Thème n°5 : L'Administration des Réseaux

L'administration de l'Internet:

SNMP

(Simple Network Management Protocol)

www.Mcours.com
Site N°1 des Cours et Exercices Email: contact@mcours.com

UV B : Complément Réseaux de Transport et Applications

Année 95/96

Laurence Duchien
CNAM-Cedric, 292, rue st Martin
75141 Paris Cedex 03
tel : 40 27 25 83
e_mail : duchien@cnam.fr
http://tulipe.cnam.fr/personne/duchien/poly.html

Laurence Duchien

Introduction

Le réseau est devenu une ressource indispensable (voir vitale) au bon fonctionnement d'une organisation, une entreprise, ...

L'administration du réseau met en oeuvre un ensemble de moyens pour :

- offrir aux utilisateurs un service de qualité,

- permettre l'évolution du système en incluant des nouvelles fonctionnalités

- optimiser les performances des services pour les utilisateurs

- permettre une utilisation maximale des ressources pour un coût minimal.

Laurence Duchien

Administration= partie opérationnelle d'un réseau

Les fonctions d'administration doivent permettre

- **l'extraction** des informations des éléments du réseau au moyen d'outils
  => récolte un grand nombre d'information,

- la **réduction** du volume d'information au moyen de filtres
  => sélection d'information significatives,

- le **stockage** des informations retenues dans une base de données d'administration,

- des **traitements** sur ces informations,

- offrir des **interfaces** (utilisateur d'administration administration, opérateur réseau).

Laurence Duchien

Les standards

Pour être utiliser par une large gamme de produits (systèmes terminaux, ponts, routeurs, équipement de télécommunication quelconque) et dans un environnement multi-constructeurs,

On trouve deux grandes familles de standards :

- SNMP :

- regroupe un ensemble de standards incluant un protocole, une spécification de la structure de la base de données et 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), la spécification de la structure de la base de données et un ensemble d'objets.

Laurence Duchien

Les attendus d'une administration de réseau

Les cinq domaines fonctionnels de l'administration tel que définis dans l'OSI:

- **La gestion des pannes :** permet la détection, la localisation, la réparation de pannes et le retour à une situation normale dans l'environnement.

- **La comptabilité** : permet de connaître les charges des objets gérés, les coûts de communication, ...
Cette évaluation est établie en fonction du volume et de la durée de la transmission. Ces relevés s'effectuent à deux niveaux : Réseau et Application.

- **La gestion des configurations** : permet d'identifier, de paramétrer les différents objets.
Les procédures requises pour gérer une configuration sont la collecte d'information, le contrôle de l'état du système, la sauvegarde de l'état dans un historique

- **L'audit des performances** : permet d'évaluer les performances des ressources du système et leur efficacité. Les performances d'un réseau sont évaluées à partir de quatre paramètres : le temps de réponse, le débit, le taux d'erreur par bit et la disponibilité.

- **La gestion de la sécurité :** une des fonctions de gestion concerne le contrôle et la distribution des informations utilisées pour la sécurité. Un sous-ensemble de la MIB concerne les informations de sécurité (SMIB). Il renferme le cryptage et la liste des droits d'accès.

L'organisation d'une administration

Qui a besoin d'administration et pour quoi faire ?

Il existe différents types de décision d'administration :

- **décisions opérationnelles :** décision à court terme, concernant l'administration au jour le jour et opérations temps réel sur le système

- **décisions tactiques :** décision à moyen terme concernant l'évolution du réseau et l'application des politiques de long terme

- **décisions stratégiques :** décision de long terme concernant les stratégies pour le futur en exprimant les nouveaux besoins et désirs des utilisateurs.

Ces niveaux déterminent différents niveaux d'administration:

- **le contrôle opérationnel réseau** pour les décisions opérationnelles

- **la gestion réseau** pour les décision tactiques

- **l'analyse de réseau** pour les décision tactiques et stratégiques

- **la planification** pour les décisions stratégiques

www.Mcours.com
Site N°1 des Cours et Exercices Email: contact@mcours.com

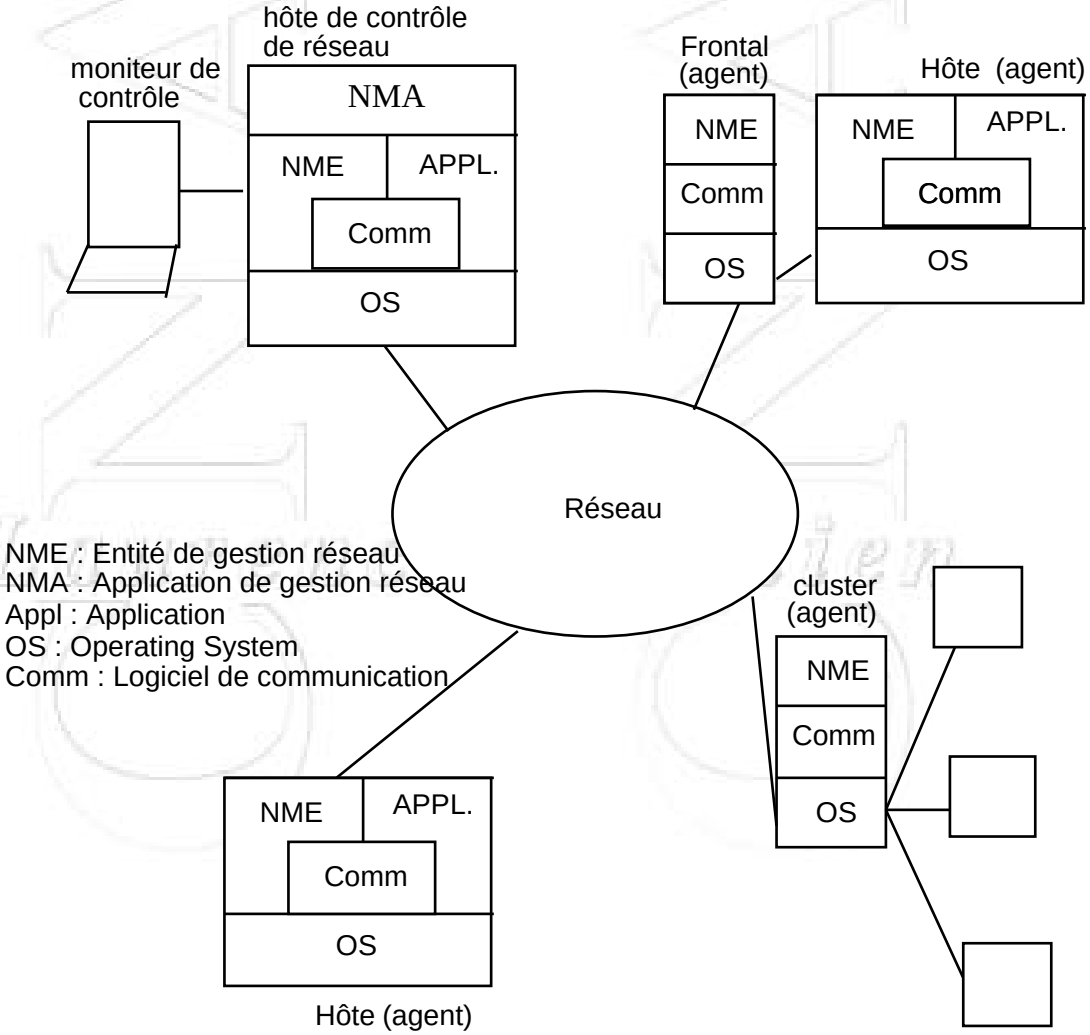
Les systèmes de gestion de réseau

Un système de gestion réseau est une collection d'outils pour contrôler et gérer le réseau qui comprend:

- une interface pour opérateur avec un ensemble de commandes pour exécuter la plupart des tâches d'administration de réseaux.

- un minimum d'équipements supplémentaire intégré au système existant.

La configuration d'un environnement de réseau géré

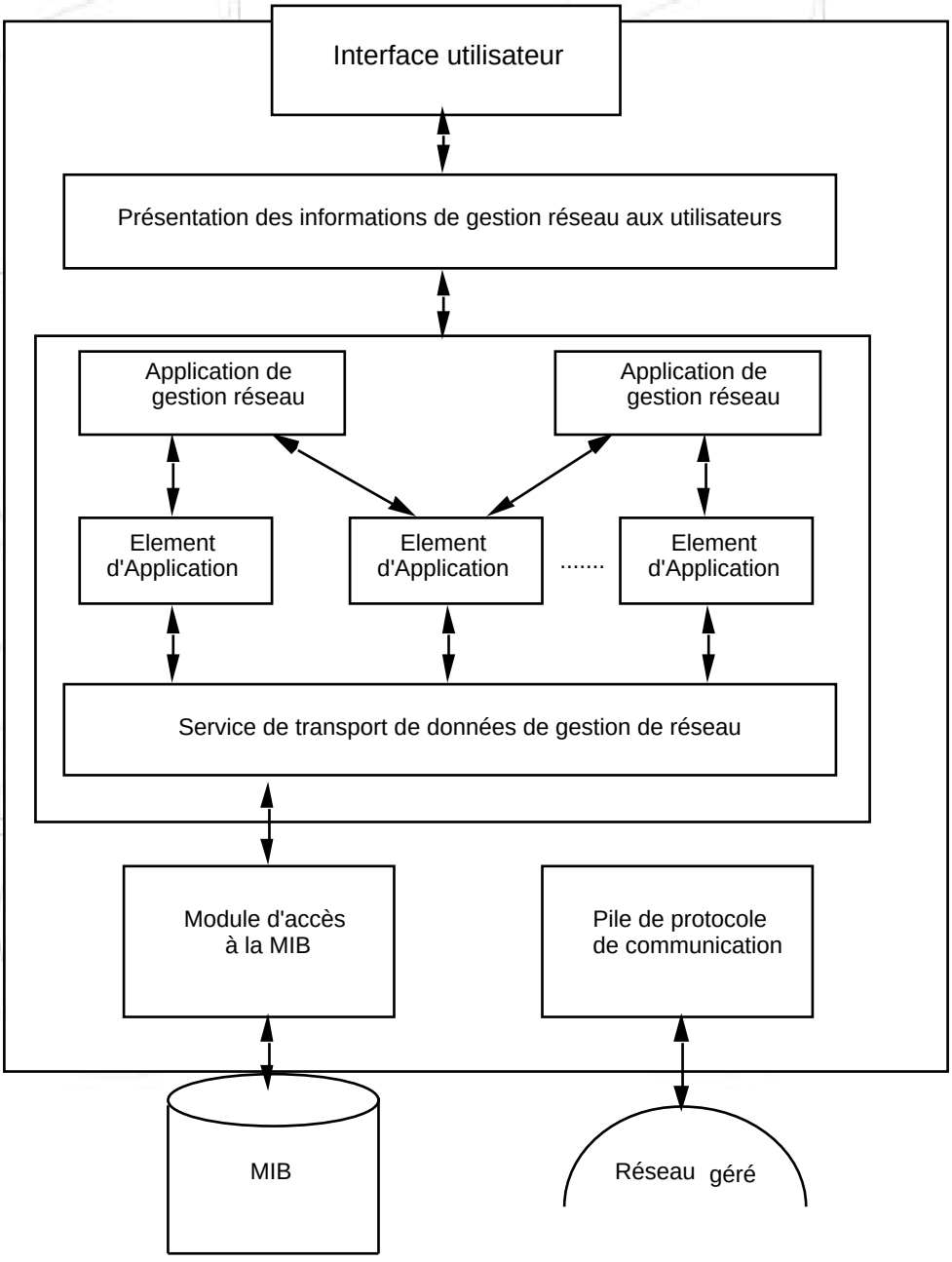


L'architecture d'un logiciel d'administration de réseau

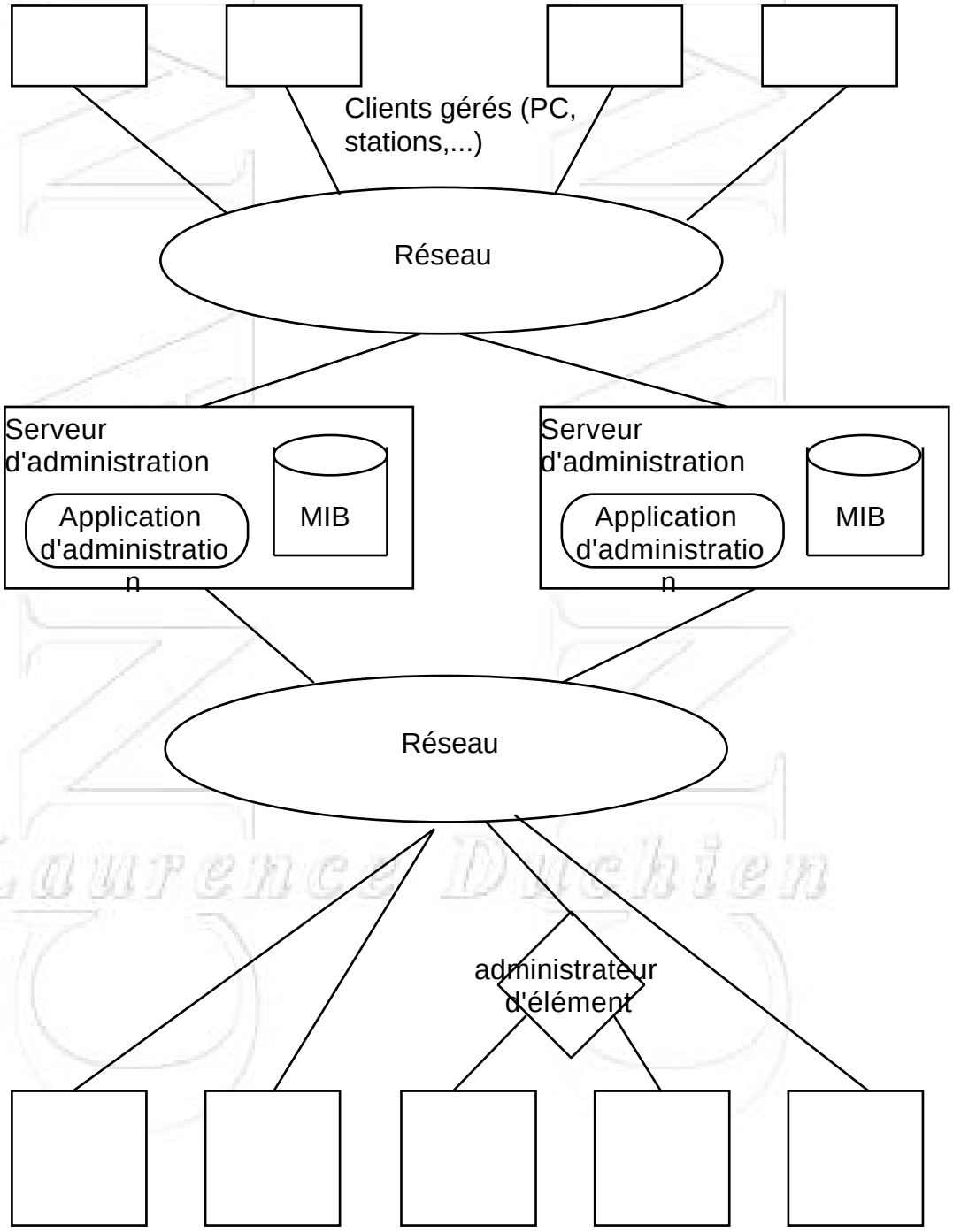
L'architecture de l'application dans un gestionnaire ou dans un agent va varier en fonction des fonctionnalités de la plate-forme.

Une vue générique d'une plate-forme divisé en trois grandes catégories :

- le logiciel utilisateur
- le logiciel de gestion réseau
- le logiciel de communication et de support des données



La gestion distribuée d'un réseau



Ressources Réseau (serveurs, routeurs, hotes) avec des agents d'administration

1. Les concepts de SNMP

- Protocole d'administration de machine supportant TCP/IP

- Conçu en 87-88 par des administrateurs de réseau
- Réponse à un appel d'offre de l'OSF selon le modèle DCE
- RMON MIB1-91, Secure SNMP-92, SNMPv2 - 93.

- 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 différentes machines
  - le modèle fonctionnel pour la surveillance et pour la gestion est extensible
  - indépendant de l'architecture des machines administrées

Laurence Duchien

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

- une **MIB** (Management Information Base) décrivant les informations gérées.

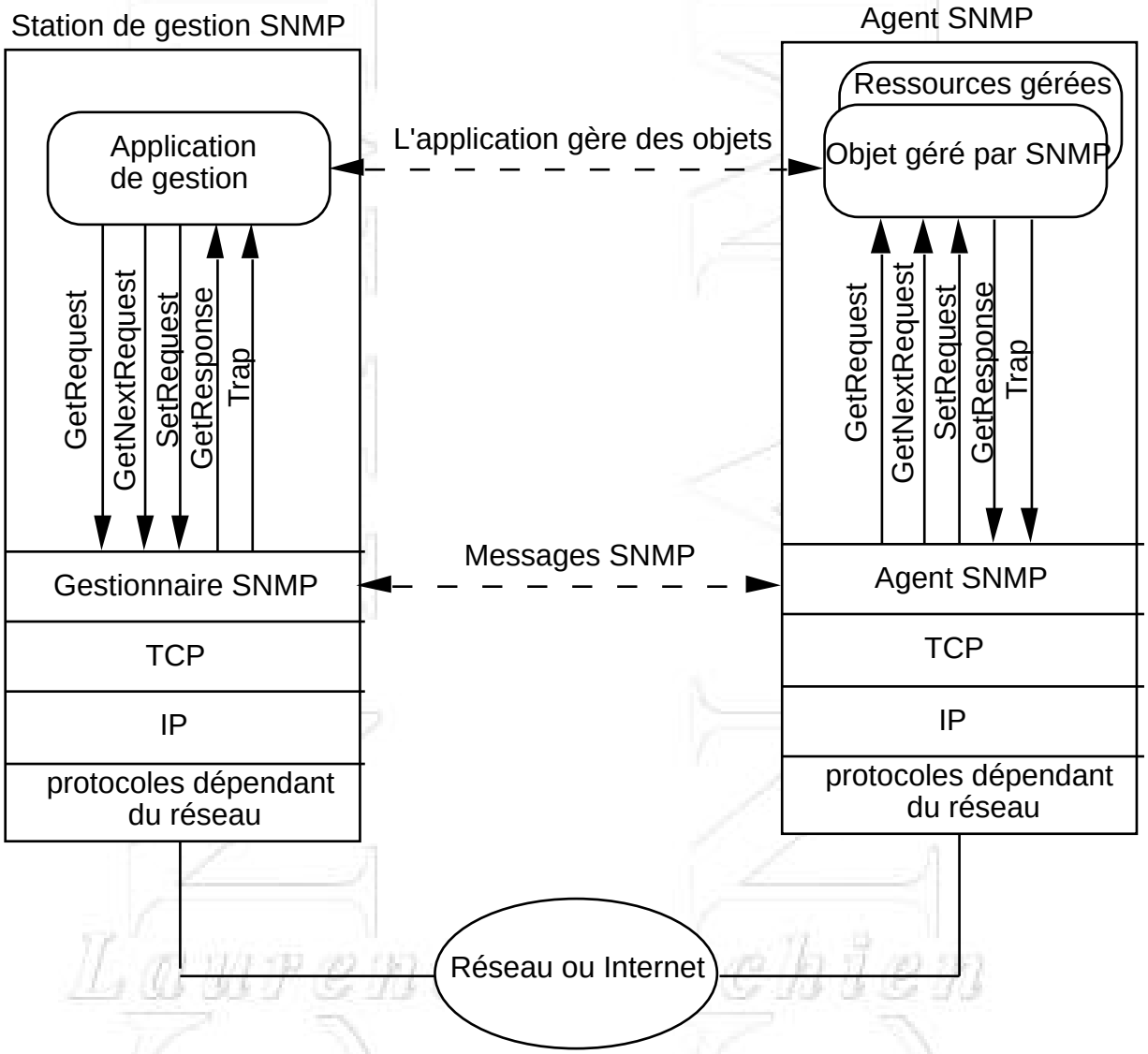
Un **protocole** activé par une API permet la supervision, le contrôle et la modification des paramètres des éléments du réseau.

Les fonctionnalités :

- **get** : permet à la station d'interroger un agent,
- **get_next** : permet la lecture de l'objet suivant d'un agent sans en connaitre le nom
- **set** : permet de modifier les données d'un agent
- **trap** : permet de transmettre une alarme

www.Mcours.com
Site N°1 des Cours et Exercices Email: contact@mcours.com

Le Modèle (2)



Architecture de SNMP

Le Modèle (3)

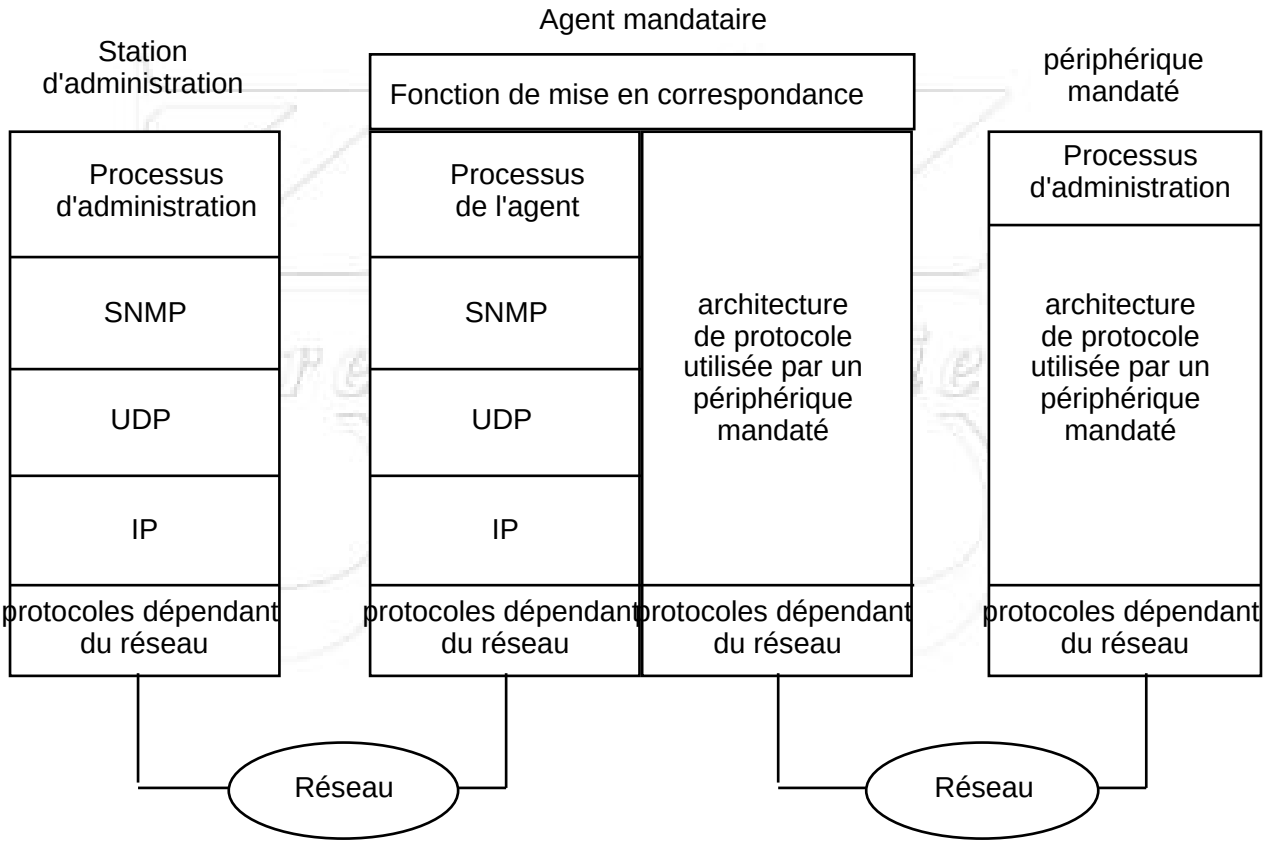
L'utilisation de SNMP suppose que tous les agents et les stations d'administration supportent IP et UDP.

Ceci limite l'administration de certains périphériques qui ne supportent pas la pile TCP/IP.

De plus, 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 périphériques:



2. La MIB (Management Information Base)

=> Modèle de données associé à SNMP:

. SMI (Structure of Management information) - méta modèle
. MIB = liste des variables reconnues par les agents

=> Base de données contenant les informations sur les éléments du réseau à gérer

=> 1 ressource à gérer = 1 objet

- MIB = Collection structurée d'objets
- chaque noeud dans le système doit maintenir une MIB qui reflète l'état des ressources gérées
- une entité d'administration peut accéder aux ressources du noeud en lisant les valeurs de l'objet et en les modifiant.

=> 2 objectifs

- Un schéma commun : SMI (Structure of Management Information)
- Une définition commune des objets et de leur structure

Laurence Duchien

SMI (Structure of de spécification des informations d'administration)

=> donne les règles de définition, d'accès et d'ajout des objets dans la MIB (méta-modèle)

Objectif : encourager la simplicité et l'extension de la MIB

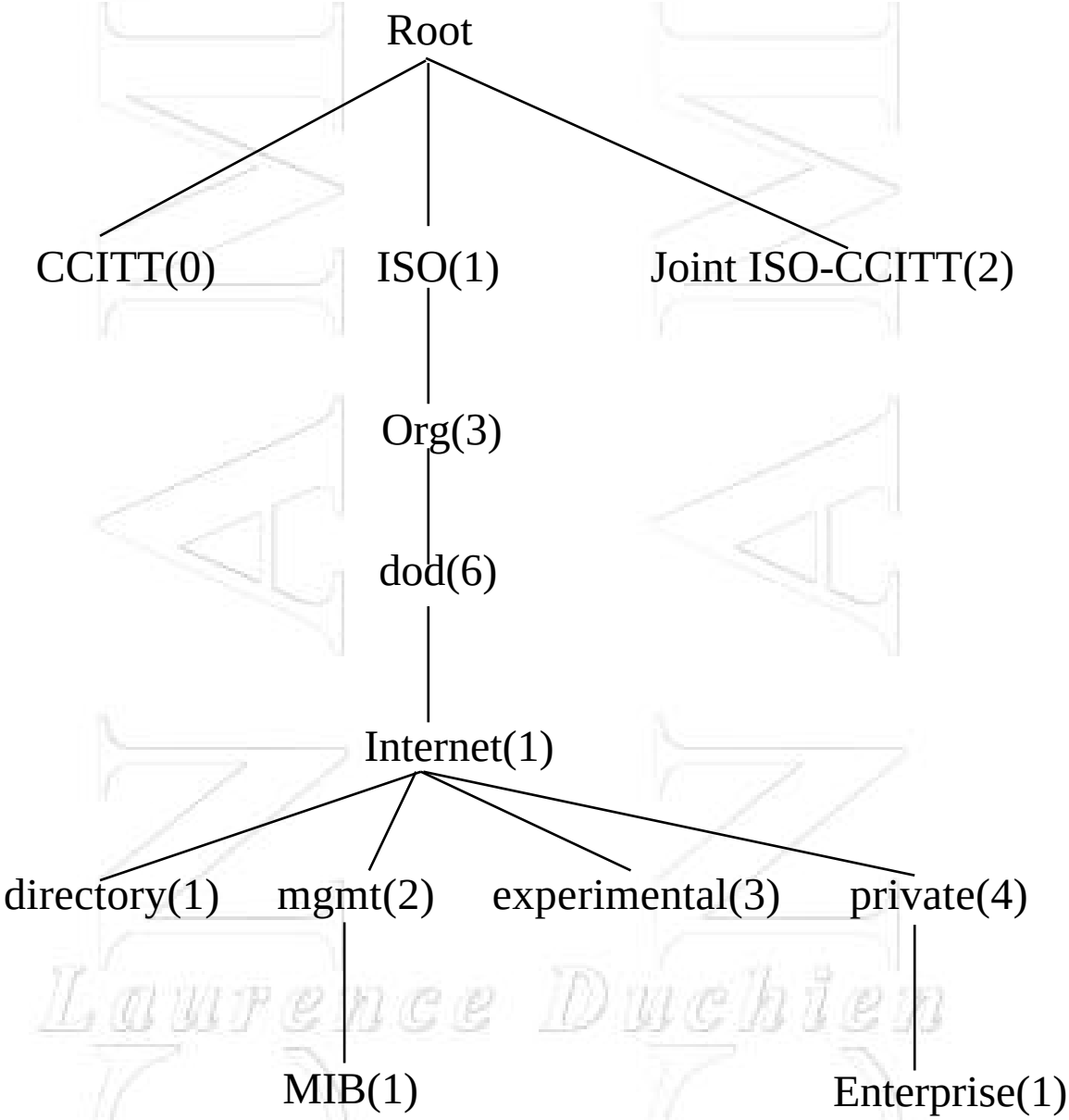
- rendre un objet accessible de la même manière sur chaque entité du réseau
- posséder une représentation identique des objets

- La MIB contient des éléments simples (scalaire et tableaux à deux dimensions de scalaires)
- SNMP ne permet que des interrogations de scalaires

≠ OSI permet des structures et des modes de recherche complexes

Laurence Duchien

La spécification de l'arbre des MIB accessibles.



On utilise la syntaxe ASN.1 pour décrire les données.
Chaque objet est représenté par un "object identifier"

Exemple : Internet Object Identifier ::= {ISO org(3) dod(6) 1}
soit en notation pointée 1.3.6.1 pour le noeud Internet.

Exemple : directory Object Identifier ::= {internet 1}

mgmtObject Identifier ::= {internet 2}

www.Mcours.com
Site N°1 des Cours et Exercices Email: contact@mcours.com

Laurence Duchien

Les types

- Des types simples : INTEGER, OCTET STRING, OBJECT IDENTIFIER, NULL, SEQUENCE, SEQUENCE OF
- Les types dérivés ou applicatifs [RFC 1155]

Exemple de types applicatifs :

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

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

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

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

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

Opaque ::=      -- représente un encodage arbitraire
    [APPLICATION 4]
    IMPLICIT Octet String
```

+ 2 types construits :

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

<table> ::= SEQUENCE OF <list>
```

Les objets décrits utilisent la macro suivante :

```
OBJECT-TYPE MACRO ::=
BEGIN
   TYPE NOTATION ::=
         "SYNTAX" type (TYPE ObjectSyntax)
         "ACCESS"  Access
         "STATUS"  Status
   VALUE NOTATION ::= value (VALUE ObjectName)
   Access ::= "read-only"
                |"read-write"
                |"write-only"
                |"not-accessible"
   Status ::= "mandatory"
                |"optional"
                |"obsolete"
                |"deprecated"
   END
```

Exemple d'objets défini par le SMI du RFC1155

OBJECT
---
atIndex {atEntry 1}
Syntax : INTEGER
Definition : The interface number for the physical address
Access : read-write
Status : mandatory

OBJECT
---
atPhysAddress {atEntry 2}
Syntax : OCTET STRING
Definition : The media-dependant physical address
Access : read-write
Status : mandatory

OBJECT
---
atEntry {atTable 1}
Syntax :
AtEntry::= SEQUENCE {
atIndex INTEGER,
atPhysAddress OCTET STRING,
atNetAddress NetworkAddress,
}
Definition : an entry in the translation table
Access : read-write
Status : mandatory

OBJECT
---
atTable{at 1}
Syntax : SEQUENCE OF AtEntry
Definition : The address translation table
Access : read-write
Status : mandatory

Autres objets intéressants :

atIndex OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS mandatory
::= {atEntry 1}

atPhysAddress OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-write
STATUS mandatory
::= {atEntry 2}

atNetAddress OBJECT-TYPE
SYNTAX NetWorkAddress
ACCESS read-write
STATUS mandatory
::= {atEntry 3}

atEntry OBJECT-TYPE
SYNTAX AtEntry
ACCESS read-write
STATUS mandatory ::= {atTable 1}

Mise jour de la structure

- le nom de la MIB concernée ne change pas mais son no de version évolue (exemple mgmt version-number)

- les anciens objets sont déclarés comme obsolètes s'il y a besoin mais sont préservés

- augmentation de la définition d'un objet en ajoutant de nouveaux objets dans la structure

- ou création complète d'un objet

=> **Évolution** : pas de modification des objets existants dans les nouvelles versions

Laurence Duchien

Les MIBs

Version 2 de la MIB

mib-2 Object Identifier ::= {mgmt 1}

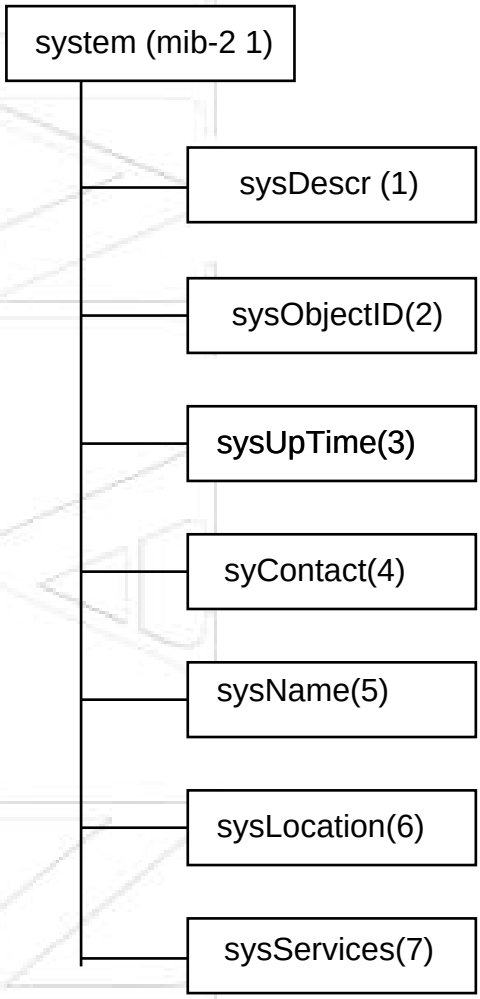
=> groupe de travail "SNMP Working Group"

MIB II : 10 sous ensembles qui sont :

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

Laurence Duchien

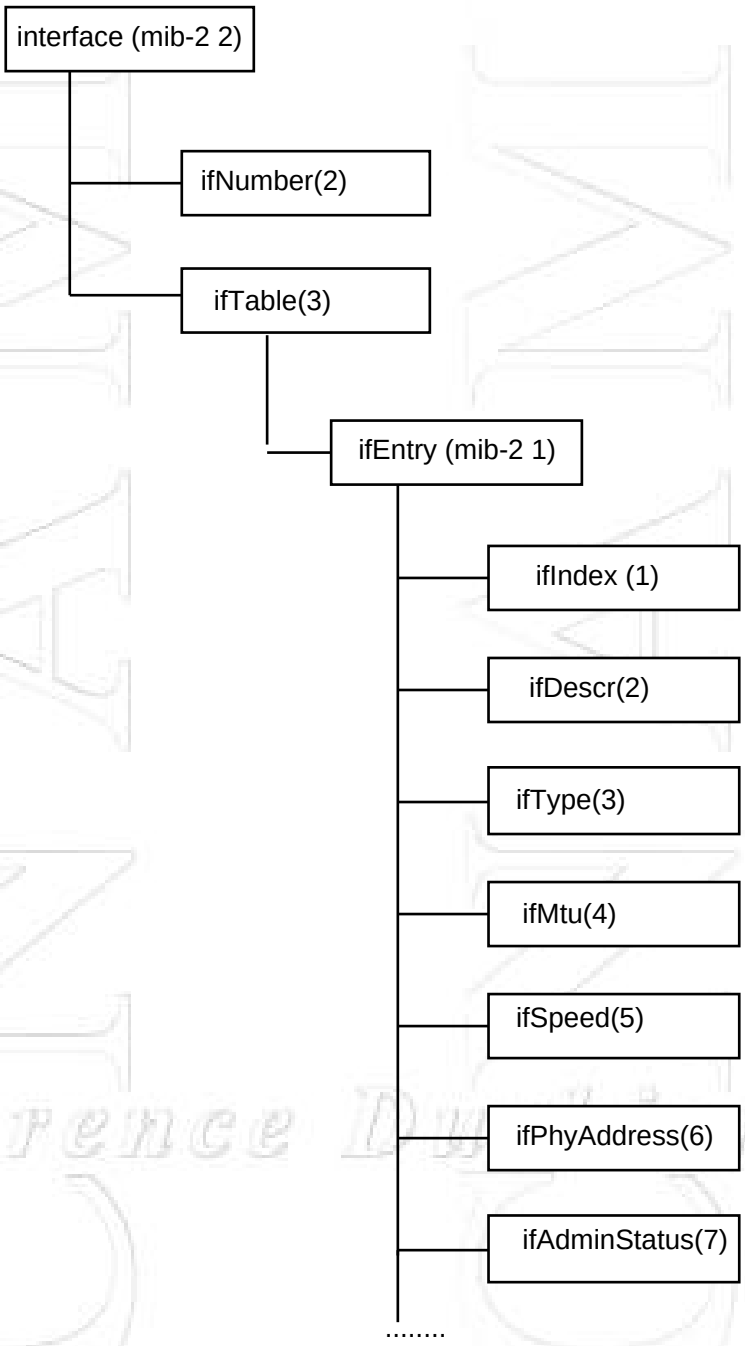
system : correspond au nom de l'agent, no de version, type de la machine, nom du système d'exploitation, type de logiciel réseau en ASCII imprimable



exemple d'interrogation : Accès à des variables d'administration sur une passerelle appletalk-internet

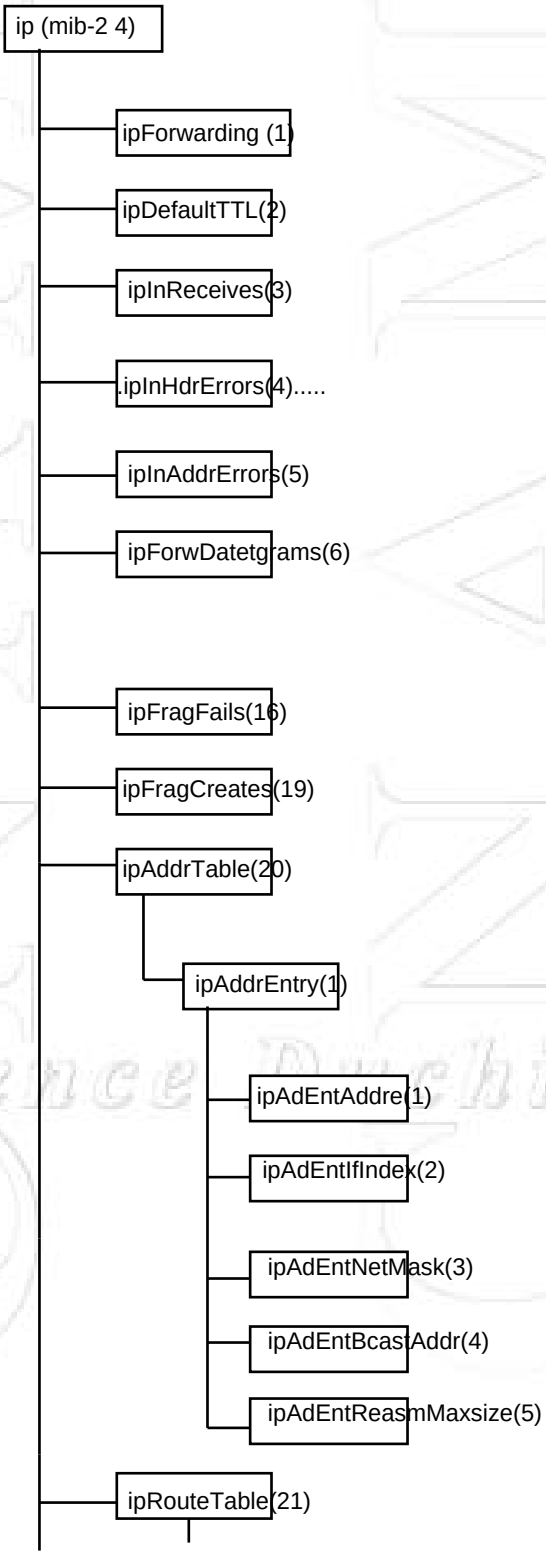
```
% echo "internet[]" | snmp-table verne.cnam.fr |more
sysDescr[0]="Beholder running on Ultrix"
sysObjectID[0]=1.3.6.1.4.1.464.1
sysUpTime[0]=449144
sysContact[0]="Stephane Bortzmeyer"
sysName[0]="verne.cnam.fr"
sysLocation[0]="My office"
sysServices[0]=127
```

interface : ≠ interfaces réseau d'une machine (nombre d'interface, type des interfaces et nom du fabricant, vitesse des interfaces, nombre de paquets entrants, sortants, en erreur,...)



at : conservé pour des raisons de compatibilité avec MIB-I. gère une table de translation entre des adresses réseau de niveau logique (IP) et adresses spécifiques (Ethernet). équivalent à la table ARP.

ip : ≠ paramètres (durée de vie par défaut des paquets IP, nb de paquets reçus ou envoyés, nb de paquets réassemblés avec succès ainsi que le nb de fragments crées, la table de routage si elle existe, le masque sous-réseau, l'adresse physique, etc.
(la partie de la MIB la plus importante)



icmp : 26 compteurs

- pour chaque message icmp, 2 compteurs pour compter les messages reçus et émis
- 4 compteurs pour compter le nombre total de messages icmp reçus, reçus par erreur ou non envoyés,

tcp : rend compte des connexions TCP en cours et des paramètres de type nombre max de connexions simultanées permises, nombre d'ouverture active,...et l'état de chaque connexion (écoute, time-wait,...).

udp :

- 4 compteurs renseignent sur le nombre de datagramme UDP envoyés, reçus, en erreur, ...
- la table gère la liste des applications utilisant UDP ainsi que le pour correspondant

- **egp** : gère le protocole egp (External gateway protocol)(routage des paquets entre routeurs). on a le nbre de paquets entrants, sortants, en erreur, la table des routeurs adjacents, des infos sur les routeurs...

- **transmission** : ne contient que

type Object Identifier ::={transmission number}

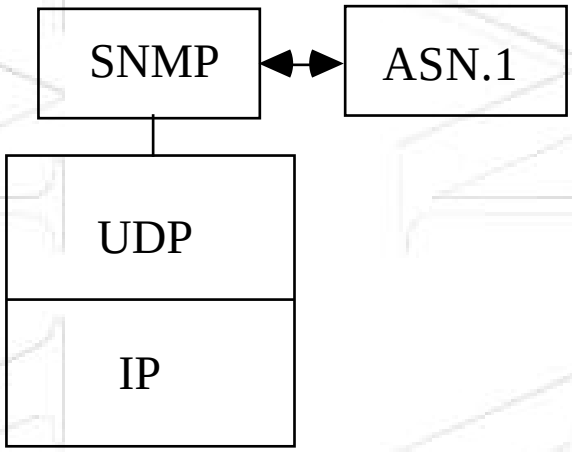
qui permet d'identifier le type de media utilisé pour la transmission.

- **snmp** : requis pour chaque entité mettant en oeuvre le protocole SNMP. contient le nombre de message SNMP entrants et sortants, le nombre de mauvaises versions reçues ou de nom de communauté invalide, la répartition du type de requêtes reçues et envoyées (get, get_next, set et trap)

www.Mcours.com
Site N°1 des Cours et Exercices Email: contact@mcours.com

3. Le Protocole SNMP

L'architecture du réseau utilisé :



Format d'un message SNMP :

- un identificateur de version : no de version SNMP
- un nom de communauté
- une PDU

Les opérations de SNMP :

- get : une station d'administration lit la valeur d'un compteur, d'une variable d'un agent géré
- set : mise à jour d'une variable sur un agent
- trap : un agent envoie une valeur d'une variable de manière implicite vers la station d'administration.

Quelques règles :

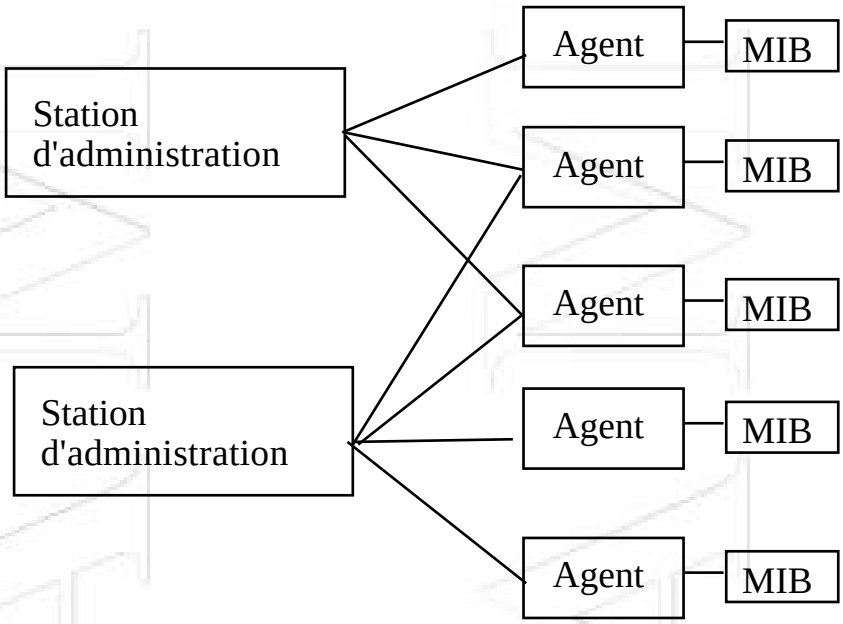
- il n'est pas possible de changer la structure de la MIB par ajout ou retrait d'instances.
- L'accès aux objets est possible uniquement sur les objets-feuilles de l'arbre des identificateurs d'objets.
- Par convention, il est possible d'exécuter des opérations sur des tables à deux dimensions.

=>D'un côté ces restrictions simplifient l'implantation de SNMP

=> De l'autre côté ils limitent la capacité du système d'administration.

Laurence Duchien

Communautés et Nom 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

=> Concerne la sécurité
=> d'où la création de **communauté SNMP**

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 est local à un 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 opérations get et set.
- Une station d'administration garde la liste des noms de communauté donnés par les différents agents.

Laurence Duchien

- **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 mis dans des communautés à part avec utilisation de cryptage et décryptage.

- **La politique d'accès :**

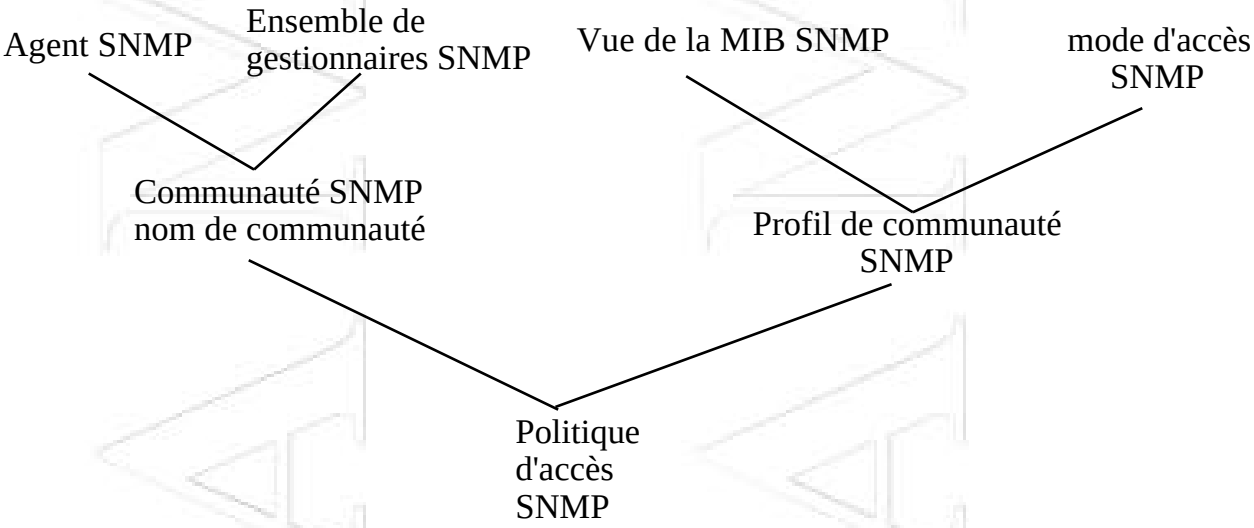
  => Un agent limite l'accès à sa MIB à une sélection de stations d'administration
  => Il peut fournir plusieurs types d'accès en définissant plusieurs communautés
  => Ce contrôle d'accès a deux aspects :
  - une vue de la MIB : un sous-ensemble des objets de la MIB. Différentes vues de la MIB peuvent 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)

Les concepts d'administration



Laurence Duchien

L'identification d'instance

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

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

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

- une technique d'accès par série :
  on utilise l'ordre lexicographique des objets de la structure de la MIB.

- une technique d'accès direct

Laurence Duchien

L'accès direct dans une table

Définition de 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'objet et chaque type a la forme suivante:

<descriptor><syntax>

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

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

tcpConnState	tcpConnLocal Address	tcpConnLocal Port	tcpConnRemA ddress	tcpConnRemP ort	
(1.3.6.1.2.1.6 .13.1.1)	(1.3.6.1.2.1.6 .13.1.2)	(1.3.6.1.2.1.6 .13.1.3)	(1.3.6.1.2.1.6 .13.1.4)	(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	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	

Exemple d'instance d'une table de connexion TCP
Les trois instances de tcpConnState ont le même identificateur : 1.3.6.1.2.1.6.13.1.1

L'index de table

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

La règle de construction de l'identificateur de l'instance d'une instance de colonne d'objet est la suivante :

Soit un objet dont l'identificateur d'objet est y, dans une table avec des objets INDEX i1, i2,..., iN, alors l'identificateur d'instance pour une instance d'objet y dans un rang particulier est

y.(i1).(i2)...(iN)

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

tcpConnState	tcpConnLocal Address	tcpConnLocalP ort	tcpConnRemA ddress	tcpConnRemPo rt
(1.3.6.1.2.1.6. 13.1.1)	(1.3.6.1.2.1.6. 13.1.2)	(1.3.6.1.2.1.6. 13.1.3)	(1.3.6.1.2.1.6. 13.1.4)	(1.3.6.1.2.1.6. 13.1.5)
x.1.10.0.0.99.1 2.9.1.2.3.15	x.2.10.0.0.99.1 2.9.1.2.3.15	x.3.10.0.0.99.1 2.9.1.2.3.15	x.4.10.0.0.99.1 2.9.1.2.3.15	x.5.10.0.0.99.1 2.9.1.2.3.15
x.1.0.0.0.0.99. 0.0	x.2.0.0.0.0.99. 0.0	x.3.0.0.0.0.99. 0.0	x.4.0.0.0.0.99. 0.0	x.5.0.0.0.0.99. 0.0
x.1.10.0.0.99.1 4.89.1.1.42.84	x.2.10.0.0.99.1 4.89.1.1.42.84	x.3.10.0.0.99.1 4.89.1.1.42.84	x.4.10.0.0.99.1 4.89.1.1.42.84	x.5.10.0.0.99.1 4.89.1.1.42.84

Identificateurs d'instance pour les objets de la table précédente

x=1.3.6.1.2.1.6.13.1 = identificateur de l'objet tcpConnEntry qui est l'identificateur de tcpConnTable
i = le dernier sous-identificateur de la colonne (sa position dans la table)
(name) = valeur du nom de l'objet

Toutes les identificateurs d'instances de tcpConnTable ont la forme :
x.i.(tcpConnLocalAddress).(tcpConnLocalPort).(tcpConnRemAddress).(tcpConnRemAddress)

L'ordre lexicographique

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

=> un identificateur 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 noeuds "fils" sont définis en ajoutant un entier à l'identificateur du père et en visitant l'arbre de bas en haut et de gauche à droite.

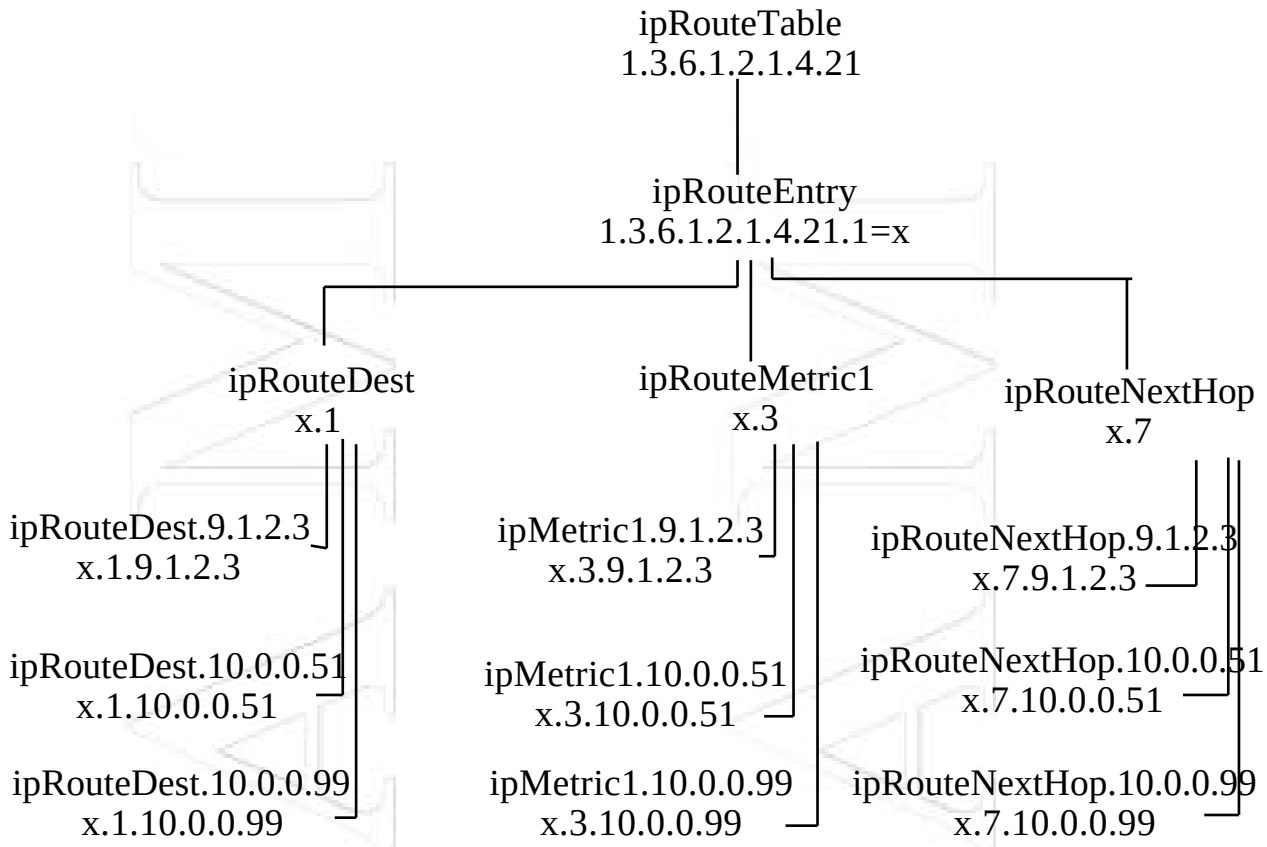
=> Cela permet d'accéder aux différents objets de la MIB sans vraiment en connaître le nom spécifique de l'objet

=> la station d'administration peut donner un identificateur d'objet ou un identificateur d'instance d'objet et demander l'instance de l'objet qui est le suivant dans l'ordre.

Laurence Duchien

www.Mcours.com
Site N°1 des Cours et Exercices Email: contact@mcours.com

exemple :



Object	Identificateur 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

La spécification du protocole

Les formats SNMP :

version	communauté	SNMP PDU

Message SNMP

type PDU	id-request	0	0	variable

GetRequestPDU, GetNextRequestPDU, SetRequestPDU

type PDU	id-request	etat erreur	index erreur	variable

GetResponse PDU

type PDU	enterprise	addr.gen	trap génér	trap specif	time-stamp	variable

Trap PDU

nom1 valeur1	nom2 valeur2		nomn valeurn

la partie variable

Laurence Duchien

Émission d'un message :

- construction de la PDU via ASN.1,

- ajout d'un nom de communauté, adresse source, adresse destination, numéro de version,

- envoi de datagramme contenant l'objet ASN.1 spécifié

Réception d'un message :

- réception du message

- analyse du message

- message ASN.1 correct ?=> non => fin

- version OK ? => non => fin
- Examen de la communauté et des données contenues dans le message
- OK ?
    oui :
    - examen de la PDU reçue (analyse syntaxique)
    - OK ?
        oui :
        - construction d'une nouvelle PDU correspondant à la requête reçue.
        - construction du message et envoi
    non :
    Signale l'erreur d'authentification
    Archive l'erreur et trap éventuel

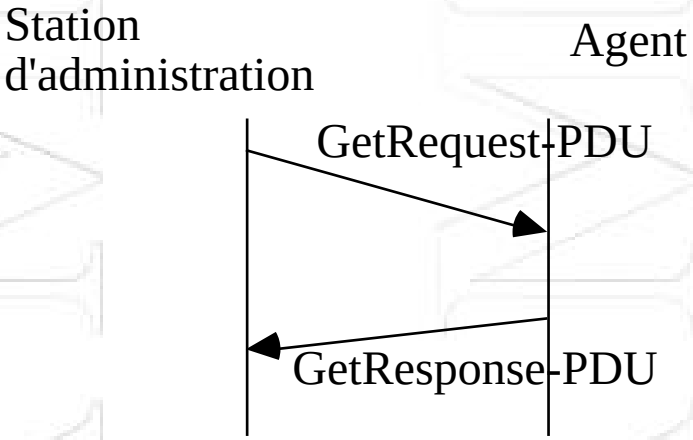
```
RFC1157-SNMP DEFINITIONS ::= BEGIN
IMPORTS
        ObjectName, ObjectSyntax, NetworkAddress, IpAddress, TimeTicks
                FROM RFC1155-SMI;

Message ::= SEQUENCE {
                version INTEGER  {version-1 (0)},-- Version 1 for this RFC
                community OCTET STRING,          -- Community name
                data ANY                         -- e.g. PDUs
                        }
-- 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
                }
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-- Only one type of network adresses
                                          -- IP 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, -- Elapse time since the last reinitialization of the enti
                variable-binding VarBindList  -- "Interesting" information
                }

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

Echange sur le réseau au niveau du service

- Get



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<-- Valeur fausse dans le message reçu,

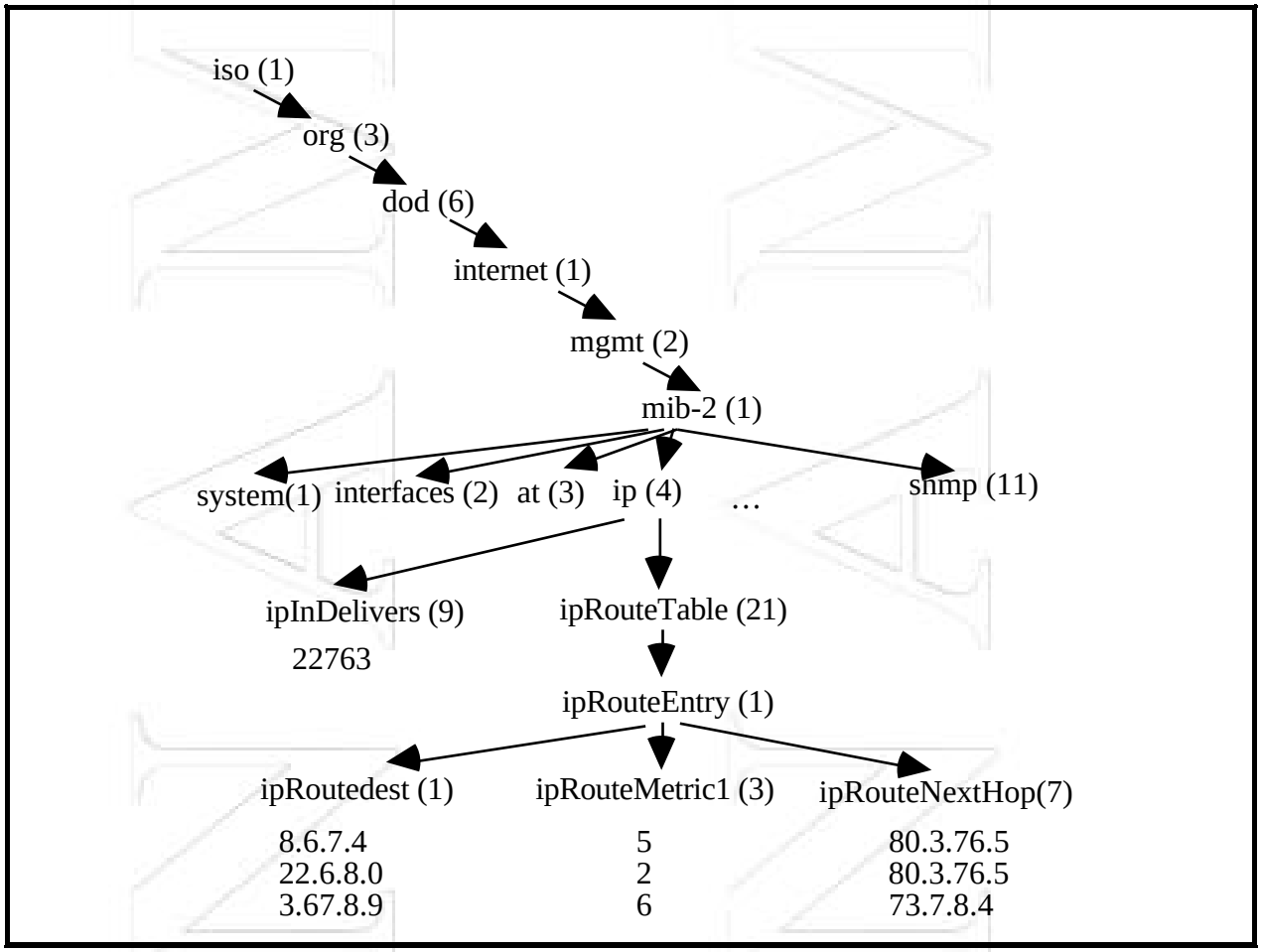
- Si GetResponse-PDU > Limitation locale alors envoi de GetResponse-PDU avec ErrorStatus<--too big et ErrorIndex<-0

- 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

www.Mcours.com
Site N°1 des Cours et Exercices Email: contact@mcours.com

Exemple



La désignation absolue de l'objet ipInDelivers est:

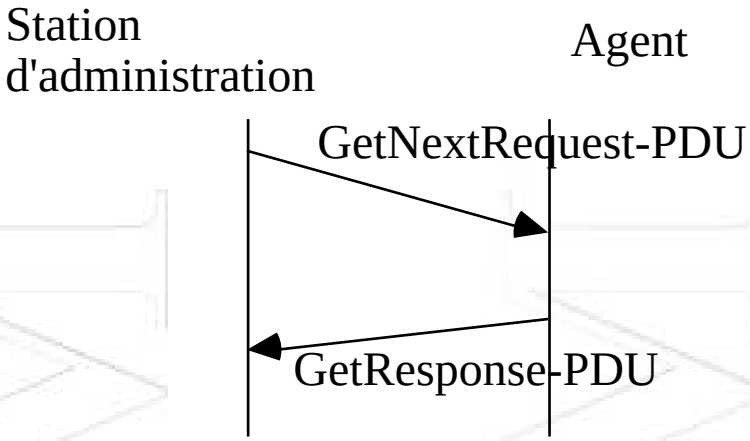
iso.org.dod.internet.mgmt.mib-2.ip.ipInDelivers soit 1.3.6.1.2.1.4.9

La valeur de la variable ipInDelivers est obtenue par la commande:

GetRequest (1.3.6.1.2.1.4.9)

qui produit la réponse GetResponse ((ipInDelivers = 22763))

- Get-Next



- 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 <--too big 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.22.6.8.0="22.6.8.0"),
(ipRouteMetric1.22.6.8.0="2")
(ipRouteNextHop 22.6.8.0="80.3.76.5"))

On continue la consultation de la table par :

GetNextRequest (ipRoutedest.22.6.8.0, ipRouteMetric1.22.6.8. ipRouteNextHop.22.6.8.0)

On obtient la réponse suivante :

GetResponse ((ipRoutedest.3.67.8.9="3.67.8.9"),
(ipRouteMetric1.3.67.8.9="6")
(ipRouteNextHop.3.67.8.9="73.7.8.9"))

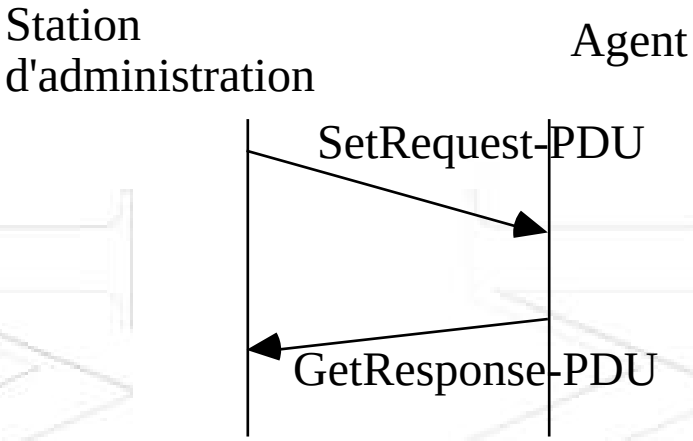
Enfin, par la dernière requête de la consultation de la table :

GetNextRequest (ipRoutedest.3.67.8.9, ipRouteMetric1.3.67.8. ipRouteNextHop.3.67.8.9)

on obtient la réponse suivante :

GetResponse ((ipRoutedest.8.6.7.4=".8.6.7.4"),
(ipRouteMetric1..8.6.7.4="5")
(ipRouteNextHop..8.6.7.4="80.3.76.5"))

- Set




```
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

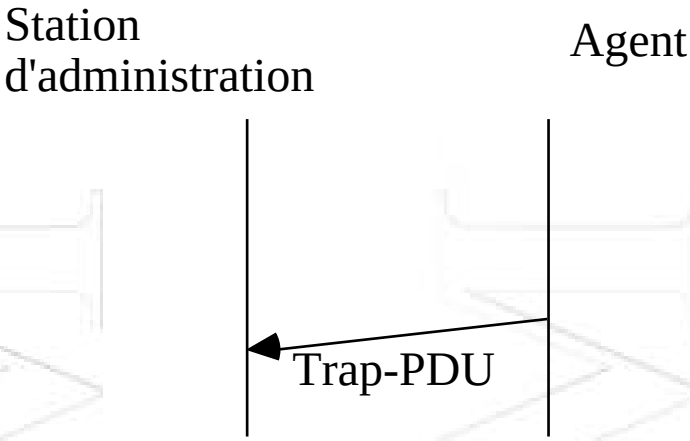
- Si, pour chaque objet du champ Variable-Bindings, l'objet n'est pas accessible pour l'opération demandée alors la PDU GetResponse-PDU (de forme identique) est envoyée avec
  ErrorStatus<--NoSuchName et ErrorIndex <-- pointeur sur la variable en erreur

- 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 <-- veriable 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 : Error_status<-- 0 et ErrorIndex<--0

• Trap



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,
   time-stamp         TimeTicks     -- temps depuis reboot
   variable-bindings      VarBindList
   }
```

www.Mcours.com
Site N°1 des Cours et Exercices Email: contact@mcours.com

=> 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 l'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.

Laurence Duchien

Conclusion provisoire

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

Avantages :

- simple

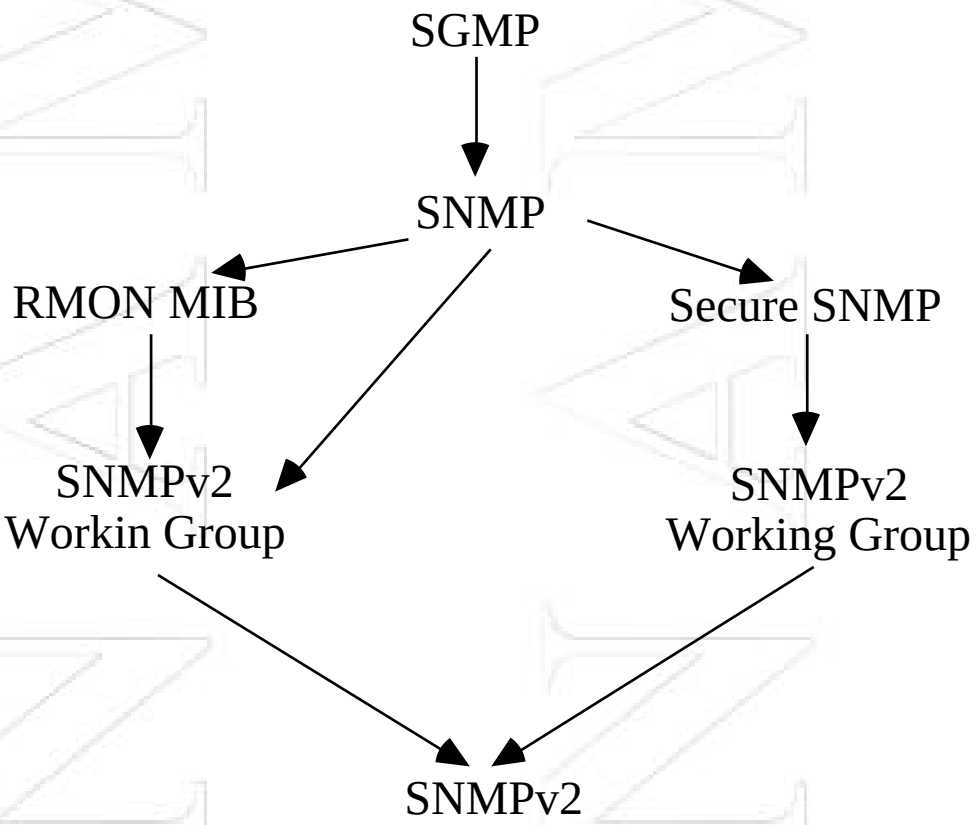
Faiblesses :

- 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

Laurence Duchien

4. SNMP v2

Historique de SNMPv2



SGMP : Simple Gateway-Monitoring Protocol
RMON : Remote Network Monitoring
CMOT : CMIP au dessus de TCP/IP

Ce qui change par rapport à SNMP :

- SNMPv2 est capable de gérer de manière distribuée un réseau: opérations entre stations d'administration
- sécurité renforcée
- nouvelles opérations

SMI : Structure de l'information d'administration

1. Définition des objets

Quelques changements mineurs :

- redéfiniton de certains types

Counter devient Counter32 ou Counter64

- La clause ACCESS devient MAX-ACCESS:

- permet d'indiquer que c'est un niveau maximum d'accès

- Quatre possibilités : pas d'accès, lecteur seule, lecture-écriture, lecture-création.

- Introduction de nouveaux mots-clés (Unit)

- La clause STATUS n'inclut plus les catégories optionnel et obligatoire

2. Les tables

Les droits de création, de destruction et d'accès :

- Les tables protégées :

Elles ne peuvent être ni crées ni détruites par une station de gestion. Ces tables sont contrôlées par l'agent.

Le maximum d'un type d'accès alloué pour cette table et Read-write.

Ces tables sont pratiques lorsqu'elles correspondent à un nombre fixe d'attributs comme le nombre d'interfaces physiques par exemple.

- Les tables non protégées :

Certaines tables peuvent être crées ou détruites. Elles peuvent être initialisées avec un nombre de rangs égal à 0.

```
snmpORTable OBJECT_TYPE
        SYNTAX          SEQUENCE OF SnmpOREntry
        MAX_ACCESS      not-accessible
        STATUS          current
        DESCRIPTION
                "the conceptual table listing the dynamically-configurable objet
resources in a SNMPv2 entity acting in an agent role. SNMPv2 entities which do not
support dynamically-configurable objetc resources will never have any instances of
the columnar objetc in this table"
        ::= {snmpOR 2}

snmpOREntry OBJECT-TYPE
        SYNTAX    SnmpOREntry
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION
                "An entry (conceptual row) in the snmpORTable"
        INDEX       {snmpORIndex}
        ::= {snmpORTable 1}
```

Création et destruction d'un rang dans un tableau

- La méthode createAndWait :

- La station de gestion commence à ordonner à l'agent de créer un nouveau rand dans la table avec une instance d'identification ("index value")

- Si l'agent accepte, il crée le rang et assigne des valeurs par défaut aux objets du rang,

- Si tous les objets de type read-write possèdent des valeurs par défaut, le rang est placé dans l'état notInservice

- si il existe des objet de type read-write qui n'ont pas des valeurs par défaut, le rang est placé dans l'état notready

- Le gestionnaire envoie une requête de type "Get" pour déterminer l'état de chaque objet dont le type d'accès est read-create dans le rang.

- L'agent envoie chaque valeur de chaque objet. Si l'objet ne possède pas de valeur, il envoie NoSuchInstance.

- La station d'administration doit alors envoyer un SetRequest pour assigner des valeurs aux objets. Elle peut ensuite envoyer une requête de type "Set" pour activer les objets non actifs.

Création et destruction d'un rang dans un tableau

- La méthode createAndGo :

Plus simple, mais plus restrictive car elle permet de travailler sur des tables dont les objets sont contenus dans une seule PDU.

De plus la station d'administration ne connaît pas les valeurs par défaut attribuées aux différentes colonnes.

La station d'administration envoie un Get-PDU pour déterminer les objets de type "read-create" possédant le type noSuchInstance.

Elle envoie ensuite un Set-PDU pour créer un nouveau rang et assigner des valeurs aux objets ayant le type d'accès "read-create" dans ce rang.

Si le Set réussit, l'agent active ces objets.

Laurence Duchien

Exemple de création de ligne d'une table

La commande "ping" qui fournit un echo distant
Les messages utilisés dans ICMP sont echo et echo_reply

=> La station d'administration peut mettre à jour un rang pour dire à l'agent de faire un ping sur un autre système à intervalle régulier:

L'agent possède initialement la table :

Index	IpAddress	Delai	Remanient	Total	Received	Rtt	Status
1	128.2.13.21	1000	0	10	9	3	active

La station d'administration souhaite ajouter un nouveau rang en utilisant la méthode createAndWait.

Elle détermine que le prochain index est 2 et souhaite que le nouveau rang ait les valeurs suivantes :

Index	IpAddress	Delai	Remanient	Total	Received	Rtt	Status
1	128.2.13.99	1000	20	20	0		active

Pour ajouter cette dernière entrée, la station de gestion commence par envoyer une commande Set à l'agent :

SetRequest(pingStatus.2=createAndWait)

En cas d'acceptation, l'agent répond :

Response(pingStatus.2,=notInService)

La station de gestion envoie un Get pour lire le nouveau rang :

GetRequest(pingIpAdress.2, pingDelay.2, ping.Remaining.2, pingStatus.2, pingSize.2)

L'agent répond :

Response ((pingIpAdress.2=noSuchInstance), (pingDelay.2=1000), (ping.Remaining.2=5), (pingStatus.2=UnderModification), (pingSize.2=noSuchObject))

Certaines valeurs ont été affectées par défaut. Il faut alors compléter....par un SetRequest((pingIpAddress.2=128.2.13.99),....)

Le protocole

Quelques modifications :

PDU Type	Request_id	0	0	Partie variable

GetRequest, GetNextRequest, SetRequest, Trap, InformRequet PDU

PDU Type	Request-id	error_status	error_index	Partie Variable

Response PDU

PDU Type	Request-id	non repeaters	max repetitions	partie variable

GetBulkRequest

Laurence Duchien

- **GetBulkRequest :**

  But : minimiser le nombre d'échange à travers le réseau

  Permet à une station d'administration de solliciter de la part d'un agent une réponse contenant le maximum d'information pouvant être contenu dans un message (limitation par la taille du message)

  Possibilité de spécifier des successeurs multiples lexicographiques.

Fonctionnement :

GetBulkRequest inclut une liste de (N+R) variables dans le champ "partie variable".

Pour les N noms, la récupération est faite comme dans GetNextRequest

Pour chaque variable de la liste, la variable suivante dans l'ordre lexicographique ainsi que sa valeur sont retournées.

Si il n'y a pas de suivant lexicographique, la variable nommée et la valeur "endOfMibView sont retournées.

les champs "non-repeaters" et "max-repetition" indiquent le nombre de variables contenu dans la liste "partie variable" et le nombre de successeurs dans être retournées pour les variables restantes.

www.Mcours.com
Site N°1 des Cours et Exercices Email: contact@mcours.com

Soient

L: nombre de variables dans partie variable
N : nombre de variables dans partie variable avec demande(variable)=un seul successeur
R : nombre de variables, succédant les N premières variables pour lesquelles de multiples successeurs lexicographiques sont demandées
M : nombre de successeurs lexicographiques sollicités pour chacune des dernières R variables

N=MAX(MIN(non-repeaters,L),0)
M=MAX(max_repeatetions,0)
R=L-N

Si N> 0 , alors les N premières variables son traitées comme pour un GetNextRequest

Si R>0 et M>0 alors pour chacun des R dernières variables, ces M successeurs lexicographiques sont renseignées.

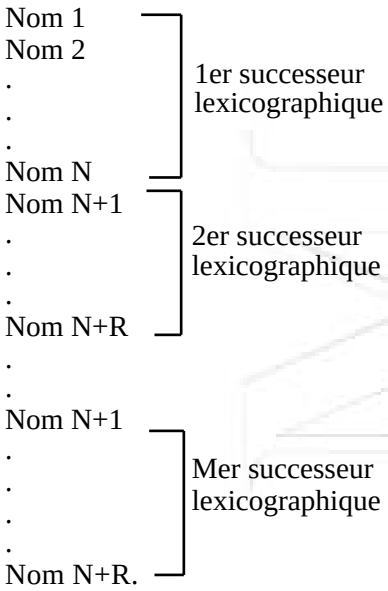
Pour chaque variable, cela signifie :

- obtenir la valeur du successeur lexicographique de la variable considérée

- obtenir la valeur du successeur lexicographique de l'instance objet obtenu à l'étape précédente

- ainsi de suite, jusqu'à ce que M instances objets soient extraites

Exemple



Ordonnancement des variables-bindings dans la réponse GetBulkrequest

Soit la table suivante :

Interface-Number	Network-Address	Physical-Address	Type
1	10.0.0.51	00.00.10.01.23.45	static
1	9.2.3.4	00.00.10.54.32.10	dynamic
2	10.0.0.15	00.00.10.98.76.54	dynamic

La station de gestion envoie:

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

L'agent répond :

Response
((sysUpTime.0="123456"),(ipNetToMediaPhysAddress.1.9.2.3.4="000010543210"),(ipNetToMediaType.1.9.2.3.4="dynamic",(ipNetToMediaPhysAddress.1.10.0.0.51="00001012345"),(ipNetToMediaType.1.10.0.0.51="static"))

La station de gestion envoie:

GetBulkRequest [non-repeaters=1, max-repetitions=2]
(sysUpTime, ipNetToMediaPhysAddress.1.10.0.0.51,
ipNetToMediaType.1.10.0.0.51)

L'agent répond :

Response
((sysUpTime.0="123466"),(ipNetToMediaPhysAddress.2.10.0.0.51="0000109887654"),(ipNetToMediaType.2.10.0.0.51="dynamic",(ipNetToMediaNetAddress.1.9.2.3.4="9.2.3.4"),(ipRoutingDiscards.0="2"))

Possibilité de station d'administration à station d'administration

- **InformRequest-PDU**

But : permet à une station de gestion d'envoyer des informations vers une station d'administration qui centralise des informations contenues dans la MIB "manager-to-manager"

Le message a le même format que Get, Set,...

La MIB permet de spécifier des paramètres tels que :

- l'intervalle de temps devant séparer 2 "InformRequest_PDU"

- le nombre d'"InformRequest-PDU" voulues

- description de l'événement à rapporter

- la date de l'événement

- …

Cette PDU étend le mécanisme de Trap de SNMP1.

La MIB

2 nouvelles MIB sont définies :

- SNMPv2 Management Information Base

- Manager-To-Manager

- **SNMPv2 Management Information Base :** permet de décrire le comportement des agents SNMP du réseau. composé de 5 groupes :

  1. SNMPv2 Statistics group : contient des informations relatives au protocole SNMPv2 comme le nombre total de paquets reçus au niveau transport, le nombre de paquets mal codés, le nombre de requêtes PDU GetRequest, GetNextRequest,....

  2. SNMPv1 Statistics group : informations relatives au protocole SNMPv1 . Par exemple, le nombre de messages ayant un mauvais nom de communauté, nombre de message demandant une opération non autorisée,...

  3. Object resource group : utilisé par l'agent SNMPv2 pour décrire les objets susceptibles d'être configurés par une station d'administration. on y trouve le nom de l'objet, sa description,...

  4. Traps group : gère les "traps" générés par un agent

  5. Set group : se compose d'un seul objet qui permet de résoudre 2 problèmes : la sérialisation des opérations de type Set émises par une station de gestion et la gestion de la concurrence d'accès par de multiples stations de gestion

- **Manager-To-Manager MIB**

  - Alarm group : permet de spécifier les paramètres de configuration des alarmes : intervalles entre les alarmes, instances ou objet ayant provoqué l'alarme,...

  - Event group : permet de renseigner une station de gestion sur un ensemble d'événements choisis, sur l'instant où ils se produisent,...

Laurence Duchien

La compatibilité entre SNMP et SNMPv2

La coexistence des 2 versions est facilitée par le fait que SNMPv2 est un sur ensemble de SNMPv1.

=> La manière la plus simple de gérer le passage de V1 à V2 est de passer la station d'administration à la version 2, qui peut ainsi gérer à la fois des stations en V2 (en cas de gestion répartie) et des agents en V1 et V2.

Il est nécessaire des équivalences dans :

- la manière dont sont gérées les informations (SMI)
- le protocole

- **Le SMI :**

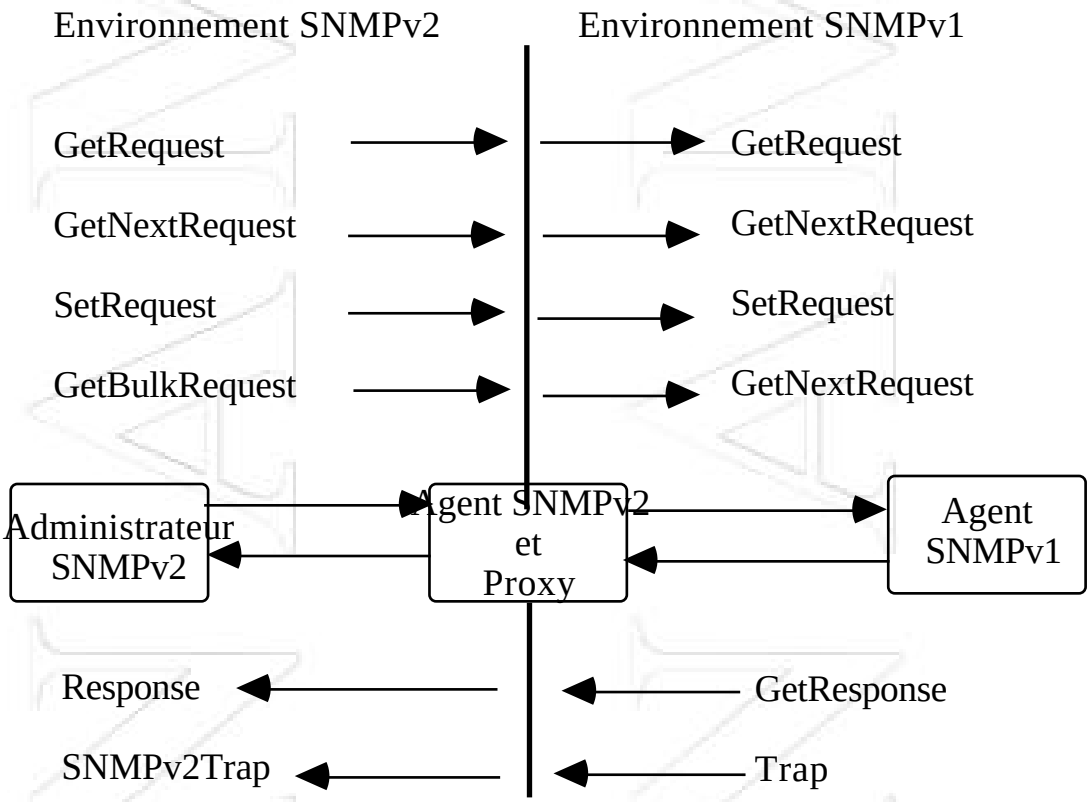
  Pour assurer la compatibilité, les correspondances suivantes sont nécessaires :

  - INTEGER défini sans restriction devient Integer32
  - Counter devient Counter32
  - Gauge devient Gauge32
  - ACCESS devient MAX-ACCESS
  - ….

- **Le protocole :**

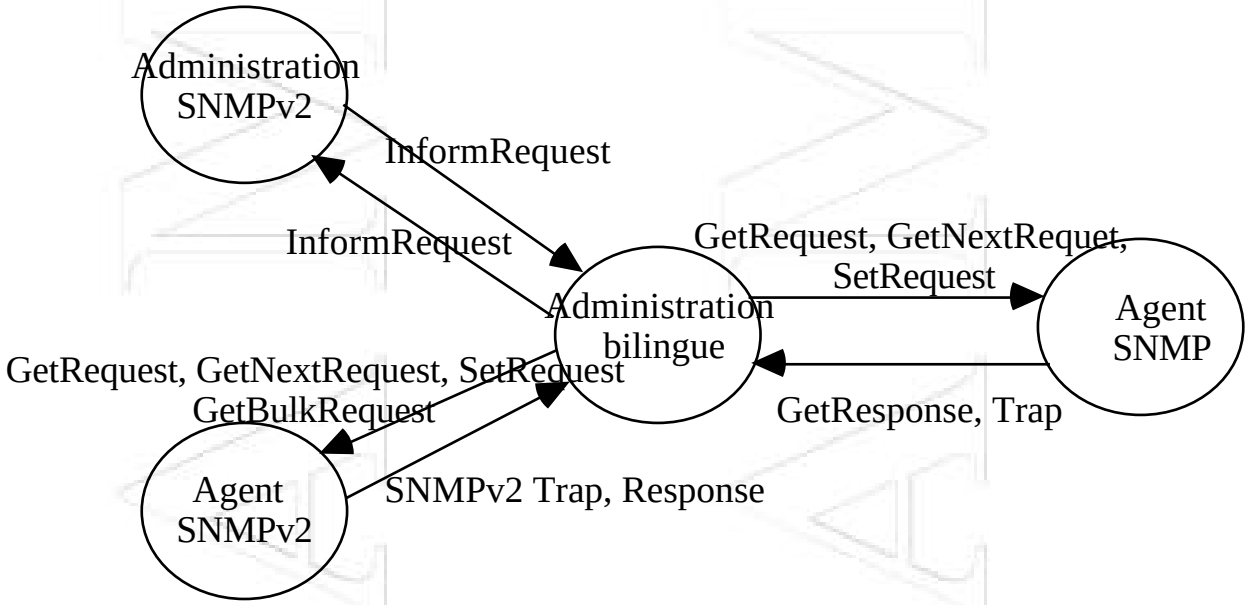
  SNMPv2 gère des PDU supplémentaires

  On prévoit l'utilisation d'un agent proxy qui assure la traduction des PDU entre les 2 versions.



- noSuchName, readOnly et badValue ne sont pas utilisables par un agent en version 2 mais interprétable par une station d'administration
- l'agent proxy assure la gestion des messages ne pouvant pas être contenues dans une seule PDU.

- Possibilité de faire cohabiter 2 versions SNMP- le gestionnaire utilise au choix le protocole 1 ou 2 :



Laurence Duchien

La sécurité dans SNMP 2

Dans la version 1 => utilisation de la notion de communauté pour définir la visibilité accordée à une station par un agent.

Dans la version 2 => **notion de groupe** :

```
SnmpParty ::= SEQUENCE {
        partyIdentify OBJECT IDENTIFIER, -- identifiant du groupe
        partyDomain OBJECT IDENTIFIER, -- type de couche transport
        partyAddress OCTET STRING, -- adresse de niveau transport
        partyMaxMessageSize INTEGER, -- taille max des messages
        partyAuthProtocol OBJECT IDENTIFIER, -- nomme le protocole
d'authentification utilisé
        partyAuthClock INTEGER, -- période valide pour le groupe
        partyAuthPrivate OCTET STRING, -- clé privée d'authentification
        partyAuthPublic OCTET STRING, -- clé publique d'authentification
        partyAuthLifeTime INTEGER, -- durée de vie des messages
        partyPrivProtocol OBJECT IDENTIFIER, --identification du protocole
utilisé (PGP par exemple)
        partyPrivPrivate OCTET STRING, -- clé privée
        partyPrivPublic  OCTET STRING, -- clé publique
}
```

Un élément actif sur le réseau agit de la manière suivante :

- exécute uniquement les opérations permises par le groupe,
- maintient une petite base de données qui contient tous les groupes reconnus par l'entité, les opérations pouvant s'effectuer directement et celles qui font appel à un agent de proxy, les ressources accessibles (notion de contexte)

=> Chaque entité maintient donc l'ensemble des données définissant le concept de "politique d'accès"

Le Contexte :

se définit comme étant l'ensemble des ressources accessibles (objets) par une entité SNMPv2

Il existe deux types de contexte :

- **local :**

Le gestionnaire accède directement aux informations dans l'agent

Le gestionnaire envoie une opération de gestion qui contient :

- un groupe source (srcParty) (le gestionnaire)
- un groupe destination (dstParty) (agent)
- un contexte
- PDU (Get, Set,...)

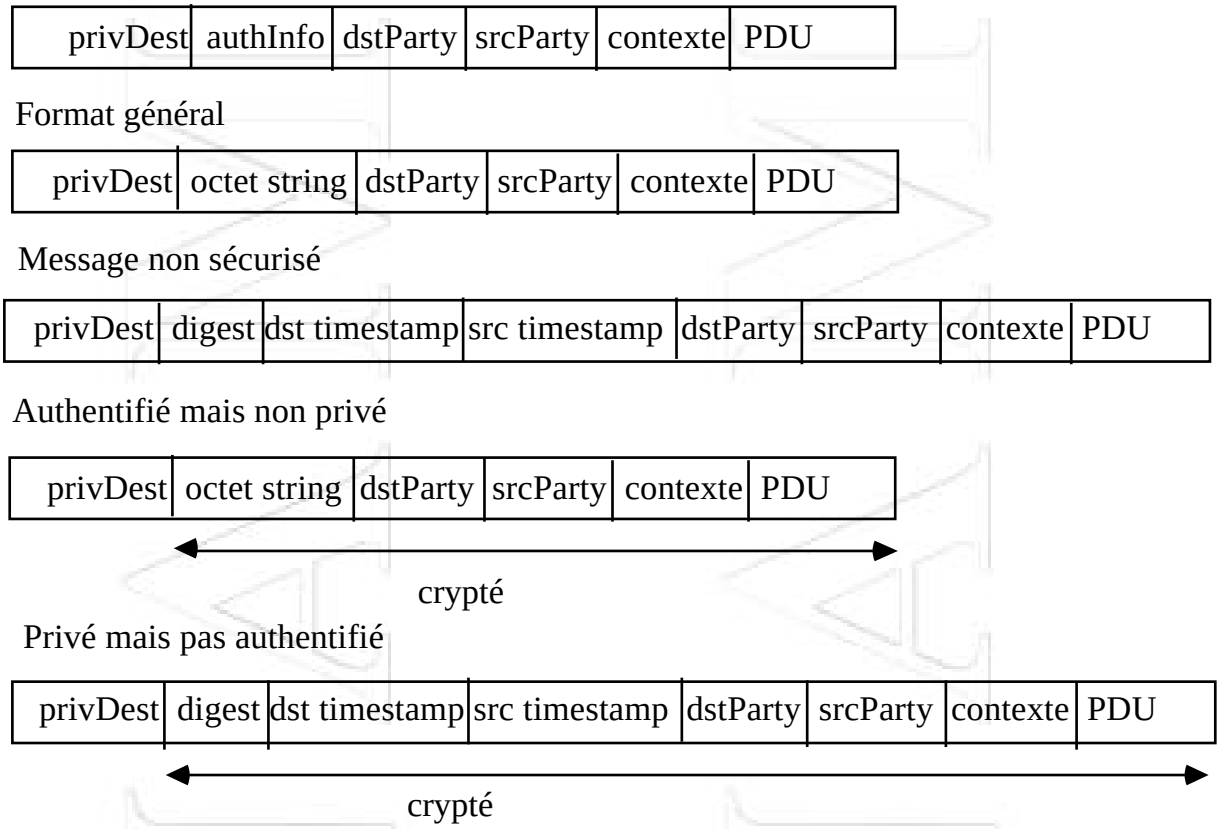
l'agent consulte l'entité ACL (Access Control List) et détermine si les opérations sont permises.

- **distant :**

L'agent intervient comme médiateur entre une station d'administration et une entité distante.

L'agent agit comme un proxy qui gère les droits d'accès.

Format des messages sécurisés



privDst : désigne le groupe pour lequel le message est destiné
authInfo: protocole d'authentification utilisé

Laurence Duchien

Émission d'une requête sécurisée

- construction d'un message:
  srcParty<-groupe d'émission
  dstParty<- groupe de réception
  contexte <- contexte voulu
  PDU <- Get, set,...

- La base de données locale de l'entité émettrice est consultée pour récupérer entre autre le type de protocole authentification utilisé

- un message authentifié est construit :
  authInfo<- type de protocole

- La base de données locale de l'entité émettrice est consultée pour récupérer les caractéristiques du protocole

- un message privé est construit
  privDest<- identifie le destinataire
  message crypté

- transmission au destinataire

Laurence Duchien

Exemples d'agents

configuration d'un agent non sécurisé

Identity	gracie (agent)	george (manager)
Domain	snmpUDPDomain	snmpUDPDomain
Address	1.2.3.4, 161	1.2.3.5, 2001
Auth Prot	noAuth	noAuth
Auth Priv Key	""	""
Auth Pub Key	""	""
Auth clock	0	0
Auth lifetime	0	0
PrivProt	noPriv	noPriv
PrivPrivKey	""	""
PrivPubKey	""	""

Base de données de l'agent

Target	Subject	Context	Privileges
gracie	george	local	35 (Get,GetNext &GetBulk)
george	gracie	local	132(Response et SNMPv2-Trap)

configuration d'un agent sécurisé

Identity	ollie (agent)	stan (manager)
Domain	snmpUDPDomain	snmpUDPDomain
Address	1.2.3.4, 161	1.2.3.5, 2001
Auth Prot	v2md5AuthProtocol	v2md5AuthProtocol
Auth Priv Key	"0123456789AZ"	"GHIJKLM45"
Auth Pub Key	""	""
Auth clock	0	0
Auth lifetime	300	300
PrivProt	desPrivProtocol	desPrivProtocol
PrivPrivKey	"MNOPIU89"	"BNJIUY78"
PrivPubKey	""	""

Base de données de l'agent

Target	Subject	Context	Privileges
ollie	stan	local	35 (Get,GetNext &GetBulk)

stan ollie local 132(Response et SNMPv2-Trap)

Laurence Duchien

Algorithme de synchronisation des horloges

Pour cet algo on utilise un nouvel objet :

```
AuthInformation ::= [2] IMPLICIT SEQUENCE {
    authDigest      OCTET STRING,
    authDstTimestamp UInteger32,
    authSrcTimestamp UInteger32
    }
```

Lorsqu'un message est transmis, il inclut les valeurs d'horloges de l'émetteur et du récepteur.

Ces horloges sont synchronysées de telle manière que l'horloge la plus lente soit égale à l'horloge la plus rapide.

Considérons deux groupes : "AgentParty" et "MgrParty" contenant respectivement un agent et une station de gestion.

4 cas de figure sont envisageables :

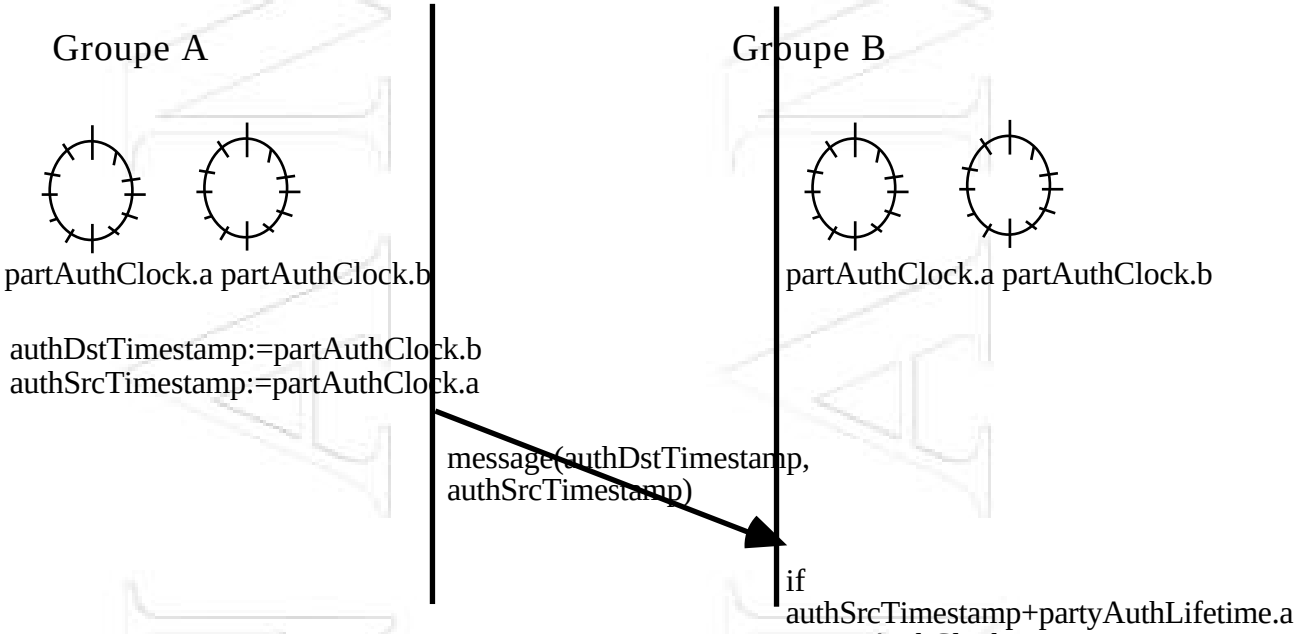
- l'estimation de l'horloge "AgentParty" qu'a la station de gestion dépasse la valeur qu'en a l'agent

- l'estimation de l'horloge "MgrParty" qu'a la station de gestion dépasse la valeur qu'en a l'agent

- l'estimation de l'horloge "AgentParty" qu'a l'agent dépasse la valeur qu'en a la station de gestion

- l'estimation de l'horloge "MgrParty" qu'a l'agent dépasse la valeur qu'en a la station de gestion

Algorithme de synchronisation des horloges(2)



5. Conclusion

- Snmpv2 : plus efficace, plus sécurisé que SNMP
mais pas encore de migrations complètes
de tous les sites

- Approche OSI marginale, non à cause des concepts, mais du fait des investissements déjà réalisés par les administrateurs sur SNMP

-> recherche d'un protocole compatible avec la V1 de SNMP

- Quelques inconnues :
  - le devenir du modèle OSI (CMISE/CMIP)
  - la forte évolution des réseaux et l'émergence de nouveaux protocoles
  - problèmes législatifs concernant le cryptage

- 2 phénomènes qui devraient aussi influencer l'administration de réseau:

  - la technologie orientée objet
  - la notion d'agent intelligent (sachant prendre des décisions sans en référer à la station de gestion)

www.Mcours.com
Site N°1 des Cours et Exercices Email: contact@mcours.com

6. 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 Bouquins :

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)

.....