Standard de l'Administration de Réseaux

SNMP

Simple Network Management Protocol

Partie 1

Université François Rabelais de Tours
Faculté des Sciences et Techniques
Antenne Universitaire de Blois

Licence Professionnalisée

Mohamed Taghelit
taghelit@univ-tours.fr

Fonctions d'Administration

(partie opérationnelle d'un réseau)

- **Extraction** des informations des éléments du réseau au moyen d'outils
  → récolte d'un grand nombre d'informations

- **Réduction** du volume d'information au moyen de filtres
  → sélection des informations significatives

- **Stockage** des informations pertinentes dans une base de données d'administration

- **Traitement** de ces informations

- **Convivialité** par l'offre d'interfaces (utilisateur d'administration, opérateur réseau)

Les Standards de l'Administration

Utilisation par une gamme la plus large de produits (systèmes terminaux, ponts, routeurs, équipement de télécommunication quelconque) et dans un environnement multi-constructeurs,

- SNMP

Regroupe un ensemble de standards incluant

◊ un protocole,
◊ une spécification de la structure de la base de données,
◊ un ensemble d'objets.

C'est le standard pour TCP/IP.

- L'administration de systèmes OSI

Regroupe un grand ensemble de standards qui décrivent

◊ une architecture générale d'administration,
◊ un service et un protocole de gestion (CMISE/CMIP),
◊ une spécification de la structure de la base de données,
◊ un ensemble d'objets.

Domaines Fonctionnels de l'Administration

- La gestion de configurations

  Mécanismes permettant le contrôle, l'identification, la collecte depuis et l'émission d'informations vers les objets gérés dans le but d'assurer une continuité des services d'interconnexion.

- La gestion des pannes

  Mécanismes permettant la détection, la localisation, la réparation et le retour à une situation normale dans l'environnement.

- La gestion de la sécurité

  Mécanismes essentiels, d'un point de vue sécurité, à une administration cohérente et à la protection des objets gérés.

- La gestion des performances

  Mécanismes nécessaires à l'évaluation du comportement des objets gérés et des activités de communication effective.

- La gestion de la comptabilité

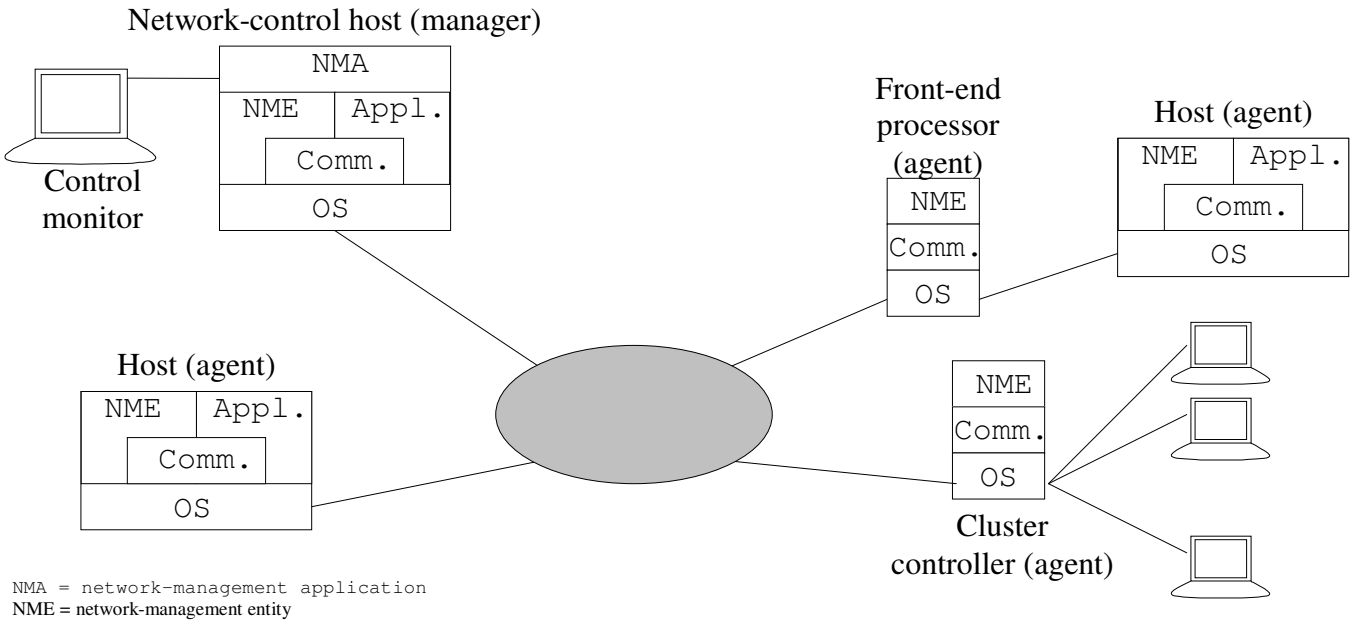
  Mécanismes permettant d'établir la charge des objets gérés et d'évaluer le coût d'utilisation de ces objets.

Les Systèmes d'Administration de Réseaux

Un système d'administration réseaux est une collection d'outils pour la surveillance et le contrôle de réseaux qui comprend :

- une interface pour opérateur avec un ensemble de commandes pour exécuter la plupart ou toutes les tâches d'administration de réseaux

- un minimum d'équipements supplémentaires intégrés au système existant

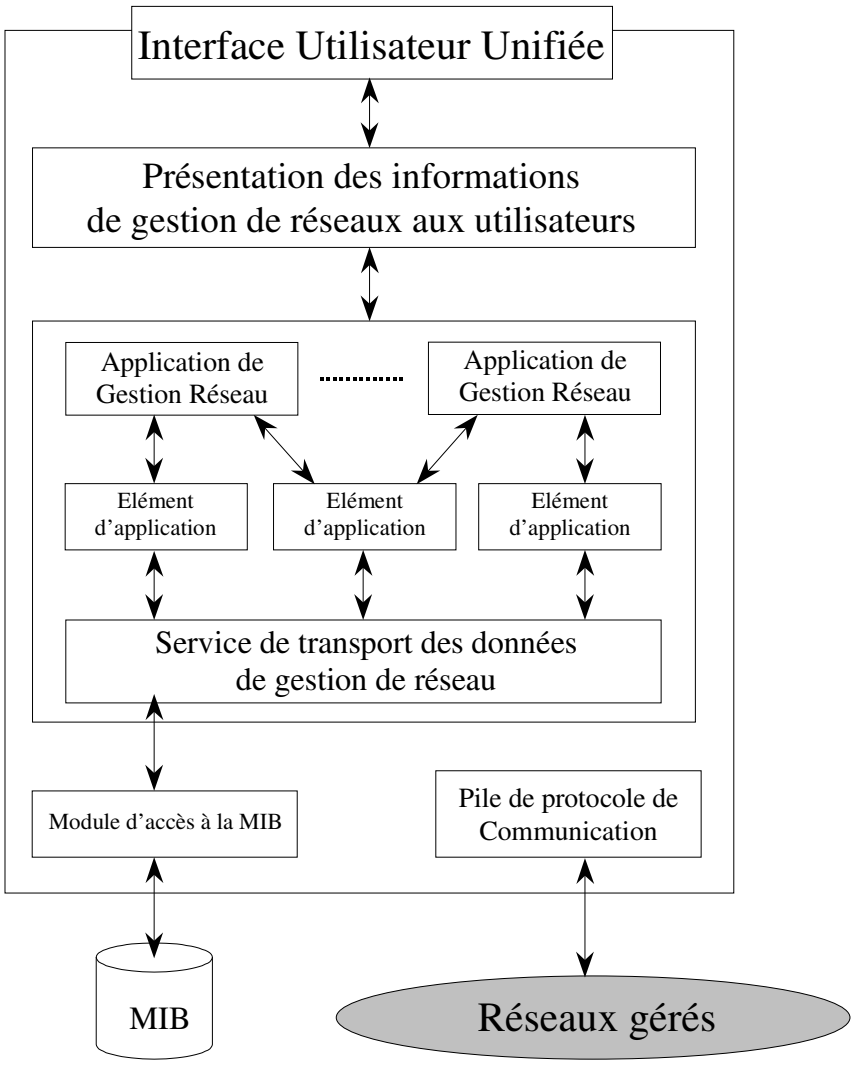


Éléments d'un système de gestion de réseaux

L'Architecture d'un Logiciel d'Administration de Réseaux

L'architecture de l'application dans un gestionnaire ou dans un agent va varier en fonction des fonctionnalités de la plate-forme et des capacités d'administration.
Une vue générique d'une architecture divisée en trois grandes catégories:

- ◊ le logiciel de présentation à l'utilisateur
- ◊ le logiciel de gestion réseau
- ◊ les logiciels de communication et de support des données



Principes de l'Administration de Réseaux

Fonctions de l'Administration de Réseau :

Supervision
Observation et analyse de l'état et du comportement de la configuration et de ses composants.

- Access to monitored information
- Design of monitoring mechanisms
- Application of monitored information

Contrôle
Altération des paramètres de divers composants de la configuration entraînant ces composants à réaliser des actions prédéfinies.

Les Concepts de SNMP

- Protocole d'administration de machines supportant TCP/IP
- Défini en 1988
- MIB-1 - 1991
- Secure SNMP – 1992
- SMP - 1992
- SNMPv2 - 1993

Permet de répondre à un grand nombre de besoins

- disposer d'une cartographie du réseau
- fournir un inventaire précis de chaque machine
- mesurer la consommation d'une application
- signaler les dysfonctionnements

Avantages

- protocole très simple, facile d'utilisation
- permet une gestion à distance des différents équipements
- le modèle fonctionnel pour la surveillance et pour la gestion est extensible
- indépendant de l'architecture des équipements administrés

Le Modèle

Une administration SNMP est composée de **trois types d'éléments**

- des agents chargés de superviser un équipement. On parle d'agent SNMP installé sur tout type d'équipement,
- une ou plusieurs stations de gestion capable d'interpréter les données. On parle de manager SNMP,
- une MIB (Management Information Base) décrivant les informations gérées.

Un protocole activé par une API permet la supervision, le contrôle et la modification des paramètres des éléments du réseau.

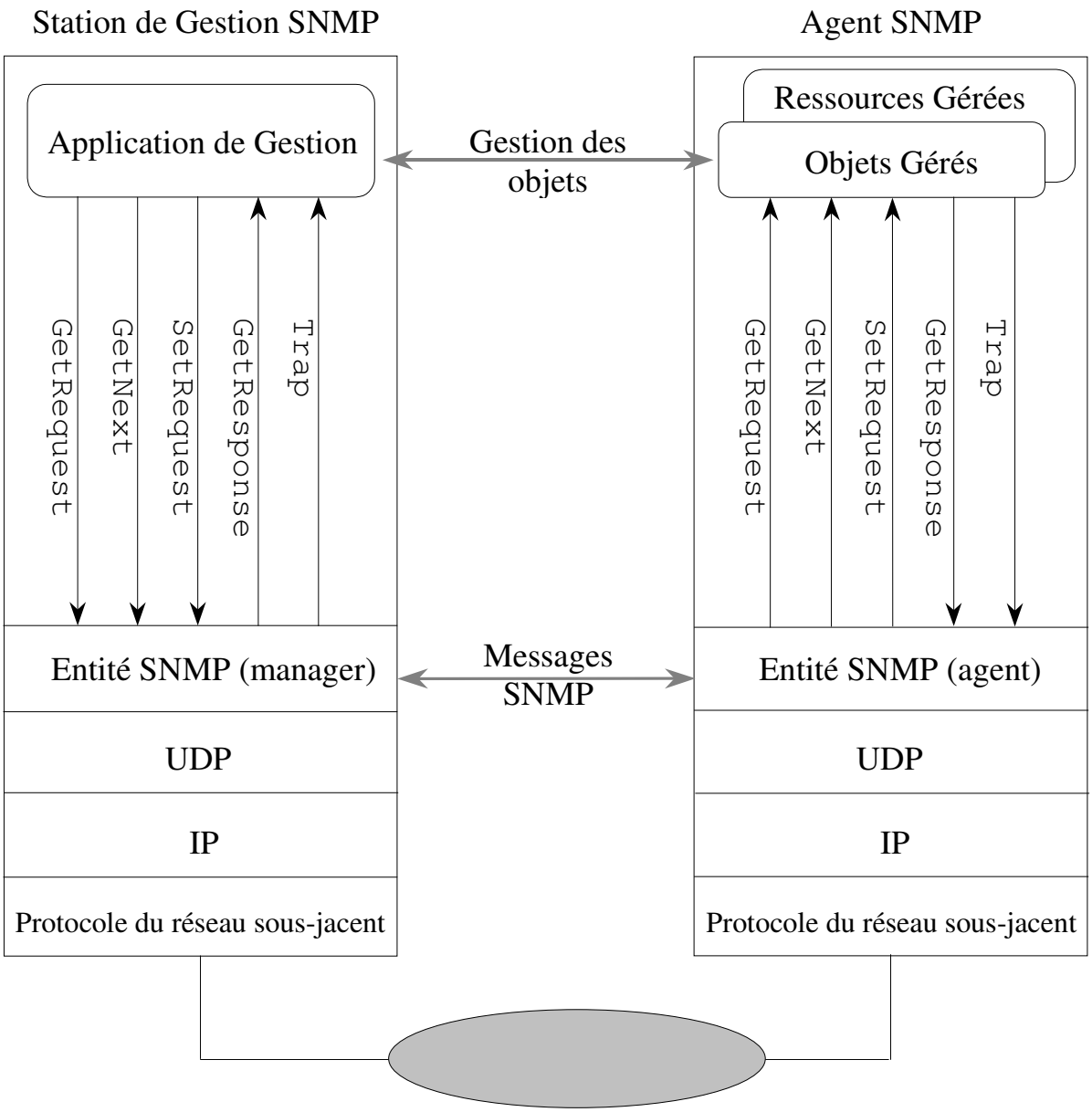
SNMP fournit trois opérations simples

- get : permet à la station manager d'interroger un agent,
- set : permet de modifier les données d'un agent,
- trap : permet à un agent de notifier au manager l'occurrence d'un événement significatif

SNMP permet des accès multiples avec une seule opération

L'ajout et la suppression d'instances d'objets ne sont pas normalisés.

Le Modèle (2)



- Protocole sans connexion,
- Aucune garantie sur l'acheminement du trafic d'administration,

→ opérations orientées connexion doivent être construites dans des couches supérieures, si la fiabilité est nécessaire

Le Modèle (3)

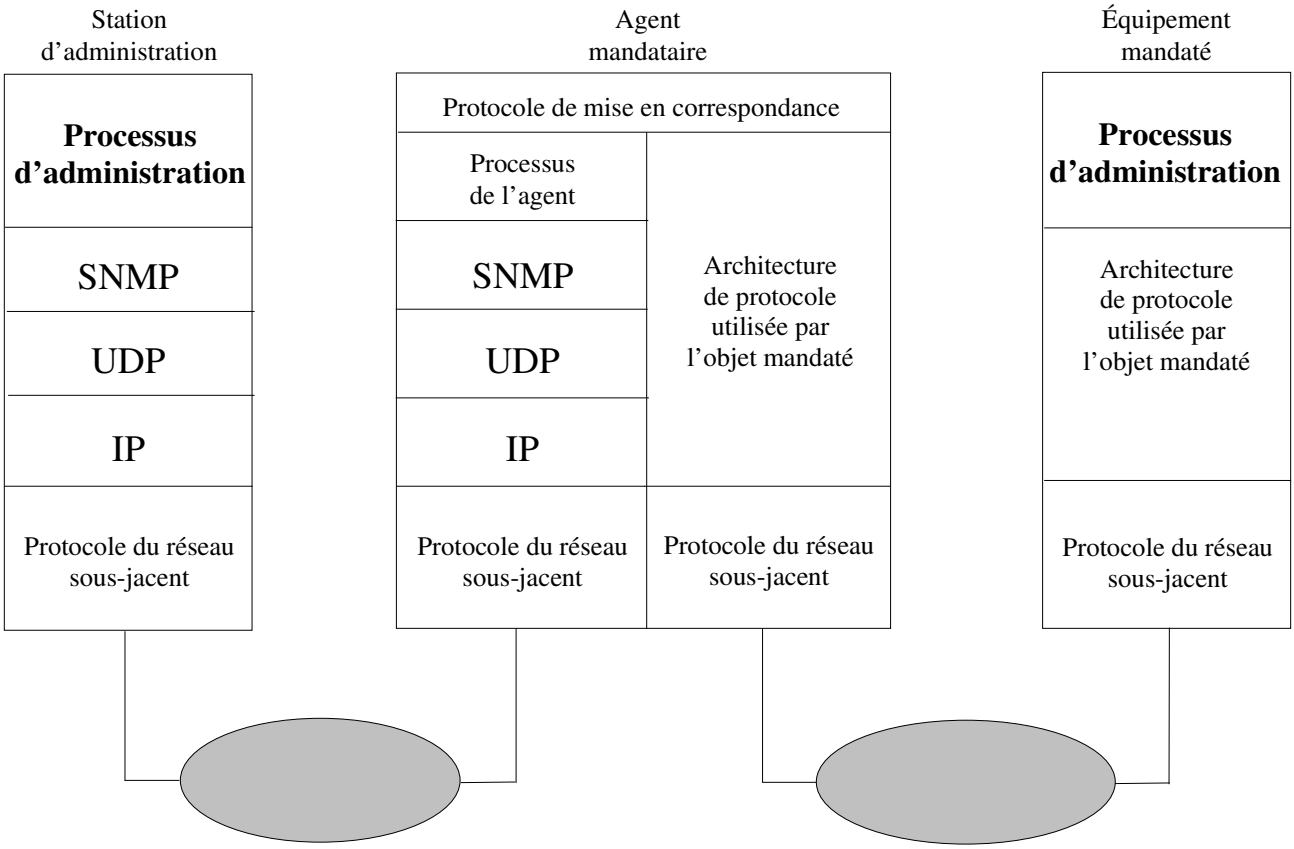
L'utilisation de SNMP suppose que tous les agents et les stations d'administration supportent IP et UDP.

→ limitation de l'administration de certains équipements

Pour certaines machines (ordinateur personnel, station de travail, contrôleur programmable, ...) qui implantent TCP/IP pour supporter leurs applications, mais qui ne souhaitent pas ajouter un agent SNMP.

⟶ utilisation de la gestion mandataire (les proxies)

Un agent SNMP agit alors comme mandataire pour un ou plusieurs équipements.



Informations d'Administration SNMP

- Système de gestion de réseaux → Base de données
  → MIB
- Ressource gérée → objet

- MIB → collection structurée d'objets
  SNMP : BD structurée en arbre

- Chaque nœud dans le système doit maintenir une MIB qui reflète l'état des ressources gérées en ce nœud

- Une entité d'administration peut surveiller les ressources du nœud en lisant les valeurs des objets dans la MIB et les contrôler en modifiant leurs valeurs

- Objectifs d'une MIB (pour répondre aux besoins d'un système de gestion)
  - Un schéma commun de représentation doit être utilisé pour supporter l'interopérabilité
  → donne les règles de définition, d'accès et d'ajout des objets dans la MIB
  → encourage la simplicité et l'extension de la MIB

  - L'objet utilisé pour représenter une ressource particulière doit être le même sur chaque système
  → donne une représentation identique des objets
  → rend un objet accessible de la même manière sur chaque entité du réseau

Structure d'Informations d'Administration (RFC 1155)

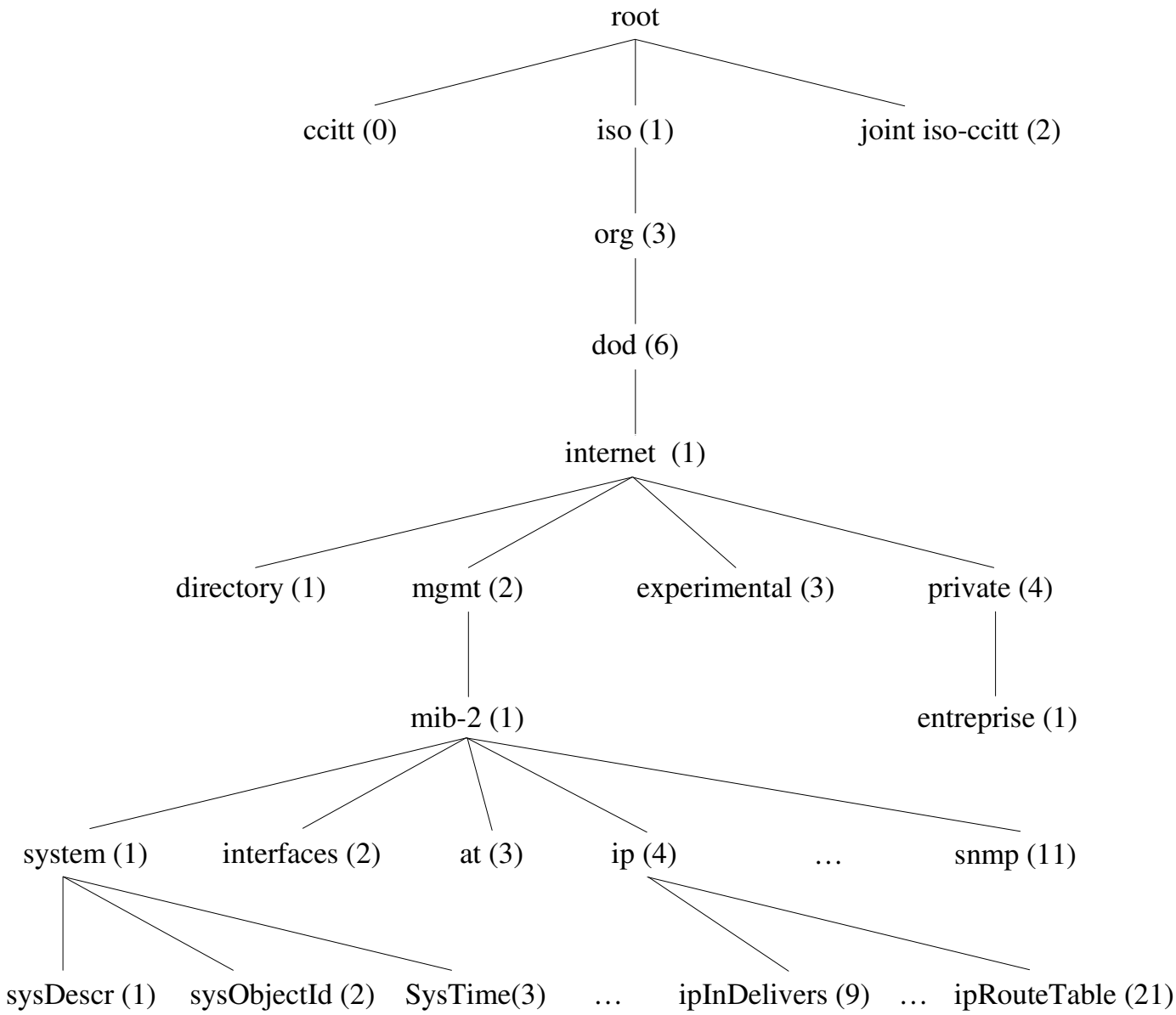
Identifie les types de données qui peuvent être utilisés dans une MIB et spécifie comment les ressources de la MIB sont représentées et nommées.

Pour fournir un schéma normalisé de représentation d'informations de gestion, le SMI doit fournir des techniques normalisées pour :

- définir la structure d'une MIB particulière,
- définir les objets (syntaxe et valeur de chaque objet),
- encoder les valeurs des objets.

Spécification de l'Arbre des MIBs Accessibles

Chaque objet est représenté par un "OBJECT IDENTIFIER" (OID).



Définition de Nouveaux Objets dans une MIB

Pour autoriser toutes les évolutions, le SMI Internet standard définit trois mécanismes d'extensibilité :

- Addition d'objets standardisés

  Le sous arbre `MIB-2` peut-être étendu ou remplacé par une nouvelle version. Un nouveau sous arbre est défini.

- Addition d'objets largement répandus mais non standard

  Une MIB peut-être définie, pour une application spécifique, par l'intermédiaire du sous arbre `experimental`.

- Addition d'objets privés

  Des extensions peuvent être ajoutées au sous arbre `enterprises`.

→ **Évolution** : pas de modification des objets existants dans les nouvelles versions.

Syntaxe des Objets

Les Types (ASN.1 pour SNMP)

- UNIVERSAL TYPES

```
INTEGER (UNIVERSAL 2)
OCTET STRING (UNIVERSAL 4)
NULL (UNIVERSAL 5)
OBJECT IDENTIFIER (UNIVERSAL 6)
SEQUENCE, SEQUENCE OF (UNIVERSAL 16)
```

- APPLICATION-WIDE TYPES

```
NetwokAddress ::= CHOICE {internet IpAddress}
```
-- adresse réseau

```
IpAddress ::= [APPLICATION 0] IMPLICIT OCTET STRING (SIZE 4)
```
-- type de données représentant une adresse IP

```
Counter ::= [APPLICATION 1] IMPLICIT INTEGER (0..4294967295)
```
-- repasse à 0 lorsque = Max

```
Gauge ::= [APPLICATION 2] IMPLICIT INTEGER (0..4294967295)
```
-- ne repasse pas à 0

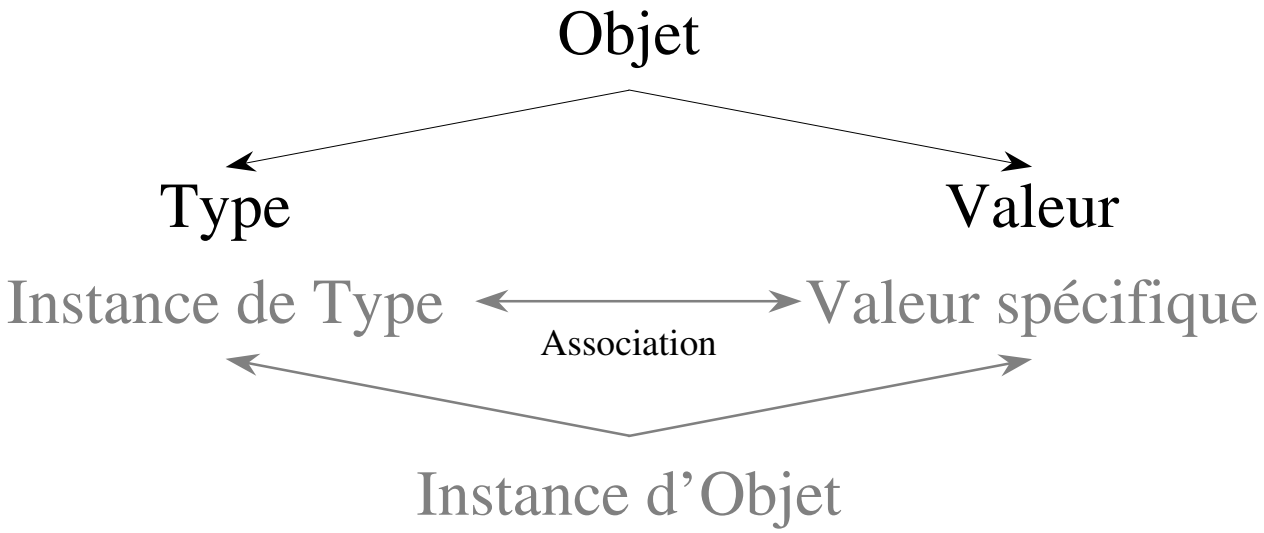
```
TimeTicks::= [APPLICATION 3] IMPLICIT INTEGER (0..4294967295)
```
-- compte le tps en centième de sec depuis une époque donnée

```
Opaque ::= [APPLICATION 4] IMPLICIT OCTET STRING
```
-- représente un encodage arbitraire

- Deux types construits

<list> ::= SEQUENCE { <type 1>...<type n>}
<table> ::= SEQUENCE OF <list>

Définition d'Objets



- Définition d'un nouveau type "Object " ?
  - chaque objet de la MIB sera de ce type !
  → définitions peu maniables

- Définition d'un ensemble ouvert de nouveaux types ?
  - un pour chaque catégorie d'objet géré
  - seule restriction, utilisation de ASN.1
  → variations dans le format des définitions d'objets

- Utilisation d'une macro pour définir un ensemble de types liés, utilisés pour définir les objets gérés !

Utilisation des macros

→ Trois niveaux de définition

- Macro definition
  - définit les instances de macros autorisées,
  - définit la syntaxe d'un ensemble de types liés.

- Macro instance
  - une instance générée à partir d'une définition de macro spécifique en fournissant des arguments aux paramètres,
  - spécifie une type particulier.

- Macro instance value
  - représente une entité spécifique avec une valeur spécifique.

```
< macroname > MACRO ::=
BEGIN
    TYPE NOTATION ::= <new-type-syntax >
                          -- décrit le nouveau type
    VALUE NOTATION ::= < new-value-syntax >
                          -- décrit les valeurs du nouveau type

    < supporting-productions >
END
```

Format d'une définition de macro

Macro pour la définition des objets gérés

```
IMPORT     ObjectName, ObjectSyntax FROM RFC-1155-SMI

OBJECT-TYPE MACRO ::=
BEGIN
        TYPE NOTATION       ::=     "SYNTAX" type (TYPE ObjectSyntax)
                                    "ACCESS" Access
                                    "STATUS" Status
                                    DescPart
                                    ReferPart
                                    IndexPart
                                    DefValPart
        VALUE NOTATION      ::=     value (VALUE ObjectName)

        Access ::= "read-only" | "read-write" | "write-only" | "not-accessible"

        Status ::= "mandatory" | "optional" | "obsolete" | "deprecated"

        DescPart ::= "DESCRIPTION" value (description DisplayString) | empty

        ReferPart ::= "REFERENCE" value (reference DisplayString) | empty

        IndexPart ::= "INDEX" "{" IndexTypes "}"

        IndexTypes ::= IndexType | IndexType  "," IndexType

        IndexType ::= value (indexobject ObjectName) | type (indextype)

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

        DisplayString ::= OCTET STRING SIZE (0..255)
END
IndexSyntax ::= CHOICE {        number INTEGER (0..MAX),
                                string OCTET STRING,
                                object OBJECT IDENTIFIER,
                                address NetworkAddress,
                                IpAddress IpAddress      }
```

Exemple de définition d'objet

tcpMaxConn OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "The limit on the total number of TCP connections the entity can support. In entities where the maximum number of connections is dynamic, this object should contain the value –1"
    ::= { tcp 4 }

Les MIBs

Version 2 de la MIB
mib-2 Object Identifier ::= {mgmt 1}

Critères de conception de la MIB-2

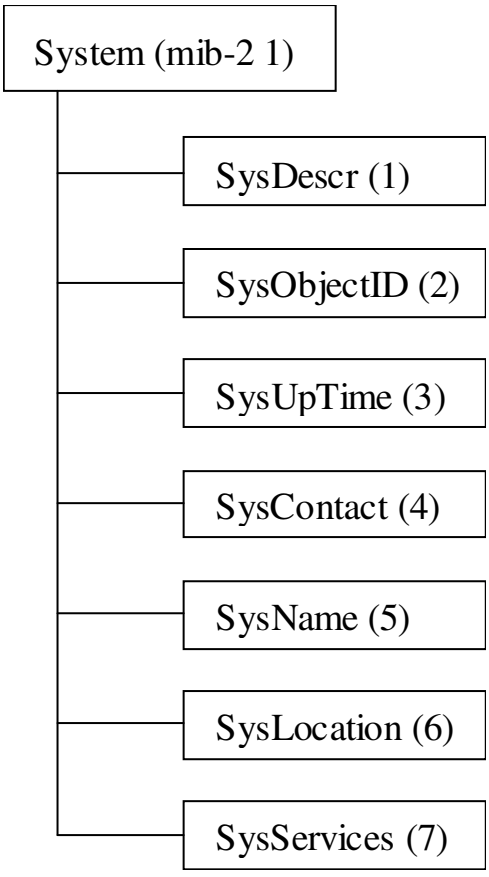
- Un objet doit être essentiel soit pour la gestion des pannes soit pour la gestion des configurations.
- Seuls les objets "faiblement" contrôlables sont permis.
- Une utilité et un usage courant évidents doivent être requis.
- Plus de limitation à 100 objets.
- Éviter les variables redondantes.
- Les objets spécifiques à des implémentations ont été exclus.

MIB-2 : 10 sous ensembles

- system
- interfaces
- at
- ip
- icmp
- tcp
- udp
- egp
- transmission
- snmp

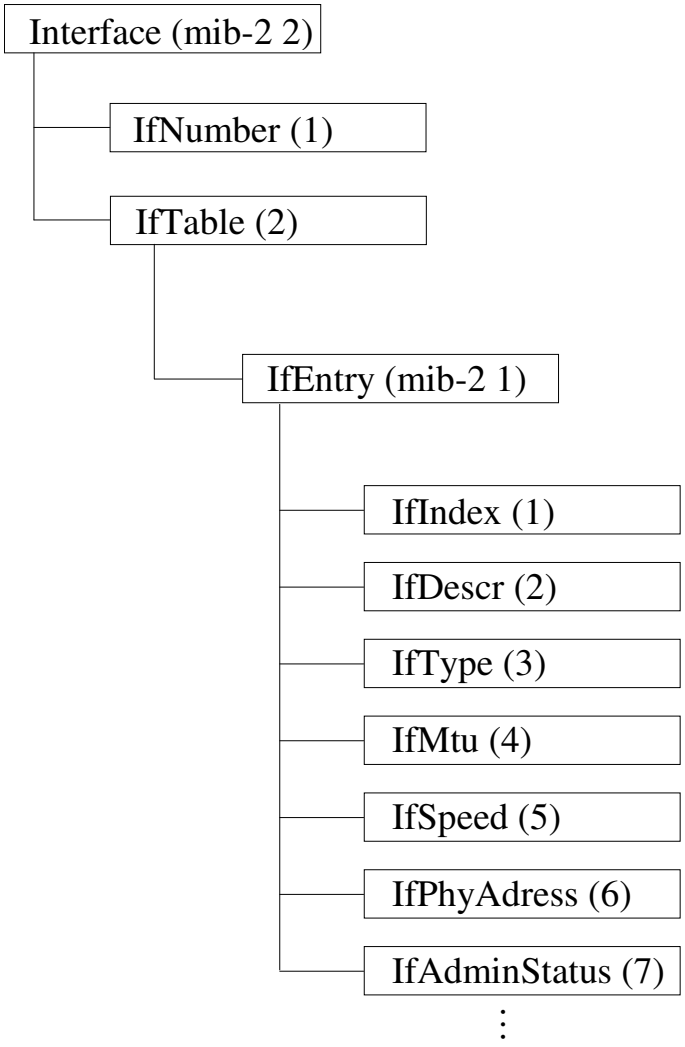
- Groupe *system*

  Donne des informations générales sur l'entité gérée : description du type de matériel, type de système d'exploitation, durée depuis la dernière réinitialisation, identification de la personne à contacter pour ce nœud, nom assigné, localisation physique et valeur indiquant l'ensemble des services offerts.



- Groupe *interface*

  Contient des informations concernant les différentes interfaces réseau de l'entité gérée, incluant des informations de configuration et des statistiques sur les occurrences de certains événements sur chaque interface : nombre d'interfaces, type des interfaces et nom du fabricant, vitesse des interfaces, nombre de paquets entrants, sortants, en erreur, ...



- Groupe *at*

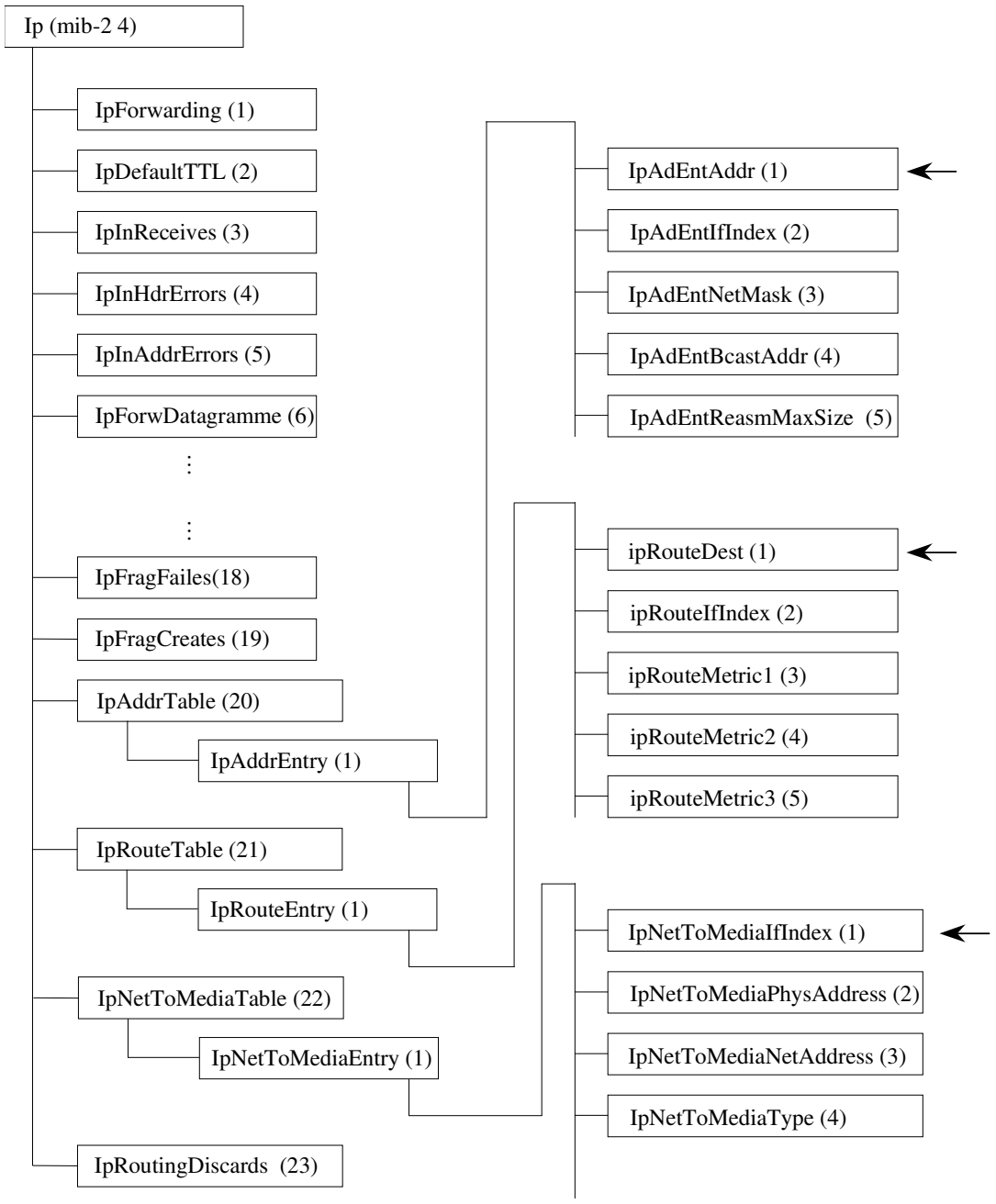
  Conservé pour des raisons de compatibilité avec MIB-I. Constitué d'une seule table de translation entre des adresses réseau de niveau logique (IP) et des adresses physiques (MAC, X.121, …). Équivalent à la table ARP.

- Groupe *Ip*

  Contient des informations sur l'implantation et les opérations IP. Tous les objets ne sont pas nécessairement significatifs pour tous les nœuds.

  Trois tables sont incluses : `ipAddrTable`, `ipRouteTable` et `ipNetToMediaTable`.

  (durée de vie par défaut des paquets IP, nb de paquets reçus ou émis, nb de paquets réassemblés avec succès, …) la partie de la MIB la plus importante.



- Groupe *icmp*

  Constitué uniquement de compteurs (26) pour comptabiliser les messages ICMP émis et reçus.

- Groupe *tcp*

  Rend compte des connexions TCP en cours et des paramètres de type nombre max de connexions simultanées permises, nombre d'ouvertures actives, ... et l'état de chaque connexion (écoute, time-wait, ...).

  Contient une seule table `tcpConnTable`.

- Groupe *udp*

  Contient des informations relatives à l'implantation et aux opérations UDP. En plus des informations sur les datagrammes émises, reçues et en erreur, le groupe inclut la table `udpTable` (n° IP, n° port) qui renseigne sur les entités pour lesquelles une application locale accepte des datagrammes.

- Groupe *EGP*

  Gère le protocole EGP (External Gateway Protocol - routage des paquets entre routeurs). Rend compte du nombre de paquets entrants, sortants, en erreur. Inclut la table `egpNeighTable` qui contient des informations sur chacun des routeurs voisins connus.

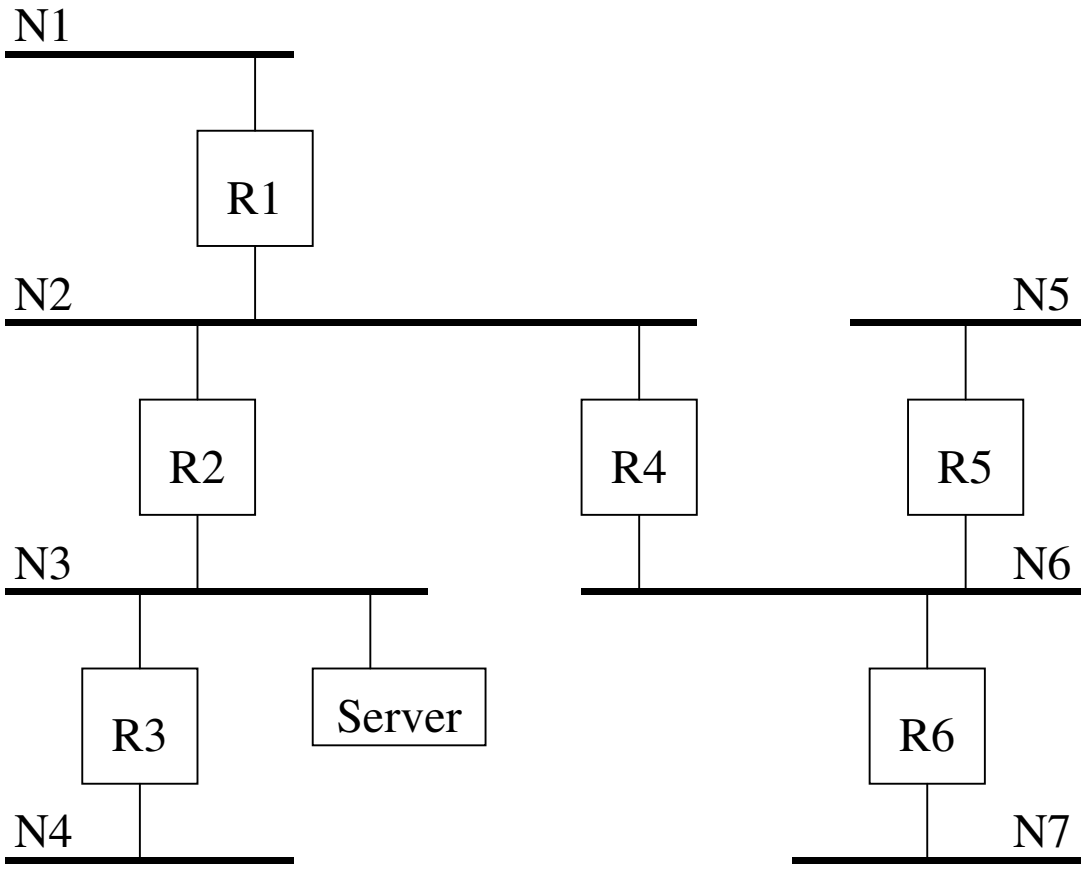
- Groupe *transmission*

  Ne contient que type Object Identifier ::={transmission number} qui permet d'identifier le type de media utilisé pour la transmission, pour chacune des interfaces.

- Groupe *snmp*

  Contient le nombre de messages SNMP entrants et sortants, de mauvaises versions reçues ou de nom de communauté invalide, la répartition du type de requêtes (get, get_next, set et trap). Constitué que de compteurs en lecture, excepté l'objet `snmpEnableAuthenTraps`.

Limitation de la MIB-II



Solution : un ensemble d'objets "plus fins"
→ un contrôle plus précis (grand) du réseau,
mais nécessite
→ plus de capacité mémoire au sein des agents,
→ plus de traitement au sein des agents,
→ plus de trafic SNMP.
ce qui est contraire à l'esprit de conception de la MIB-II, qui est de minimiser la surcharge des systèmes et des réseaux.

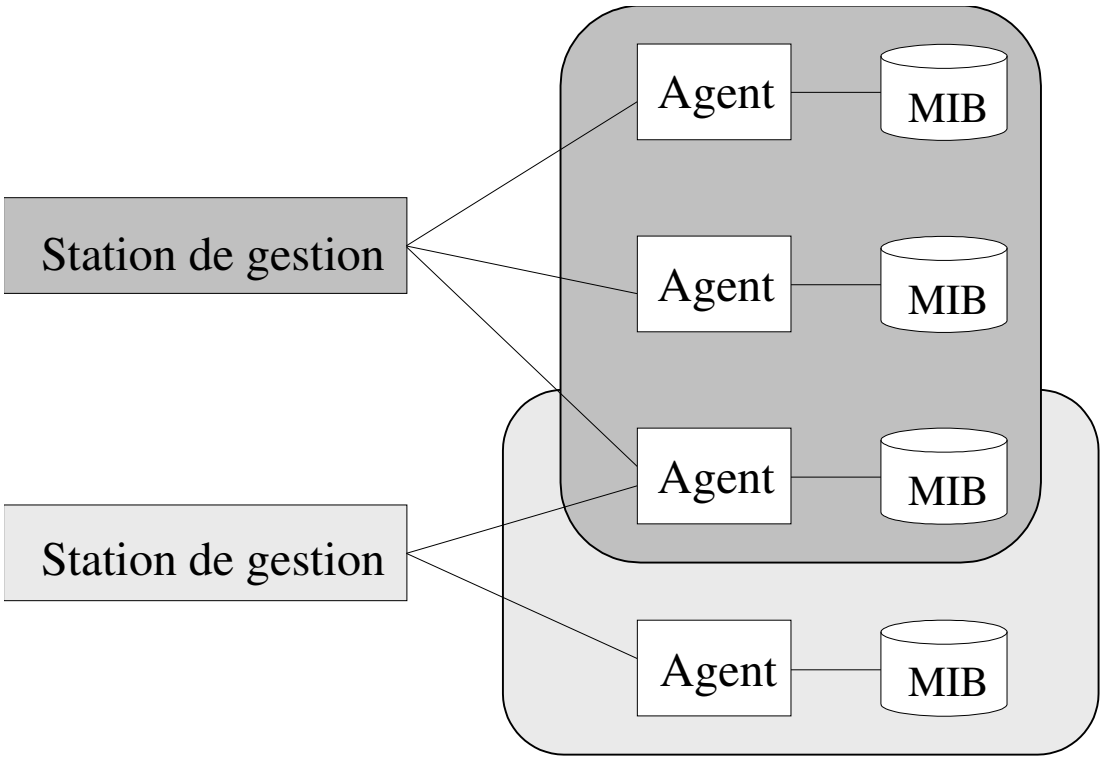
Règles d'Utilisation

- Il n'est pas possible de changer la structure de la MIB par ajout ou retrait d'instances.
- Il n'est pas possible d'émettre une commande pour l'exécution d'une action.
- L'accès aux objets est possible uniquement sur les objets-feuilles de l'arbre des identificateurs d'objets.
- Il n'est pas possible d'accéder à la totalité, ou à une rangée, d'une table par une opération atomique.
- Par convention, il est possible d'exécuter des opérations sur des tables à deux dimensions.

→ Restrictions simplifiant l'implantation de SNMP.

→ Restrictions limitant la capacité du système d'administration.

Communautés et Noms de Communautés



Le contrôle d'accès par les différentes stations d'administration à la MIB de chaque agent comporte trois aspects :

- **un service d'authentification** : un agent peut souhaiter limiter les accès à la MIB aux stations d'administrations autorisées,
- **une politique d'accès** : un agent peut donner des privilèges différents aux différentes stations d'administration,
- **un service de mandataire** (proxy) : un agent peut agir comme un proxy pour d'autres stations gérées.

→ Concernent la sécurité
→ Concept de communauté

Définition de la Communauté

- La communauté SNMP est une relation, entre un agent et les stations d'administration, qui définit l'authentification, le contrôle d'accès et les caractéristiques des proxys.

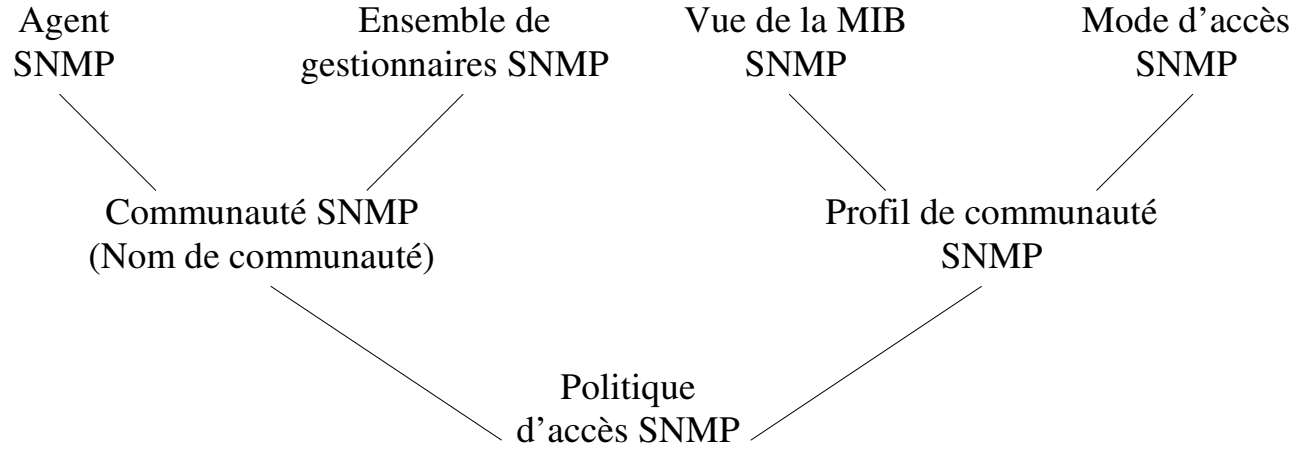
- Le concept de communauté est local à un agent, et donc défini au sein de l'agent.

- Un agent établit une communauté pour chaque combinaison d'authentification, de contrôle d'accès et de caractéristiques de proxys.

- Chaque communauté définie entre un agent et ses stations d'administration a un nom unique (pour l'agent) employé lors des requêtes SNMP.

- Une station d'administration garde la liste des noms de communauté donnés par les différents agents.

Les Concepts d'Administration



L'authentification

- ⇨ Doit assurer l'agent que le message vient bien de la source citée dans le message.
- ⇨ SNMP fournit un schéma d'authentification simple : chaque message d'une station d'administration comporte le nom de la communauté.
- ⇨ Ce nom fonctionne comme un mot de passe, et le message est dit authentifié si l'émetteur connaît le mot de passe.
- ⇨ Léger ! ce qui fait que les opérations `set` et `trap` sont mises dans des communautés à part.

La politique d'accès

- ⇨ Un agent limite l'accès à sa MIB à une sélection de stations d'administration.
- ⇨ Il fournit plusieurs types d'accès en définissant plusieurs communautés.
- ⇨ Ce contrôle d'accès a deux aspects :
  - une vue de la MIB : un sous-ensemble des objets de la MIB. Différentes vues de la MIB peuvent être définies pour chaque communauté.
  - un mode d'accès SNMP : un élément de l'ensemble {read-only, read-write}. Il est défini pour chaque communauté.

La vue de la MIB et le mode d'accès forment ce que l'on appelle le profil de la communauté SNMP.

Le service de proxy

- ⇨ C'est un agent SNMP qui agit pour d'autres périphériques (qui ne supportent pas, par exemple, TCP/IP).
- ⇨ Pour chaque périphérique représenté par le système de proxy, celui-ci doit maintenir une politique d'accès.
- ⇨ Le proxy connaît quels sont les objets MIB utilisés pour gérer le système mandaté (la vue de la MIB et les droits d'accès).

Identification d'Instance

Nous avons vu que chaque objet de la MIB a un identifiant unique qui est défini par sa position dans la structure en arbre de la MIB.

Quand un accès est fait à une MIB, via SNMP, c'est une instance spécifique d'un objet qu'on veut et non un type d'objet.

SNMP offre deux moyens pour identifier une instance d'objet spécifique dans une table :

- Une technique d'accès séquentiel

  L'ordre lexicographique des objets dans la structure de la MIB est utilisé.

- Une technique d'accès direct

Définition d'une Table

Une table a la syntaxe suivante :

```
SEQUENCE OF <entry>
```

Un rang a la syntaxe suivante :

```
SEQUENCE { <type1>, ... <typeN> }
```

les types définissent chaque colonne d'objets et chaque type a la forme suivante:

<descriptor> <syntax>

<descriptor> : nom de la colonne
<syntax> : valeur de la syntaxe

Exemple de table

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


tcpConnEntry OBJECT-TYPE
     SYNTAX    TcpConnEntry
     ACCESS    not-accessible
     STATUS    mandatory
     DESCRIPTION
          "Information about a particular TCP connection."
     INDEX     {tcpConnLocalAdress,
                tcpConnLocalPort,
                tcpConnRemAdress,
                tcpConnRemPort,
     ::= { tcpConnTable 1 }


TcpConnEntry ::= SEQUENCE {
                    tcpConnState        INTEGER,
                    tcpConnLocalAdress  IpAddress,
                    tcpConnLocalPort    INTEGER(0..65535),
                    tcpConnRemAdress    IpAddress,
                    tcpConnRemPort      INTEGER(0..65535)}


tcpConnState OBJECT-TYPE
     SYNTAX    INTEGER { closed(1), listen(2), …, deleteTCB(12)}
     ACCESS    read-write
     STATUS    mandatory
     DESCRIPTION "… "
     ::= {tcpConnEntry 1}


tcpConnLocalAdress OBJECT-TYPE
     SYNTAX    IpAddress
     ACCESS    read-only
     STATUS    mandatory
     DESCRIPTION "… "
     ::= {tcpConnEntry 2}
```

Exemple de table (suite)

```
tcpConnLocalPort OBJECT-TYPE
     SYNTAX    INTEGER(O..65535)
     ACCESS    read-only
     STATUS    mandatory
     DESCRIPTION "… "
     ::= {tcpConnEntry 3}

tcpConnRemAdress OBJECT-TYPE
     SYNTAX    IpAddress
     ACCESS    read-only
     STATUS    mandatory
     DESCRIPTION "… "
     ::= {tcpConnEntry 4}

tcpConnRemPort OBJECT-TYPE
     SYNTAX    INTEGER(O..65535)
     ACCESS    read-only
     STATUS    mandatory
     DESCRIPTION "… "
     ::= {tcpConnEntry 5}
```

Chaque colonne d'objets est définie de la manière habituelle avec une macro `OBJECT-TYPE`.
Chaque élément a un identifiant unique.

tcpConnState 1.3.6.1.2.1.6.13.1.1	tcpConnLocalAddress 1.3.6.1.2.1.6.13.1.2	TcpConnLocal 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	TcpConnEntry 1.3.6.1.2.1.6.13.1
2	0.0.0.0	99	0.0.0.0	0	TcpConnEntry 1.3.6.1.2.1.6.13.1
3	10.0.0.99	14	89.1.1.42	84	TcpConnEntry 1.3.6.1.2.1.6.13.1
	↑ INDEX	↑ INDEX	↑ INDEX	↑ INDEX	

Les trois instances de `tcpConnState` ont le même identifiant : `1.3.6.1.2.1.6.13.1.1`

L'index de table

- La clause INDEX définit un rang. Elle détermine sans ambiguïté la valeur de l'objet.

- La règle de construction de l'identifiant de l'instance d'une instance de colonne d'objets est la suivante :

  Soit un objet dont l'identifiant d'objet est y, dans une table avec des objets INDEX $i_1, i_2, \ldots, i_n$, alors l'identifiant d'instance pour une instance d'objet y dans un rang particulier est

  $$y.(i_1).(i_2)...(i_n)$$

Règles de conversion de la valeur d'une instance d'objet en un ou plusieurs sous-identifiants :

- entier →
- chaîne de taille fixe →
- chaîne de taille variable →
- identificateur d'objet →
- adresse IP →

On distingue par les index les différentes colonnes. On combine l'identifiant de l'objet pour une colonne et un ensemble de valeur de l'Index pour obtenir le rang.

tcpConnState 1.3.6.1.2.1.6.13.1.1	tcpConnLocalAddress 1.3.6.1.2.1.6.13.1.2	TcpConnLocalPort 1.3.6.1.2.1.6.13.1.3	TcpConnRemAddress 1.3.6.1.2.1.6.13.1.4	TcpConnRemPort 1.3.6.1.2.1.6.13.1.5	
x.1.10.0.0.99.12. 9.1.2.3.15	x.2.10.0.0.99.12. 9.1.2.3.15	x.3.10.0.0.99.12. 9.1.2.3.15	x.4.10.0.0.99.12. 9.1.2.3.15	x.5.10.0.0.99.12. 9.1.2.3.15	TcpConnEntry 1.3.6.1.2.1.6.13.1
x.1.0.0.0.0.99. 0.0.0.0.0	x.2.0.0.0.0.99. 0.0.0.0.0	x.3.0.0.0.0.99. 0.0.0.0.0	x.4.0.0.0.0.99. 0.0.0.0.0	x.5.0.0.0.0.99. 0.0.0.0.0	TcpConnEntry 1.3.6.1.2.1.6.13.1
x.1.10.0.0.99.14. 89.1.1.42.84	x.2.10.0.0.99.14. 89.1.1.42.84	X.3.10.0.0.99.14. 89.1.1.42.84	x.4.10.0.0.99.14. 89.1.1.42.84	x.5.10.0.0.99.14. 89.1.1.42.84	TcpConnEntry 1.3.6.1.2.1.6.13.1
	↑ INDEX	↑ INDEX	↑ INDEX	↑ INDEX	

`x` = 1.3.6.1.2.1.6.13.1 = identifiant de l'objet `tcpConnEntry` qui est l'identifiant d'un rang de `tcpConnTable`
`i` = le dernier sous-identificateur de la colonne (sa position dans la table)
(`name`) = valeur de l'objet `name`
Tous les identifiants d'instances de `tcpConnTable` ont la forme :

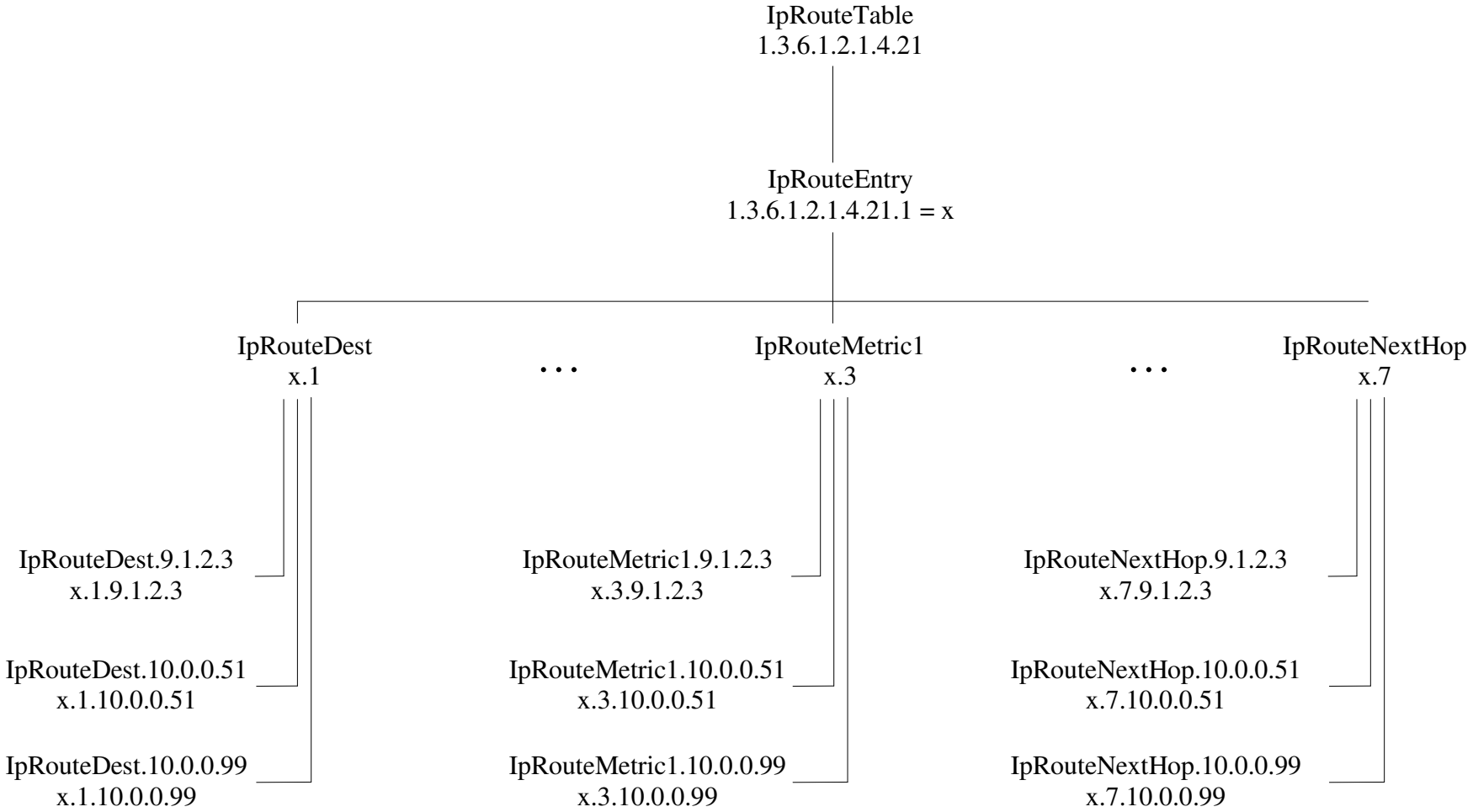
```
x.i.(tcpConnLocalAddress).(tcpConnLocalPort).(tcpConnRemAddress).(tcpConnRemPort)
```

L'ordre lexicographique

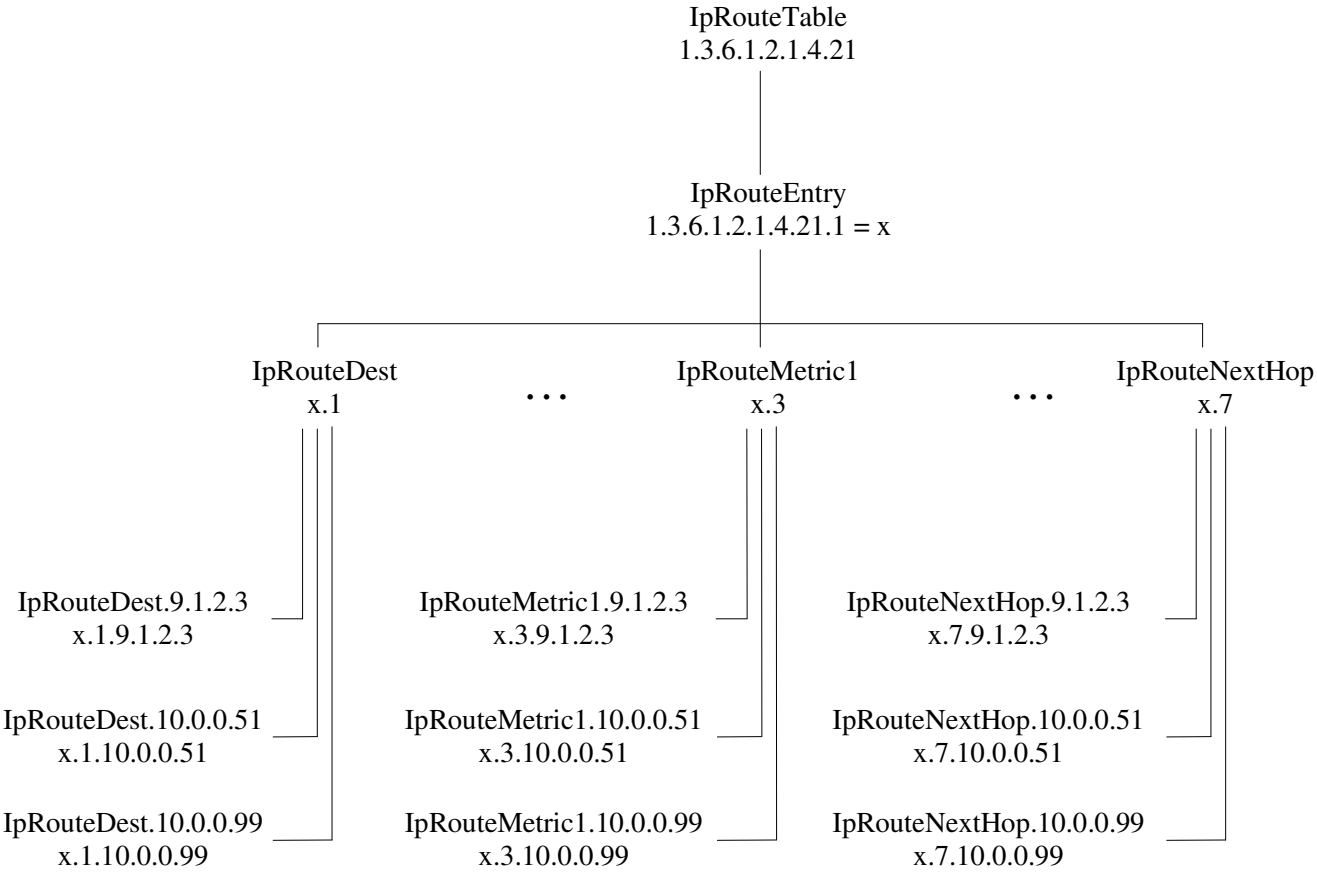
L'identifiant d'objet est une séquence d'entiers qui reflète une structure hiérarchique des objets de la MIB.

- un identifiant d'objet pour un objet donné peut être dérivé par la trace du chemin de la racine à l'objet,
- l'utilisation d'entiers apporte un ordre lexicographique,
- la règle : les nœuds "fils" sont définis en ajoutant un entier à l'identifiant du père et en visitant l'arbre de haut en bas et de gauche à droite,
- cela permet d'accéder aux différents objets de la MIB sans vraiment en connaître le nom spécifique de l'objet,
- la station d'administration peut donner un identifiant d'objet ou un identifiant d'instance d'objet et demander l'instance de l'objet qui est le suivant dans l'ordre.

Exemple


IpRouteTable
1.3.6.1.2.1.4.21
IpRouteEntry
1.3.6.1.2.1.4.21.1 = x
IpRouteDest
x.1
IpRouteMetric1
x.3
IpRouteNextHop
x.7
IpRouteDest.9.1.2.3
x.1.9.1.2.3
IpRouteDest.10.0.0.51
x.1.10.0.0.51
IpRouteDest.10.0.0.99
x.1.10.0.0.99
IpRouteMetric1.9.1.2.3
x.3.9.1.2.3
IpRouteMetric1.10.0.0.51
x.3.10.0.0.51
IpRouteMetric1.10.0.0.99
x.3.10.0.0.99
IpRouteNextHop.9.1.2.3
x.7.9.1.2.3
IpRouteNextHop.10.0.0.51
x.7.10.0.0.51
IpRouteNextHop.10.0.0.99
x.7.10.0.0.99

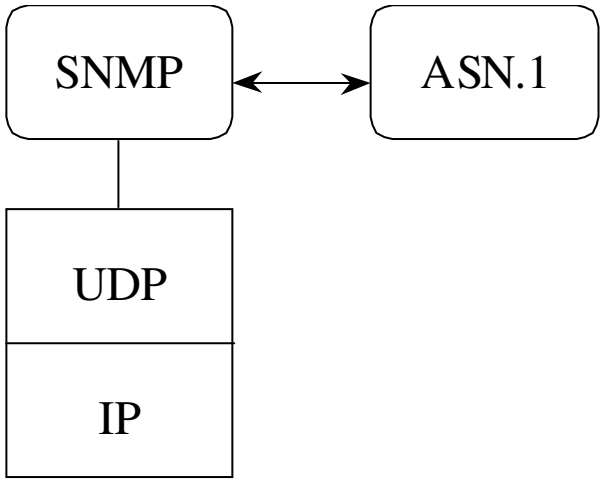
Exemple



Objet	Identifiant d'objet	Prochaine instance d'objet dans l'ordre lexicographique
ipRouteTable	1.3.6.1.2.1.4.21	1.3.6.1.2.1.4.21.1.1.9.1.2.3
ipRouteEntry	1.3.6.1.2.1.4.21.1	1.3.6.1.2.1.4.21.1.1.9.1.2.3
ipRouteDest	1.3.6.1.2.1.4.21.1.1	1.3.6.1.2.1.4.21.1.1.9.1.2.3
ipRouteDest.9.1.2.3	1.3.6.1.2.1.4.21.1.1.9.1.2.3	1.3.6.1.2.1.4.21.1.1.10.0.0.51
ipRouteDest.10.0.0.51	1.3.6.1.2.1.4.21.1.1.10.0.0.51	1.3.6.1.2.1.4.21.1.1.10.0.0.99
IpRouteDest.10.0.0.99	1.3.6.1.2.1.4.21.1.1. 10.0.0.99	1.3.6.1.2.1.4.21.1.3.9.1.2.3

Le Protocole SNMP

- L'architecture du réseau utilisé



- Format d'un message SNMP

  - un identificateur de version : n° de version SNMP,
  - un nom de communauté,
  - une PDU.

Version	Communauté	PDU SNMP

Les Requêtes SNMP

L'interaction entre un manager et les objets gérés se fait par l'intermédiaire d'un agent. Un manager peut émettre à un agent les requêtes suivantes :

- Get

  Récupère les valeurs d'un ensemble de variables dont les OIDs sont donnés directement.
- Set

  Modifie les valeurs d'un ensemble de variables dont les OIDs sont donnés directement.
- Get-Next

  Récupère les valeurs d'un ensemble de variables situé après les variables données dans la requête et suivant l'ordre lexicographique (par rapport aux OIDs).
- Get-Bulk

  Est un Get-Next compacté permettant de récupérer les valeurs d'un ensemble de variables d'une portée donnée, suivant l'ordre lexicographique (v2).
- Response

  Renvoie à un manager les réponses aux requêtes SNMP citées plus haut.
- Trap

  Est un message non sollicité émis par l'agent vers le manager. Son utilisation est très limitée.
- InformRequest

  Est un message permettant la communication entre deux managers SNMP (v2).

La spécification du protocole

- Les formats SNMP

◊ SNMP message

Version	Community	SNMP PDU

◊ `GetRequest` PDU, `GetNextRequest` PDU, `SetRequest` PDU

PDU Type	request-id	0	0	variablebindings

◊ `GetResponse` PDU

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

◊ `Trap` PDU

PDU Type	enterprise	agent-addr	generic-trap	specific-trap	time-stamp	variable bindings

◊ `variablebindings`

name1	value1	name2	value2	...	namen	valuen

Émission/Réception d'un Message

- Émission d'un message

  Une entité SNMP accomplit les actions suivantes pour transmettre l'un des 5 types de PDU à une autre entité SNMP :

  1. construction de la PDU en utilisant la structure ASN.1,
  2. la PDU est transmise à un service d'authentification, avec les adresses source et destinataire ainsi que le nom de communauté
  3. l'entité du protocole construit un message constitué d'un champs version, le nom de communauté et le résultat de l'étape 2,
  4. ce nouvel objet ASN.1 est encodé, en utilisant BER, et passé au service transport.

- Réception d'un message

  Une entité SNMP accomplit les actions suivantes sur réception d'un message SNMP :

  1. message ASN.1 correct ? Non → fin
  2. version OK ? Non → fin
  3. l'entité du protocole passe la PDU et les adresses source et destinataire à un service d'authentification.
     a. si l'authentification échoue, information de l'entité de protocole SNMP qui génère éventuellement une trap, et supprime le message,
     b. si l'authentification réussit, retourne la PDU sous forme d'objet ASN.1
  4. l'entité du protocole vérifie la syntaxe de la PDU.

     si échec, suppression du message,

     sinon, en utilisant le nom de communauté, la politique d'accès SNMP appropriée est sélectionnée et la PDU est traitée.

```
RFC1157-SNMP DEFINITIONS ::= BEGIN
IMPORTS
   ObjectName, ObjectSyntax, NetworkAddress, IpAddress, TimeTicks
   FROM RFC1155-SMI;
```

-- top-level message

```
Message ::= SEQUENCE
   { version INTEGER {version-1 (0)},   -- Version 1 for this RFC
     community OCTET STRING,            -- Community name
     data ANY                           -- e.g. PDUs
   }
```

-- Protocol data units

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

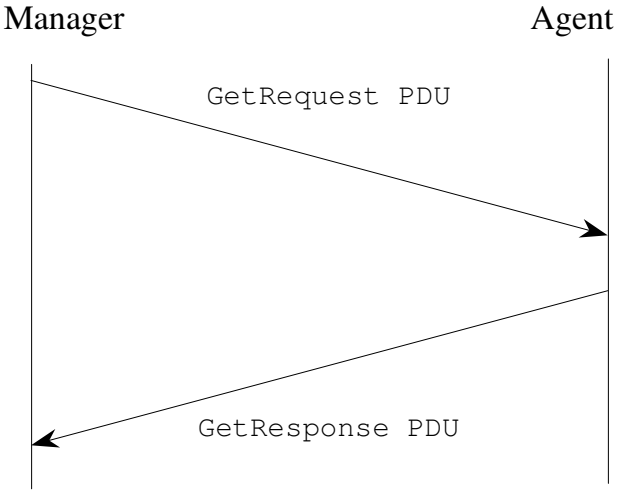
-- PDUs

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

PDU ::= SEQUENCE
   { request-id INTEGER,                -- Request identifier
     error-status INTEGER {             -- Sometimes ignored
        noError (0),
        toobig (1),
        noSuchName (2),
        badValue (3),
        readOnly (4),
        genError (5)},
     error-index INTEGER,               -- Sometimes ignored
     variable-binding VarBindList}      -- Values are sometimes ignored


Trap-PDU ::= [4] IMPLICIT SEQUENCE
   { enterprise OBJECT IDENTIFIER,      --Type of object generating trap
     agent-addr NetworkAddress,         -- adress of object generating trap
     generic-trap INTEGER               -- generic trap type
        {
          coldStart (0),
          warmStart (1),
```

```
          linkDown (2),
          linkUp (3),
          authenticationFailure (4),
          egpNeighborLoss (5),
          enterpriseSpecific (6),
        },
    specific-trap INTEGER,              -- Specific code
    time-stamp TimeTicks,               -- time elapsed between the last
                                        --      (re)initialization of the network
    variable-binding VarBindList        -- "Interesting" information
  }

-- variable binding

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

END
```

Échange sur le réseau au niveau du service

- `GetRequest` PDU




```
GetRequest-PDU ::= [0]
   IMPLICIT SEQUENCE
   {   request-id RequestID,
       error-status ErrorStatus,        -- à 0
       error-index ErrorStatus,         -- à 0
       Variable_Binding VarBindList  }
```

Réception d'une GetRequest-PDU

Si pour chaque objet de la VarBindList l'objet ne correspond pas, alors envoi d'un GetResponse-PDU avec : ErrorStatus ← noSuchName et
ErrorIndex ← pointe sur la variable non ok dans la demande

Si GetResponse-PDU > limitation locale, alors envoi de GetResponse-PDU avec :
ErrorStatus ← tooBig et
ErrorIndex ← 0

Si une des variables demandées ne peut pas être obtenue, alors envoi de GetResponse-PDU avec :
ErrorStatus ← genErr et
ErrorIndex ← variable en erreur

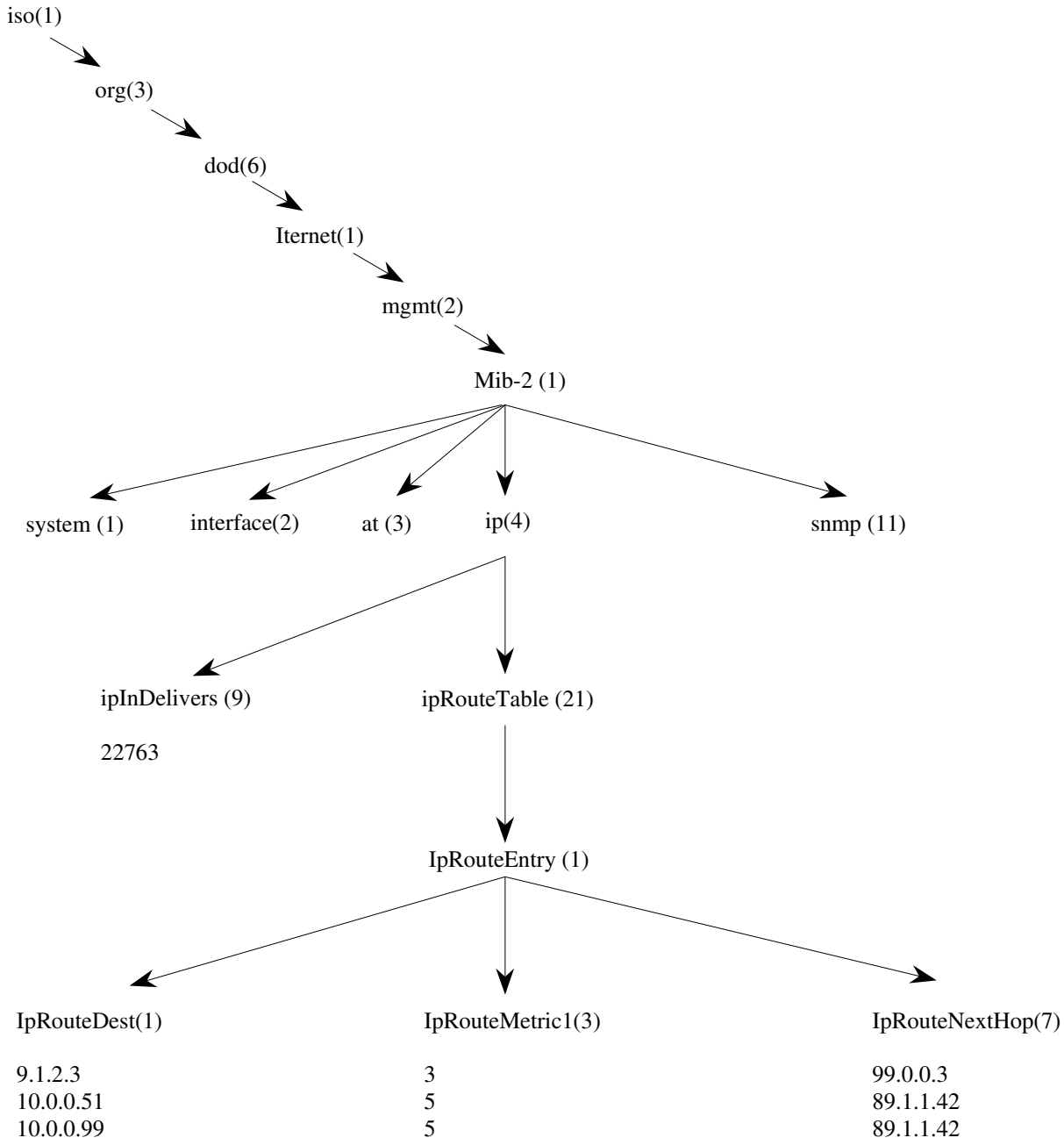
Si tout est OK, envoi d'un GetResponse-PDU où les variables sont associées aux valeurs.

Exemple

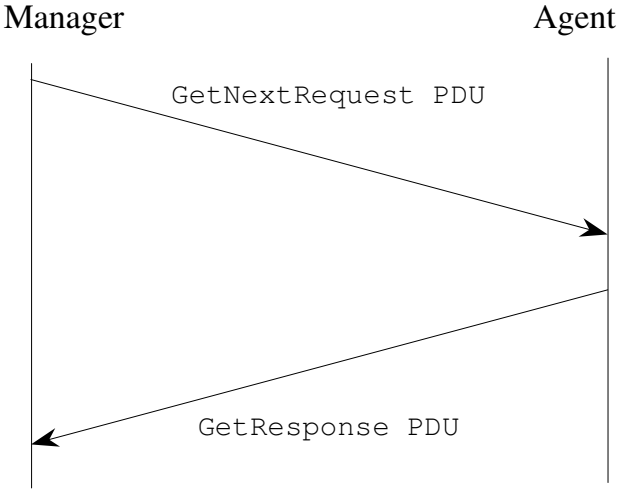
La désignation absolue de l'objet ipInDelivers est :

iso.org.dod.internet.mgmt.mib-2.ip.ipInDelivers
soit 1.3.6.1.2.1.4.9

La valeur de la variable ipInDelivers est obtenue par la commande GetRequest (1.3.6.1.2.1.4.9.0) qui produit la réponse GetResponse ((ipInDelivers.0 = 22763))



- GetNextRequest PDU




```
GetNextRequest-PDU::= [1]
    IMPLICIT SEQUENCE
    {  request-id RequestId,
       error-status ErrorStatus,
       error-index ErrorIndex,
       Variable_Bindings VarBindList   }
```

Réception d'un GetNextRequest PDU

Si, pour un objet de la VarBindList, le nom ne précède pas (lexicographiquement) le nom d'un objet accessible par un get, alors un GetResponse-PDU est renvoyé avec le même contenu et :

ErrorStatus ← noSuchName et
ErrorIndex ← pointe sur la variable non ok dans la demande

Si GetResponse-PDU > limitation locale, alors envoi de GetResponse-PDU avec :

ErrorStatus ← tooBig et
ErrorIndex ← 0

Si une des variables demandées ne peut pas être obtenue, alors envoi de GetResponse-PDU avec :

ErrorStatus ← genErr et
ErrorIndex ← variable en erreur

Si tout est OK, envoi d'un GetResponse-PDU où les variables sont associées à des valeurs.

Suite de l'exemple

On consulte la table de routage par la commande **GetNextRequest**

GetNextRequest (ipRouteDest, ipRouteMetric1, ipRouteNextHop)

On obtient alors la réponse suivante :
GetResponse ((ipRoutedest.9.1.2.3 = "9.1.2.3"),
(ipRouteMetric1.9.1.2.3 = "3"),
(ipRouteNextHop.9.1.2.3 = "99.0.0.3"))

On continue la consultation de la table par :
GetNextRequest (ipRoutedest.9.1.2.3, ipRouteMetric1.9.1.2.3,
ipRouteNextHop.9.1.2.3)

et on obtient la réponse suivante :
GetResponse ((ipRoutedest.10.0.0.51 = "10.0.0.51"),
(ipRouteMetric1.10.0.0.51 = "5"),
(ipRouteNextHop. 10.0.0.51 = "89.1.1.42"))

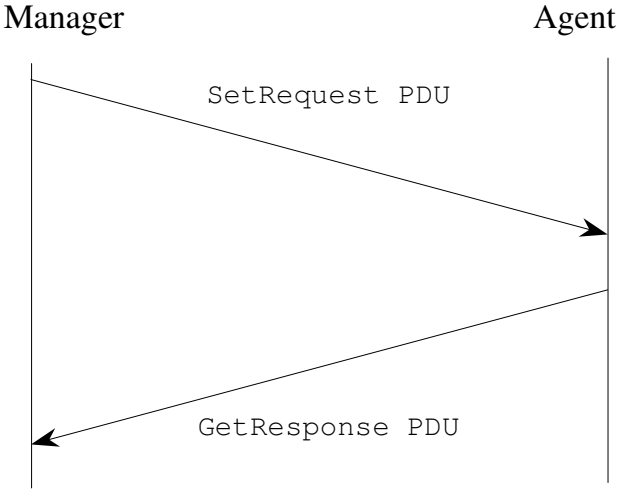
Enfin, par la dernière ligne de la consultation de la table :
GetNextRequest (ipRoutedest.10.0.0.51, ipRouteMetric1.10.0.0.51,
ipRouteNextHop. 10.0.0.51)

on obtient la réponse suivante :
GetResponse ((ipRoutedest.10.0.0.99 = "10.0.0.99"),
(ipRouteMetric1.10.0.0.99 = "5"),
(ipRouteNextHop.10.0.0.99 = "89.1.1.42"))

Si, à nouveau, le manager émet une requête
GetNextRequest (ipRoutedest.10.0.0.99, ipRouteMetric1.10.0.0.99,
ipRouteNextHop. 10.0.0.99)

Il obtient
GetResponse ((ipRouteMetric1.9.1.2.3 = "3"),
(ipRouteNextHop.9.1.2.3 = "99.0.0.3"),
(ipNetToMediaIfIndex.1.3 = "1"))

- SetRequest PDU




```
SetRequest-PDU ::= [3]
    IMPLICIT SEQUENCE
    {  request-id RequestId,
       error-status ErrorStatus,        -- à 0
       error-index ErrorIndex,          -- à 0
       variable_bindings VarBindList  }
```

Réception d'un message SetRequest-PDU

Si, pour chaque objet du champ Variable-Bindings, l'objet n'est pas accessible pour l'opération demandée, alors la PDU GetResponse-PDU (de forme identique) est envoyée avec :

ErrorStatus ← noSuchName et
ErrorIndex ← pointeur sur la variable en erreur

Si, pour chaque objet de la VarBindList, la valeur ne correspond pas au type attendu (longueur, valeur,..), alors la PDU GetResponse-PDU (de forme identique) est envoyée avec :

ErrorStatus ← BadValue et
ErrorIndex ← variable en erreur

Si GetResponse-PDU > limitation locale, alors envoi de GetResponse-PDU avec :

ErrorStatus ← tooBig et
ErrorIndex ← 0

Si une des variables de la VarBindList ne peut pas être mise à jour, alors GetResponse-PDU est envoyée avec :

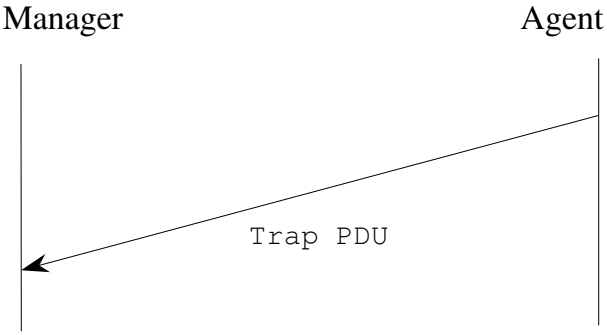
ErrorStatus ← genErr et
ErrorIndex ← variable en erreur

Si tout est OK, les variables citées dans la VarBindList sont mises à jour et un GetResponse-PDU est envoyé (avec un contenu identique) avec :

ErrorStatus ← 0 et
ErrorIndex ← 0

• Trap PDU

```
Trap-PDU::= [4] IMPLICIT SEQUENCE
{  enterprise Object Identifier,
   agent-addr NetworkAddress,        --addr générateur trap
   generic-trap INTEGER
   {     coldstart (0),
         warmstart (1),
         linkdown (2),
         linkup(3),
         authentificationfailure(4),
         egpneigborloss(5),
         enterprisespecific(6)     }
   specific-trap INTEGER,
   variable-bindings VarBindList     }
   time-stamp TimeTicks              -- temps depuis reboot
   variable-bindings VarBindList     }
```



Valeurs utilisées dans le "generic-trap"

coldstart
: l'agent envoyant le trap se réinitialise suite à un incident (crash, erreur majeure, …). Le redémarrage n'était pas prévu.

warmstart
: l'agent envoyant le trap se réinitialise suite à une altération de ses données.

linkdown
: signale une erreur sur une voie de communication de l'agent. Le premier élément de la VarBindList précise l'interface en erreur.

linkup
: signale qu'une voie de communication de l'agent est mise en service. Le premier élément de la VarBindList précise l'interface activée.

authentificationfailure
: signale que l'agent a reçu un message non authentifié.

egpneihborloss
: le routeur voisin de l'agent qui communiquait avec lui via EGP vient d'être stoppé.

enterprisespecific
: indique qu'un événement spécifique vient de se produire. Le "specific-trap" indique le numéro de trap concerné.

Conclusion

- ◊ modélisation par groupe d'objets ou variables,
- ◊ scrutation des agents,
- ◊ mode non connecté,
- ◊ 5 opérations : GetRequest, GetNextRequest, GetResponse, SetRequest, Trap - Opérations atomiques.

- Avantage
  - ◊ simple
- Faiblesses
  - ◊ interrogation périodique : polling → limite le nombre d'agents pouvant être supervisés,
  - ◊ pas d'initiatives des agents, sauf exceptions,
  - ◊ mode non connecté : sécurité des messages mal assurée,
  - ◊ comptabilité entre MIB propriétaires,
  - ◊ pas ou peu de sécurité : nom de communauté.

Bibliographie

Les divers RFC sur SNMP accessibles sur le serveur web de l'Urec ou à Pasteur (www.urec.fr, www.pasteur.fr).

SNMPv1
RFCs 1089,1140, 1147, 1155, 1156, 1157, 1158, 1161, 1212, 1213, 1215, 1298.

SNMPv2
RFCs 1441, 1442, 1443, 1444, 1445, 1446, 1447, 1448, 1449, 1450, 1451, 1452.

Les livres

The Simple Book : An Introduction to management of TCP/IP-based Internets by Marshall Rose, Prentice Hall, 2nd edition, 1994

SNMP, SNMPv2, CMIP, The pratical Guide to Network-Management Standards, William Stallings, Addison Wesley, 1995

Rapport de valeur C 94-95 "SNMP", O. Porte, M. Izadpahan

Mémoire d'ingénieur
"Mise en oeuvre du protocole SNMP pour un outil de gestion hétérogène",
G. Ndjeudji, 1992

Les sites qui offrent des logiciels SNMP

lancaster.andrew.cmu.edu:/pub/snmp-dist/*
snmp2.1.2.tar
CMU SNMPv2 source library agent, mid-level agent, net management routines

ftp.ics.uci.edu:mrose/isode-snmpv2/isode-snmpv2.tar.Z
4BSD/ISODE 8.0 SNMPv2 package

dnpap.et.tudelft.nl:/pub/btng
contient RMON agent pour OS/2, SUN OS 4.1.x &Ultrix
Tricklet (perl based SNMP tool pour Unix ou OS/2)
…