SNMP – *Simple Network Management Protocol*

Plan : Partie 1 : généralités, historique

Partie 2 : architecture

Partie 3 : protocole

Partie 4 : représentation des données

Généralités – SNMP

- SNMP permet
  - de surveiller « à chaud » les éléments du réseau (routeur, station de travail, imprimante…) en collectant des informations sur leurs états et leurs comportements
  - de gérer la configuration des éléments du réseau
- SNMP est un protocole d'administration se voulant simple et donc facile à implémenter

Généralités - SNMP

- En terme de modélisation, SNMP définit un format universel de représentation des informations d'administration
- En terme de transport, SNMP définit un protocole permettant l'échange de ces informations pour n'importe quel terminal
  - Initialement, fournit un protocole ciblant IP
  - Puis, supporte l'ajout de mécanismes pour d'autre architectures réseaux

SNMP - historique

- SNMP est issu du protocole SGMP (*Simple Gateway Monitoring Protocol*)
- SNMP est un standard de l'Internet :
  - 3 versions coexistent
  - des stratégies de coexistence sont définies (RFC 3584)
- Le protocole et la structure des informations d'administrations sont standardisés par l'IETF au niveau de RFC distinctes

Historique – version initiale SNMPv1, 1998

- Est devenue et reste le standard *de facto* dans la communauté Internet
- Cette version est largement critiquée du fait du manque de sécurité :
  - L'authentification se base sur un mot de passe (champs *community string*) transporté en clair sur le réseau
  - 1992 : volonté de sécuriser SNMP avec SNMPsec

Historique - SNMPv2

- SNMPv2 reprend le modèle de SNMPv1 et comble ces lacunes fonctionnelles :
  - Introduit une communication entre managers
  - Améliore la modélisation des objets (qui reste proche de SNMPv1)
  - Introduit une primitive limitant le nombre d'objets regroupés et envoyés sur le réseau
  - En améliore la structure des messages de notifications
  - SNMPv2C est la version expérimentale/intermédiaire majoritairement utilisée. Or, elle n'inclut pas le nouveau modèle de sécurité de l'époque qui est très controversé

Historique - SNMPv3

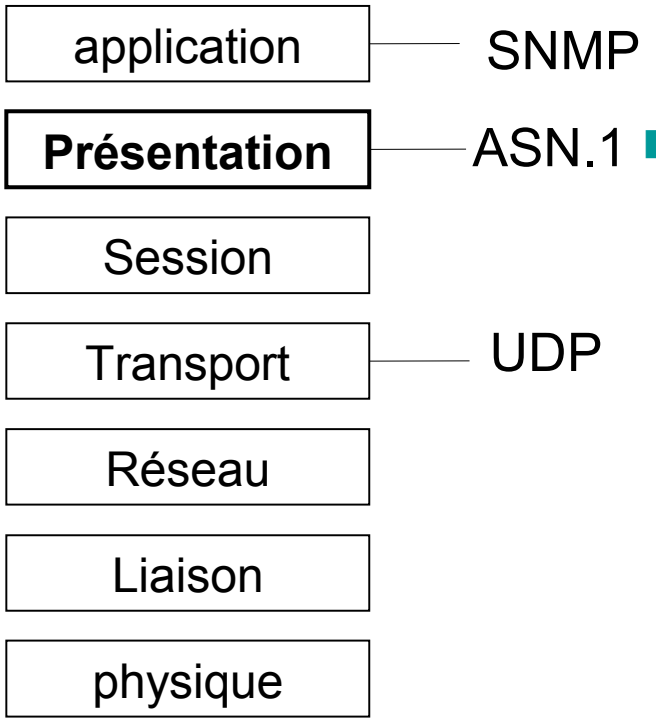
- Fait basculer le modèle client-serveur de SNMPv1/v2 dans le modèle peer-to-peer et modulaire
  - la notion de maître/esclave est remplacée au profis de celle d'entité
  - L'ensemble des fonctionalités est décomposée en modules
- Fournit un réel support en matière de sécurité : confidentialité, authentification, intégrité des messages

Partie 1 : Architecture SNMP

Architecture SNMP – client-serveur

- SNMP a pour but d'administrer en intervenant à distance
- SNMP est donc (comme la majorité des systèmes d'administration) de type client – serveur

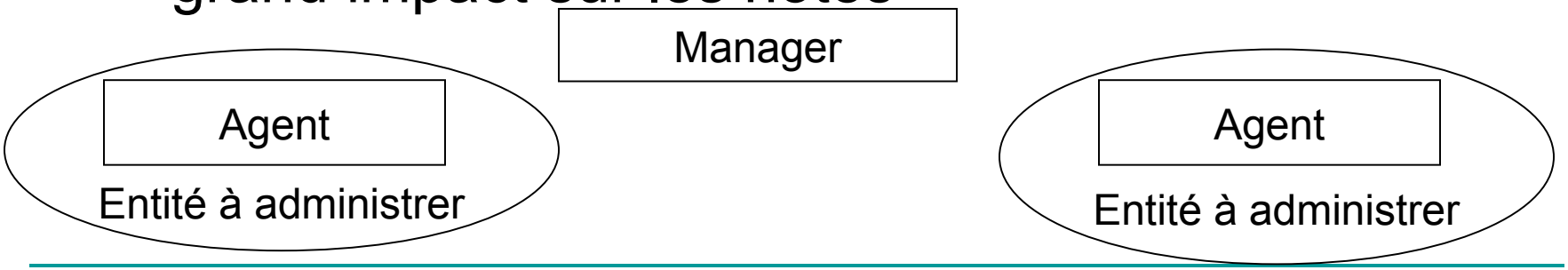
SNMP et le modèle OSI



- Un protocole de niveau applicatif permettant à deux entités SNMP de dialoguer
- SNMP s'appuie sur
  - UDP (communication non fiable, sans connexion) : un message SNMP est un paquet transporté via UDP sur un port (port standard : 161 pour les commandes et 162 pour les *traps*)
  - ASN.1

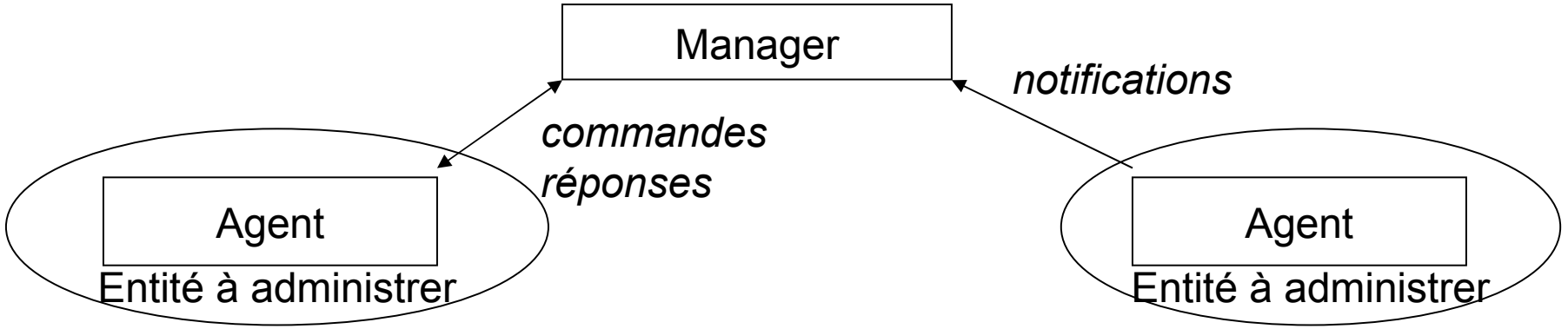
Architecture SNMPv1/v2 – manager/agent

- Un **manager** (entité logicielle) centralise les informations en provenance des équipements du réseau par l'intermédiaire **d'agents** agissant pour le compte du **manager**
  - L'intelligence se trouve dans le manager
  - Ainsi, les agents sont simples et légers sans grand impact sur les hôtes

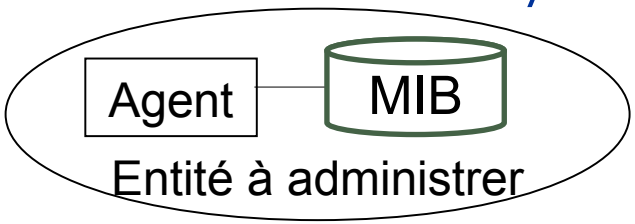


Architecture SNMPv1/v2 – manager

- Le **manager :**
  - Envoie des **commandes** (de contrôle) et reçoit les réponses y afférant : approche de type *pull* (question-réponse)
  - Reçoit des **notifications** non sollicitées. Il incombe au manager d'émettre des commandes pour obtenir plus de détail (approche *push*)



Architecture SNMPv1/v2 – agent

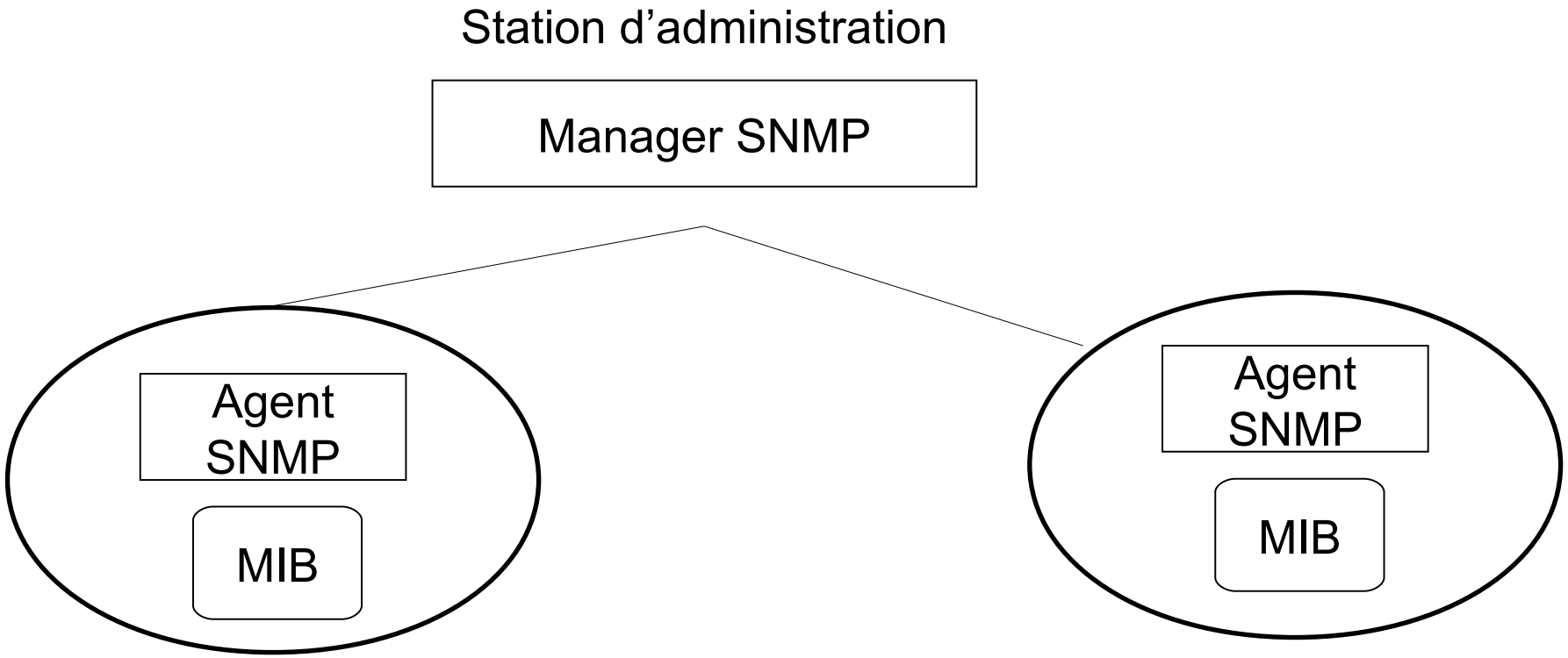


- L'agent fournit un accès à un objet local MIB qui reflète l'état des ressources et l'activité de l'hôte
- L'agent répond aux commandes du manager en retournant les valeurs de la MIB et en assignant des valeurs
- Exemples d'objets :
  - un compteur comptabilise le nombre de messages reçus sur un lien (permet au manager de surveiller l'activité/la charge du réseau au niveau d'un nœud)
  - L'état du lien peut être placé en mode inactif en assignant la valeur correspondante

Architecture SNMP v1, v2

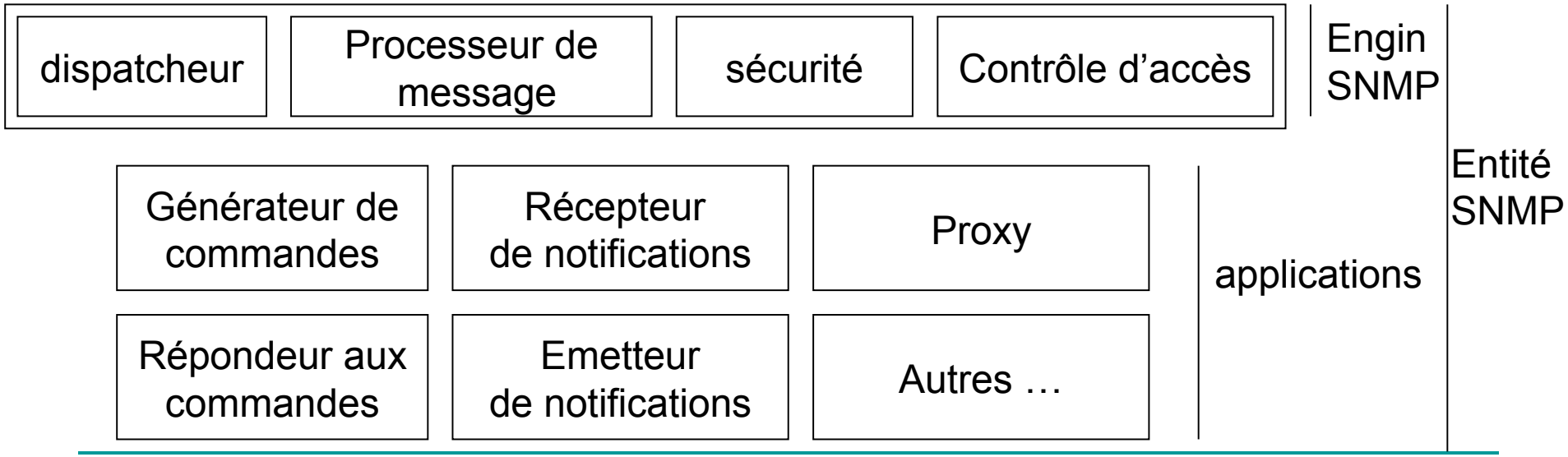
- L'interopérabilité entre SNMPv1 et SNMPv2c n'est pas préservée
  - Le format des messages change (nouvelles notifications)
  - Des commandes supplémentaires sont introduites
- 2 solutions :
  - un agent SNMPv2 joue le rôle de *proxy* et traduit
  - un manager supporte SNMPv1 et v2

Architecture SNMPv1,v2



Architecture SNMP V3

- Propose une nouvelle terminologie : on ne parle plus de manager, d'agent mais d'entité
- Constitue une transition vers une architecture p2p et modulaire : une entité est composée d'un engin de base et d'applications



SNMPv3 et la sécurité

- Une entité SNMP est exposée à plusieurs dangers :
  - Modification des informations
  - Mascarade
  - Modification du flux de messages
- A cet effet, SNMP propose
  - Des mécanismes de sécurité
  - Une configuration standardisée des mécanismes de sécurité

Mécanismes de sécurisation de SNMPv3

- Les algorithmes de hachage MD5 et Secure Hash (SHA) sont utilisés pour calculer des hashs
  - Se prémunir contre des attaques modifiant les données
  - Fournir indirectement une authentification de l'origine
  - Se défendre contre des attaques de type mascarade

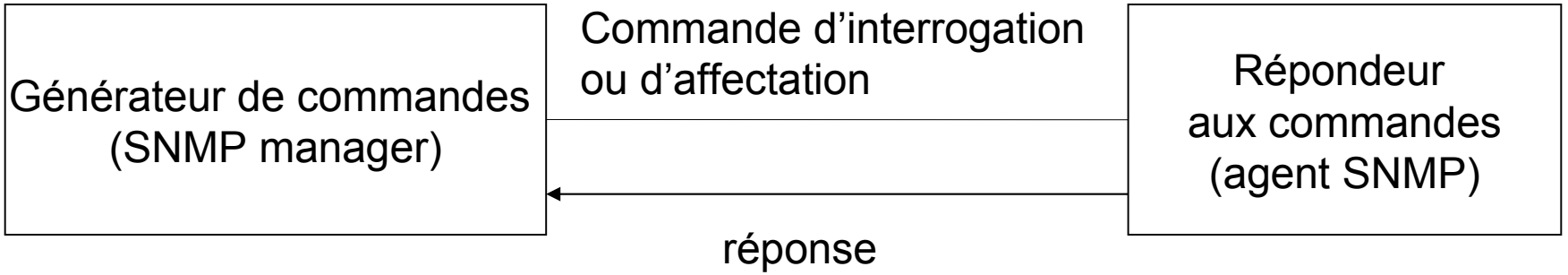
Mécanismes de sécurisation de SNMPv3

- Une synchronisation avec des indicateurs de temporisation permet de se prémunir de certaines attaques de modification de flux
  - A cet effet, un mécanisme de synchronisation d'horloge sans intervention d'un tiers est proposé
- Les messages sont chiffrés en ayant recours à DES (*Data Encryption Standard*)
- Un contrôle d'accès aux informations et aux terminaux administrés est proposé

Partie 2 – Protocole SNMP v1/2

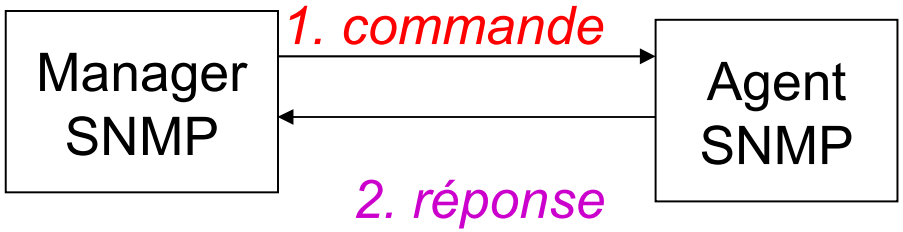
Modèle d'interaction

- SNMP se base sur 5 primitives lui permettant de réaliser des actions sur une MIB



- Ces primitives offrent deux fonctionnalités fondamentales : l'interrogation et l'assignation
  - Avantage : 1. simplicité, 2. ces primitives peuvent être facilement gérées par un protocole asynchrone

Messages (1/2)

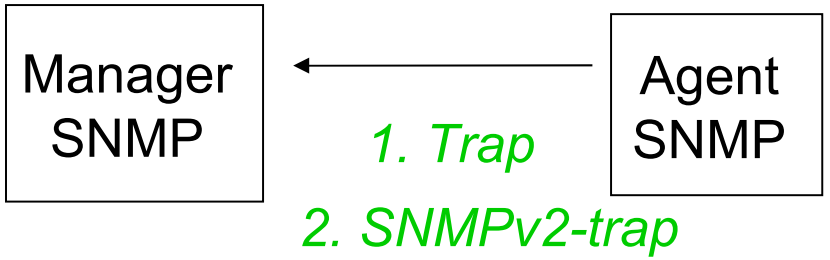


- Commandes d'interrogation (`Getxxx`)
  - `GetRequest` : récupère des informations portant sur un objet administré identifié par une OID
  - `GetNextRequest` : récupère des valeurs d'objets se suivant lexicographiquement dans l'arbre de nommage
    - Cette commande peut générer un trafic important (d'où la commande `GetBulk` qui limite le nombre des objets retournés)
  - `GetBulkRequest` : est introduit par SNMPv2
- `SetRequest` : commande permet d'affecter une valeur à un ou plusieurs objets (`Setxxx`)
- Réponse à une commande :
  - `GetResponse` (SNMPv1)
  - `Response` (SNMPv2)

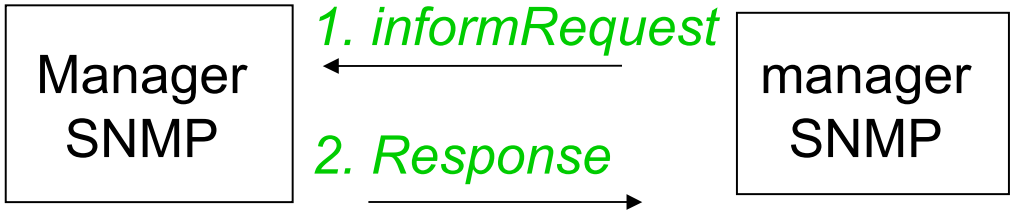
Extrait de la RFC 1157 (SNMPv1)

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

Messages (2/2)



- Notifications = message non sollicité (asynchrone) envoyé par l'agent au manager
  - `Trap` : indication de déroutement
  - `SNMPv2-trap` : indication de déroutement provenant d'un agent SNMPv2
- Information non sollicitée envoyée entre 2 **managers**
  - `InformRequest` (avec SNMP v2)



Format d'un message SNMPv1/2 d'une commande

- Le message SNMP dans son ensemble est une séquence de 3 champs :

Version (*Integer*)	Communauté (*Octet String*)	Protocol Data Unit, ou PDU (*type construit*)

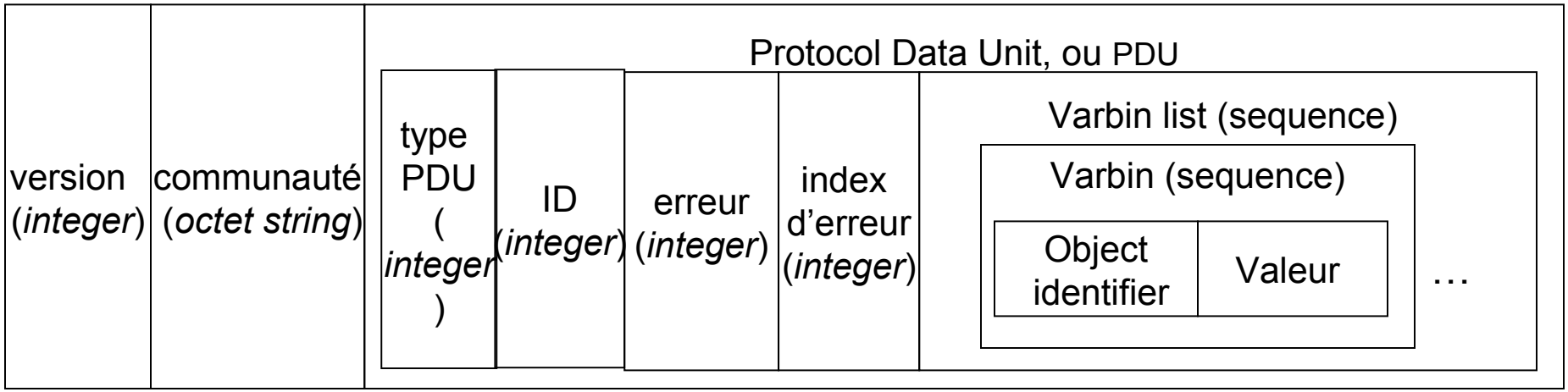
  - La **version** du protocole (=0 pour SNMPv1)
  - La **communauté** : un environnement d'accès pour un groupe de manager. Un agent ne connaissant pas le nom de la communauté est exclu
  - Le **PDU** est un type construit (dans le jargon ASN.1), et est donc constitué de plusieurs champs
    - Son contenu varie selon qu'il s'agisse d'une commande-réponse ou d'une trap

Extrait de la RFC 1157 (SNMPv1)

```
Message ::= SEQUENCE {
version -- version-1 for this RFC
 INTEGER { version-1(0) },
  community OCTET STRING, --
  community name
data -- e.g., PDUs if trivial
ANY -- authentication is being used
  }
```

Format PDU – commande et réponse

- Le PDU est constitué de 5 champs dont varbin List (pour variable binding list) qui est de type construit (séquence de Varbin)



Extrait de la RFC 1157 (SNMPv1)

```
PDU ::= SEQUENCE {
request-id INTEGER,
error-status -- sometimes ignored
INTEGER { noError(0), tooBig(1),
  noSuchName(2), badValue(3),
  readOnly(4), genErr(5) },
 error-index INTEGER, -- sometimes
  ignored
 variable-bindings -- values are
  sometimes ignored VarBindList }
```

Format PDU – commande et réponse

- **ID** : identifiant de la requête (et donc aussi sa réponse) qui est attribué lors de la requête
- **Erreur** : code d'erreur
  - Est placé par l'agent SNMP si une erreur survient,
  - Est initialisé à 0 dans la requête envoyée par le manager
- **Index d'erreur** :
  - index pointant sur le premier objet ayant causé l'erreur
  - Est égale à 0x00 si aucune erreur ne survient
- Remarque : la primitive GetBulk utilise le champs erreur *et indexerreur* pour stocker les valeurs `nonrepeater` et `maxrepetition`

Format - Value

- **Varbind list** :
  - Séquence constituée de la paire OID-valeur
  - Représente une liste de variables
- Suivant le type de message, la valeur diffère. Par exemple, pour
  - `SetRequest` : la valeur correspond à celle de l'OID spécifiée
  - `GetRequest` : valeur est `NULL`
  - `GetResponse` la valeur correspond à celle de l'OID de l'agent SNMP spécifié

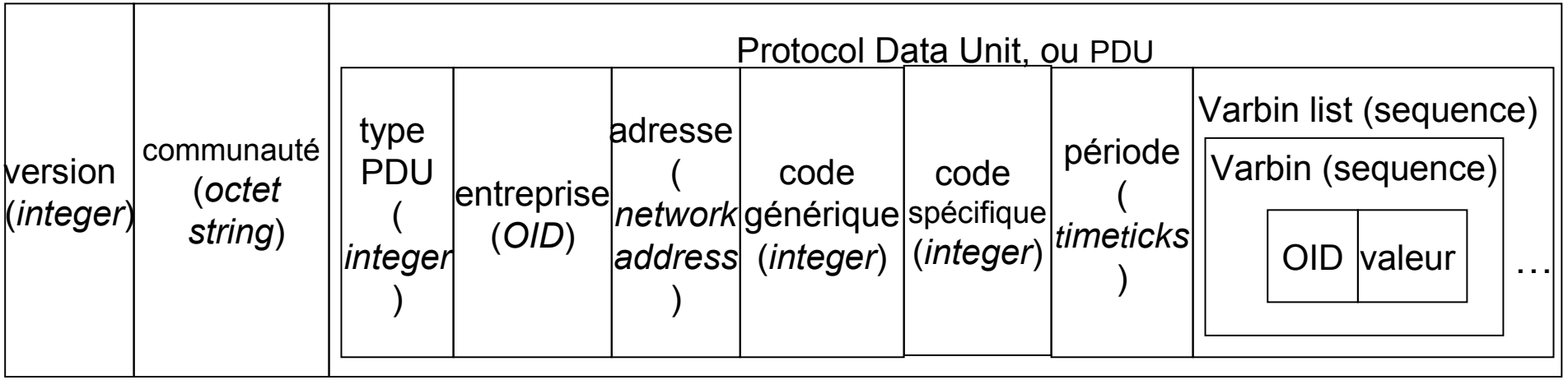
Code d'erreur – SNMPv1

Valeur du code d'erreur	explication
0x00	aucune erreur n'est survenue
0x01	réponse trop longue pour être transportée
0x02	le nom de l'objet demandé n'a pas été trouvé
0x03	le type de donnée de la requête ne « matche » pas celui se trouvant dans l'agent SNMP
0x04	le manager a tenté d'assigner une valeur à un paramètre accessible ne lecture seulement
0x05	0x05 – Erreur générale

Code d'erreur – SNMPv2

Code d'erreur	Explication
0x06	Accès non permis
0x07	e type n'est pas le bon
0x08	a taille de la valeur excède la taille permise
0x09	roblème d'encodage
0x10	aleur fausse/impossible
0x11	ssignation à une variable n'existant pas dans la MIB
0x12	'objet de la MIB est inconsistant et n'accepte pas d'affection
0x13	es ressources nécessaires à l'affection sont indisponibles
0x14	rreur lors de l'assignation

Format d'une trap – SNMPv1



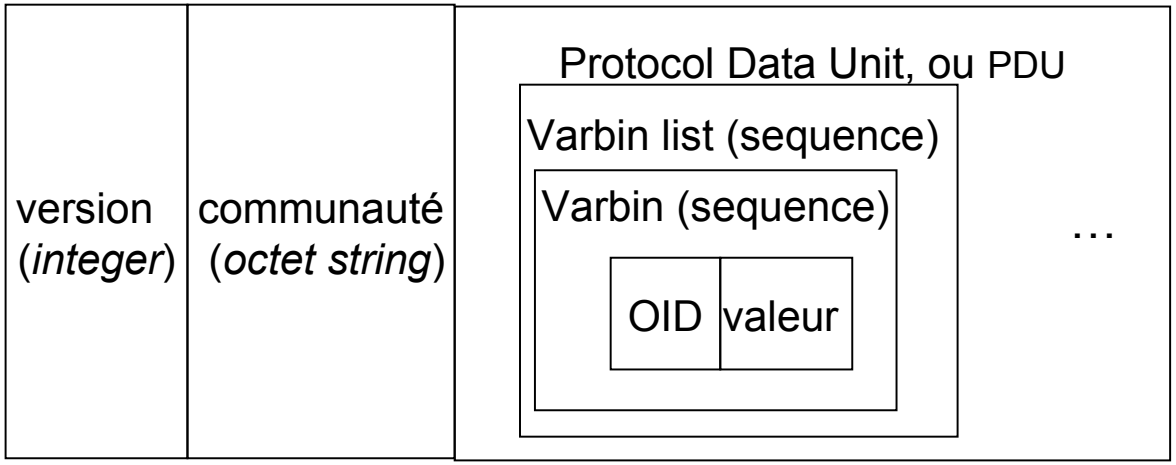
- `Entreprise` : identification de l'objet administré générant la trap
- `Adresse` de l'agent ayant généré la trap
- le `code générique` (c -à-d standardisé) et `spécifique` (non conventionnel) de la trap
- `Période` écoulée depuis la dernière trap ou réinitialisation

Extrait de la RFC 1157 (SNMPv1)

```
Trap-PDU ::= [4] IMPLICIT SEQUENCE {
Enterprise  OBJECT IDENTIFIER,
agent-addr NetworkAddress, -- trap
generic-trap  INTEGER { coldStart(0),
  warmStart(1), linkDown(2), linkUp(3),
  authenticationFailure(4),
  egpNeighborLoss(5),
  enterpriseSpecific(6) },
specific-trap INTEGER,
time-stamp TimeTicks
variable-bindings VarBindList }
```

Format d'une trap – SNMPv2

- Avec SNMPv2, la structure d'une trap se simplifie : les informations sont stockées dans le champs Varbin list sous la forme d'attributs (OID-valeur)



Architecture SNMP - dialogue standardisé

- SNMP formalise les échanges possibles entre les « entités » SNMP : agent(s) et un (ou plusieurs) manager(s)
- SNMP modélise aussi les informations d'administration. A cet effet, il définie la structure des objets administrés (**MIB**) en se basant sur la syntaxe **SMI** (Structure of Management Information)

Part 3 – Représentation des informations. MIB et SMI

Management Information Base (MIB)

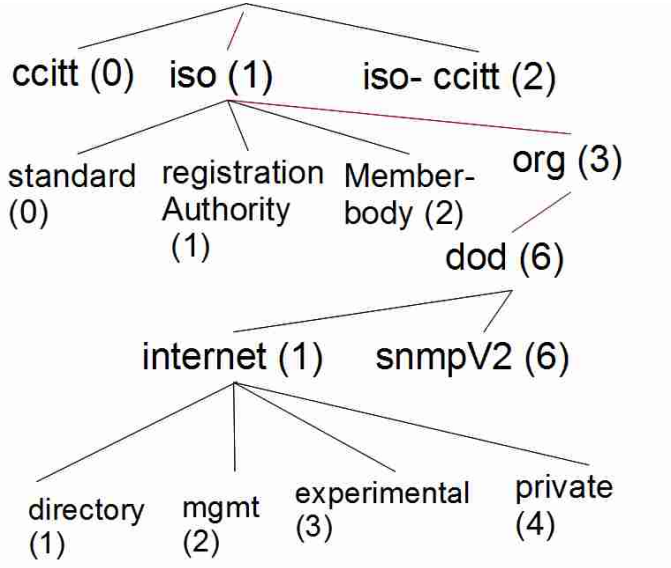
- Une **MIB** est une collection d'informations
  - organisées de façon hiérarchique
  - Accédées par un protocole (SNMP)
- Une MIB comprend un ensemble d'objets administrés (ou objet MIB ou simplement par abus de langage, MIB)
- Un objet MIB représente une caractéristique du terminal administré

Management Information Base (MIB)

- Une MIB est une spécification orientée data définissant le nommage, le type, le format, les actions auprès des objets administrés
- La MIB forme une interface d'accès auprès de l'instrumentation sous-jacente ; les objets étant accédés via la MIB (sorte de base de données **virtuelle**)

Organisation hiérarchique de la MIB

- Les objets gérés au travers d'une MIB sont organisés sous la forme d'une hiérarchie
- Cette hiérarchie (ou arbre) a
  - une racine sans nom
  - Des feuilles correspondant aux objets supervisés
- Les différents niveaux de la hiérarchie sont gérés par différentes organisations (ISO, CCITT au niveau 1)



Organisation hiérarchique de la MIB

- La hiérarchie est organisée en deux parties :
- la partie en dessous de 1.3.6 (dod)
  - est la seule à être utilisée par SNMP
  - est prévue pour le dod, département de la défense américain à l'origine de l'Internet
- Private (4) : les vendeurs peuvent définir des branches privées incluant les objets MIB de leurs propres produits

Représentation unifiée des informations

- Nécessité : fournir une représentation unifiée des composants à administrer, indépendante de la plateforme matérielle des terminaux
- Approche : le standard ASN.1 est utilisé en tant que syntaxe de base de travail
  - Un sous ensemble de la syntaxe ASN.1 appelé SMI, est défini
  - SMI introduit aussi des macros ASN.1 et devient en cela aussi un sur-ensemble de ASN.1

Objets MIB

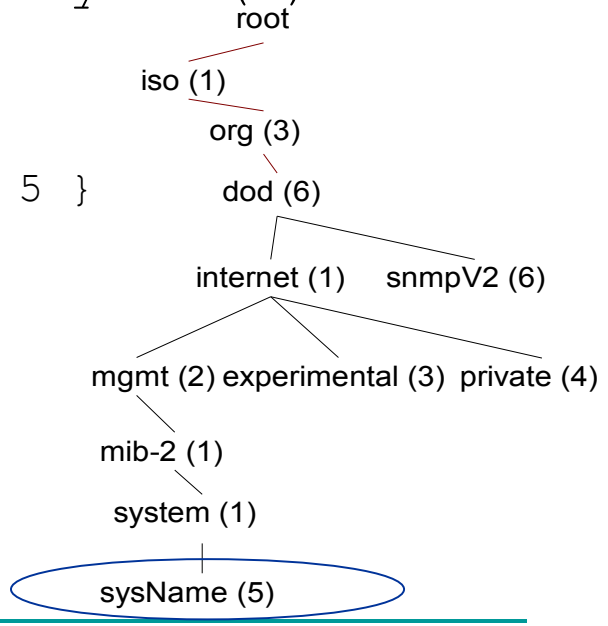
- Dans cette hiérarchie, un objet de la MIB est identifié de façon non ambiguë par
  - un nom
  - un identifiant de type OID (Object Identifier) qui identifie l'objet de façon unique dans la hiérarchie (c'est-à-dire dans l'espace de nommage globale)

Exemple - nom d'une machine

- Le codage de l'OID suit les règles de codage définies par ASN.1
- On considère l'objet nommé `sysName`
  - Son **OID** est `sysName OBJECT IDENTIFIER ::= { iso org(3) dod(6) internet(1) mgmt(2) mib-2(1) system(1) sysName(5)}`
    - Alternative : OID défini à partir du père

      `sysName OBJECT IDENTIFIER ::= { system 5 }`
    - Forme abrégée :

      `.1.3.6.1.2.1.1.5`

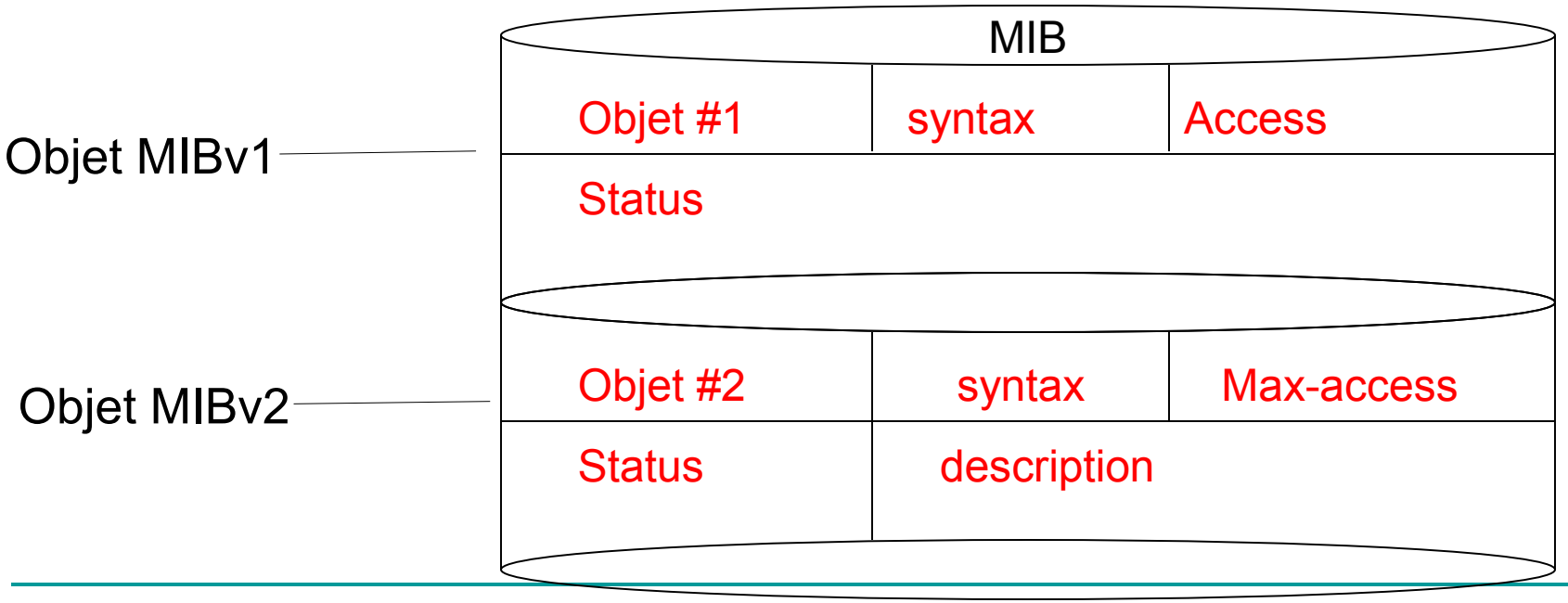


Avantage des OID

- Chaque information d'administration est identifiée de façon **unique** et **globale** (c'est-à-dire au sein d'une même espace de nommage)
- Fournit une façon de **déléguer** les autorités administratives
- Fournit un nommage flexible (homme/machine) alliant nombre et nom

MIB

- Une MIB contient un ensemble d'objets
- La structure des objets évolue entre les versions 1 et 2 de SNMP (et SMI)
  - Avec v1 : 4 éléments décrivent un objet
  - Avec v2 : 5 éléments décrivent un objet



Objet MIB

- Un objet contenu dans une MIB est défini par
  - son nom
  - Son type (`SYNTAXE`)
  - Son rôle (`DESCRIPTION` : chaîne de caractère indiquant le rôle de l'objet avec SNMPv2)
  - Ses droits d'accès (`ACCESS` avec SMPv1 et `MAX-ACCESS` avec SNMPv2), par exemple, `read-only`
  - Son état de (`STATUS`) (par exemple `optional`)

Syntaxe SMIv1 d'un objet MIB – extrait de la RFC 1155

```
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" END
```

Syntaxe SMIv2 d'un objet MIB – extrait de la RFC 2578

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

Suite syntaxe SMIv2 d'un objet MIB – extrait de la RFC 2578

```
UnitsPart ::= "UNITS" Text | empty

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

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

ReferPart ::= "REFERENCE" Text | empty

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

IndexTypes ::= IndexType | IndexTypes ","
IndexType

IndexType ::= "IMPLIED" Index | Index
```

Droits d'accès, SNMPv1 versus SNMPv2

Droit d'accès	Description
`read-only`	Accès en lecture
`read-write`	Accès en lecture et écriture
`write-only`	Accès en écriture
`not-accessible`	Non accessible
`read-create`	Accès à la création (avec SMIv2),
`accessible-for-notify`	Accès pour trap (avec SMIv2)

Status, SNMPv1 versus SNMPv2

	Description
`mandatory`	Présence obligatoire (SNMPv1)
`optional`	Présence optionnelle (SNMPv1)
`obsolete`	Est obsolète (SNMPv1 et SNMPv2)
`deprecated`	est toujours supporté mais deviendra obsolète dans le futur ou sera remplacé par un objet plus adapté (SNMPv2)
`current`	actuel (SNMPv2)

Exemple

Nom de l'objet

```
rptrMonitorTransmitCollisions OBJECT-TYPE
  SYNTAX Counter32
  MAX-ACCESS read-only
  STATUS deprecated
  DESCRIPTION « ... »
  REFERENCE « [IEEE 802.3 Mgt], 30.4.1.8,
    aTransmitCollisions. "» ::=
  { rptrMonitorRptrInfo 1 }
```

de type counter

Accessible en lecture seulement

Objet désuet avec SMIv2, mais pouvant continuer à être utilisé pour des raisons de comptabilité

Description textuelle

OID

Types primitifs

Nom	Type	Nombre d'octets	Signification
INTEGER	ASN.1	4	Entier
BIT STRING	ASN.1 défini par SNMPv2	≥ 0	Suite de 1 à 32 bits
OCTET STRING	ASN.1	≥ 0	Chaîne de caractères (0 to 65,535 octets)
OBJECT INDENTIFIER	ASN.1	➢ 0	Identifiant
NULL	ASN.1		NULL

Types applicatifs

Nom	Type	Nombre d'octets	Signification
Counter Counter32, Counter64	Entier pour SMIv1, Entier positif pour SNMPv2	4 ou 5	Compteur augmentant jusqu'à ce qu'il atteigne son maximum et soit alors réinitialisé.
TimeTicks	Entier	4	Compte une durée en centième de seconde depuis un instant donné
Gauge, Gauge32	Entier positif Même chose que Gauge	4	Entier positif dont la valeur ne passe jamais à 0 en cas de dépassement
IpAddress	OCTET STRING	4	Adresse IPv4 sous forme décimale pointée. Pas de IPv6 en SMIv1 et 2
NetworkAdress	OCTET STRING	4	Même chose que IpAdress mais peut représenter différents types d'adresses

Types applicatifs

Nom	Type	Nombre d'octets	Signification
Opaque	OCTET STRING		donnée opaque utilisée pour représenter des informations arbitraires non-conformes aux autres types SMI. Devient obsolète avec SMIv2
Integer32	INTEGER	4	Entier codé sur 32 bits même pour une unité centrale de 64. SMIv2
Unsigned32		4	Entier positif codé sur 32 bits. SMIv2
BITS			Énumération de bits

Types construits

Type ASN.1	Description
SEQUENCE	
SEQUENCE OF	

SMI

- Au niveau le plus bas, les variables SNMP sont définies en tant
  - qu'objets individuels pour décrire les objets administrés
  - que notifications
- Les objets apparentés sont réunis dans des groupes assemblés en modules
  - Ainsi un agent SNMP ne gère que les groupes et modules dont il a besoin

Groupes dans les MIB-2

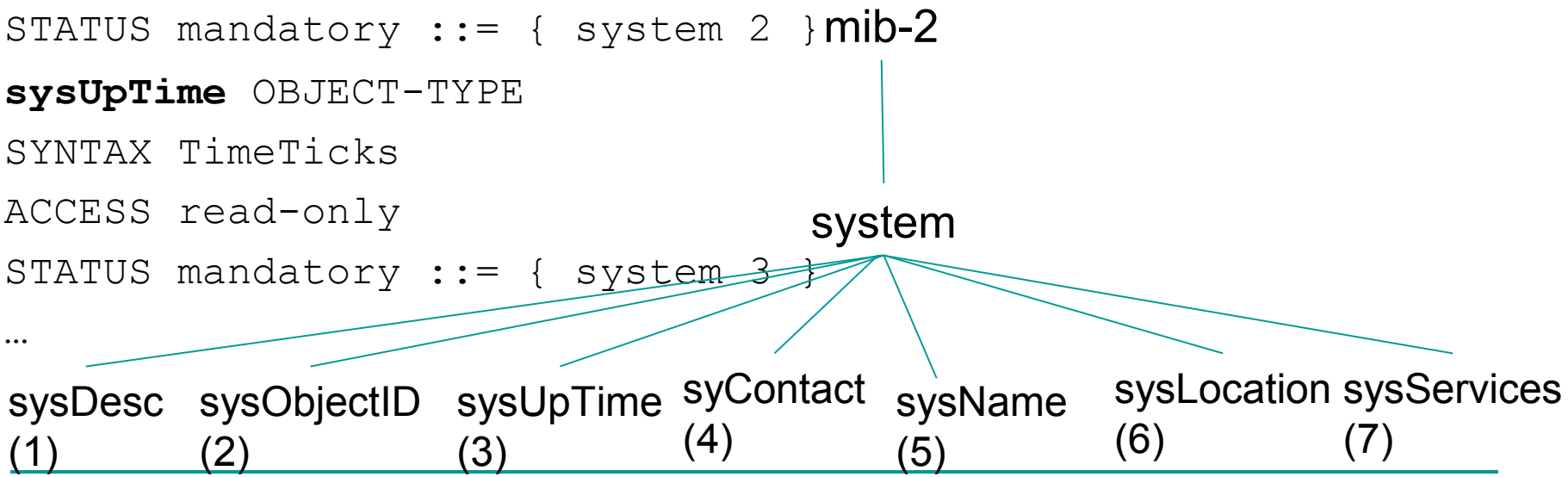
Groupe	Description
System	Nom, emplacement et description de l'entité
Interfaces	Interfaces réseau et trafic mesuré
AT	Traduction d'adresse (usage peu recommandé)
IP	Statistiques sur les paquets IP
ICMP	Statistiques sur les messages ICMP reçus
TCP	Algorithmes, paramètres et statistiques TCP
UDP	Statistiques de trafic UDP
EGP	Statistique de trafic EGP (usage historique)
Transmission	Réservé
SNMP	Statistiques de trafic SNMP

Exemple - MIBv1 RFC 1066

```
RFC1066-MIB { iso org(3) dod(6) internet(1) mgmt(2)
1 }
DEFINITIONS ::= BEGIN IMPORTS mgmt,
OBJECT-TYPE, NetworkAddress, IpAddress, Counter,
Gauge, TimeTicks FROM RFC1065-SMI;
mib OBJECT IDENTIFIER ::= { mgmt 1 }
system OBJECT IDENTIFIER ::= { mib 1 }
interfaces OBJECT IDENTIFIER ::= { mib 2 }
at OBJECT IDENTIFIER ::= { mib 3 }
ip OBJECT IDENTIFIER ::= { mib 4 }
icmp OBJECT IDENTIFIER ::= { mib 5 }
tcp OBJECT IDENTIFIER ::= { mib 6 }
udp OBJECT IDENTIFIER ::= { mib 7 }
egp OBJECT IDENTIFIER ::= { mib 8 } END
```

Description ASN.1 du groupe System

```
sysDescr OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-only
STATUS mandatory ::= { system 1 }
sysObjectID OBJECT-TYPE
SYNTAX OBJECT IDENTIFIER
ACCESS read-only
STATUS mandatory ::= { system 2 }
sysUpTime OBJECT-TYPE
SYNTAX TimeTicks
ACCESS read-only
STATUS mandatory ::= { system 3 }
…
```



Modules

- Les objets se regroupent dans un module qui
  - définit ses relations avec les autres modules (clauses `IMPORT` et `EXPORT`)
  - S'identifie en appelant la macro `MODULE-IDENTITY` dont les paramètres fournissent le nom, l'adresse du développeur ect…
  - Spécifie son emplacement dans l'arbre (clause `OBJET-IDENTITY`)
  - Définit les objets MIB en faisant appel à la macro `OBJECT-TYPE` qui indique les variables à gérer et leur propriétés
  - Définit ses exigences en matière d'implémentation

Exemple de module extrait de la RFC 3418

```
SNMPv2-MIB DEFINITIONS ::= BEGIN
IMPORTS MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE, …
FROM SNMPv2-SMI DisplayString, …
MODULE-IDENTITY
LAST-UPDATED "200210160000Z"
ORGANIZATION "IETF SNMPv3 Working Group"
CONTACT-INFO "WG-EMail: snmpv3@lists.tislabs.com
Subscribe: snmpv3-request@lists.tislabs.com Co-Chair: Russ
Mundy Network Associates Laboratories… "
DESCRIPTION "The MIB module for SNMP entities…. This
version of this MIB module is part of RFC 3418... "
REVISION "200210160000Z"
DESCRIPTION "This revision of this MIB module was
published as RFC 3418."
REVISION "199511090000Z"  DESCRIPTION "…"
```

Suite de l'exemple

```
 ::= { snmpModules 1 }
snmpMIBObjects OBJECT IDENTIFIER ::= { snmpMIB 1 }
-- the System group
system OBJECT IDENTIFIER ::= { mib-2 1 }
 sysDescr OBJECT-TYPE
SYNTAX DisplayString (SIZE (0..255))
MAX-ACCESS read-only
STATUS current
DESCRIPTION "A textual description of the entity.
This value should include the full name and version
identification of the system's hardware type,
software operating-system, and networking software."
::= { system 1 }
```

….

Notifications

- Une notification (ou trap)
  - s'identifie (`OBJECT IDENTIFIER`) par
    - par un OID
    - Un numéro qui est soit générique (generic-trap) soit spécifique et défini dans un autre module
  - Peut se voir attachée des objets de la MIB et leurs valeurs respectives

Exemple de trap générique - extrait de la RFC1215

```
coldStart TRAP-TYPE
ENTERPRISE snmp
DESCRIPTION "A coldStart trap
signifies that the sending protocol
entity is reinitializing itself
such that the agent's configuration
or the protocol entity
implementation may be altered." ::=
0
```

Exemple de trap spécifique à une entreprise - extrait de la RFC1215

```
myEntreprise OBJECT IDENTIFIER :=
  {enterprise 9999}
myLinkDown TRAP-TYPE
ENTERPRISE myEnterprise
VARIABLES {ifIndex}
DESCRIPTION « myLinkDown trap
  significates that the application
  recognises a failures among one
  communication links within the agent
  configuration »
:=2
```

OID

Type générique : 6

La trap indique un changement à d'état down d'une interface dont le numéro est donné par ifIndex

2 est le numéro de trap spécifique

Numéro de traps génériques

Numéro génériques de trap	Description
`coldStart(0)`	indique un reset complet de l'agent (et probablement de l'équipement dans son entier
`warmStart (1)`	indique une réinitialisation
`linkDown (2)`	indique le passage à l'état down d'une interface
`linkUp (3)`	indique le passage à l'état up d'une interface
`authentificationFailure(4)`	une interrogation SNMP sur l'agent à donné lieu à une mauvaise authentification
`egpNeigborLoss(5)`	indique la perte d'un voisin EGP, quand le protocole de routage est supporté par l'équipement

Conclusion – SNMP (1/2)

- SNMP est le standard de facto d'administration des réseaux (IP)
- SNMP est largement implanté (routeurs, ponts, équipement de télécommunication) par les constructeurs/vendeurs
- On trouve d'autres standards pour l'administration de systèmes OSI comme CMIP

Conclusion – administration

- SNMP ne répond qu'à quelques unes des exigences fonctionnelles de l'administration réseau qui englobent plus généralement
  - Gestion des pannes
  - Gestion de la comptabilité
  - Gestion de la configuration
  - Gestion de la sécurité

Quelques RFC

RFC	Description
STD16, RFC 1155	Définit la structure des informations d'administration (syntaxe **SMIv1**)
STD16, RFC 1212	Décrit de façon plus concise la structure des informations d'administration (syntaxe SMIv1) et les **traps**
STD15, RFC 1157	Définit le protocole **SNMP**
STD17, RFC 1213	Décrit la structure d'une **MIB-II**
RFC 1215	Décrit les **traps**

Quelques RFC

RFC	Description
STD58, RFC2578	Définit les types fondamentaux, model d'objet MIB SMIv2
STD 58, RFC 2579	Conventions textuelles de SMIv2
RFC 1905	SNMPv2

RFC	Description
RFC 2570	Introduction à SNMPv3
RFC 2571, 2572, 2573	Architecture SNMP (emphase sur la securité)
RFC 2574	Modèle de sécurité de SNMPv3
RFC 2575	Contrôle d'accès