Administration du réseau

(Z:\Polys\Internet_gestion_reseau\5.SNMP.fm- 4 décembre 2007 14:05)

PLAN

- Introduction
- L'administration de réseau sous Internet
- La base des données administrables
- Le protocole SNMP
- Contrôle d'accès et protection
- Conclusion
- L'administration de réseau dans l'OSI

Bibliographie

- W. Stallings. "SNMP, SNMPv2 & CMIP". Addison-Wesley. 1993.
- A. Leinwand, K. F. Conroy, "Network management", Addisson-Wesley, 1996

1. Introduction

Buts de l'administration de réseau :
. configuration (configuration management)
. sécurité (security management)
. panne (fault management)
. audit (performance management)
. comptabilité (accounting management)

Le réseau est hétérogène :
. un petit ensemble d'opérations simples
. un unique protocole d'administration (SNMP)
. une base d'informations répartie (MIB)

Indépendance vis-à-vis des applications et des interfaces
. opérations élémentaires

Le réseau est réparti :
. administration à distance
. s'appuie sur le réseau lui-même (IP + UDP)

2. L’administration de réseau sous Internet

2.1. Introduction

SNMP (Simple Network Management Protocol) :

- SNMPv1 :
  - rfc 1156, 1157 (1990), MIB-1 : rfc 1158
- SNMPv2 :
  - introduction à SNMPv2 : rfc 1441
  - SMI (“Structure of Management Information”) de SNMPv2 : rfc 1442
  - le modèle administratif de SNMPv2 : rfc 1445, 1446, 1447
  - le protocole de SNMPv2 : rfc 1448
  - la MIB de SNMP : rfc 1450
- SNMPv3 : faiblement utilisé

Normes OSI d'administration de réseau (ISO 7498) :

- Common Management Information Service/Protocol
  - CMIS/CMIP : ISO 9595 et 9596

    => CMOP (CMIP over TCP) : rfc1189

2.2. Architecture générale

Architecture basée sur le client/serveur.

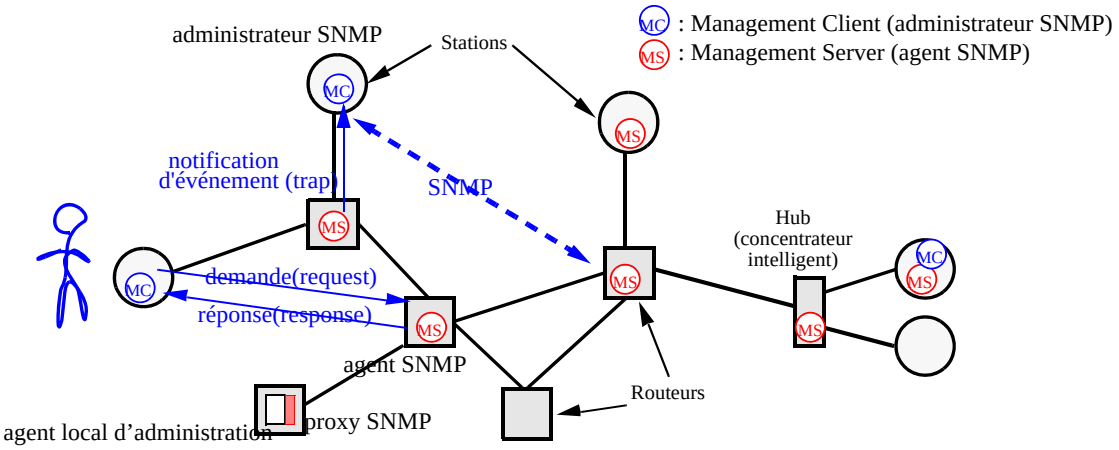
- L'agent SNMP est chargé de gérer les équipements :
  - il en propose une certaine vue (la MIB)
  - il met certains paramètres de l’équipement à la disposition du monde externe
- L'administrateur SNMP peut interroger/piloter les équipements par l'intermédiaire des agents.

Les échanges de données s’appuient sur IP/UDP et sont contrôlés par le protocole SNMP qui possède 3 types de message :

- request message
- response message
- trap message

Des proxys peuvent être utilisés pour réaliser une adaptation :

- d'équipement ou protocolaire



3. La base de données administrables

3.1. Introduction

Management Information Base (MIB)

- définit les objets administrables dans les équipements administrés
  - leur nom, leur type, leur sémantique, etc.
  - ex : ipInReceives - entier - nombre de datagrammes reçus
- différentes MIB :
  - la MIB standard +
  - des MIB spécialisées, adaptées à chaque type d'équipement/ nouveau réseau
  - . par ex. MIB pour Ethernet (rfc 1398), FDDI (rfc 1512), ou le pontage (rfc 1493)
- les valeurs de ces objets pourront être interrogées/modifiées par les administrateurs
  - exemple : quel est le nombre de datagrammes reçus par la station "poseidon" ?

    => *sendto(poseidon(get.req(ipInReceives)))*

3.2. Le type des objets

Les objets de la MIB peuvent être :
- simple : ipInReceives = [0 à 2^{32}-1]
- complexe : ipRouteTable !!!
- typée : ipAdresss (4 octets)

Les représentations existantes sont nombreuses. Par exemple les entiers :
- complément-à-1, complément-à-2, …
- sur un octet ou plusieurs, sur un mot, …
- longueur fixe ou variable, …

=> ASN-1 (Abstract Syntax Notation : X409)
définit le type de objets et leur représentation (codage)

Exemple : (information sur les interfaces d'une station)

```
ipAddrTable ::= SEQUENCE OF IpAddrEntry --liste des interfaces d'un noeud
IpAddrEntry ::= SEQUENCE {
  ipAdEntAddr   IpAddress, -- @IP de l'interface
  ipAdEntIfIndex   INTEGER, -- index vers l'interface correspondante
  ipAdEntNetmask   IpAddress, -- netmask associé
  ipAdEntBcastAddr   IpAddress, -- @IP de diffusion
  ipAdEntReasmMaxSize   INTEGER(0...65535) -- longueur max. du datag.
}
```


3.3. Définition des objets administrables

Un objet administrable est défini par :

o Son nom

o Le type de l'objet :
- en syntaxe ASN1
- préfixé par SYNTAX
- soit un type simple :
  - soit un type universel,
    - par ex : INTEGER ::= [UNIVERSAL 2]
  - soit un type spécifique à l'application, ici SNMP, et défini dans le rfc 1155
    - par ex : ipAddress ::= [APPLICATION 0] IMPLICIT OCTET STRING (SIZE(4))
- soit un type composé
  - par ex. : ipAddrTable

o Le niveau d'accès autorisé :
- préfixé par MAX-ACCESS
- 4 niveaux d'accès possibles : read-only, read-write, write-only, non-accessible
- c'est le niveau d'accès maximum possible

- les règles d'accès définissent le niveau d'accès réel

o Le statut de la définition de l'objet :

- préfixé par STATUS
- parmi les 3 statuts suivants : current, deprecated, obsolete

o La description textuelle (sémantique) de l'objet

o Son OID ("object identifier")

Exemple :

```
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 ..."
::= {system 1}
```


3.4. Les différentes catégories des objets

Pour faciliter leur gestion, les objets administrables de la MIB standard sont regroupés en plusieurs catégories.

catégorie	
system	système et informations générales
interface	interface d'accès au réseau (coupleur, contrôleur, …)
addr.trans.	adressage (ARP, …)
ip	protocole IP
tcp	protocole TCP
udp	protocole UDP
egp	protocole EGP
trans	informations sur les lignes de transmission (X25, FR, etc.)
snmp	protocole SNMP
...	pont, hub, imprimantes, RIP-2, etc.

Plus précisément, les objets des MIB spécialisées sont regroupés au sein de modules ASN1 :

- les objets d'un même module doivent être soit en totalité absents soit en totalité présents

3.5. Exemples d'objets administrables de la MIB standard

Nom	Catégorie	Sémantique
sysUpTime	system	durée depuis le démarrage
ifNumber	interfaces	nombre d'interfaces d'accès
ifMtu	interfaces	MTU (maximum transfer unit) d'une interface d'accès
ipInReceives	ip	nombre de datagrammes reçus
ipFragOKs	ip	nombre de fragments correctement reçus
ipRouteTable	ip	table de routage
tcpRtoMin	tcp	durée minimale du temporisateur de retransmission
udpInDatagrams	udp	nombre de paquets UDP reçus

3.6. Identification des objets

Identifie les objets administrables :

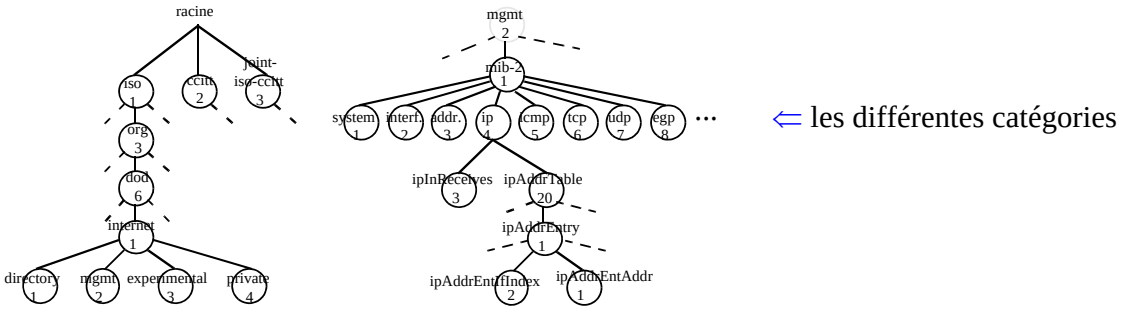
- tous les objets actuels et futurs
- tous les types d'objets :

  - normes de protocole, compteurs d'événements, paramètres de configuration, …

Délégation d'attribution des identificateurs

- efficacité, souplesse et maintien de la cohérence

  => un espace global, arborescent : MIT ("Management Information Tree")



⇐ les différentes catégories

3.7. L'OID ("Object identifier")

Nécessité d'avoir une identification universelle et non ambiguë des objets

- l'objet est identifié par la branche allant de la racine au noeud correspondant à l'objet
- la suite de labels (de numéros) identifiant chaque noeud intermédiaire
- exemple : iso.org.dod.internet.mgmt.mib.ip.ipAddrTable

  => 1.3.6.1.2.1.4.20

  ou encore

  ipAddrTable OBJECT-TYPE [...] ::= {ip 20}

Un ordre total est établi entre tous les noms d'objets administrables :

iso.org.dod.internet.mgmt.mib.ip.ipInReceives : 1.3.6.1.2.1.4.3
< iso.org.dod.internet.mgmt.mib.ip.ipAddrTable : 1.3.6.1.2.1.4.20
< iso.org.dod.internet.mgmt.mib.ip.ipAddrTable.ipAddrEntry.ipAddrEntAddr : 1.3.6.1.2.1.4.20.1.1
< iso.org.dod.internet.mgmt.mib.ip.ipAddrTable.ipAddrEntry.ipAddrEntIndex : 1.3.6.1.2.1.4.20.1.2

L'obtention des différentes instances successives (même en nombre variable) d'un objet utilise le sens de parcours défini par cet ordre.

3.8. Exemple de définition d'objet administrable

```
ip OBJECT IDENTIFIER ::= {mib-2 4}      -- La catégorie ip

ipInReceives OBJECT-TYPE           -- L'objet scalaire ipInReceives
    SYNTAX COUNTER                 -- son type
    Max-access READ-ONLY           -- droit d'acces
    Status Mandatory ...           -- ...
    ::= {ip 3}                     -- son OID
ipAddrTable OBJECT-TYPE            -- L'objet vectoriel ipAddrTable
    SYNTAX SEQUENCE OF ipAddrEntry -- composé de lignes de type ipAddrEntry
    ...                            -- ...
    ::= {ip 20}                    -- son OID
ipAddrEntry OBJECT-TYPE            -- L'objet correspondant aux lignes
    SYNTAX IpAddrEntry             -- son type
    ...                            -- ...
    INDEX {ipEntAddr}              -- l'index permet d'identifier chaque instance
    ::= {ipAddrTable 1}            -- son OID
ipAddrEntry ::= SEQUENCE{          -- Définition de l'enregistrement ipAddrEntry
    ipAdEntAddr ipAddress          -- adresse IP
    ipAdEntIfIndex INTEGER         -- l'entrée correspondante dans ifTable
    ipAdEntNetMask ipAdMask        -- masque de sous-réseau associé
    ipAdEntReasmMaxSize            -- taille du plus grand paquet possible
    ...
}
ipAdEntAddr OBJECT-TYPE            -- L'objet correspondant aux colonnes ipAdEntAddr
    SYNTAX    ipAddress
    ...
    ::= {ipAddrEntry 1}
```

3.9. L'identification des instances d'objets

Les objets peuvent avoir une ou plusieurs instances :

- scalaire (une seule instance) :

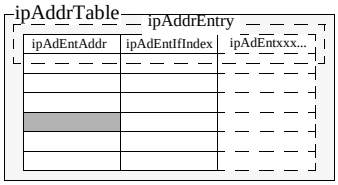
  - suffixé par 0 !

    Ex : ipInReceives = 1.3.6.1.2.1.4.20.3.0

- vecteur (plusieurs instances d'un même objet)

  - suffixé par l'index de l'entrée recherchée

    Ex : ipAddrEntAddr.5 = 1.3.6.1.2.1.4.20.1.1.5



L'ordre établi sur les objets est généralisé aux instances.

Nota : SNMP ne permet d'accéder qu'aux instances d'objets simples.

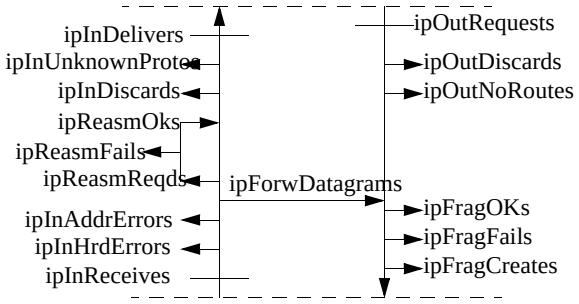
Nota : pas de tableaux de tableaux dans SNMP.

3.10. Les objets administrables du groupe d'objets IP

```
ip OBJECT IDENTIFIER ::= {mib-2 4}

ipForwarding     {ip 1}, Counter, RO, M, l'équip. fonctionne en tant que routeur
ipDefaultTTL     {ip 2}, Counter, RO, M, durée de vie par défaut des paquets
ipInReceives     {ip 3}, Counter, RO, M, nombre total de paquets (fragments) reçus
ipInHdErrors     {ip 4}, Counter, RO, M, nombre de paquets détruits à cause de l'entête
ipInAddrErrors   {ip 5}, Counter, RO, M, nombre de paquets détruits à cause de l'adresse
ipForwDatagrams  {ip 6}, Counter, RO, M, nombre de paquets routés
ipInUnknownProtos{ip 7}, Counter, RO, M, nombre de paquets détruits à cause de protocole inconnu
ipInDiscards     {ip 8}, Counter, RO, M, nombre de paquets détruits par manque de ressource (place)
ipInDelivers     {ip 9}, Counter, RO, M, nombre de paquets remis
ipOutRequests    {ip 10}, Counter, RO, M, nombre de paquets à émettre
ipOutDiscards    {ip 11}, Counter, RO, M, nombre de paquets détruits
ipOutNoRoutes    {ip 12}, Counter, RO, M, nombre de paquets détruits par routage déficient
ipReasmTimeout   {ip 13}, Counter, RO, M, nombre de paquets détruits lors du réassemblage
ipReasmReqds     {ip 14}, Counter, RO, M, nombre de paquets nécessitant un réassemblage
ipReasmOKs       {ip 15}, Counter, RO, M, nombre de paquets correctement réassemblés
ipReasmFails     {ip 16}, Counter, RO, M, nombre d'échecs lors du réassemblage
ipFragOKs        {ip 17}, Counter, RO, M, nombre paquets fragmentés avec succès
ipFragFails      {ip 18}, Counter, RO, M, nombre de fragmentations nécessaires mais impossibles
ipFragCreates    {ip 19}, Counter, RO, M, nombre de fragments émis
ipAddrTable      {ip 20}, Counter, NA, M, les différentes adresses IP de la station (cf ci-dessus)
ipRouteTable     {ip 21}, seq of ipRouteEntry, RO, M, la table de routage (remplacé par ipForward)
ipRouteDest      {ip 21.1.1}, ipAddress,RW,m, adresse de destination
ipRouteIfIndex   {ip 21.1.2}, INTEGER, RW, M, le numero d'interface sert d'index
ipRouteMetric1   {ip.21.1.3}, INTEGER, RW, M, la cout de la route
...
ipRouteNextHop   {ip 21.1.7}, ipAddress, RW, M, adresse du prochain routeur
...
ipNetToMediaTable{ip 22}, Counter, RO, M, la table de résolution d'adresses
...
<<M : mandatory (objet obligatoire)>>
```

Diagramme de Case pour le groupe IP :



4. Le protocole SNMP

4.1. Introduction

Le protocole SNMP : “Simple network management protocol”

Echange d’informations sur les objets des MIB

Opérations sans mémoire (“selfcontent message”) :

- stabilité, simplicité, flexibilité

Opérations atomiques :

- cohérence

Les opérations SNMP :

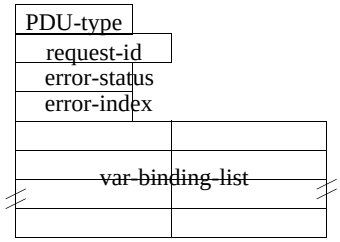
- get : demande d'obtention de la valeur d'une instance d’objet
- get-next : demande de la valeur de l’instance suivante
- response : réponse à une demande
- set : demande de stockage d'une valeur dans une instance d’objet
- get-bulk : get-next multiple
- trap : notification d'évènement
- inform : échange entre administrateurs

Les messages SNMP utilise le protocole UDP :

=> numéros de port 162 (trap) et 161 (autres)

4.2. Le format général des messages SNMP

Vision simplifiée :



. en ASN-1 !

```
SNMP-PDUs ::= CHOICE{
    get-request GetRequest-PDU,
    get-next-request GetNextRequest-PDU,
    get-bulk-request GetBulkRequest-PDU,
    get-response GetResponse-PDU,
    set-request SetRequest-PDU,
    inform-request InformRequest-PDU,
    trap Trap-PDU
}
max-bindings INTEGER ::= 2147483647 -- nombre maximum de variables dans la liste
```



```
PDU ::= SEQUENCE {
    request-id Integer32,
    error-status INTEGER {noError(0), tooBig(1), noSuchName(2), badValue(3), rea-
dOnly(4), genErr(5), noAccess(6), wrongType(7), wrongLength(8), wongEncoding(9),
wrongValue(10), noCreation(11), inconsistantValue(12), resourceUnavailable(13), co-
mitFailed(14), undofailed(15), authorizationError(16), noWritable(17), inconsisten-
tName(18)}, -- parfois inutilisé; 2,3,4 pour assurer la compatibilité SNMPv1
    error-index INTEGER (0..max-bindings), -- parfois inutilisé
    variable-bindings VarBindList
}
VarBindList ::= SEQUENCE (SIZE (0 .. max-bindings)) OF VarBind
VarBind ::= SEQUENCE {
    name ObjectName,
    CHOICE {
         value ObjectSyntax,
         unSpecified NULL,
         noSuchObject [0] IMPLICIT NULL,
         noSuchInstance [1] IMPLICIT NULL,
         endofMibView [2] IMPLICIT NULL
    }
}
```

Il existe plusieurs types de PDU :

- GetRequest-PDU, GetNext-PDU, GetResponse-PDU,SetRequest-PDU et Trap

RequestID :

- un type entier sur 4 octets

  - association entre demande (get, get-next, get-bulk, set, inform) et réponse (response)

ErrorStatus :

- type entier sur un seul octet
- initialisé à zéro lors d'une requête

  - la cause du mauvais déroulement de la demande

ErrorIndex :

- type d'entier sur un seul octet
- initialisé à zéro lors d'une requête

  - la variable de la liste qui a causé le mauvais déroulement de la demande

VarBindList :

- liste de variables
- chaque élément de la liste est un couple nom et valeur de la variable

  - liste des objets dont on veut obtenir la valeur
  - la valeur est “null” lors d’une requête

4.3. L’opération *get*

```
GetRequest-PDU ::= [0] IMPLICIT PDU
GetResponse-PDU ::= [2] IMPLICIT PDU
```

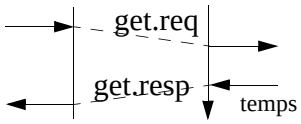
Enchaînement :

- Envoi d’une demande :

  - choix d’un identificateur unique de demande
  - error-status = noError
  - aucune, une ou plusieurs instances dans la liste des objets demandés, chacune accompagnée de la valeur *unSpecified*

- Envoi d’une réponse

  - le même identificateur d’objet
  - indication d’erreur si nécessaire
  - les objets demandés avec leur valeur si possible



- Exemple :

```
get.req({ipInReceives.0, unspecified})
=> get.resp[error-status=noError, error-index=0]({ipInReceives.0, 1237})
```

4.4. L'opération *get-next*

```
GetNextRequest-PDU ::= [1] IMPLICIT PDU
```

Enchaînement :

- similaire à l'opération get,
- mais la réponse fournit le nom (et la valeur) de la prochaine plus proche instance

  - parcours de l'arbre MIT (instances en nombre inconnu)
- Exemple :

```
get-next(ipRouteDest)
=> response [<noerr>]({ipRouteDest.0.0.0.0, 191.56.38.89})
get-next(ipRouteDest.0.0.0.0)
=> response [<noerr>]({ipRouteDest.192.33.4.0, 191.56.38.89})
get-next(ipRouteDest.192.33.4.0)
=> response [<noerr>]({ipRouteIfIndex, 125})
```

- possibilité d'accéder à des instances dont on ne connaît pas le nom a priori

  . Exemple :

```
get-next(0)
```


4.4.1 Trace get-next

On capture un requête SNMP à destination d'une machine

```
SNMP: len 38 version: int(1) 0x00 comm: string(6) public type: GET-NEXT
req-id: int(4) 0x00005e31 error: int(1) 0x00 error-index: int(1) 0x00
var: obj(7) 1 3 6 1 2 1 2 1 val: empty(0)
```

On capture une réponse SNMP en provenance de la machine

```
SNMP: len 40 version: int(1) 0x00 comm: string(6) public type: RESPONSE
req-id: int(2) 0x5e31 error int(1) 0x00 error-index: int(1) 0x00
var: obj(8) 1 3 6 1 2 1 2 1 0 val: int(1) 0x06
```

4.5. L'opération *get-bulk*

```
GetBulkRequest-PDU ::= [5] IMPLICIT Bulk-PDU
```

Extraction massive d'informations :

- minimisation du nombres d'échanges
- (n'existait pas en SNMPv1 !)

Opérations *get-next* multiples non-répété sur une 1ère liste d'objets, puis répété sur une 2ème.

Format du PDU compatible mais légèrement différent :

```
Bulk-PDU ::= SEQUENCE {
    request-id Interger32,
    non-repeaters INTEGER,          -- nb de variables non répétées
    max-repetitions INTEGER,        -- nb de répétition des variables répétées
    variable-bindings VarBindList
}
```

Exemple :

```
get-bulk[ non-repeaters=1, max-repetitions=4] (sysUpTime,ipNetToMediaPhyAddress, ipNetToMediaTy-
pe)
=> response [<<noerror>>]({sysUpTime.0,123456}, {ipNetToMediaPhyAddress.10.20.30.40,
010203040506}, {ipNetToMediaType.10.20.30.40, dynamic}, {ipNetToMediaPhyAddress.10.20.30.41,
010203040507}, {ipNetToMediaType.10.20.30.41, static}, {ipNetToMediaNetAddress.10.20.30.40,
10.20.30.40, {ipRouteDiscards.0, 2})
```


4.6. L'opération *set*

```
SetRequest-PDU ::= [3] IMPLICIT PDU
```

L'opération *set* ne réussit que si toutes les variables de la liste sont mises à jour :

- concept de validation en deux phases ("two step commitment")

  - vérification de chaque variable :
    - . la variable n'est pas accessible : `noAccess`
    - . la variable existe, l'instance ne peut pas être crée : `noCreation`
    - . la variable existe, la valeur de l'instance ne peut pas être modifiée : `notWritable`
    - . la valeur fournie est incorrecte : `wrongType, wrongLength, wrongEncoding, wrongValue`
    - . inconsistance : `inconsistentName, inconsistentValue`
    - . ressource indisponible : `resourceUnavailable`
  - validation de l'ensemble de l'opération :
    - . effective : `commit`
    - . incorrecte : `commitFailed, undoFailed`

Attention : le protocole de transport peut dupliquer ou déséquencer les demandes

- synchronisation entre entités d'administration coopérantes

  - verrou de synchronisation : l'objet snmpSetSerialNo de type TestAndIncr

4.7. L'opération *trap*

```
Trap-PDU ::= [4] IMPLICIT PDU
```

Une notification (trap) est générée d'un agent SNMP vers un administrateur SNMP :

- l'agent détecte un évènement extraordinaire,
- structure similaire aux PDU des autres opérations,
- mais pas de réponse !
- les deux premières variables de la liste sont spécifiques :

  - sysUpTime.0 : date d'apparition de l'évènement
  - snmpTrapOID.0 : identificateur d'objet identifiant l'évènement

Exemple :

```
trap ({sysUpTime.0, 67}, {snmpTrapOID.0, linkUP}, {ifIndex.7, 3})
```

Sept types de traps :

. démarrage à froid, démarrage à chaud, lien coupé, lien remarche, erreur d'au=thentification, perte de voisinange de routeur, spécifique à une application

4.8. L'opération *Inform*

```
Inform-PDU ::= [7] IMPLICIT PDU
```

Echange d'informations entre 2 administrateurs SNMP :

- structure similaire au PDU des autres opérations
- les deux premières variables de la liste sont spécifiques (comme l'opération Trap):

  - sysUpTime.0 : date d'apparition de l'évènement
  - snmpTrapOID.0 : identificateur d'objet identifiant l'évènement

Le module MIB *snmpv2-M2M* contient les objets qui déterminent les moments où un évènement se produit et quelles entités SNMP doivent recevoir l'opération *inform*.

4.9. L’encodage

La description des objets par ASN1 est volontairement conceptuelle (abstraite).

Pour transmettre les objets, il faut définir des règles d’encodage pour chaque type ASN1.

Plusieurs règles d’encodage peuvent être définies :

- BER est la plus commune

BER (“Basic Encoding Rules”) :

- technique d’encodage TLV : Type, Longueur, Valeur
- le premier octet code le type de l’objet :

  - à chaque type est associée une représentation standard
- le deuxième octet code la longueur du champ Valeur
- les octets suivants contiennent la valeur de l’objet

Exemple :

- Entier (complément à 2) sur 4 octets de valeur 17

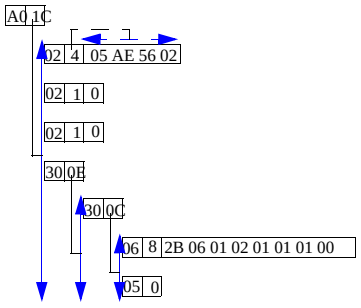
  - 0x020400000011

4.10. Exemple d’encodage d’une opération

Encodage du message SNMP GetReq(<SysDescr.0>) !

```
GetReq_PDU   len=28                    -- type dépendant du contexte spécifique
   A0          1C                      -- de l'application, ici : SNMP
     INTEGER    len=4     value_request_id
        02        4        05 AE 56 02
     INTEGER    len=1     value_errror_status
        02        1        00
     INTEGER    len=1     value_error_index
        02        1        00
     SEQUENCE   len=14              -- une liste …
        30        0E
        SEQUENCE   len=12          -- …. de couples
           30        0C
           Object_id     len=8   value_object=1.3.6.1.2.1.1.1.0   -- les deux 1er labels
              06          8      2B 06 01 02 01 01 01 00         -- sont encodés ensemble
           NULL          len=0
              05          0
```

La structure de données correspondante :



5. Contrôle d'accès et protection

5.1. Présentation

Les informations contenues par la MIB sont sensibles :

- identifier et contrôler des applications habilitées à effectuer certaines opérations d'administration
- identifier et contrôler des opérations d'administration applicables sur certains objets
- identifier et contrôler des objets accessibles et manipulables par certaines applications

Dans SNMPv1, la notion de communauté ("community"), identifiée par une chaîne de caractères, permettait de gérer simplement ces contrôles :

- une application connaissant le nom de la communauté, affectée à certaines opérations sur certains objets, pouvait effectuer ces opérations sur ces objets
- le nom de la communauté circulait en clair dans les messages SNMP !

SNMPv2 permet de protéger la MIB :

- authentification des intervenants
- confidentialité des informations échangées

SNMPv2, définit la notion de groupe ("party") qui comporte :

- des attributs de transport
- des attributs d'authentification
- des attributs de confidentialité

Les messages SNMP présentent 4 structures imbriquées qui permettent respectivement la mise en oeuvre :

- de la confidentialité (chiffrement)
- de l'authentification (signature numérique)
- du contrôle d'accès
- de la transmission des informations

5.2. La confidentialité

Le service de confidentialité assure qu'un tiers interceptant le message n'est pas capable de comprendre les informations échangées.

Principe :

- L'émetteur et le récepteur partagent un secret,
- l'émetteur chiffre ("crypte") le message à émettre en utilisant comme paramètre le secret,
- le message chiffré est transmis,
- seul le récepteur, partageant le même secret, est capable de déchiffrer le message.

Afin de protéger toutes les informations contenues dans le message, le chiffrement s'applique sur la totalité du message : en dernier à l'émission, et le déchiffrement : en premier à la réception.

SNMP propose d'utiliser optionnellement le service de confidentialité :

- pas de confidentialité : pas de chiffrement
  - privData == SnmpAuthMsg
- confidentialité : chiffrement
  - privData == chiffrement (SnmpAuthMsg)
  - l'algorithme de chiffrement (par défaut) :
    - DES-CBC ("Data Encryption Standard - Cypher Block Chaining")

5.3. L'authentification

Lorsque le service d'authentification est assuré, le récepteur du message a la certitude que :

- le vrai émetteur du message est celui annoncé (authentification proprement dit)
- le message reçu est celui qui a été émis (intégrité du message)

Principe :

- l'émetteur et le récepteur partagent un secret,
- l'émetteur produit un condensé du message (signature numérique) en utilisant un algorithme paramétré avec le secret,
- le message et le condensé sont transmis,
- le récepteur vérifie que le message reçu produit un condensé qui correspond au condensé.

L'authentification est optionnelle.

Parmi les méthodes d'authentification possibles :

- l'algorithme MD5 qui produit un condensé du message sur 16 octets.

Remarques :

- le message contient des informations identifiant l'émetteur.
- pour éviter le rejeu : l'horodatage est utilisée lors du calcul.

5.4. Structure des messages SNMP

4 structures imbriquées :

```
SnmpPrivMsg ::= [1] IMPLICIT SEQUENCE {
    privDst OBJECT IDENTIFIER,
    privData [1] IMPLICIT OCTET STRING       -- le message SnmpAuthMsg chiffré
}
SnmpAuthMsg ::= [1] IMPLICIT SEQUENCE {
    authinfo CHOICE {                        -- 2 algos d'authentification :
         noAuth OCTET STRING (SIZE(0)),      -- soit l'algo vide
         v2md5AuthProtocol [2] IMPLICIT SEQUENCE {-- soit l'algo MD5
              authDigest OCTET STRING (SIZE(16)),-- signature
              authDstTimestamp UIntenger32,
              authSrcTimestamp UInterger32
         }
    }
    authData SnmpMgmtCom                     -- message authentifié
}
SnmpMgmtCom ::= [2] IMPLICIT SEQUENCE {
    dstParty OBJECT IDENTIFIER,
    srcParty OBJECT IDENTIFIER,
    context OBJECT IDENTIFIER,
    operation SNMP-PDUs                      -- le PDU !
}
```


5.5. Droits d'accès

Définition de la liste des opérations autorisées :

- le nombre d'opérations possibles est limité
- la liste est codée par un entier (pour des raisons de compatibilité historique) :

  - chaque opération est codée par un multiple de deux :
    - get=1, get-next=2, response=4, set=8, get-bulk=32, inform=64, trap=128
  - la liste est codée comme la somme des codes de chaque opération autorisée
    - Exemple : 35 = get + get-next + get-bulk
- la liste vide (aucune opération autorisée) est codée 0.

5.6. Les vues de la MIB

Les objets administrables sont regroupés pour faciliter la gestion de leur droit d'accès :

- un vue regroupe plusieurs sous-arbres d'objets

Un sous arbre est défini par

- un identificateur d'objet qui indique l'origine du sous-arbre
  - ex. : 1.3.6.1.2.1.1 ({system})
- un masque qui sélectionne certains labels de l'identificateur
  - ex. : 1111111 que l'on note 'fe'H ou encore ''H

Une instance d'objet administrable appartient a un sous-arbre

- si le sous-arbre et l'instance ont des labels identiques pour tous les labels où le masque binaire est à un
  - ex. : 1.3.6.1.2.1.2.2.1.1.7({ifIndex.7}) ∉ 1.3.6.1.2.1.1 ({system})/1111111
  - ex. : 1.3.6.1.2.1.2.2.1.1.7({ifIndex.7}) ∈ 1.3.6.1.2.1.2 ({interfaces})/1111111
  - ex. : 1.3.6.1.2.1.2.2.1.1.7({ifIndex.7})∉1.3.6.1.2.1.2.2.1.0.5({ifEntry.0.5})/11111111101
  - ex. : 1.3.6.1.2.1.2.2.1.1.7({ifIndex.7})∈1.3.6.1.2.1.2.2.1.0.7({ifEntry.0.7})/11111111101

Une vue est définie par l'ensemble des sous-arbres qui la composent et l'ensemble des sous-arbres qui n'y appartiennent pas (qui en sont explicitement exclus).

6. Conclusion

SNMP

- Architecture générale :
  - client/serveur (administrateur/agent)
- Protocole simple, minimal et sans mémoire
  - implémentation efficace, cohérence assurée
- Des opérations simples sur les instances d'objets administrables :
  - obtenir une valeur (get, get-next, get-bulk/response), modifier la valeur (set/response), notifier l'apparition d'un évènement (trap) et s'informer entre admin. (inform)
- Protection :
  - authentification, confidentialité, contrôle d'accès

MIB

- Base de données générique et répartie
  - référençant tous les objets administrables, au sein d'une arborescence (MIT)
- Les objets et leurs instances sont identifiés par leur OID
  - le type des objets est défini à l'aide d'ASN-1
  - leur valeur est encodée par BER

7. L'administration de réseau de l'OSI

7.1. Gestion de la configuration

Principaux rôles :

- inventaire des ressources
- initialisation des équipements
- gestions des noms et des adresses
- mise à jour des paramètres des ressources

Procédures :

- collecter les informations
- contrôler l'état du système
- sauvegarder l'historique ("log")
- présenter l'état du système = synoptique

7.2. Gestion de la sécurité

Nécessaire pour protéger le réseau contre :

- un dysfonctionnement, une inadvertance, une malveillance

7.2.1 Attaques

Attaque passive :

- écoute de messages, observation du trafic

Attaque active :

- mascarade,
- duplication de message, modification de message,
- perturbation d'un service, modification d'un service.

7.2.2 Quelques mécanismes de protection

Enregistrement de l'activité des utilisateurs :

- les évènements significatifs, les actions interdites ou sensibles

Une base de données spécifique :

- Security MIB : rassemble (protège) les informations utilisées pour assurer la sécurité

Filtrage (“firewall”)

- trie des flux de données circulant entre deux parties du réseau (inter/intranet)
- sur les adresses IP, le sens d’entrée/sortie, le type du protocole, le numéro de port,
- les applications accessibles doivent être protégées

7.2.3 Principaux services

Authentification :

- de l’utilisateur de service (mot de passe), de l’émetteur de message (signature)
- notaire (tierce partie certifiante)

Intégrité :

- condensé du message (“digest”), signature numérique
- utilise une fonction (de hachage) paramétrée par un secret partagé entre émetteur et récepteurs

Confidentialité :

- chiffrement (cryptage) des messages
- système à clef secrète (symétrique) ou publique (asymétrique)
- chiffrement à la source ou par un serveur

7.3. Gestion des pannes

Défauts :

- systématique : panne d’un équipement, rupture d’un lien
- dépendant : fonction de l’état de l’environnement, congestion, etc.

Phase de traitement d’un défaut :

- détection d’un fonctionnement anormal
- localisation/diagnostic
- réparation
- vérification

Détection :

- messages d’erreur (quoi, qui(où), quand)
- tests (de contrôle routinier, de diagnostic)
- seuils (dénombrement des évènements)

Diagnostic :

- exploitation de l’historique (suite d’évènements, ensemble d’évènements)
- tests de diagnostic

Réparation :

- reconfiguration
- remplacement

Vérification :

- contrôle du retour à la normale

7.4. Audit des performances

Performances des ressources du réseau :

- délai, débit, taux d'erreur, disponibilité

Evaluation des performances effectuée à partir de mesures statistiques :

- Collecte,
- Contrôle,
- Stockage,
- Présentation

Analyse :

- Détection de comportements symptomatiques
- Prévision

7.5. Gestion de la comptabilité

Informations permettant d'évaluer le coût des communications.

- En fonction de la durée, du volume
- Au niveau du réseau ou de l'application

Exemple :

- nombre d'octets transmis, durée de connexion