família, de subarbres, on cada subarbre pot estar inclòs o exclòs de la MIB view. S'utilitza per a restringir l'accés a determinats objectes a un grup concret.

Per cada fila de la taula *vacmAccesTable* hi ha tres MIB views associades, una per a cada tipus d'accés: lectura, escriptura i notificació.

5.3.2.1.5 Política d'accés

Finalment, la política d'accés als objectes de la MIB local dependrà de:

- L'estació de gestió (principal) que està efectuant la comanda.
- El nivell de seguretat amb el que s'ha tramés el missatge SNMPv3.
- El model de seguretat utilitzat per a processar el missatge.
- El context MIB utilitzat.
- La instància d'objecte específica per a la que s'efectua la comanda.
- El tipus d'accés requerit (lectura, escriptura o notificació).

5.3.2.2 Processament del control d'accés

Les definicions donades per separat no donen una idea de conjunt del processament del control d'accés. Per a veure-ho tot més clar és planteja una possible VACM MIB i l'algoritme utilitzat per a determinar el control d'accés als objectes. Les columnes subratllades per a les taules constitueixen l'índex

vacmContextName

vacmContextName
""

vacmSecurityToGroupTable

vacmSecurityModel	*vacmSecurityName*	*vacmGroupName*
3 (USM)	initial	initial

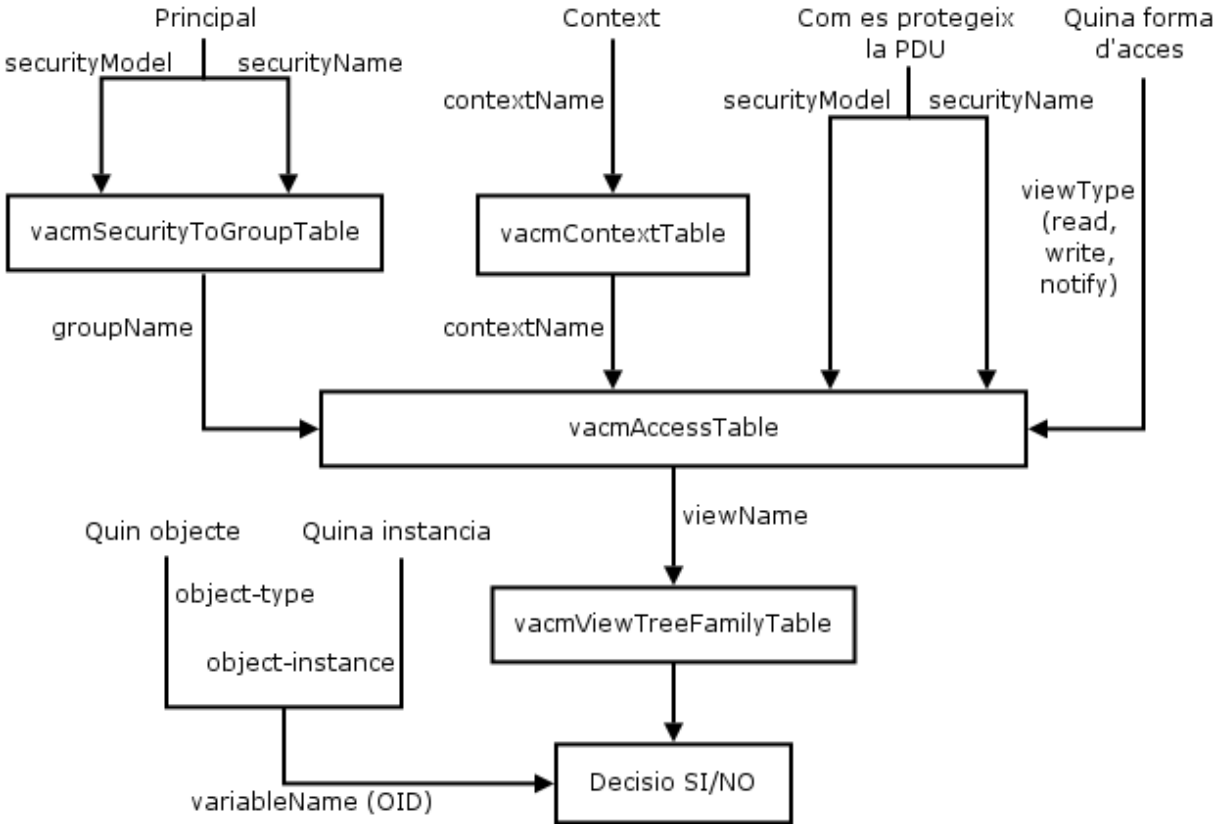
vacmAccessTable

vacmGroup Name	*vacmAcces ContextPrefix*	*vacmAccess SecurityModel*	*vacmAccess Security Level*	*vacmAccess ReadView Name*	*vacmAccess WriteView Name*	*vacmAccess NotifyView Name*
initial	""	3 (USM)	noAuthNoPriv	restricted	""	restricted
initial	""	3 (USM)	authNoPriv	internet	internet	internet
initial	""	3 (USM)	authPriv	internet	internet	internet

vacmViewTreeFamilyTable

vacmViewTreeFamily Name	*vacmViewTreeFamily Subtree*	*vacmViewTreeFamily Mask*	*vacmViewTreeFamily Type*
internet	1.3.6.1	“”	1 (inclòs)
restricted	1.3.6.1.2.1.1	“”	1 (inclòs)
restricted	1.3.6.1.2.1.11	“”	1 (inclòs)
restricted	1.3.6.1.6.3.7.2.1	“”	1 (inclòs)

Il·lustració 5.7: Exemple de VACM MIB



Il·lustració 5.8: Lògica VACM de processament

5.4 Avantatges i inconvenients

SNMPv3 aporta les següents millores front les versions anteriors:

- Defineix una arquitectura modular que defineix perfectament una entitat SNMP, alhora que permet evolucionar una part d'SNMP sense desencadenar una nova versió de tot el protocol.
- Aporta seguretat a la transmissió de dades entre entitats, tot incloent autenticació i xifrat.
- Defineix un nou sistema de control d'accés a les dades molt més potent i configurable que les versions anteriors.
- La nova arquitectura permet conviure SNMPv3 amb les versions anteriors.
- Els avantatges de les versions 1 i 2 continuen presents a SNMPv3.

No obstant, encara hi ha presents alguns inconvenients:

- L'ús de missatges crítics mitjançant un Trap no garanteix que el gestor rebi el missatge donat l'ús del transport UDP.
- No permet enviar comandes que desencadenin una acció a l'estació agent.
- No està tant implementat per part dels fabricants com la versió SNMPv2c, segurament per la gestió de seguretat que en aquesta versió esdevé un element força pesat de gestionar.

Tot i aquests inconvenients, SNMPv3 defineix una molt bona arquitectura que li ofereix la possibilitat d'evolucionar amb facilitat. A més a més, l'arquitectura preveu la convivència amb entitats de versions anteriors. Aquests aspectes fan que sigui idoni per a la implementació d'un gestor SNMP. No oblidem tampoc, que en essència, SNMPv3 manté la senzillesa de les versions anteriors en quant a la manera de gestionar la informació.

2a. part

Llibreria NetSNMP.NET

6 ANÀLISI I DISSENY

6.1 Antecedents

La segona part d'aquest document es centra exclusivament en l'aplicació de la teoria vista sobre SNMP, concretament s'analitzarà, dissenyarà i implementarà una llibreria SNMP.

Analitzant les llibreries SNMP disponibles en el mercat, s'ha trobat que és un sector poc difós, i en conseqüència les bones llibreries, que implementen tota la funcionalitat d'SNMP en les seves tres versions, són de llicència propietària. Les llibreries trobades amb llicència de lliure ús i distribució han estat molt poques i sols en una d'elles s'implementa SNMPv3, la resta es limiten a SNMPv1 i SNMPv2 i no sempre en la seva totalitat.

La millor llibreria, amb diferència, d'entre totes les de llicència de lliure ús i distribució, ha estat Net-SNMP (http://net-snmp.sourceforge.net). Aquesta llibreria implementa les tres versions d'SNMP en la seva totalitat, està àmpliament provada donat que els seus orígens s'ubiquen a l'any 1992 i a més a més està disponible per a diverses plataformes, entre elles Unix, Linux i Windows. Aquests factors la situen com una de les millors llibreries, no obstant té un problema: el seu ús.

La llibreria està escrita en C, i el seu ús és força complicat. Aquí no entrarem en anàlisi de tendències de programació, no obstant és indiscutible que els llenguatges orientats a objecte s'estan imposant clarament front els llenguatges estructurats, motiu pel qual és un problema l'ús d'aquesta llibreria en entorns orientats a objecte. A més a més, i a títol d'opinió personal, sempre he estat un entusiasta del C, no obstant aquest és un entusiasme poc compartit, factor que també dificulta l'ús de la llibreria. Per a solucionar aquest problema la llibreria té recobriments a altres llenguatges com Perl, PHP, Java, Tk/Tcl o Python.

Tornant a les llibreries amb llicència de lliure ús i distribució, comentar que per a la plataforma .NET no s'ha trobat cap llibreria amb unes garanties mínimes de funcionament. En quant a les llibreries amb llicència propietària per a .NET, dir que són molt poques i poc provades.

Així doncs, la trama està servida ..., l'escenari és idoni. Disposem d'una llibreria excel·lent, però que no es pot utilitzar en entorns .NET, a més a més, no hi ha cap llibreria amb llicència lliure per aquesta plataforma, així doncs tot porta cap a un recobriment per a .NET de la llibreria Net-SNMP.

6.2 Requeriments

Els requeriments per a la nova llibreria són els següents:

1. Ha de suportar sessions SNMPv1, SNMPv2 i SNMPv3, acomplint a la perfecció les especificacions dictades en els seus estandards.
2. Com a mínim ha d'implementar les operacions *GetRequest*, *GetNextRequest*, *GetBulkRequest*, *SetRequest* i òbviament *GetResponse*.
3. Per a sessions SNMPv3 ha d'implementar com a mínim el model de seguretat USM.
4. Les operacions especificades s'han de poder cridar de forma síncrona o asíncrona.
5. S'ha d'aconseguir que el seu ús sigui extremadament senzill, amb un bon recobriment orientat a objectes. Cal abstraure amb la màxima precisió possible el conceptes SNMP per a portar-los al model de negoci de la llibreria.
6. El tractament d'objectes i instàncies SNMP s'ha de fer d'una forma àgil, de tal manera que la tasca d'instanciar objectes de la MIB sigui el més simple possible.
7. La llibreria ha de funcionar en entorns POSIX, concretament s'ha d'assegurar el seu correcte funcionament en GNU/Linux. També s'ha de considerar que, donat el cas, ha de resultar força fàcil portar-la a sistemes Windows.
8. Ha d'estar programada per a la plataforma .NET per tal de cobrir l'absència de llibreries SNMP amb llicència de lliure ús i distribució que existeix en aquest entorn.
9. S'ha de lliurar amb llicència de lliure ús i distribució.
10. La llibreria ha d'estar degudament documentada, aportant eines per a consultar fàcilment aquesta documentació.

6.3 NetSNMP.NET

En base als antecedents exposats i als requeriments sol·licitats, la solució passa per fer un recobriment a la llibreria Net-SNMP per a la plataforma .NET. Aquest recobriment ha d'assolir com a mínim els requeriments indicats en els punts 1, 2, 3 i 4.

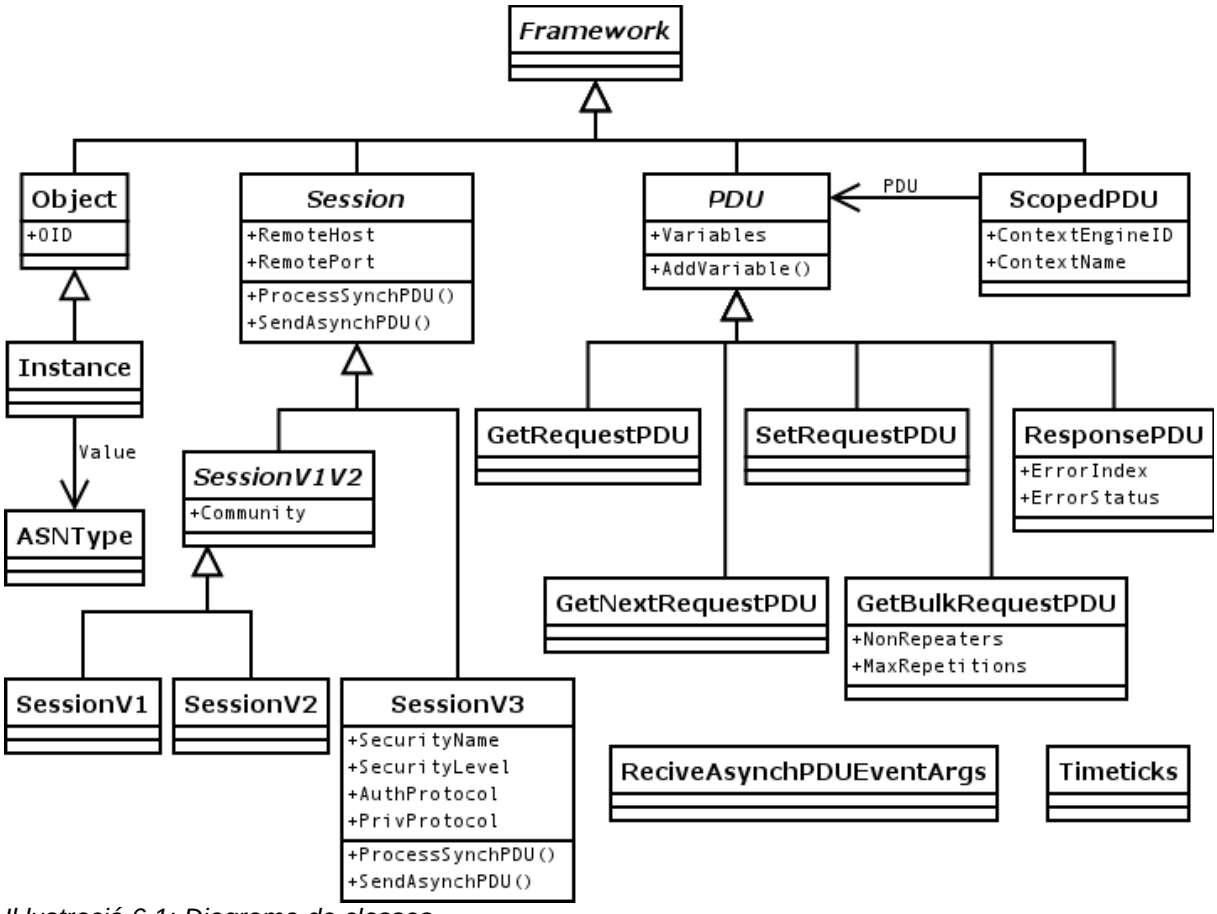
Com ja es pot intuir, la nova llibreria s'anomenarà NetSNMP.NET. A partir d'ara, per tal de no crear confusió, sempre que ens adrecem a la llibreria Net-SNMP escrita en C, ho farem amb el nom de Net-SNMP.C o simplement llibreria nativa, mentre que per indicar la nova llibreria, la desenvolupada en aquest projecte, parlarem de NetSNMP.NET.

NetSNMP.NET es programarà per a la plataforma .NET en llenguatge C#, concretament s'utilitzarà l'entorn .NET lliure Mono. El seu desenvolupament es durà a terme sota el sistema operatiu GNU/Linux, i les proves de la llibreria es faran en aquest mateix entorn. Amb això es cobriran els requeriments 7 i 8.

Per últim, NetSNMP.NET, tal i com especifica el requeriment 9, es lliurarà sota llicència *GNU General Public License* (GNU GPL).

6.4 Model de negoci

El model de negoci de la llibreria ha de proveir d'una bona definició de classes per tal d'assolir els requeriments 5 i 6. Tot seguit s'indica un primer diagrama de classes de la llibreria.



Il·lustració 6.1: Diagrama de classes

7 IMPLEMENTACIÓ

7.1 Plataforma operativa

La plataforma requerida per al desenvolupament de NetSNMP.NET consta de les parts següents:

- *Sistema operatiu GNU/Linux*. Qualsevol distribució serà bona, concretament la llibreria es desenvoluparà sobre la distribució *Debian Sid*.
- *Binaris de la llibreria Net-SNMP.C versió 5.2.1 o superior.* NetSNMP.NET funciona sobre Net-SNMP.C, d'aquí que sigui imprescindible tenir instal·lada la llibreria Net-SNMP.C.
- *Capçaleres de programació de la llibreria Net-SNMP.C versió 5.2.1 o superior*. Tot i que NetSNMP.NET estarà programada majoritàriament en C#, algunes parts de la llibreria estaran programades en C, motiu pel que es necessari tenir instal·lades les capçaleres de Net-SNMP.C.
- *Entorn d'execució .NET Mono versió 1.1.10 o superior.* Mono constituirà el CLR (C*ommon Language Runtime*) per al programari .NET.

A l'annex a NetSNMP.NET es donen detalls sobre com instal·lar els components.

7.2 Entorn de desenvolupament

La plataforma requerida per al desenvolupament de NetSNMP.NET consta de:

- *Compilador de C# Mono versió 1.1.10 o superior.*
- *Compilador de C gcc versió 4 o superior*.
- *IDE Monodevelop versió 0.9 o superior*. Es pot substituir aquest IDE per qualsevol altre com per exemple Eclipse, no obstant NetSNMP.NET es desenvoluparà amb *Monodevelop*.
- *Monodoc versió 1.1.10 o superior*. Eina per a construir i visualitzar la documentació.
- *Monodocer*. Eina per a generar una documentació preliminar.

A l'apartat "Instal·lació" es donen detalls sobre com instal·lar els components.

7.3 Estructuració de NetSNMP.NET

7.3.1 Projectes

La solució NetSNMP.NET consta de 4 projectes, aquests són:

- *NetSNMP.Native.* Constitueix el recobriment de la llibreria Net-SNMP.C a C#. Consta de diverses classes per accedir a les estructures natives, més la classe Invoke que engloba totes les crides a funcions de la llibreria nativa. El projecte genera la llibreria NetSNMP.Native.dll.
- *libnetsnmp_to_dotnet.* Llibreria escrita en C que dóna suport a NetSNMP.Native. La llibreria nativa Net-SNMP.C principalment s'ataca des de NetSNMP.Native, no obstant, algunes de les funcionalitats de Net-SNMP.C, requereixen d'un tractament especial que justifica aquesta llibreria. El projecte s'ha de compilar amb l'script build_lib.sh, que forma part del mateix projecte, des de la línia de comandes. La llibreria generada és libnetsnmp_to_dotnet.so.
- *NetSNMP*. Constitueix la llibreria SNMP que utilitzarà l'usuari (programador) final. El projecte defineix tota l'estructuració de classes de la llibreria NetSNMP.NET. NetSNMP utilitza NetSNMP.Native per accedir a Net-SNMP.C, ocultant així al programador la complexitat de Net-SNMP.C, alhora que oferint un model de classes fàcil d'utilitzar i entenedor. El projecte genera la llibreria NetSNMP.dll.
- *DocGen*. Generació de documentació. El projecte consta d'un únic fitxer script que s'encarrega d'iniciar la documentació per a *Monodoc* amb la utilitat *monodocer*. La documentació generada a partir d'aquí es completa amb l'eina *Monodoc*.
- *Test*. Aquest projecte conté diversos exemples d'utilització de la llibreria. Aquests exemples constitueixen un bon punt de partida per estudiar la utilització de la llibreria.

7.3.2 Arbre de directoris

Tot el projecte s'engloba dins el directori NetSNMP.NET, dins trobarem:

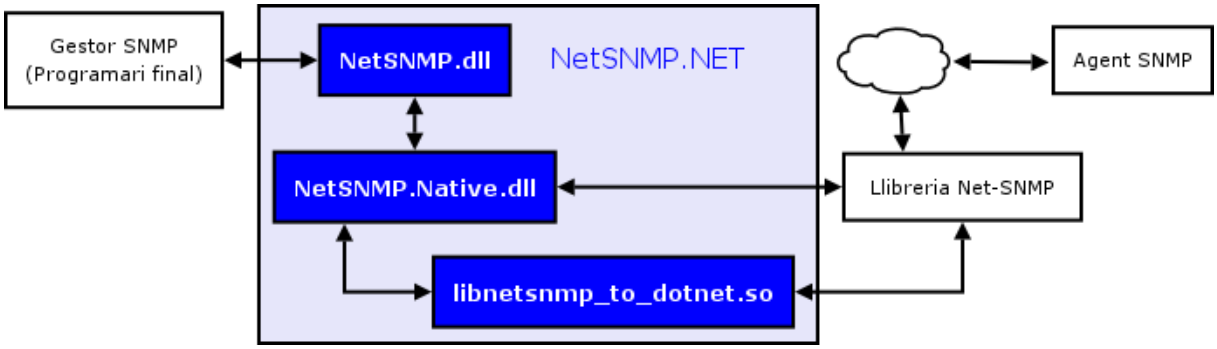
- *NetSNMP.NET.mds*. Fitxer solució de *Monodevelop*. Obrint aquest fitxer amb l'IDE *Monodevelop* accedirem a l'entorn de programació format pels 4 projectes descrits. S'observa que no tots els projectes són de C#, no obstant es presenten tots 4 sota la mateixa solució.
- *NetSNMP.Native.* Directori del projecte NetSNMP.Native.

- *libnetsnmp_to_dotnet*. Directori del projecte libnetsnmp_to_dotnet.
- *NetSNMP*. Directori del projecte NetSNMP.
- *DocGen*. Directori del projecte DocGen.
- *Test*. Directori del projecte Test.
- *doc*. Directori de documentació. Aclarir que DocGen tant sols genera l'esquelet de la documentació en aquest directori. A partir d'aquí la documentació es treballa en aquest directori amb l'eina *Monodoc*.
- *bin*. Directori de binaris. Els projectes NetSNMP.Native, libnetsnmp_to_dotnet, NetSNMP i Test generen els seus binaris aquí.

7.3.3 Espais de noms

- *NetSNMP.Native*. Aquest espai de noms engloba tot el recobriment per a la llibreria nativa Net-SNMP.C. L'usuari final no ha d'utilitzar per a res aquest espai de noms, sent aquest d'ús intern per a NetSNMP.NET. Es desenvolupa íntegrament en el projecte NetSNMP.Native i en conseqüència, s'ubica exclusivament a la llibreria NetSNMP.Native.dll.
- *NetSNMP.Constants.* Espai de noms per a la definició de constants utilitzades en l'entorn SNMP. Es programa al projecte NetSNMP.Nativa i per tant es troba a la llibreria NetSNMP.Native.dll.
- *NetSNMP*. Espai de noms principal de NetSNMP.NET. Engloba tota la funcionalitat que ofereix la llibreria, així, l'usuari final ha d'utilitzar aquest espai de noms. Es desenvolupa íntegrament en el projectes NetSNMP, i en conseqüència, s'ubica a la llibreria NetSNMP.dll.

7.3.4 Interacció de components



Il·lustració 7.1: Interacció de NetSNMP.NET en una aplicació SNMP

El diagrama mostra com interaccionen els components de NetSNMP.NET entre ells, i també com ho fan amb altres mòduls d'una aplicació SNMP.

Conclusions i bibliografia

8 CONCLUSIONS I LÍNIES DE FUTUR

El primer que podem dir d'SNMP és que és un bon protocol per a la gestió de xarxes TCP/IP. Característiques de les versions 1 i 2 com:

- Simplicitat d'ús i implementació.
- Consum de pocs recursos i ample de banda.
- Implementació dels grans fabricants en els seus aparells.
- Bona portabilitat entre diferents marques.
- Facilitat per a la definició d'objectes propis de monitoratge i control.

fan d'SNMP una excel·lent eina per a la gestió de xarxes.

D'altra banda ens trobem front un protocol amb desenvolupament dinàmic i fàcilment escalable. Així és demostra en les successives versions del protocol que han evolucionat per a cobrir les necessitats del mercat i superar les seves mancances.

La tercera versió d'SNMP acaba de madurar plenament el protocol dotant-lo de:

- Bona gestió de seguretat amb la implementació del model de seguretat basat en l'usuari (USM) i el control d'accés basat en vistes (VACM).
- Una excel·lent arquitectura modular que defineix perfectament totes les parts d'una implementació SNMP. Aquesta estructura permet que fàcilment pugui evolucionar cada mòdul per separat sense haver de definir una nova versió del protocol i mantenint així la compatibilitat.

Remarcar també que SNMP, a més a més de mirar endavant, sempre ha tingut cura del passat en la seva evolució, tot permetent la coexistència entre versions. Tot i així SNMP encara manté algunes mancances com la impossibilitat de confirmar la recepció d'una comanda *Trap* o desencadenar una acció a través d'una comanda.

Finalment, en quant a la llibreria NetSNMP.NET, s'estudiarà la possibilitat de publicar el projecte a SourceForge.net per tal de continuar el desenvolupament. Juntament amb això, donar coneixement de NetSNMP.NET, com a recobriment de la llibreria Net-SNMP, a la comunitat de desenvolupament del projecte Net-SNMP. Remarcar que per emprendre aquestes accions caldria millorar NetSNMP.NET amb aspectes com implementació de *Traps* o gestió d'excepcions.

9 BIBLIOGRAFIA

9.1 Documents impresos

- William Stalling. *SNMP, SNMPv2, SNMPv3 and RMON 1 and 2*. Tercera edició. Addison-Wesley, 1996. ISBN 0-201-48534-6.
- Marshall T.Rose; Keith McCloghrie. *How to manage your network using SNMP*. Primera edició. Prentice-Hall, Inc., 1995. ISBN 0-13-141517-4.

9.2 Documents electrònics

- RFC 1155, 1157, 1212, 1213, 1354, 1573, 1643, 1901-1908, 2271-2275. The RFC Archive.
- NET-SNMP.
- Javier Fernández-Sanguino Peña. "Artículo para la revista Linux Actual número 17: Gestión SNMP con Linux"
- "Understanding Network Performance".
- SNMPLink.org.
- Javi Tobal. SNMP.
- PAESSLER. PRTG Traffic Grapher.
- CSCARE.Inc. Active SNMP.
- Ben Rockwood. "The Cuddletech Guide to SNMP Programming".

Annexos a Estudi d'SNMP

10 La MIB II (SNMPv1)

La MIB-II es defineix a l'RFC 1213. En aquest apartat es donarà un visió global de la MIB-II, per aprofundir en la seva estructura es recomana consultar l'RFC 1213. S'ha escollit aquesta MIB per a donar un exemple de MIB perquè és una de les MIB més utilitzades.

La MIB-II succeeix la MIB-I tot afegint-hi alguns objectes sobre la primera. Ambdues MIB tenen el mateix OID, això significa que les dues alhora no poden coexistir en un mateix sistema.

La MIB-II defineix 10 grups amb funcions específiques:

- *system*
- *interfaces*
- *at*
- *ip*
- *icmp*
- *tcp*
- *udp*
- *egp*
- *dot3 (transmissió)*
- *snmp*

Tot seguit es descriu breument cada grup.

10.1 Grup system

Aquest grup proporciona informació general sobre el sistema gestionat. Està format per set objectes escalars que permeten obtenir la següent informació:

- Descripció de l'entitat (hardware i software).
- Identificació del subsistema de gestió de xarxa contingut a l'entitat.
- Temps que el dispositiu porta en funcionament.
- Identificació i informació de contacte de la persona que administra el node.
- Nom administratiu del node.
- Localització física del node.
- Valor (nombre entre 0 i 127) que indica el conjunt de serveis que ofereix el node de l'arquitectura TCP/IP o OSI.

10.2 Grup interfaces

El grup conté informació sobre les interfícies físiques de l'entitat, incloent informació de configuració i estadística sobre els events esdevinguts a cada interfície. La implementació d'aquest grup es obligatòria. El grup inclou un objecte escalar (*ifNumber*) que representa el nombre d'interfícies associades a l'entitat i una taula (*ifTable*) que conté una fila per cada interfície. A la taula es pot distingir informació sobre:

- Descripció de la interfície.
- Tipus d'interfície (frame-relay, fddi, ppp, ethernet, loopbac, ...).
- MTU.
- Estimació de l'ample de banda.
- Estat de la interfície (*up, down, testing*).
- Adreça física.
- Paquets amb error.
- Nombre de bytes entrants i sortints.

10.3 Grup at

El grup *address translation (at)* proporciona el mapatge d'adreces de xarxa a adreces físiques. A la MIB-II aquest grup es troba definit amb l'estat *deprecated*, i s'inclou amb l'únic propòsit de mantenir la compatibilitat amb la MIB-I. A la MIB-II la informació sobre mapatge d'adreces es troba a cada grup de protocol de xarxa.

10.4 Grup ip

El grup ip conté informació rellevant sobre l'operació i implementació del protocol IP en un node.

El grup està format per 20 objectes escalars útils per al monitoratge de rendiment i control d'errades, i per tres taules: *ipAddrTable, ipRouteTable* (substituïda pel grup *ipForward*) i *ipNetToMediaTable*.

La taula *ipAddrTable* conté informació la llista d'adreces IP associades a l'entitat.

La taula *ipRouteTable* conté la taula d'enrutament. La taula té una entrada per a cada ruta coneguda per l'entitat. Aquesta taula presentava el problema de que sols estava indexada per l'adreça de destí amb l'inconvenient que la taula d'enrutament pot tenir varies rutes amb el mateix destí. Per a solucionar aquest entrebanc, i fer més flexible la taula d'enrutament, es va eliminar la taula i fou substituïda per noves definicions a l'RFC 1354.

L'RFC 1354 defineix un grup dins el grup *ip* anomenat *ipForward* que consta d'un objecte escalar i d'una taula anomenada *ipForwardTable*. L'objecte escalar emmagatzema el nombre de files de la taula. La taula *ipForwardTable* és la taula que substitueix la taula *ipRouteTable*. La gran diferència de la nova taula està en l'indexat de les seves files per tal de suportar totes les possibilitat d'una taula d'enrutament.

La taula *ipNetToMediaTable* permet la traducció d'adreces entre l'adreça física i l'adreça IP.

10.5 Grup icmp

El protocol ICMP, definit a l'RFC 792, és part integral de la suite de de protocols TCP/IP. Es requerida la seva presència junt a IP, per tant, tots els sistemes que implementin IP han d'implementar ICMP. El grup *icmp* conté proporciona informació sobre la implementació i l'operació d'ICMP en cada node.

10.6 Grup tcp

El grup tcp conté informació rellevant sobre l'operació i implementació del protocol TCP en un node. El grup està format per la taula *tcpConnTable* i per 14 objectes escalars que contenen informació sobre els segments rebuts, enviats, erronis, retransmesos, nombre de connexions TCP, ...

10.7 Grup udp

El grup *udp* conté informació rellevant sobre l'operació i implementació del protocol UDP en un node. Aquest grup conté 4 objectes escalars que representen informació sobre datagrames enviats i rebuts. El grup també conté la taula *udpTable* que emmagatzema informació sobre els ports UDP disponibles.

10.8 Grup egp

El grup egp conté informació rellevant sobre l'operació i implementació del protocol *External Gateway Protocol* EGP en un node. Aquest grup està format per 5 objectes escalars que emmagatzemen informació sobre els missatges EGP enviats i rebuts, i una taula (*egpNeightTable*) que gestiona informació sobre cadascun dels *gateways* veïns coneguts per l'entitat.

10.9 Grup dot3 (transmissió)

Aquest grup està dissenyat per a contenir objectes que proporcionin detalls sobre el medi de transmissió subjacent per a cada interfície del sistema. De fet, no és un grup, sinó simplement un node de la jerarquia MIB-II, sota el qual es troben varis grups d'interfícies específiques. A títol d'exemple, es pot citar MIB EtherLike que es troba definida a l'RFC 1643.

10.10 Grup snmp

El grup *snmp* està format per 30 objectes escalars que emmagatzemen informació rellevant sobre l'operació i implementació del protocol SNMP. Alguns exemples sobre la informació recollida són els següents:

- Quantitat de comandes *GetRequest*, *GetNextRequest* i *SetRequest* enviades.
- Quantitat de comandes *GetResponse* rebudes.
- Nombre de *traps* rebuts.
- Quantitat de paquets rebuts error (*tooBig*, *BadCommunityName*, ...)

11 SNMPv2 MIB

SNMPv2 aporta millores a la MIB-II d'SNMPv1 en els grups següents:

- system
- interfaces
- SNMP

A més a més, SNMPv2 defineix fora de l'àmbit de la MIB-II el grup següents:

- MIB objects

11.1 Grup system

El grup *system* definit a la MIB SNMPv2 és el mateix grup que es definí a la MIB-II amb l'addició d'alguns objectes nous. Els nous objectes afegits comencen amb el prefix sysOR i, entre d'altres, s'encarreguen de descriure quins objectes controla un agent.

11.2 Grup interfaces

El grup *interfaces* de la MIB-II és millorat mitjançant una extensió especificada a l'RFC 1573 tot utilitzant la definició SMIv2. L'RFC 1573 principalment afegeix quatre noves taules al grup *interfaces*, aquestes són:

- *ifXTable:* És una extensió de la taula ifTable que es definix a partir d'aquesta mitjançant la clàusula *AUGMENTS*.
- *ifStackTable*: Identifica la relació entre files de la taula *ifTable* per a una mateixa interfície.
- *ifTestTable:* Defineix objectes que permeten al gestor efectuar determinades proves sobre les interfícies de l'agent.
- *ifRcvAddressTable:* Conté una taula amb les adreces origen dels paquets rebuts per cada interfície.

11.3 Grup SNMP

Al grup SNMP s'eliminen la majoria dels objectes definits a la MIB-II i se n'afegeixen dos de nous: *snmpSilentDrops* i snmpProxyDrops.

11.4 Grup MIB objects

Aquest grup no penja de la branca mib-2 (1.3.6.1.2.1), sinó que o fa de la branca snmpV2 (1.3.6.1.6), i més concretament, de l'objecte snmpMIB d'aquesta branca.

El grup conté informació per al control d'objectes MIB. Aquest grup està format per dos subgrups *snmpTrap* i *snmpSet*.

12 SNMPv3 MIB

SNMPv3 defineix tres MIB per aplicacions SNMPv3 a l'RFC 2273: *Management Traget MIB*, *Notification MIB* i *Proxy MIB*.

12.1 Management Target MIB

La MIB SNMP-TARGET conté objecte per a la definició de destins de gestió i inclou un únic grup (*snmpTargetObjectsGroup*).

Un destí de gestió és una entitat de gestió que pot ser receptora de missatges generats per generadors de notificacions i per proxies. Això requereix un mecanisme per a determinar com i on han de ser enviats els missatges generats. Es requereix dos tipus d'informació:

- *Informació de destí:* Consisteix en el domini i direcció de transport. Aquest parell es coneix com *transport endpoint*.
- *Paràmetres de missatge SNMP:* Informació sobre el model de processament de missatges, model de seguretat, nivell de seguretat i nom de seguretat.

12.2 Notification MIB

Aquesta MIB conté objectes per a la configuració remota dels paràmetres utilitzats per una entitat SNMP en la generació de notificacions. Conté un grup format per tres taules: *snmpNotifyTable*, *snmpNotifyFilterProfileTable* i *snmpNotifyFilterTable*.

12.3 Proxy MIB

Aquesta MIB conté objectes per a la configuració remota dels paràmetres utilitzats per una entitat SNMP en les operacions de reenviament de missatges. La MIB inclou un únic grup (*snmpProxyObjectsGroup*) que està format per la taula *snmpProxyTable*. Cada fila de la taula és un conjunt de paràmetres de traducció utilitzats pel proxy.

Annexos a NetSNMP.NET

13 INSTAL·LACIÓ I CONFIGURACIÓ

13.1 Sistema operatiu GNU/Linux

En aquest document no s'entra en com instal·lar un S.O. GNU/Linux, senzillament s'indica que el S.O. utilitzat ha estat Debian Sid, no obstant qualsevol altre distribució serà bona.

13.2 Llibreria Net-SNMP

13.2.1 Entorn d'execució

- *Debian.* `apt-get install snmp`
- *Altres.* Cal descarregar els binaris de la web de la llibreria.
  1 – http://net-snmp.sourceforge.net/download.html
  2 – Seleccionar l'enllaç Sourceforge download page - current releases
  3 – Seleccionar l'enllaç net-snmp binaries
  4 – Triar el paquet escaient.

13.2.2 Entorn de desenvolupament

- *Debian.* `apt-get install libsnmp9-dev`
- *Altres.* Cal descarregar els fonts de la web de la llibreria.
  1 – http://net-snmp.sourceforge.net/download.html
  2 – Seleccionar l'enllaç Sourceforge download page - current releases
  3 – Seleccionar l'enllaç net-snmp
  4 – Triar el paquet escaient.

13.2.3 Agent

- *Debian.* `apt-get install snmpd`
- *Altres.* La instal·lació de l'entorn d'execució ja instal·la l'agent *snmpd*.

13.2.3.1 Configuració

Per a l'agent cal configurar els paràmetres de seguretat per tal de que els jocs de proves s'executin correctament.

El primer que és farà serà crear un usuari per al model de seguretat USM de SNMPv3. Les accions a fer són:

1. Assegurar-nos de que l'agent està aturat. En una configuració Debian ho podem fer amb la comanda `/etc/init.d/snmpd stop`.
2. Executar la comanda:
   `net-snmp-config --create-snmpv3-user -a clauAutenticacio -x clauXifrat -A MD5 -X DES UOC`

Tot seguit cal retocar el fitxer de configuració de l'agent. El fitxer de configuració s'anomena *snmpd.conf* i en una distribució debian s'ubica al directori */etc/snmp/*, en altres distribucions es molt probable que també es trobi en aquest directori. Retocarem l'apartat *Access Control*, que ens ha de quedar de la següent manera:

```
####
# First, map the community name (COMMUNITY) into a security name
# (local and mynetwork, depending on where the request is coming
# from):

#       sec.name  source          community
com2sec readonly  default         public
com2sec UOC       default

####
# Second, map the security names into group names:

#                        sec.model  sec.name
group grReadonly  v1         readonly
group grReadonly  v2c        readonly
group grUOC       usm        UOC
####
# Third, create a view for us to let the groups have rights to:

#               incl/excl subtree                                    mask
view all        included  .1                                         80
view system     included  .iso.org.dod.internet.mgmt.mib-2.system

####
# Finally, grant the 2 groups access to the 1 view with different
# write permissions:

#                  context sec.model sec.level match  read   write      notif
access grReadonly  ""      any       noauth    exact  system none       none
access grUOC       ""      usm       authpriv  exact  all    all        none
# ----------------------------------------------------------------------------
```

Finalment cal engegar l'agent amb la comanda `/etc/init.d/snmpd start`.

13.3 Mono

13.3.1 Entorn d'execució

- *Debian*. `apt-get install mono`
- *Altres*. Cal descarregar de la web de mono. Els pasos són:
  1 - http://www.mono-project.com/Downloads
  2 – Seleccionar el paquet per a la distribució GNU/Linux utilitzada. També hi ha disponible un instal·lable per a totes les distribucions.

13.3.2 Monodevelop

- *Debian*. `apt-get install monodevelop monodoc`
- *Altres*. La instal·lació de l'entorn d'execució ja l'inclou.

13.3.2.1 Configuració

Es recomana visualitzar com a mínim les finestres *Task list*, *Solution*, *Classes*, *Build output* i *Application output*. Per a fer-ho cal activar-les des del menú *View*.

13.4 NetSNMP.NET

13.4.1 Entorn de desenvolupament

Tots els fitxers del projecte es troben a l'arxiu comprimit NetSNMP.NET.zip, per tant cal descomprimir aquest per disposar dels fonts i binaris de NetSNMP.NET. Tot el projecte es troba ubicat sota el directori ./NetSNMP.NET.

13.4.2 Entorn d'execució

Per a utilitzar la llibreria NetSNMP.NET cal disposar dels fitxers NetSNMP.dll, NetSNMP.Native.dll i libnetsnmp_to_dotnet.so. Aquests fitxers s'han de distribuir de la següent manera:

- *NetSNMP.dll i NetSNMP.Native.dll*. S'han d'ubicar al directori d'execució del programari o també es poden instal·lar al GAC.
- *libsnmp_to_dotnet.so*. S'ha d'ubicar al directori d'execució del programari o en qualsevol altre que es trobi al path.

14 Manual d'ús

14.1 Entorn de desenvolupament

Tot els fitxers del projecte NetSNMP.NET s'inclouen a l'arxiu comprimit NetSNMP.NET.zip.

Per a obrir l'entorn de desenvolupament de NetSNMP.NET cal engegar l'IDE *Monodevelop* amb la comanda `monodevelop` i tot seguit carregar la solució NetSNMP.mds.

Per a compilar els projectes NetSNMP.Native, NetSNMP i Test és pot fer des del mateix IDE amb alguna de les opcions de compilació del menú Run. La manera més fàcil és fer F5 que engega el projecte i si s'escau prèviament compila.

Per a compilar el projecte libsnmp_to_dotnet cal executar des de una consola l'script `./build_lib.sh` al directori del projecte (./libsnmp_to_dotnet).

14.2 Documentació

La documentació s'ha de llegir amb la utilitat Monodoc. Per tal de simplificar el procés d'instal·lació es consultarà la documentació en mode edició, no obstant dir que s'hagués pogut compilar i instal·lar la documentació.

Per a consultar la documentació cal obrir una consola, situar-se al directori NetSNMP.NET i tot seguit escriure la comanda `monodoc --edit ./doc`. En l'arbre de temes apareix la branca NetSNMP.NET, cal consultar aquesta branca.

14.3 Utilització de la llibreria NetSNMP.NET

Per a poder utilitzar NetSNMP.NET com a llibreria en un projecte extern cal disposar dels arxius NetSNMP.dll, NetSNMP.Native.dll i libnetsnmp_to_dotnet.so.

La llibreria libnetsnmp_to_dotnet.so és pot ubicar al directori d'execució del programa que l'utilitza o en algun directori que estigui al path del sistema.

Les llibreries NetSNMP.dll i NetSNMP.Native.dll es poden ubicar al directori d'execució del programari que les utilitza o també es poden instal·lar al GAC.

La programació de la llibreria és molt simple, cal seguir els passos següents:

1. Crear una sessió SNMP tot instanciant SessionV1, SessionV2c o SessionV3.
2. Crear una PDU, que segons l'operació desitjada, serà una instància de GetRequestPDU, GetRequestNextPDU, GetBulkRequestPDU o SetRequestPDU. Tot seguit afegir variables a la PDU creada amb algun dels mètodes AddVariable.
3. Passar la PDU a la sessió per a que sigui processada. Es disposa de dos tipus de procés: síncron i asíncron.

   3.1. *Procés síncron*. Cridar al mètode ProcessSynchPDU de l'objecte sessió tot passant-li la PDU creada al punt 2. El mètode retornarà una PDU de tipus ResponsePDU.

   3.2. *Procés asíncron*. Destacar que el procés asíncron engega un altre fil d'execució per a rebre els resultats, no obstant, tot aquest procés és absolutament transparent al programador.

      3.2.1. Crear un mètode amb l'estructura del delegat Session.ReciveAsynchPDU. Aquest mètode rebrà la PDU resultat.

      3.2.2. Assignar a l'event OnReciveAsynchPDU de l'objecte sessió el mètode creat.

      3.2.3. El mètode creat rebrà un objecte de tipus ReciveAsynchPDUEventArgs que entre d'altres conté la PDU de resposta (ResponsePDU) i un camp status que cal iniciar al valor 1 per a indicar que s'ha llegit el resultat; si no s'inicialitza aquest valor es tornarà a disparà l'event de forma successiva.
4. Llegir la ResponsePDU si s'escau. Remarcar que cal verificar que no s'hagi produït cap error abans d'intentar llegir les variables de la PDU. Destacar que en el procés asíncron aquest pas és fa dins el mètode creat per a respondre a l'event d'arribada de resultat de PDU.
5. Per al cas síncron s'ha de destruir la ResponsePDU tot invocant el mètode Dispose. Si aquest pas no es fa es farà automàticament quan el recol·lector d'escombraries de Mono destrueixi l'objecte, tot i així és preferible fer-ho per alliberar recursos el més aviat possible.
6. Cal tancar la sessió invocant el mètode Dispose. Si aquest pas no es fa es farà automàticament quan el recol·lector d'escombraries de Mono destrueixi l'objecte, tot i així és preferible fer-ho per alliberar recursos el més aviat possible.

Per a comprendre bé el procés es recomana aferrissadament consultar el exemples donats a la secció d'exemples.

14.4 Exemples

14.4.1 Entorn d'execució

Els exemples que es donen sempre s'intenten connectar a un agent SNMP ubicat a la mateixa màquina, per tant la IP de connexió és 127.0.0.1 i el port UDP el 161, que és el port per defecte de SNMP.

L'agent utilitzat ha estat el de Net-SNMP i cal que estigui configurat segons s'indica a la secció "Instal·lació i configuració".

14.4.2 Manual d'ús

Els exemples es troben al projecte Test. Aquest projecte conté una classe principal anomenada Test, que és la responsable d'engegar l'exemple desitjat. Per a fer-ho cal indicar al mètode main quin exemple es vol engegar. Per exemple, per engegar l'exemple 3, caldrà escriure Test3.Run ().

14.4.3 Exemple 1

Objectiu: Mostrar com crear una sessió SNMPv1 i sol·licitar dades de forma síncrona a través d'una PDU GetRequest.

Paraules clau: SNMPv1, llegir, PDU GetRequest, PDU GetResponse, procés síncron.

Funcionament: Obre una sessió SNMPv1 i crea una PDU GetRequest per llegir el nom i la descripció del sistema de forma síncrona.

Resultats

```
1 - SNMPv2-MIB::sysName.0 = ASN_OCTET_STR : estudion
2 - SNMPv2-MIB::sysDescr.0 = ASN_OCTET_STR : Linux estudion 2.6.10 #1
Wed Feb 16 20:49:09 CET 2005 i686
```

14.4.4 Exemple 2

Objectiu: Mostrar com crear una sessió SNMPv2c i sol·licitar dades de forma síncrona a través d'una PDU GetNextRequest.

Paraules clau: SNMPv2c, llegir, PDU GetNextRequest, PDU GetResponse, procés síncron.

Funcionament: Obre una sessió SNMPv2c i crea una PDU GetNextRequest per llegir les instàncies següents, segons l'ordre lexicogràfic de la MIB, a sysName.0 i sysDescr.0. El procés de la PDU és fa de forma síncrona.

Resultats

```
1 - SNMPv2-MIB::sysLocation.0 = ASN_OCTET_STR : Unknown (configure
/etc/snmp/snmpd.local.conf)
2 - SNMPv2-MIB::sysObjectID.0 = ASN_OBJECT_ID : NET-SNMP-
MIB::netSnmpAgentOIDs.10 (.1.3.6.1.4.1.8072.3.2.10)
```

S'observa que l'objecte *sysLocation* no està assignat.

14.4.5 Exemple 3

Objectiu: Mostrar que el procés de PDU en sessions SNMPv1 és atòmic mentre que en SNMPv2 no.

Paraules clau: SNMPv1, SNMPv2c, PDU GetRequest, PDU GetResponse, procés síncron, procés atòmic de PDU.

Funcionament: Obre dos sessions, una SNMPv1 i l'altra SNMPv2c, tot seguit demana el valor de dos instàncies per a cada sessió a través d'una PDU GetRequest. D'aquestes dos instàncies una s'hi pot accedir (sysName.0) i l'altra no (ifInOctets.1). El procés es fa de forma síncrona.

Resultats

```
*** SNMPv1 ***
Error: (2) (noSuchName) There is no such variable name in this MIB.

*** SNMPv2c ***
1 - SNMPv2-MIB::sysName.0 = ASN_OCTET_STR : estudion
```

S'observa que SNMPv1 no retorna cap resultat i per tant el procés de la PDU s'ha produït de forma atòmica, mentre que en SNMPv2c s'ha retornat el valor de la instància que s'ha pogut llegir, i en conseqüència el procés de la PDU no ha estat atòmic.

14.4.6 Exemple 4

Objectiu: Mostrar com crear una sessió SNMPv3 i sol·licitar dades de forma síncrona a través d'una PDU GetBulkRequest.

Paraules clau: SNMPv3, llegir, PDU GetBulkRequest, PDU GetResponse, procés síncron.

Funcionament: Obre una sessió SNMPv3, amb autenticació i xifrat, i crea una PDU GetBulkRequest per llegir les instàncies següents a sysName.0 i sysDescr.0 i les 5 següents a interfaces.0. El procés es fa de forma síncrona.

Resultats

```
1 - SNMPv2-MIB::sysLocation.0 = ASN_OCTET_STR : Unknown (configure
/etc/snmp/snmpd.local.conf)
2 - SNMPv2-MIB::sysObjectID.0 = ASN_OBJECT_ID : NET-SNMP-
MIB::netSnmpAgentOIDs.10 (.1.3.6.1.4.1.8072.3.2.10)
3 - IF-MIB::ifNumber.0 = ASN_INTEGER : 2
4 - IF-MIB::ifIndex.1 = ASN_INTEGER : 1
5 - IF-MIB::ifIndex.2 = ASN_INTEGER : 2
6 - IF-MIB::ifDescr.1 = ASN_OCTET_STR : eth0
7 - IF-MIB::ifDescr.2 = ASN_OCTET_STR : lo
```

14.4.7 Exemple 5

Objectiu: Mostrar com es pot modificar el valor de les instàncies a través d'una PDU SetRequest.

Paraules clau: SNMPv2c, SNMPv3, escriure, PDU SetRequest, PDU GetResponse, procés síncron.

Funcionament: Obre dos sessions, una SNMPv2 i l'altra SNMPv3. Les dos sessions intenten modificar les instàncies sysName.0 i ifAdminStatus.1 a través de la PDU GetRequest. El procés es fa de forma síncrona.

Resultats

```
*** SNMPv2c ***
Error: (17) notWritable (That object does not support modification)

*** SNMPv3 ***
1 - SNMPv2-MIB::sysName.0 = ASN_OCTET_STR : superEstudion
2 - IF-MIB::ifAdminStatus.1 = ASN_INTEGER : 1
```

S'observa que la comunitat *public* no té permisos per modificar instàncies, en canvi l'usuari UOC si que en té com es pot comprovar. Aclarir que, el fet de poder modificar o no, no depèn de la versió d'SNMP. També s'hagués pogut configurar que una comunitat pogués modificar instàncies, no obstant ho faria sense autenticació i sense xifrat, condicionants que ho fan altament desaconsellable.

14.4.8 Exemple 6

Es recomana analitzar atentament aquest exemple per tal de comprendre el funcionament asíncron.

Objectiu: Mostrar el procés d'una PDU de forma asíncrona.

Paraules clau: SNMPv3, llegir, PDU GetRequest, PDU GetResponse, procés asíncron.

Funcionament: Obre una sessió SNMPv3 i crea una PDU GetRequest per a obtenir informació de la interfície 1. El procés de la PDU és fa de forma asíncrona.

Resultats

```
Executant altres operacions
1 - IF-MIB::ifDescr.1 = ASN_OCTET_STR : eth0
2 - IF-MIB::ifType.1 = ASN_INTEGER : 6
3 - IF-MIB::ifSpeed.1 = ASN_GAUGE : 10000000
4 - IF-MIB::ifMtu.1 = ASN_INTEGER : 1500
5 - IF-MIB::ifInOctets.1 = ASN_COUNTER : 186007949
Executant altres operacions
6 - IF-MIB::ifInErrors.1 = ASN_COUNTER : 0
7 - IF-MIB::ifOutOctets.1 = ASN_COUNTER : 293533054
8 - IF-MIB::ifOutErrors.1 = ASN_COUNTER : 0
```

En el resultat és pot veure clarament com hi ha dos fils d'execució.