MEMOIRE DE FIN D'ETUDES

Alexandre Fenyö

3A C.C.

Etude et implémentation d'outils SNMPv2

Juin 1994

3ième année - Option IDL

1. Préface

Ce mémoire est le résultat du travail que j'ai effectué de février 1994 à juin 1994 sur le protocole d'administration de réseau SNMP[1], et sur sa nouvelle forme, SNMPv2.

Ce mémoire s'adresse à des personnes ayant une connaissance de base dans le domaine des réseaux. De plus, nous allons souvent être amenés à parler du langage de description ASN.1[2] utilisé très largement dans les normalisations OSI. L'objet de ce mémoire n'étant pas de rappeler les définitions formelles de ce langage, nous utiliserons largement ASN.1 au cours de ce rapport, sans avoir défini au préalable les règles qui le régissent. Les lecteurs peu avertis quant à ASN.1 pourront néanmoins saisir les développements qui vont suivre, car sa lecture n'est pas une chose complexe, même s'il faut une bonne expérience afin de pouvoir écrire soi-même des documents suivant cette syntaxe.

Mes activités autour de SNMP ont été de types très variés, l'objectif étant d'acquérir une connaissance théorique ainsi qu' un savoir faire technique dans le domaine.

Le travail effectué peut être classé en 4 types d'activités :

1. Etude bibliographique sur le protocole SNMP
2. Découverte et utilisation de logiciels SNMP
3. Implémentation d'outils pour une utilisation simplifiée de SNMP
4. Implémentation de nouveaux objets SNMP

L'étude bibliographique a principalement consisté à analyser les dizaines de RFCs[3] relatifs aux modèles d'administration réseau et en particulier aux différentes versions successives de SNMP.

La découverte et l'utilisation de logiciels a consisté à rechercher différents logiciels domaine public implémentant SNMP, et à analyser leur fonctionnement. Les logiciels qui ont particulièrement retenu mon attention ont été *tricklet*, *xmib*, *scotty* et *tkined*. De plus, j'ai étudié les fonctionnalités du package ISODE déjà installé sur les machines du département Informatique, notemment les librairies perrmettant de développer des agents SNMP, et le module gawksnmp, une extension de gawk permettant d'interroger de manière conviviale les agents SNMP.

L'implémentation d'outils pour une utilisation simplifiée de SNMP a consisté à implémenter un service d'accès à des ensembles de variables SNMP via le service World Wide Web. J'ai pour cela écrit une interface W3 permettant d'interroger des agents SNMP en utilisant gawksnmp.

L'implémentation de nouveaux objets a consisté à définir un ensemble de variables SNMP, et à implémenter un agent servant ces variables, dans le but d'administrer une entité locale à une machine. Les entités administrables sont souvent des périphériques, tels les disques ou les imprimantes. Le choix ici s'est porté sur la conception d'un agent SNMP permettant

---

[1]Simple Network Management Protocol
[2]Abstract Syntax Notation One
[3]Request For Comments, documents définissant les protocoles des réseaux du modèle ARPANET.

d'administrer un manager local du système de gestion de mémoire partagée en Distributed C[4]. Avant d'implémenter ce manager, il a fallu, comme toujours avec SNMP, définir en ASN.1. un ensemble de variables organisé, appelé MIB. Par la suite, quand nous aurons besoin d'exemples afin de commenter des définitions, nous utiliserons le module MIB DCC-MIB qui a été conçu pour DCC.

Remarque préliminaire : dans ce mémoire, le terme *SNMP* désignera indifférement la première et la deuxième version du protocole. Lorsqu'on voudra parler précisément de l'une ou de l'autre, on utilisera explicitement la notation *SNMPv1*, ou *SNMPv2*.

[4]Il s'agit ici du système de gestion de mémoire partagée SHM DCC, objet du mémoire de Samuel Tardieu. En SHM DCC, un manager local est une entité qui offre les services distribués de mémoire partagée aux différents processus qui s'exécutent sur la machine locale.

2. Analyse de SNMP

2.1. Historique

Le début des années 80 a été marqué par la prolifération de réseaux d'interconnexion de machines. Les différents fournisseurs de ressources réliées par ces réseaux (machines, passerelles, ponts, périphériques divers tels les imprimantes ou les scanners), ont commencé par fournir leurs propres outils d'administration. Ainsi, par exemple, chaque type de passerelle devait être configurée par un protocole ou une méthode propriétaire. Très vite, il est apparu qu'une méthode simple et unifiée d'administration des ressources distribuées sur un réseau s'avérait nécessaire, afin de diminuer la charge et la complexité de l'administration réseau.

En 1987, trois voies indépendantes se sont créées pour répondre à ce besoin. Il s'agissait de :

. HEMP : High-level Entity Management Protocol
. SGMP : Simple Gateway Monitoring Protocol
. CMOT : Common Management Information Protocol over TCP

HEMP fut très rapidement abandonné. SGMP était plus simple à implémenter que CMOT, mais CMOT, basé sur CMIP, donc très proche de l'architecture OSI de gestion d'informations réseau, était considéré comme supérieur. Mais à cette époque, il y avait peu de tentatives d'implémentation de CMIP, et des développements poussés dans ce sens semblaient nécessaires avant qu'il puisse être adopté par tous.

Il fut ainsi décidé d'adopter une solution mixte pour résoudre les problèmes d'administration de réseaux : SGMP devait continuer à être développé en vue d'obtenir une implémentation efficace et utilisable à court terme, tandis que parallèlement, la recherche autour de CMOT devait permettre d'aboutir à une solution finale pour le long terme.

Mais jusqu'à maintenant, CMOT/CMIP est toujours très peu implanté, alors que SGMP a su évoluer successivement sous les formes SNMPv1, puis SNMP Security et enfin SNMPv2, protocoles implémentés par de nombreux constructeurs.

2.1.1. Simple Gateway Monitoring Protocol

Comme son nom l'indique, ce protocole a été créé dans l'unique but de gérer l'augmentation croissante des passerelles qui interconnectent différents réseaux TCP/IP. Les premières idées concernant SGMP sont apparues en mars 1987. Sa conception a pris deux mois, et encore deux mois plus tard, il était implémenté sur de nombreuses plateformes.

2.1.2. Simple Network Management Protocol

En février 1988, l'IAB[5] décida de résoudre le problème de la coexistence des trois protocoles d'administration de réseaux cités précédement. Il forma donc un groupe de travail sur la question, qui proposa d'abandonner définitivement HEMP, et de faire évoluer SGMP en donnant à ses nouvelles versions le nom de Simple Network Management Protocol. SNMP

[5]Internet Architecture Board

étant appeler à terme à être remplacé par CMIP, un modèle commun d'administration de réseaux devait être développé, afin de faciliter la transition.

Six mois après le premier meeting du groupe de travail formé par l'IAB, les spécifications initiales de l'Internet-standard Network Management Framework[6] étaient mises en place, au sein de trois RFCs :

. RFC-1065 *Structure and Identification of Management Information for TCP/IP-based Internets*
Ce RFC ne décrit ni les objets gérés, ni le protocole en lui même. Il donne les bases générales du SMI[7] : mécanisme utilisé pour décrire et nommer les objets.

. RFC-1066 *Management Information Base for Network Management of TCP/IP-based internets*
Ce RFC décrit précisément la MIB[8].

. RFC-1067 *A Simple Network Management Protocol*
Ce RFC décrit le protocole SNMPv1, utilisé pour accéder via le réseau, aux objets administrés.

Depuis, il y a eu de nombreuses modifications à ce modèle initial, et la version courante de SNMPv1 est principalement décrite par les trois RFC suivants:

. RFC-1155 *Structure and Identification of Management Information for TCP/IP-based Internets*
Modifications mineures du RFC-1065.

. RFC-1213 *Management Information Base for Network Management of TCP/IP-based internets: MIB-II*
Description de la seconde version de la MIB.

. RFC-1157 *A Simple Network Management Protocol (SNMP)*
Modifications mineures du RFC-1067.

De plus, viennent s'ajouter à ces documents, quatre autres RFCs qui viennent compléter le modèle :

. RFC-1212 *Concise MIB Definitions*
Définition d'un mécanisme de description de modules de la MIB, toujours compatible avec le SMI.

. RFC-1215 *A Convention for Defining Traps for use with the SNMP*
Définition de conventions pour décrire les traps utilisées par SNMP. Ainsi, les traps ne sont pas fixées dans le modèle, seules quelques traps y sont définies, et, grâce à ce RFC, les constructeurs peuvent définir d'autres traps suceptibles d'être émises par leurs machines.

---

[6]nous le dénomerons par la suite *modèle d'administration*, ou tout simplement *modèle*
[7]Structure of Management Information : le modèle d'administration
[8]Management Information Base : l'ensemble des objets controlés par SNMP

. RFC-1303 *A Convention for Describing SNMP-based Agents*
Ce RFC fournit une convention pour définir, à l'aide d'ASN.1, les caractéristiques d'un agent SNMP, par exemple les branches de la MIB qu'il supporte, ou ses restrictions au modèle.

. RFC-1354 *IP Forwarding Table MIB*
Modifications de la table IP-Forwarding.

Le modèle continue par être étendu par de nombreuses MIBs.

2.1.3. Simple Network Management Protocol Security

La principale lacune dans SNMPv1 était le manque de sécurité dans le protocole. Chaque message SNMPv1 est accompagné d'une chaine de caractères, appellée communauté[9]. L'agent SNMPv1 décide des droits d'accès du manager qui a envoyé ce message en fonction de cette communauté. Ainsi, on retrouve le principe du mot-de-passe. Mais comme la chaine définissant la communauté est écrite en clair dans le paquet SNMPv1 échangé entre les deux entités SNMPv1, n'importe quel espion réseau peut la récupérer et acquérir du même coup des droits usurpés.

L'IETF[10] décida donc de créer un groupe de travail, dont le but était d'ajouter des notions de sécurité et d'identification au sein de SNMPv1, mais en modifiant le moins possible le protocole. Trois documents en résultèrent :

. RFC-1351 *SNMP Administrative Model*
Définition du modèle d'administration d'entités SNMP dans une optique sécurisée.

. RFC-1352 *SNMP Security Protocols*
Définition des protocoles de sécurités pouvant être utilisés dans le modèle.

. RFC-1353 *Definitions of Managed Objects for Administration of SNMP Parties*
Définitions d'une MIB servant à définir les droits et les protocoles utilisés dans les transactions entre entités SNMP.

SNMP Security a été pris de cours par les évènements : au moment de son achèvement, à un meeting de l'IETF en juillet 1992, il fut décidé que les déficiences en matières de sécurité seraient résolues dans une nouvelle version du protocole, et que le successeur de SNMPv1 devraient inclure les nouvelles extensions concernant la sécurité. Ainsi, les RFCs 1351, 1352 et 1353 sont maintenant devenus obsolètes depuis que les RFCs décrivant SNMPv2 sont parus.

2.1.4. Simple Network Management Protocol version 2

Plus les expériences d'implémentation se succédèrent, et plus les déficiences dans le modèle purent être mises à jour. Ainsi, au meeting de l'IETF d'aout 1991, une session spéciale fut réservée à l'accumulation d'idées quant aux déficiences du modèle. Puis, au meeting de

---

[9]Community
[10]Internet Engineering Task Force

l'IETF de juillet 1992, une proposition pour un nouveau modèle, appelé SMP[11], fut soumise, accompagnée de quatre implémentations indépendantes.

A ce même meeting, une deadline pour le 10 septembre 1992 fut fixée, afin de récupérer d'autres propositions. Au jour de la deadline, aucune autre proposition n'avait été reçue. Il fut donc décidé de faire entrer le SMP dans le processus de standardisation des protocoles de l'Internet. Il devait ainsi en résulter un nouveau protocole, Simple Network Management Protocol version 2 - SNMPv2. Ce protocole est décrit dans les douze documents suivants :

. RFC-1441 *Introduction to the Version 2 of the Internet-standard Network Management Framework*
. RFC-1442 *Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)*
. RFC-1443 *Textual Conventions for version 2 of the Simple Network Management Protocol (SNMPv2)*
. RFC-1444 *Conformance Statements for version 2 of the Simple Network Management Protocol (SNMPv2)*
. RFC-1445 *Administrative Model for version 2 of the Simple Network Management Protocol (SNMPv2)*
. RFC-1446 *Security Protocols for version 2 of the Simple Network Management Protocol (SNMPv2)*
. RFC-1447 *Party MIB for version 2 of the Simple Network Management Protocol (SNMPv2)*
. RFC-1448 *Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2)*
. RFC-1449 *Transport Mappings for version 2 of the Simple Network Management Protocol (SNMPv2)*
. RFC-1450 *Management Information Base for version 2 of the Simple Network Management Protocol (SNMPv2)*
. RFC-1451 *Manager-to-Manager Management Information Base*
. RFC-1452 *Coexistence between SNMPv1 and SNMPv2*

Il est à noter que les RFCs décrits dans les sections précédentes ont permis la transition vers la définition de SNMPv2; la plupart des idées nouvelles qu'ils précisaient font maintenant partie de SNMPv2.

[11]Simple Management Protocol

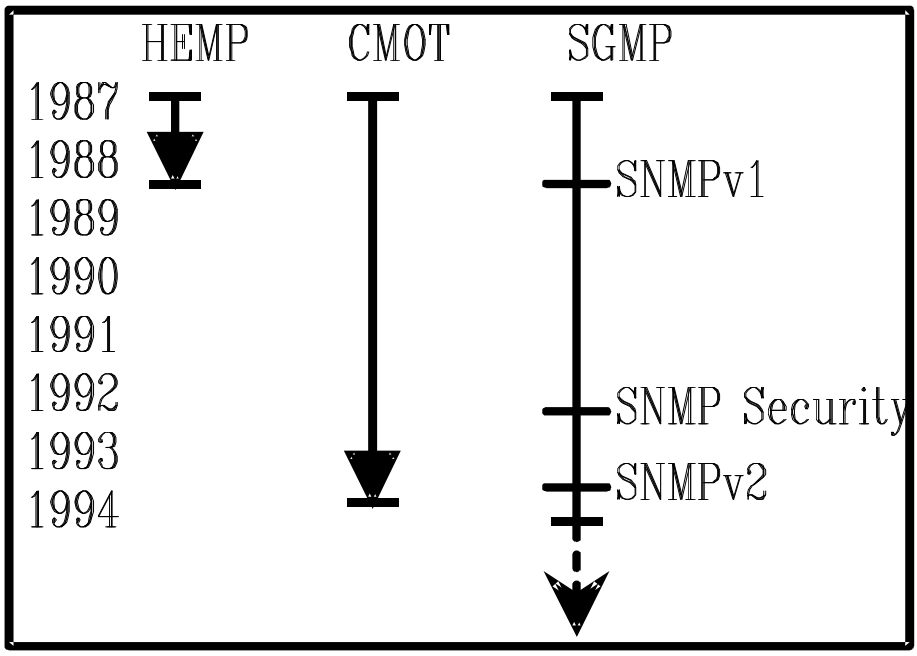


Figure 1, évolution des protocoles de gestion de réseaux

2.2. Limites dans SNMPv1

Les spécifications de SNMPv1 conduisent à considérer un ensemble de données, les MIBs[12], administrées par des entités appelées agents, qui fournissent le contenu de ces données à des entités appelées managers. Les MIBs sont des ensembles de valeurs organisées au sein d'une arborescence, décrite à l'aide d'ASN.1. Chaque branche de cet arbre porte un numéro, et à chaque noeud est associé un identificateur d'objet[13], qui correspond à la suite des numéros des branches successives qu'il faut suivre pour accéder à ce noeud en partant de la racine de l'arbre. La racine de l'arbre s'appelle iso et son OID est 1; un de ses noeuds s'appelle internet et son OID est 1.3.6.1, ce qui veut dire que pour l'atteindre, on doit sélectionner la troisième branche de la racine, puis depuis le noeud ainsi atteint, on doit sélectionner la sixième branche, et enfin la première.

Chaque objet est typé, les types étant définis à partir de types primitifs en ASN.1. Il peut ainsi y avoir par exemple des objets complexes de type tableau de structures. Supposons que l'objet 1.3.6.100 soit une tableau d'une structure comprenant trois champs, et que ses différents champs soient constitués des objets 1.3.6.100.1, 1.3.6.100.2 et 1.3.6.100.3. Supposons de plus que ces trois champs soient de types respectifs adresse IP, entier, et chaine de caractères; supposons enfin que la table 1.3.6.100 soit indexée sur son premier champs, c'est à dire sur une adresse IP. Alors, la valeur du troisième champs, pour la ligne de la table d'index 137.194.160.1, sera 1.3.6.100.3.137.194.160.1. On voit ainsi comment adresser les différentes valeurs des lignes d'une table.

SNMPv1, en plus de définir ainsi tout un ensemble de variables, fournit un protocole d'échange entre les agents et les managers. Il contient quatre opérations, qui sont *Get*, *Get-Next*, *Set* et *Trap*. Les trois premières sont générées par les managers à destination des agents SNMP, alors que la dernière est utilisée par les agents. *Get* permet de récupérer les valeurs d'un ensemble d'objets, *Get-Next* est utilisé pour parcourir les objets de type table, *Set* permet

---

[12]MIB : Management Information Base
[13]Object Identifier ou OID

de modifier la valeur d'un objet, et *Trap* est utilisée par les agent pour prévenir les managers en cas de problème grave.

Différentes limitations sont rapidement apparues à l'utilisation de la première version du protocole. On y trouve des problèmes dus à la sécurité, de la perte de bande passante, le manque de moyen de communication entre les différents managers SNMP.

2.2.1. La sécurité

Le format d'un message SNMPv1 est, en langage ASN.1, défini de la manière suivante :

```
Message ::= SEQUENCE {
                version INTERGER { version-1(0) }
                community OCTET STRING,
                data PDUs
                }
```

Format des messages SNMPv1

La seule notion de sécurité réside dans le champs *community*, qui contient une chaine censée identifier la provenance du message, et donc informer l'agent SNMP sur les opérations permises. Cette chaine correspond, en quelque sorte, à un mot-de-passe, qui est écrit en clair dans le paquet SNMP. On peut ainsi distinguer cinq différents dangers éventuels, dus au protocole SNMPv1[14] :

Analyse du traffic Les paquets n'étant pas cryptés, un intervenant extérieur, qui réussirait à écouter sur le média de transport des paquets SNMPv1, serait parfaitement en mesure de connaitre les différentes opérations soumises par les managers aux agents SNMP. Par exemple, dans un routeur CISCO, des objets de la MIB CISCO décrivent la localisation du fichier de configuration du routeur au moment du boot, notemment le protocole pour le récupérer (TFTP[15] par exemple), l'adresse IP de la machine où le fichier est maintenu, et la localisation du fichier sur cette machine. Si un utilisateur mal intentionné récupère cette information en analysant des paquets SNMP, il saura quelle machine attaquer afin d'agir indirectement sur le routeur en question.

Séquencement des messages Les messages sont la plupart du temps encapsulés dans un transport en mode non connecté (par exemple sur une couche UDP[16] dans le cadre des réseaux du modèle ARPANET), donc sans notion de séquence. Vu qu'il n'y a pas de notion de séquencement dans les messages SNMP, et que le transport est habituellement en mode non connecté, il se peut que les messages dans des paquets différents arrivent dans un ordre inverse de l'émission. Avec un transport UDP, cela arrive souvent, les paquets étant routés individuellement, et les routes

[14]RFC-1446
[15]Trivial File Transfer Protocol
[16]User Datagram Protocol

dans le modèle ARPANET étant suceptibles d'être variables en fonction de l'engorgement de certains routeurs. Les opérations SNMP prises dans un ordre différent peuvent amener à des comportements non voulus. Par exemple, un manager peut vouloir redémarrer un CISCO en lui faisant charger un fichier de configuration de test. Il doit pour cela envoyer un ordre SNMP pour fixer la variable MIB qui maintient le nom du fichier de configuration au boot, et un autre ordre pour rebooter la machine. Si ces deux ordres n'arrivent pas en séquence, mais sont intervertis, le CISCO redémarrera avec l'ancien fichier de configuration !

Altération de l'information Une passerelle intermédiaire, séparant un agent et un manager, peut parfaitement modifier le contenu d'un message, suite à une erreur de logiciel, par exemple. Cette erreur ne sera pas détectée à la réception, vu qu'il n'y a pas de contrôle de cohérence du paquet, telle une checksum.

Coupure du service Les paquets SNMP étant encapsulés dans la couche transport sous-jacente, le disfonctionnement d'une passserelle intermédiaire, ou une action d'un utilisateur du réseau mal intentionné, peut empécher les échanges entre des entités SNMP.

Identification SNMPv1 ne fournit pas de moyen pour identifier la provenance d'un paquet SNMP. Seule la communauté est suceptible de fournir une information sur la provenance. Avec un transport UDP, il est possible de connaitre la provenance du paquet, en recherchant l'adresse IP de la machine émettrice. Mais une passerelle intermédiaire peut parfaitement changer l'adresse IP source du paquet pour simuler une autre provenance.

2.2.2. Perte de bande-passante

Le protocole SNMPv1 ne fournit que deux opérateurs pour récupérer de l'information. Il s'agit de *get-request* et *get-next-request*. L'opération get-request récupère le contenu de toutes les variables dans une liste de correspondances variable/valeur fournie, alors que l'opération get-next-request récupère le contenu de tous les successeurs directs dans la MIB des variables de la liste de correspondance contenue dans le paquet. Ainsi, un manager qui veut récupérer le contenu entier d'une table de la MIB, doit nommer chaque entrée de la table qu'il veut récupérer. Il ne peut pas les mettre toutes dans la liste de correspondances d'un même PDU[17] car la plupart du temps, le manager ne connait pas l'index de la table.

Par exemple, la table de la MIB-II décrivant la table de routage d'un agent est indexée sur des adresses IP, le manager n'en a donc pas connaissance. Il doit donc utiliser des opérations get-next-request afin de découvrir successivement les indexes. Si la table est un tableau de structures contenant C champs, et si la table contient N lignes, il faudra donc que le manager génère N+1 PDUs contenant chacun une table de C correspondances, et l'agent

[17]Protocol Data Unit :paquet encapsulant une opération du protocole

génèrera en retour N+1 PDUs. Il aura donc fallu échanger 2N+1 PDUs pour récupérer le contenu d'une table contenant N lignes.

2.2.3. Pas de communication entre managers

Dans le cas de l'administration de larges réseaux, découpés en plusieurs sous-réseaux, il peut paraitre souhaitable, pour des problèmes de bande passante, et d'efficacité, d'utiliser plusieurs managers pour administrer l'ensemble du réseau. On peut par exemple prévoir un manager par sous réseau, ce qui permet d'économiser de la bande passante, mais les informations récupérées par les managers ne pourront pas être globalisées. On peut aussi prévoir plusieurs managers, chacun se chargeant d'un domaine restreint de la MIB, mais pour toutes les machines du réseau. Là aussi, l'information aura du mal à être globalisée, même si elle sera plus organisée que dans le cas précédent, mais il y aura une énorme perte de bande passante au niveau réseau, car chaque manager devra joindre chaque agent du réseau. L'absence de communication entre les différents managers est donc un frein à l'utilisation de SNMPv1.

2.2.4. Insuffisances dans les structures

La MIB du modèle SNMPv1 contient des objets maintenus par des agents SNMP, mais peu de données abstraites concernant le modèle. Il n'y a pas d'objets dans les MIBs pour décrire les marges laissées aux implémenteurs, par exemple ce qu'il est obligatoire d'implémenter et ce qui peut ne pas l'être, tout en restant conforme au modèle. Il n'y a pas non plus de moyen d'écrire dans une MIB les possibilités réelles d'un agent, et notemment ses variations face au protocole, ce qu'il implémente et ce qu'il laisse de côté.

2.2.5. Limitations dans les opérations du protocole

Les opérations sur les différentes lignes d'une table manquent d'atomicité. On ne peut par exemple pas créer une ligne nouvelle, dans un état temporaire, puis y positionner les différents paramètres, et enfin l'activer pour que le périphérique géré par l'agent puisse s'en servir. Ainsi, si un manager crée une ligne, puis en positionne les différents éléments, il se peut qu'entre temps un autre manager ait demandé le contenu de cette ligne et ait ainsi obtenu des informations incohérentes.

De plus, on a déjà vu qu'il y a de l'utilisation inutile de bande passante dans la récupération du contenu des tables. Un opérateur plus approprié que l'opérateur get-next-request serait ainsi utile.

2.2.6. Limitation du nombre de codes d'erreurs

Le nombre limité de codes d'erreur[18] les rend souvent très ambigüs. Cela oblige, dans la majorité des cas, l'entité qui les reçoit à renvoyer successivement des parties de la requête initiale afin de déterminer précisément la racine du problème.

2.3. SNMPv2

2.3.1. Structure des informations gérées.

[18]SNMPv1 définit seulement 6 codes d'erreur : noError, noSuchName, badValue, readOnly et genErr.

Le SMI[19] décrit la structure des informations gérées, c'est-à-dire les méta-informations qui vont permettre de décrire les ensembles d'objets au sein des MIBs. Il est décrit au sein du RFC-1442[20] et est divisé en trois parties : définitions pour les modules, définitions pour les objets, et définitions pour les traps.

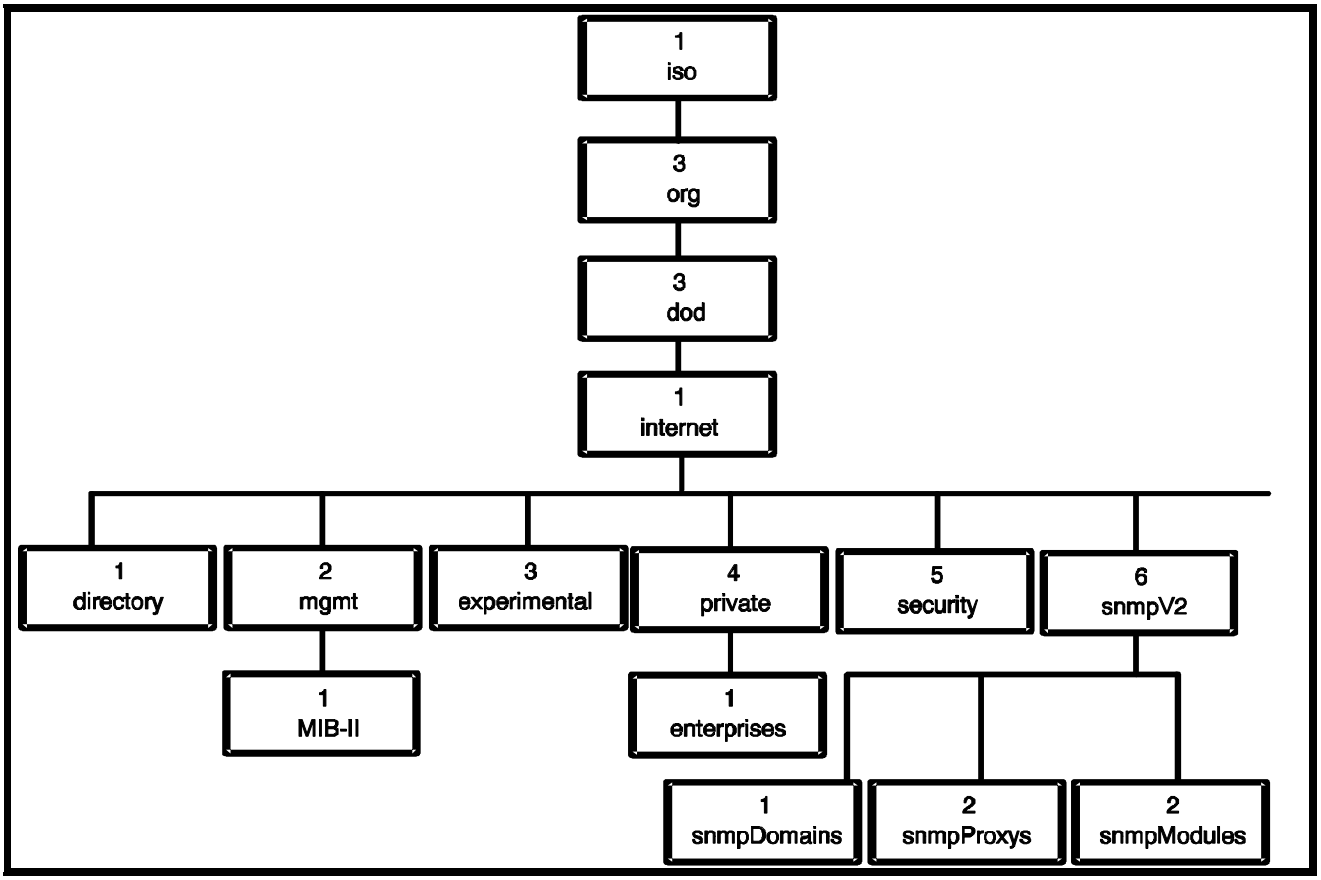
Les définitions pour les modules sont utilisées pour fournir un ensemble d'informations sur un module. Pour cela, une macro ASN.1, MODULE-IDENTITY, est utilisée pour préciser la sémantique d'un module.

Les définitions pour les objets sont utilisées pour fournir un ensemble d'informations sur les objets réellement administrés par les entités SNMP. Pour cela, une macro ASN.1, OBJECT-TYPE, est utilisée pour préciser la sémantique des objets.

Les définitions pour les traps sont utilisées pour décrire la sémantique d'informations d'administration non solicitées. Une macro ASN.1, NOTIFICATION-TYPE, est prévue à cet effet.

2.3.1.1. Base de la MIB

Le SMI définit en langage ASN.1 la racine de la MIB.



Base de l'arbre des d'identificateurs d'objets

---

[19]Structure of Management Information
[20]Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)

2.3.1.2. Définitions de types

Le SMI définit les types de base à partir desquels on pourra dériver les autres types. Ils sont classés dans la catégorie *simpleSyntax* :

. INTEGER (2^{31} .. 2^{31}-1)
. OCTET STRING
. OBJECT IDENTIFIER
. BIT STRING

De plus, d'autres types, dérivés des précédents, sont fournis dans la catégorie ApplicationSyntax :

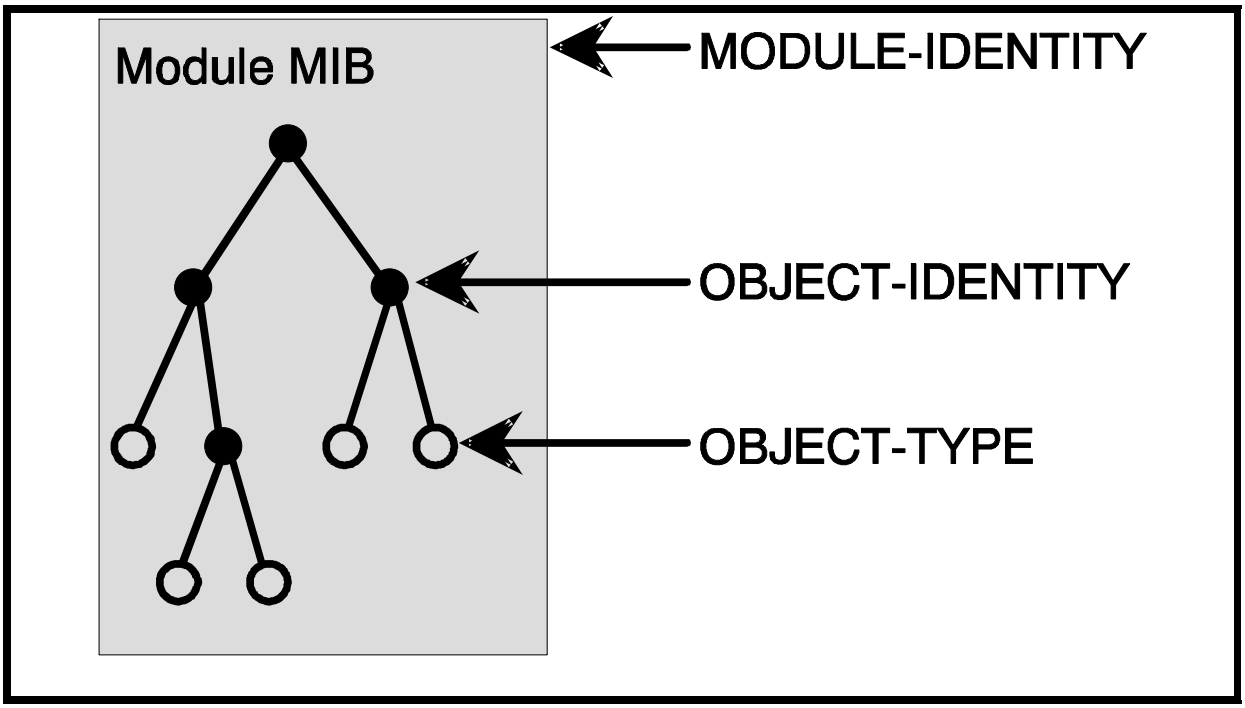
Type	Définition	Remarque
IpAddress	OCTET STRING (SIZE(4))	Adresse IP
Counter32	INTEGER (0 .. 2^{32})	Compteur en boucle
Gauge32	INTEGER (0 .. 2^{32})	Compteur
TimeTicks	INTEGER (0 .. 2^{32})	Centièmes de secondes
Opaque	OCTET STRING	Compatibilité ascendante
NsapAddress	OCTET STRING (SIZE(1|4 .. 21))	Adresses OSI
Counter64	INTEGER (0 .. 2^{64})	Compteur en boucle
UInteger	INTEGER (0 .. 2^{32})	Entier non signé

Types de la catégorie Application Syntax

2.3.1.3. Macros SNMPv2

Le SMI définit quatres macros ANS.1 pour définir des informations sur les modules MIBs :

. la macro **MODULE-IDENTITY** permet de définir un module MIB;
. la macro **OBJECT-IDENTITY** permet de définir les branches d'un arbre d'identificateurs d'objets;
. la macro **OBJECT-TYPE** permet de définir les branches d'un arbre d'identificateur d'objets;
. la macro **NOTIFICATION-TYPE** permet de définir des groupes d'objets, utilisés dans le protocole lors du transfert d'informations non solicitées.



2.3.1.3.1. Macro MODULE-IDENTITY

```
MODULE-IDENTITY MACRO ::=
BEGIN
        TYPE NOTATION ::=
                "LAST-UPDATED" value(Update UTCTime)
                "ORGANIZATION" Text
                "CONTACT-INFO" Text
                "DESCRIPTION" Text
                RevisionPart

        VALUE NOTATION ::=
                value(VALUE OBJECT IDENTIFIER)

        RevisionPart ::=
                Revisions
                | empty
        Revisions ::=
                Revision
                | Revisions Revision
        Revision ::=
                "REVISION" value(Update UTCTime)
                "DESCRIPTION" Text

        -- uses the NVT ASCII character set
        Text ::= """" string """"
END
```

Définitions ASN.1 de la macro MODULE-IDENTITY - RFC-1442[21]

Quatre clauses sont définies au sein de cette macro :

[21]Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)

LAST-UPDATED Cette clause définit la date de dernière modification apportée au module.

ORGANISATION Cette clause fournit des informations sur l'organisation qui a développé ce module.

DESCRIPTION Cette clause fournit des informations sur les objectifs et le contenu de ce module.

REVISION Cette clause, qui est optionnelle, permet de garder une trace des différentes modifications apportées au module.

Pour exemple, voici l'utilisation de cette macro au sein du module DCC-MIB dont nous avons parlé précedemment :

```
enst MODULE-IDENTITY
    LAST-UPDATED "9407030000Z"
    ORGANIZATION "Ecole Nationale Superieure des
                  Telecommunications de Paris"
    CONTACT-INFO "Alexandre Fenyo fenyo@enst.fr / Phone: +33
                  (1) 45.81.71.70"
    DESCRIPTION "The MIB module for DCC local memory manager."
::= { enterprises 69 }
```

2.3.1.3.2. Macro OBJECT-IDENTITY

```
OBJECT-IDENTITY MACRO ::=
BEGIN
        TYPE NOTATION ::=
                "STATUS" Status
                "DESCRIPTION" Text
                ReferPart

        VALUE NOTATION ::=
                value(VALUE OBJECT IDENTIFIER)

        Status ::=
                "current"
                | "obsolete"

        ReferPart ::=
                "REFERENCE" Text
                | empty

        Text ::= """ string """
END
```

Définitions ASN.1 de la macro OBJECT-IDENTITY - RFC-1442[22]

Trois clauses sont définies au sein de cette macro :

STATUS Cette clause précise si l'objet est obsolète.

[22]Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)

DESCRIPTION Cette clause permet de décrire l'object à l'aide d'une chaine de caractères.

REFERENCE Cette clause, optionnelle, permet de fournir une référence croisée pointant vers un objet défini dans un autre module.

2.3.1.3.3. Macro OBJECT-TYPE

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

        VALUE NOTATION ::=
                value(VALUE ObjectName)


        UnitsPart ::=
                "UNITS" Text
                | empty

        Access ::=
                "not-accessible"
                | "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

        Index ::=
```

```
                -- use the SYNTAX value of the
                -- correspondent OBJECT-TYPE invocation
                value(Indexobject ObjectName)

        Entry ::=
                -- use the INDEX value of the
                -- correspondent OBJECT-TYPE invocation
                value(Entryobject ObjectName)

        DefValPart ::=
                "DEFVAL" "{" value(Defval Syntax) "}"
                | empty

        -- uses the NVT ASCII character set
        Text ::= """" string """"
END
```

Définitions ASN.1 de la macro OBJECT-TYPE - RFC-1442[23]

Neuf clauses sont définies au sein de cette macro, mais seulement huit pourront apparaître simultanément, car les clauses INDEX et AUGMENTS ne peuvent être utilisées toutes les deux à la fois.

SYNTAX Cette clause définit le type des données contenues dans cet objet.

MAX-ACCESS Cette clause définit les modes d'accès autorisés pour cet objet.

STATUS Cette clause précise si cet objet est obsolète.

DESCRIPTION Cette clause fournit une description textuelle de l'objet.

UNITS Cette clause, optionnelle, décrit l'unité dans laquelle les valeurs contenues dans cet objet doivent être interprétées.

REFERENCE Cette clause, optionnelle, permet de fournir une référence croisée pointant vers un objet défini dans un autre module.

INDEX Cette clause, optionnelle, permet, lorsque l'on définit une ligne d'une table, de préciser les objets qui vont l'indexer.

AUGMENTS Cette clause, optionnelle, qui ne peut être utilisée s'il existe déja une clause INDEX, permet, lorsque l'on définit une ligne d'une table, d'identifier la ligne dont celle-ci la complète.

DEFVAL Cette clause, optionnelle, permet de fournir une valeur par défaut, utilisée au moment de la création de l'instance.

Pour exemple, voici l'utilisation de cette macro au sein du module DCC-MIB :

```
dccLoad OBJECT-TYPE
    SYNTAX INTEGER (1..2147483647)
```

[23]Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)

```
    UNITS ".001 * number of jobs in the run queue averaged over 1 minute"
    MAX-ACCESS read-write
    STATUS current
    DESCRIPTION
        "Load average."
    REFERENCE "See SunOS 4.x manyal page about w."
    ::= { dccLoadAverage 2 }
```

2.3.1.3.4. Macro NOTIFICATION-TYPE

```
NOTIFICATION-TYPE MACRO ::=
BEGIN
        TYPE NOTATION ::=
                ObjectsPart
                "STATUS" Status
                "DESCRIPTION" Text
                ReferPart

        VALUE NOTATION ::=
                value(VALUE OBJECT IDENTIFIER)

        ObjectsPart ::=
                "OBJECTS" "{" Objects "}"
                | empty
        Objects ::=
                Object
                | Objects "," Object
        Object ::=
                value(Name ObjectName)

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

        ReferPart ::=
                "REFERENCE" Text
                | empty

        -- uses the NVT ASCII character set
        Text ::= """" string """"
END
```

Définitions ASN.1 de la macro NOTIFICATION-TYPE - RFC-1442[24]

Une notification est un groupe d'objets qui sont d'un intérêt particulier quand un certain type d'évènement se produit. Quatre clauses sont définies au sein de cette macro :

STATUS Cette clause indique si la définition est obsolète.

DESCRIPTION Cette clause fournit une déscription textuelle de la notification.

OBJECTS Cette clause fournit la liste des objets associés à cette notification.

---

[24]Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)

REFERENCE Cette clause, optionnelle, permet de fournir une référence croisée pointant vers un objet défini dans un autre module.

2.3.1.4. Conventions textuelles

Le RFC-1443[25] décrit une nouvelle manière de créer des types évolués. Quand on construit un module MIB, il est souvent utile de définir de nouveaux types similaires à ceux déjà définis dans le SMI. En comparaison de ces derniers, chacun de ces nouveaux types possède un nom différent, une syntaxe similaire mais une sémantique plus riche. Ces nouveaux types sont nommés *conventions textuelles*, et sont utilisés pour le confort des individus lisant les définitions de modules MIB.

Un macro particulière, TEXTUAL-CONVENTION, est utilisée pour préciser la syntaxe et la sémantique d'une convention textuelle.

```
TEXTUAL-CONVENTION MACRO ::=
BEGIN
        TYPE NOTATION ::=
                DisplayPart
                "STATUS" Status
                "DESCRIPTION" Text
                ReferPart
                "SYNTAX" type(Syntax)

        VALUE NOTATION ::=
                value(VALUE Syntax)

        DisplayPart ::=
                "DISPLAY-HINT" Text
                | empty

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

        ReferPart ::=
                "REFERENCE" Text
                | empty

        -- uses the NVT ASCII character set
        Text ::= """" string """"
END
```

Définitions ASN.1 de la macro TEXTUAL-CONVENTION - RFC-1443

Cinq clauses sont définies au sein de cette macro :

DISPLAY-HINT Cette clause, optionnelle, indique aux applications la manière d'afficher les valeurs de ce type.

STATUS Cette clause précise si la définition est obsolète.

---

[25]Textual Conventions for version 2 of the Simple Network Management Protocol (SNMPv2)

DESCRIPTION Cette clause indique précisément le type de cette convention textuelle.

REFERENCE Cette clause, optionnelle, permet de fournir une référence croisée pointant vers un objet défini dans un autre module.

SYNTAX Cette clause précise la structure abstraite correspondant à cette convention textuelle.

Un ensemble de conventions textuelles prédéfinies sont fournies dans le RFC-1443 :

Nom	Syntaxe	Description
DisplayString	OCTET STRING (SIZE (0..255))	Information textuelle
PhysAddress	OCTET STRING	Adresse couche physique
MacAddress	OCTET STRING (SIZE (6))	Adresse 802.x
TruthValue	INTEGER { true(1), false (2) }	Valeur Booléenne
TestAndIncr	INTEGER (0 .. 2147483647)	Pour les opérations atomiques
AutonomousType	OBJECT IDENTIFIER	Valeur identifiant un type extensible
InstancePointer	OBJECT IDENTIFIER	Pointeur vers une ligne d'une table
RowStatus	INTEGER { active(1), notInService(2), notReady(3), createAndGo(4), createAndWait(5), destroy(6) }	Utilisé dans les phases de création et d'élimination des lignes dans les tables
TimeStamp	TimeTicks	Valeur de sysUpTime
TimeInterval	INTEGER (0 .. 2147483647)	délai mesuré en centièmes de seconde
DateAndTime	OCTET STRING (SIZE(8| 11))	Indication de la date et de l'heure

Convention textuelles définies dans le RFC-1443

TimeStamp permet de préciser la valeur de sysUpTime lorsqu'un évènement particulier s'est produit. *RowStatus* permet l'atomicité lors les créations de lignes dans les tables.

Pour exemple, voici l'utilisation de la macro TEXTUAL-CONVENTION dans le module DCC-MIB :

```
ProcessCount ::= TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
        "A number of processes matching a specified condition."
    SYNTAX INTEGER (1..2147483647)
```

2.3.2. Concepts d'administration

Les concepts d'administration dans SNMP sont composés du model administratif[26], de la party MIB[27], et des protocoles de sécurité[28].

Le modèle administratif permet de définir les controles d'accès, les méthodes d'authentification et de cryptage.

---

[26]RFC-1445: Administrative Model for version 2 of the Simple Network Management Protocol (SNMPv2)
[27]RFC-1447 : Party MIB for version 2 of the Simple Network Management Protocol (SNMPv2)
[28]RFC-1446 : Security Protocols for version 2 of the Simple Network Management Protocol (SNMPv2)

La party MIB définit les objets nécessaires au modèle administratif et aux protocoles de sécurité.

Les protocoles de sécurité définissent les protocoles actuellement utilisables avec SNMPv2; dans l'avenir, d'autres protocoles pourront venir s'y ajouter.

2.3.2.1. Modèle administratif

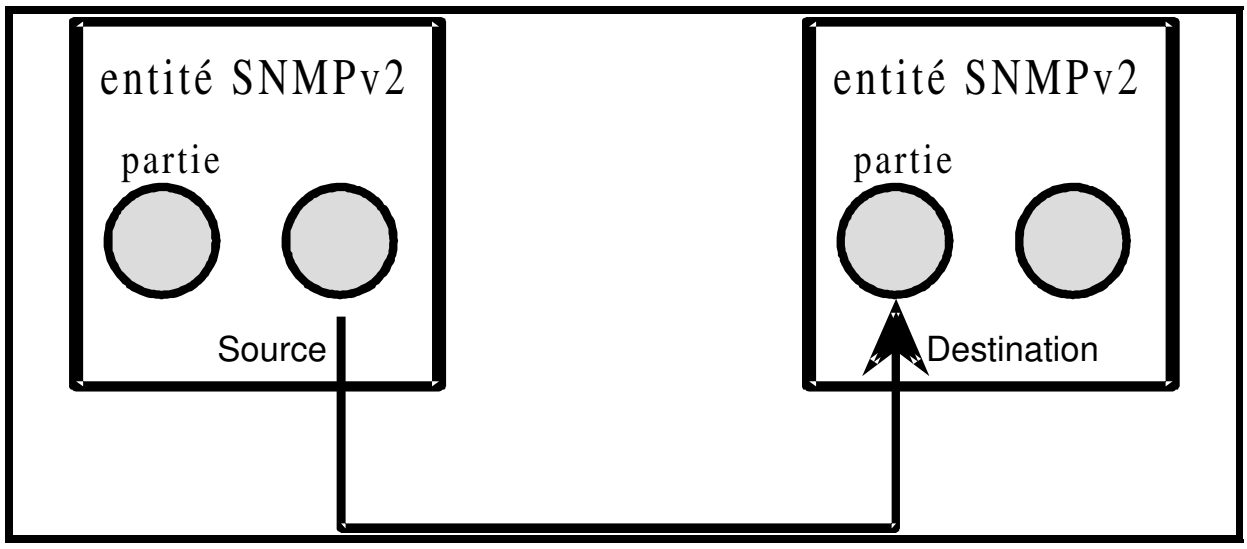
Nous allons ici décrire le modèle qui régit le comportement d'une entité SNMP face aux messages qu'elle est amenée à traiter. Quatre mécanismes coopérent pour ce faire. Il s'agit du mécanisme des parties, des contextes, des vues et des contrôles d'accès.

2.3.2.1.1. Mécanisme des parties

Une partie est un contexte d'exécution virtuel au sein duquel les opérations sont restreintes, pour des raisons de sécurité ou autres. A chaque fois qu'une entité SNMPv2 s'occupe d'un message, elle le fait en agissant en tant que partie, et est ainsi restreinte aux opérations autorisées pour cette partie.

Chaque partie SNMPv2 contient :

- un identificateur de partie unique;
- une localisation dans le réseau de l'endroit où s'exécute la partie, caractérisé par un protocole de transport et une adresse relative à ce transport;
- un protocole d'authentification;
- un protocole de cryptage.



Le concept des parties

2.3.2.1.2. Mécanisme des contextes

Tandis que le mécanisme des parties se focalise sur la transmission d'une opération, le mécanisme des contextes est un environnement d'exécution virtuel qui se focalise sur l'exécution des opérations. Il existe deux types de contextes : les contextes locaux et les contextes distants. Les contextes locaux servent à une entité à créer un environnement

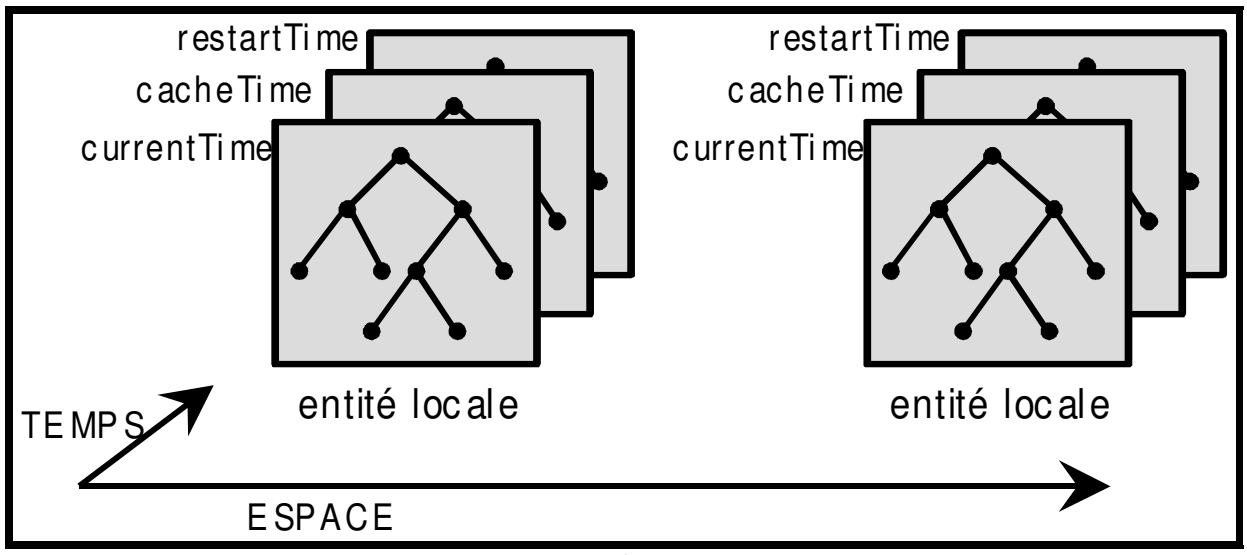
d'exécution virtuel pour le traitement d'une opération, tandis que les contextes distants servent à une entité à demander à une entité distante de se placer dans le contexte d'exécution voulu.

Quand un contexte est mis en place, il sert à effectuer l'opération localement ou via une relation proxy.

Si les données doivent être accédées via une relation proxy, le contexte fournit une nouvelle partie locale, une nouvelle partie distante et un nouveau contexte pour l'opération.

Si les données doivent être accédées localement, alors le contexte fournit une des duplications virtuelles de la MIB. Pour cela, deux champs particuliers dans la définition d'un contexte servent à préciser une position dans l'espace et dans le temps, vis-à-vis de la MIB. La position dans le temps est définie par le champs contextLocalTime, qui précise l'identificateur d'un objet dans le sous-arbre temporalDomains de la MIB des parties. Quant à la position dans l'espace, il s'agit du champs contextLocalEntity, qui précise l'entité locale qui est en relation avec ce contexte.

Attention : plusieurs entités locales peuvent être administrées par un seul agent SNMPv2.



Le concept des contextes

2.3.2.1.3. Mécanisme des vues

Le mécanisme des vues régit les objets accessibles dans les opérations SNMP. Une vue correspond à un masque placé sur une MIB. Souvent, les vues permettent de distinguer les différents modules administrés par une même entité.

2.3.2.1.4. Mécanisme des contrôle d'accès

Afin de fournir les règles du contrôle d'accès, une table de contrôle d'accès est définie. Elle est indexée sur une partie source, une partie destination, et un contexte. Elle précise les opérations permises lorsque ces trois éléments sont réunies.

2.3.2.2. La base de donnée d'administration - Party MIB

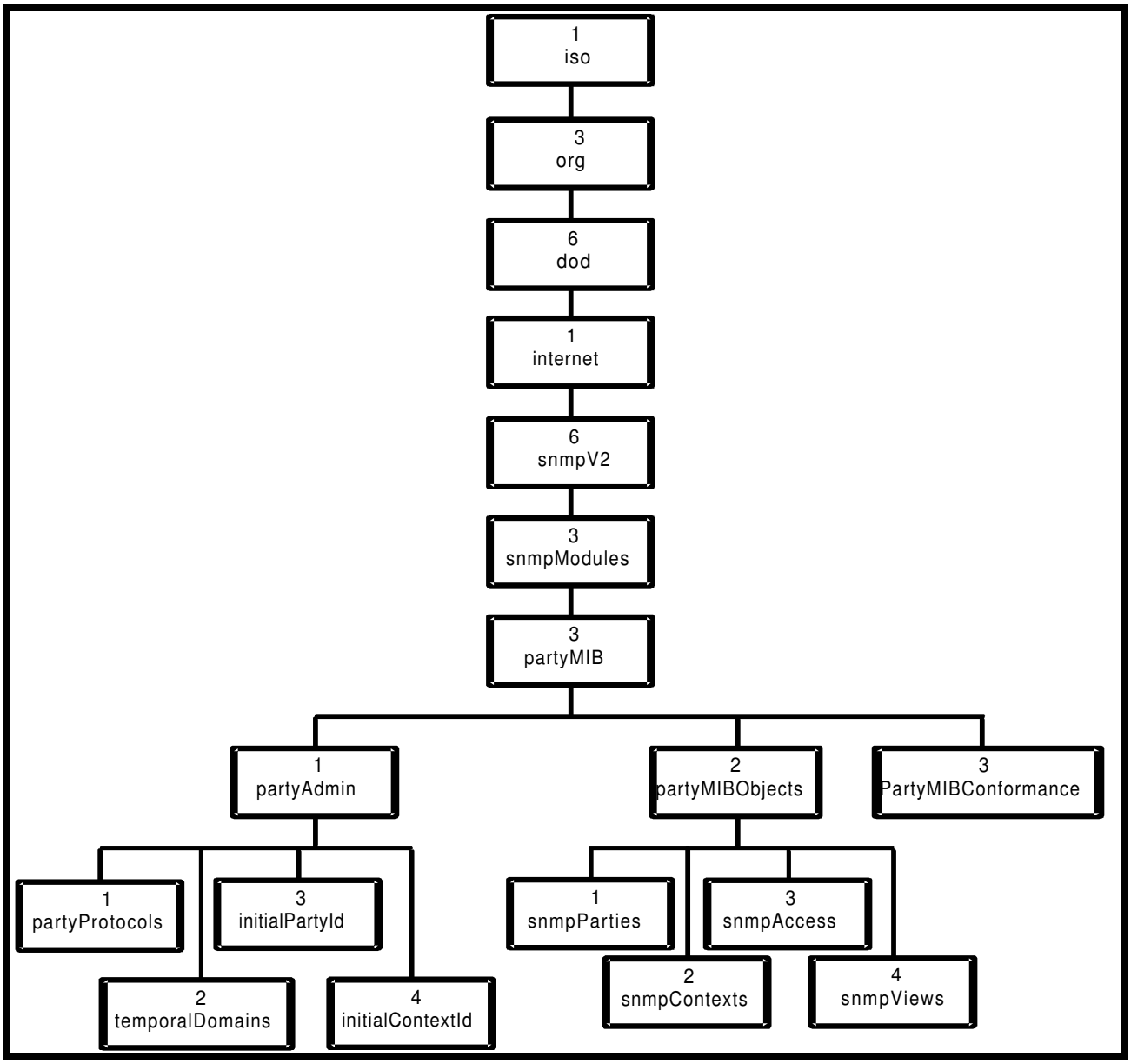
Une module MIB particulier a été défini afin de permettre d'organiser les informations concernant le modèle d'administration. Il fournit des conventions textuelles et une MIB qui s'appelle la Party MIB. Cette dernière regroupe les différentes tables qui coopèrent pour la cohérence du modèle :

. Table des parties Cette table, *partyTable*, est indexée sur un identificateur de partie unique, *partyIdentity*. Ses champs principaux renseignent sur le domaine de transport *partyTDomain* (par exemple UDP), une adresse *partyTAddress* (par exemple adresse IP et numéro de port UDP), un protocole d'authentification *partyAuthProtocol* (par exemple MD5), un protocole de cryptage *partyPrivProtocol* (par exemple DES). Les entrées de cette table décrivent les agents et les managers.

. Table des contextes Cette table, *contextTable*, est indexée sur un identificateur de contexte unique, *contextIdentity*. Ses champs principaux dépendent de la valeur du champs *contextViewIndex*. S'il est non nul, il référence alors une vue dans la table des vues, et fournit donc un masque sur les données accessibles dans ce contexte. Les champs *contextLocalEntity* et *contextLocalTime* renseignent alors sur le domaine spatial et temporel associés à cette vue. Si *contextViewIndex* est nul, ce contexte correspond à une relation proxy, dont la source, la destination, et le contexte distants sont fournis par les champs *contextProxyDstParty*, *contextProxySrcParty*, et *contextProxyContext*.

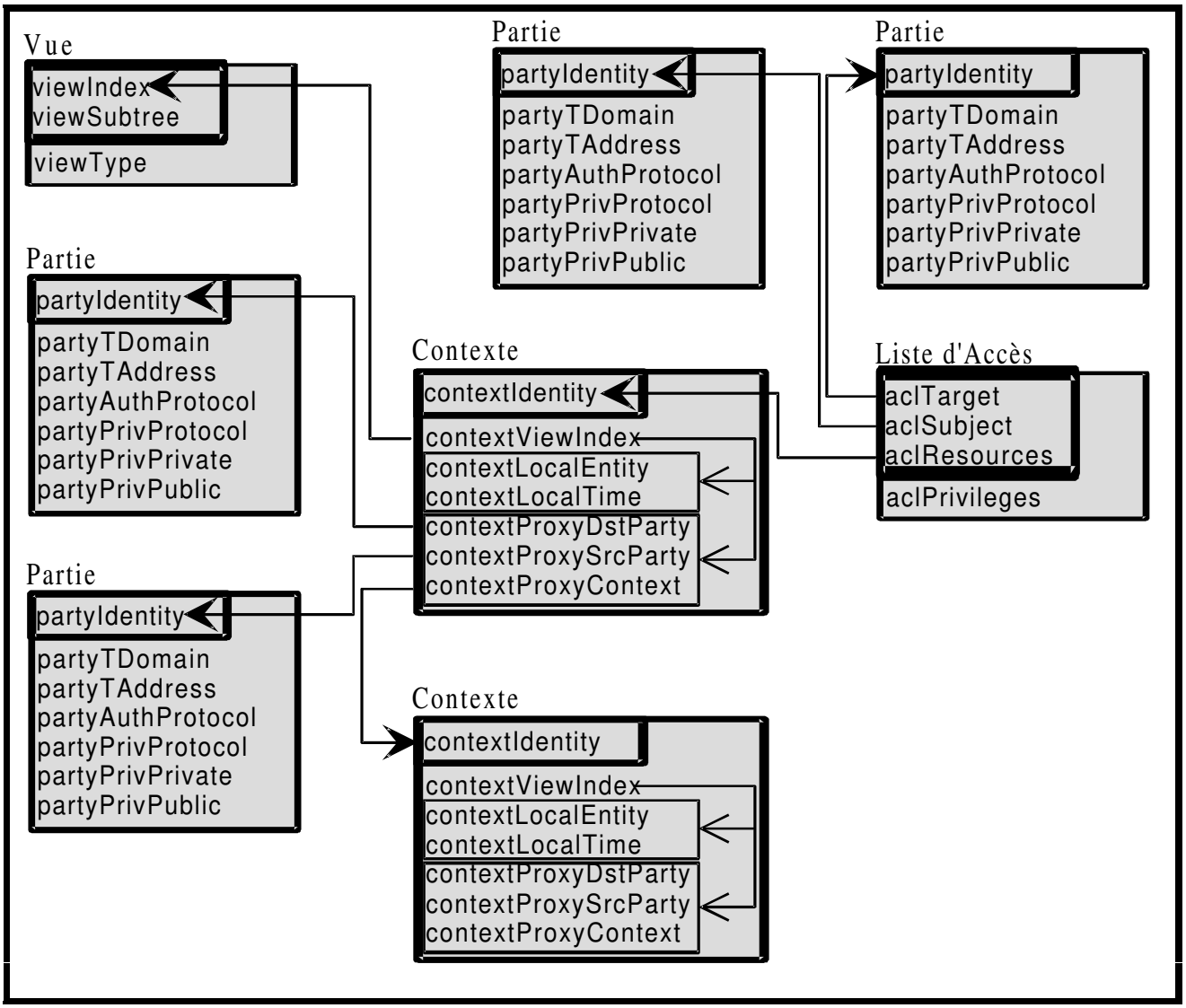
. Table des vues Cette table, *viewTable*, est indexée à la fois par un identificateur de vue unique, *viewIndex*, et par un identificateur de sous arbre *viewSubTree*. Une vue est ainsi constitué d'un ensemble de sous arbres. Pour chacun, on précise à l'aide de *viewType* s'il est inclus ou exclus de la vue.

. Table des accès Cette table, *aclTable*, est indexée par trois champs, *aclTarget* et *aclSubject* qui pointent vers des entrées de la table des parties, et *aclResources* qui pointe vers une entrée de la table des contextes. Elle contient de plus un champs *aclPrivileges* qui précise les opérations autorisées pour les messages envoyés depuis *aclTarget* vers *aclSubject* dans le contexte *aclResources*.



Vue partielle de la Party-MIB

Les champs des différentes tables pointent vers des indexes dans les autres tables. Ainsi, ces quatre tables sont complètement liées entre-elles.



Références croisées entre les différentes tables

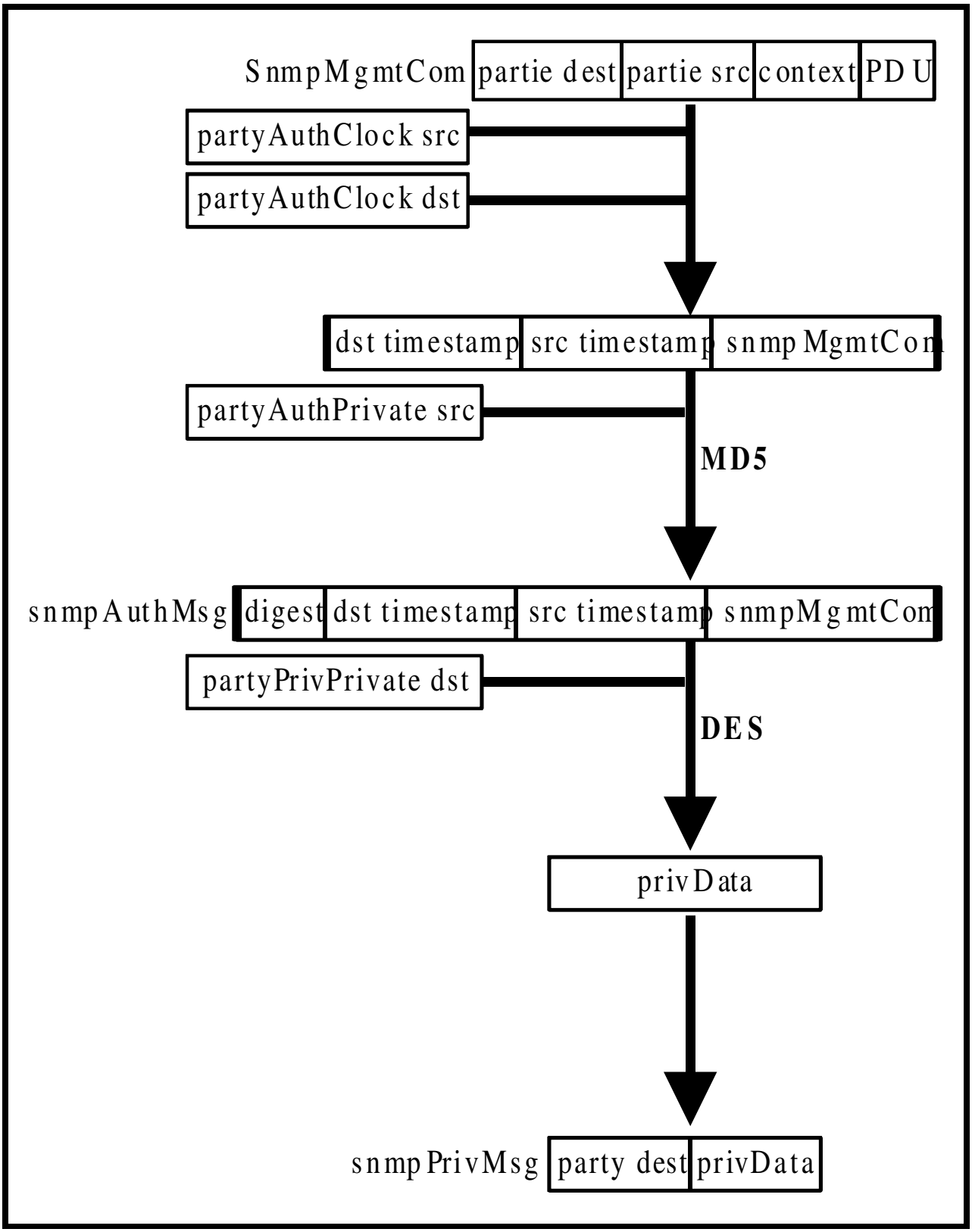
2.3.3. Sécurité

Nous avons précedemment vu que la version 1 de SNMP posait différents problèmes de sécurité, notemment en ce qui concernait l'absence de moyen d'identification de l'émetteur d'un message, la possibilité d'inversion dans le séquencement des paquets et l'absence de cryptage des paquets. Ces trois dangers ont été éliminés au sein de SNMPv2.

L'origine d'un message est décrite par une partie, et sa destination par une autre partie. La partie source d'un message fournit le protocole d'identification, et la partie destination fournit le protocole de cryptage. Les données relatives à ces deux protocoles sont situées dans les tables des parties, au sein de la base de donnée locale à chaque entité source et destination. Ces données comprennent notemment la description des protocoles utilisés, ainsi que les données necessaires à leur implémentation, c'est à dire le contenu des clefs publiques et privées pour chacune des parties en jeu.

Aucun protocole particulier n'est imposé pour l'identification ou pour le cryptage, mais SNMPv2 propose d'utiliser MD5 pour l'identification et DES pour le cryptage.

Le danger concernant le mauvais séquencement des paquets est éliminé en incluant des timestamps dans les paquets.



Opérations sur un PDU avant émission

2.3.4. Opérations du protocole

Les opérations du protocole, définies au sein du RFC-1448, sont au nombre de cinq. Il s'agit de *GetRequest*, *GetNextRequest*, *GetBulkRequest*, *SetRequest,* et *InformRequest*. Les types de messages échangés sont au nombre de sept. Il s'agit de *GetRequest-PDU*, *GetNextRequest-PDU*, *GetBulkRequest-PDU*, *SetRequest-PDU*, *Response-PDU*, *InformRequest-PDU* et *SNMPv2-Trap-PDU*.

2.3.4.1. Les cinq opérations

Les opérations sont générées par les managers, en direction des agents SNMP, sauf pour InformRequest, qui est générée par un manager, en direction d'un autre manager. Il existe trois opérations d'interrogation, ce sont les opérations de type Get, et une opération de positionnement, l'opération SetRequest. De plus, il y a une opération de transmission d'information, InformRequest.

GetRequest Cette opération génère un message de type *GetRequest-PDU*, qui contient une liste d'identificateurs d'objets, et reçoit en retour un message *Response-PDU* qui contient la liste des identificateurs ainsi que leur valeur dans le contexte du message.

GetNextRequest Cette opération génère un message de type *GetNextRequest-PDU*, qui contient une liste d'identificateurs d'objets, et reçoit en retour un message *Response-PDU* qui contient, pour chacun des identificateurs, son successeur lexicographique direct, ainsi que la valeur qui lui est affectée dans le contexte du message. Cette opération est utilisée pour récupérer le contenu des tables dont le manager ne connait pas le contenu de l'index. Par exemple, une table de routage indexée sur les adresses IP des réseaux destination, peut être récupérée par un manager qui ne connait pas les adresses réseau destination, en utilisant GetNextRequest.

GetBulkRequest Cette opération génère un message de type *GetBulkRequest-PDU*, et reçoit en retour un message *Response-PDU*. Par cette opération, on peut récupérer les valeurs d'une suite de successeurs lexicographiques d'identificateurs d'objets. Cela revient au même résultat qu'une suite de message GetNextRequest, mais la bande passante utilisée est réduite.

SetRequest Cette opération génère un message de type *SetRequest-PDU*, et reçoit en retour un message *Response-PDU*. Cela permet de positionner les valeurs d'un ensemble d'identificateurs d'objets.

InformRequest Cette opération génère un message de type *InformRequest-PDU*, qui contient une liste d'identificateurs d'objets, ainsi que leurs valeurs, et reçoit en retour un message de type *Reponse-PDU*. Cette opération est utilisée dans les communications entre managers.

2.3.4.2. Les sept types de messages

Les sept types de messages ont été définis en ASN.1 de la manière suivante :

```
GetRequest-PDU      ::= [0] IMPLICIT PDU
GetNextRequest-PDU  ::= [1] IMPLICIT PDU
```

```
Response-PDU        ::= [2] IMPLICIT PDU
SetRequest-PDU      ::= [3] IMPLICIT PDU
GetBulkRequest-PDU  ::= [5] IMPLICIT BulkPDU
InformRequest-PDU   ::= [6] IMPLICIT PDU
SNMPv2-Trap-PDU     ::= [7] IMPLICIT PDU
```

Les six premiers messages de la liste précédente sont utilisés pour les cinq opérations dont on a parlé précédemment. Le septième, SNMPv2-Trap-PDU, est utilisé par les agents, afin de transmettre aux managers des informations non sollicités concernant d'éventuels problèmes qu'il viennent de rencontrer.

Le type PDU[29], qui sert dans six des messages précédents, est défini en ASN.1 de la manière suivante :

```
max-bindings INTEGER ::= 2147483647

PDU ::=
        SEQUENCE {
                request-id Integer32,

                error-status                -- sometimes ignored
                        INTEGER {
                                noError(0),
                                tooBig(1),
                                noSuchName(2),
                                badValue(3),
                                readOnly(4),
                                genErr(5),
                                noAccess(6),
                                wrongType(7),
                                wrongLength(8),
                                wrongEncoding(9),
                                wrongValue(10),
                                noCreation(11),
                                inconsistentValue(12),
                                resourceUnavailable(13),
                                commitFailed(14),
                                undoFailed(15),
                                authorizationError(16),
                                notWritable(17),
                                inconsistentName(18)
                        },

                error-index                 -- sometimes ignored
                        INTEGER (0..max-bindings),

                variable-bindings           -- values are sometimes ignored
                        VarBindList
        }
```

Le champs *request-id* permet d'identifier la requête qui a généré un PDU de type Response-PDU. Le champs *error-status* permet d'avoir une information sur une eventuelle

[29]Protocol Data Unit

erreur, en relation avec l'identificateur d'objet d'index *error-index* dans la liste des associations *variable-bindings*.

Le type GetBulkRequest-PDU, utilisé dans les requêtes de type GetBult-Request, est défini en ASN.1 de la manière suivante :

```
BulkPDU ::=                     -- MUST be identical in
        SEQUENCE {              -- structure to PDU
                request-id Integer32,
                non-repeaters INTEGER (0..max-bindings),
                max-repetitions INTEGER (0..max-bindings),
                variable-bindings       -- values are ignored
                        VarBindList
        }
```

Ce type de PDU permet de récupérer le contenu d'une table, sans perdre de la bande passante, comme cela se produit si l'on utilise un PDU de type GetNextRequest-PDU.

Les deux types de PDU précédents font référence à une liste d'associations, de type VarBindList, permettant de relier des identificateurs d'objets à leurs valeurs. Ce type est défini en ASN.1 de la manière suivante :

```
VarBind ::=
        SEQUENCE {
                name ObjectName,

                CHOICE {
                        value ObjectSyntax,
                        unSpecified             -- in retrieval requests
                                NULL,

                        -- exceptions in responses
                        noSuchObject[0] IMPLICIT NULL,

                        noSuchInstance[1] IMPLICIT NULL,

                        endOfMibView[2] IMPLICIT NULL
                }
        }

VarBindList ::= SEQUENCE (SIZE (0..max-bindings)) OF VarBind
```

Les PDUs de type SNMPv2-Trap-PDU sont générés lors des traps. Les deux premiers identificateurs d'objets de la liste d'association sont sysUpTime.0 - le temps depuis lequel l'entité est en fonctionnement -, et snmpTrapOID.0, dont la valeur correspond à l'OID[30] de la notification correspondante, cette notification ayant été définie en ASN.1 avec la macro NOTIFICATION-TYPE. Les objets du champs OBJECTS de cette macro sont ensuite placés dans la liste d'associations, ainsi que leurs valeurs.

Le RFC-1450 définit un ensemble de traps par défaut, à l'aide de la macro NOTIFICATION-TYPE. Il s'agit de :

---

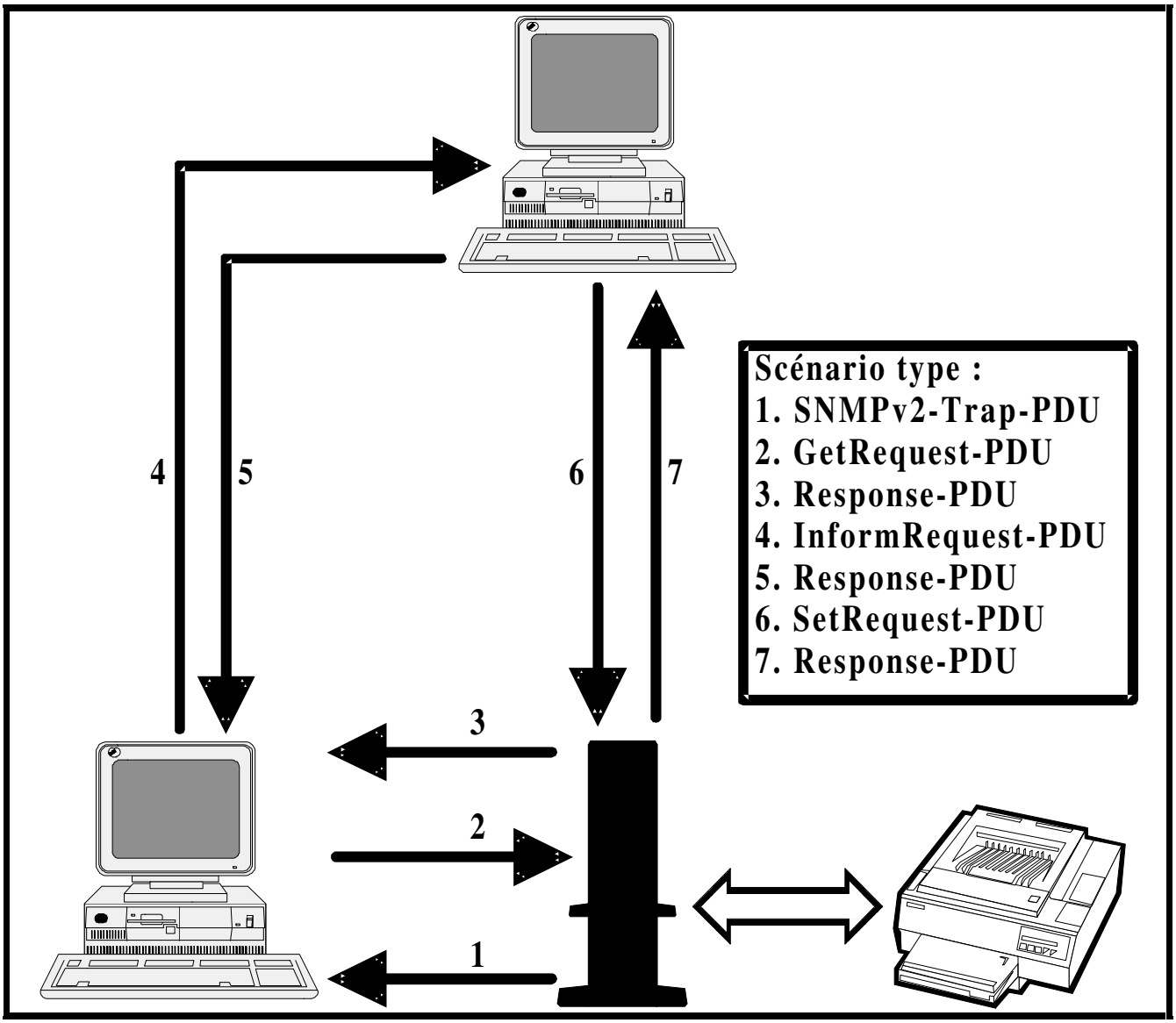
[30]Object Identifier

coldStart	L'entité vient de se réinitialiser et sa configuration peut en être altérée.
warmStart	L'entité vient de se réinitialiser mais sa configuration n'est pas altérée.
linkDown	L'entité reconnait un problème dans un de ses modules de communication.
linkUp	L'entité reconnait qu'un de ses modules de communication fonctionne à nouveau.
authentificationFailure	Un paquet non correctement authentifié vient d'être reçu.
egpNeighborLoss	Un voisin EGP ne peut plus être contacté.

Les PDUs de type InformRequest-PDU sont générés lors des échanges entre managers. Les deux premiers identificateurs de la liste d'associations sont sysUpTime.0, et snmpEventID.i, champs snmpEventID de la i-ème ligne de la table snmpEventTable, dont la valeur est celle de l'OID d'une notification définie à l'aide de la macro NOTIFICATION-TYPE. Les objets du champs OBJECTS de cette macro sont ensuite placés dans la liste d'associations, ainsi que leurs valeurs.

2.3.4.3. Scénario type

Imaginons maintenant un scénario type de résolution d'un problème par SNMP. Supposons que nous disposions d'un agent SNMP qui soit relié à une imprimante, que cet agent soit contrôlé par un manager SNMP local, ce dernier lui-même administré par un manager global, chargé d'administrer tout un ensemble de managers. Si l'agent découvre un disfonctionnement dans l'imprimante, il est amené à générer une trap (message numéro 1 sur le schéma) à destination du manager local. Ce dernier interroge alors l'agent sur un certain nombre de variables MIB concernant l'imprimante (message numéro 2), afin de déterminer précisément le problème. Ces variables du manager local pourront être des variables relatives à un contexte définissant une relation proxy dirigée vers l'agent en question. Si ces variables sortent des bornes fixées, le manager local prévient le manager global par communication entre managers (message 4). Le manager global peut alors prendre une décision qui se répercutera par une modification dans les variables de l'agent initial (message 6).



Scénario type

2.3.5. Conclusion

Nous venons de jeter un oeil sur le modèle d'administration SNMPv2, tel qu'il est défini actuellement. L'objectif initial de SNMP était d'en faire un protocole avant tout simple, afin qu'il soit implémenté par le plus grand nombre de constructeurs. C'est ce qui est arrivé pour SNMPv1, qui était réellement un protocole simple.

La simplicité de SNMPv1 a eu comme effet secondaire tout un ensemble de lacunes telles que les problèmes de sécurité. SNMPv2 a résolu ces problèmes en instaurant un nouveau protocole, à peine plus compliqué que celui de SNMPv1, qui fournit des moyens d'identification et de cryptage éprouvés. Mais SNMPv2 a aussi largement développé le nouveau concept de *modèle d'administration réseau*. Ce modèle est très complexe, et nous n'en avons vu qu'une partie. Nous n'avons pas développé par exemple tous les concepts liés aux communications entre managers, car il ne s'agissait pas ici de fournir une simple traduction littérale des RFCs décrivant SNMPv2, mais aussi car ce concept n'est pas du tout, ou que partiellement implémenté dans les logiciels actuellement.

Les difficultés qu'il y a à comprendre les fondements de SNMPv2 sont pour lui un sérieux handicap. Nombre d'administrateurs réseaux hésitent à utiliser ce protocole et les outils qui l'implémentent simplement car il nécessite un très grand investissement dans son étude avant de pouvoir en tirer parti. De plus, sa complexité ralentit les développements chez les constructeurs. On constatera que dans l'école, parmi les routeurs répondant à SNMP, seul *ulysse*, celui du département Informatique, implémente SNMPv2.

3. Les logiciels domaine public

Les logiciels domaine public qui ont attiré mon attention sont au nombre de cinq : *tricklet*, *xmib*, *scotty*, *tkined* et *gawksnmp*.

3.1. Le logiciel xmib

Le logiciel xmib est un front-end du package tricklet version 1.4 d'interrogation SNMPv1. Bien que la version 3.1 de tricklet soit actuellement sortie, permettant les interrogations suivant le protocole SNMPv2, xmib n'a pas été remis à jour pour supporter ce protocole.

Xmib est une application écrite sous l'environnement graphique Motif et permet de visualiser l'arbre d'une MIB dans un widget Motif. Etant construit au dessus de tricklet, il prend les mêmes fichiers de description de MIB que ceux générés par ISODE pour SNMPv1 (c'est à dire générés avec *mosy* dans sa version SNMPv1). Xmib ne permet pas d'interroger le contenu des tables, seuls les objets isolés peuvent être contrôlés.

L'intérêt principal de xmib est de pouvoir visualiser l'arbre d'une mib de manière graphique. Cela permet de débugger rapidement les applications SNMP, agents ou managers, et de découvrir rapidement des MIBs inconnues.

J'ai configuré tricklet afin d'intégrer dans la base de données des variables MIB le module DCC-MIB. Des exemples d'écrans xmib sur ce module sont disponibles en annexe 1.

3.2. Le logiciel tkined

Tkined est un logiciel d'administration réseau très complet de haute qualité. Il est construit au dessus du package TCL/TK, et fait largement utilisation de *scotty*, extension de TCL permettant de générer du traffic SNMPv1, ainsi que du traffic suivant d'autres protocoles réseau : ICMP pour récupérer des subnetmasks ou tracer des routes, DNS pour analyser le routage du mail ou résoudre les noms de machines en adresses IP, SunRPC pour accéder à des informations telles que les listes de systèmes de fichiers montés, exportés, ou à des statistiques diverses.

Scotty, extension d'un shell TCL, permet d'utiliser en TCL les protocoles précédemment cités. Avec ce dernier, on peut ainsi écrire rapidement des applications d'administration réseau qui vont fonctionner en faisant coexister différents protocoles. Tkined, en est ainsi un exemple. L'avenir est à ce genre d'outils, car il est clair que c'est seulement en utilisant toutes les informations disponibles sur un réseau, c'est-à-dire en pouvant communiquer par les différents protocoles implémentés sur les machines d'un réseau, que l'on peut l'administrer et le contrôler au mieux. Ce n'est pas l'utilisation d'un seul protocole qui pourra donner les meilleurs résultats.

Tkined, en utilisant les services réseau de scotty, permet de générer automatiquement des cartes des réseaux, puis de travailler sur ces cartes. Une fois la carte générée à l'aide de paquets ICMP, on peut sélectionner des entités sur le réseau et leur appliquer des opérations telles que traceroute, ou analyses des statistiques sur les disques fournies par SunRPC. On peut

aussi leur appliquer des opérations SNMP pour récupérer les tables ARP ou les tables de routage, par exemple.

J'ai utilisé Tkined afin de générer une carte du réseau de l'école. En annexe 2 se trouve une vue d'écran de Tkined, et en annexe 3 la carte du réseau de l'école. Cette carte est disponible en fichier postscript, ainsi qu'en fichier de description Tkined. Avec ce dernier, j'ai interrogé tous les routeurs et fait apparaître clairement ceux qui répondent aux requêtes SNMP. On peut ainsi utiliser Tkined sur le réseau de l'école efficacement afin de résoudre des problèmes rapidement.

3.3. Le logiciel gawksnmp

Le logiciel gawksnmp est une extension de gawk version 2.13 permettant de générer du traffic SNMP. Il est fourni avec le package ISODE-8.0 dans ses modules SNMPv1 et SNMPv2. Ainsi, il y a deux versions de gawksnmp : une qui communique par SNMPv1, une autre par SNMPv2. Ces deux versions fournissent la même interface SNMP, la seule différence est que gawksnmp v2 permet d'utiliser un protocole bien plus sécurisée. Mais comme gawksnmp v2 est très complexe à configurer - il utilise des fichiers de configuration du même type que ceux d'ISODE-8.0 pour SNMPv2 -, il est bien plus souple d'utiliser gawksnmp v1. Ce dernier souffre d'un seul défaut : on ne peut pas configurer le port destination des paquets SNMP. Il prend par défaut le port standard. J'ai créé un patch pour gawksnmp v1 permettant de modifier le port par défaut, cf annexe 4. Il suffit de rajouter sur la ligne de commande `-v PORT=25761` pour, par exemple, utiliser le port 25761.

Lorsqu'on lance gawksnmp, il lit un fichier de configuration décrivant les variables MIB accessibles, localisé par la variable d'environnement MIBDEFS.

Pour générer un tel fichier contenant toutes les MIBs standard dont les descriptions ASN.1 sont fournies par ISODE, ainsi que le module DCC-MIB, il faut tout d'abord déterminer les précédences dans les MIBs, c'est à dire quel module MIB dépend de quel autre module, puis transformer ces définitions ASN.1 pour SNMPv2 en définitions ASN.1 pour SNMPv1 à l'aide de mosy pour SNMPv2[31], puis compiler les définitions ASN.1 pour SNMPv1 à l'aide de mosy pour SNMPv1, puis concaténer tous les fichiers générés en tenant compte des précédences, afin de ne pas référencer des objets définis plus loin dans le fichier compilé final.

Après avoir analysé les modules MIB standard, j'ai pu déterminer les précédences suivants (dans la liste suivante, un module ne peut dépendre que des précédents) :

SNMPv2-SMI RFC1213-MIB SNMPv2-TC APPLICATION-MIB BRIDGE-MIB CLNS-MIB BSDUNIX-MIB DECNET-PHIV-MIB DSA-MIB RFC1285-MIB FDDI-SMT73-MIB HOST-RESOURCES-MIB MAU-MIB RFC1316-MIB RFC1382-MIB RFC1381-MIB MIOX25-MIB MTA-MIB PPP-LCP-MIB PPP-BRIDGE-NCP-MIB PPP-IP-NCP-MIB PPP-SEC-MIB RFC1229-MIB RFC1230-MIB RFC1231-MIB RFC1243-MIB RFC1253-MIB RFC1269-MIB RFC1271-MIB RFC1304-MIB RFC1315-MIB RFC1317-MIB RFC1318-MIB RFC1354-MIB RFC1389-MIB RFC1398-MIB RFC1406-MIB RFC1407-MIB RFC1414-MIB SNMP- EPEATER-MIB SOURCE-ROUTING-MIB TOKEN-RING-RMON-MIB SNMPv2-PARTY- IB SNMPv2-TM DCC-MIB.

---

[31]Cette transformation est nécessaire car certains concepts n'existent pas dans le sous-ensemble ASN.1 de SNMPv1, alors qu'elles existent pour SNMPv2. Par exemple la macro TEXTUAL-CONVENTION n'existe pas en ASN.1 pour SNMPv1.

Bien sûr, DCC-MIB est situé à la fin de la liste, car aucun autre module n'en dépend.

A chacun de ces modules est associé un fichier *.my, par exemple DCC-MIB.my pour DCC-MIB. Pour le transformer en ASN.1 pour SNMPv1, on utilise la version SNMPv2 de mosy avec l'option -1. Le fichier produit est un autre fichier ASN.1.

Supposons que le fichier ../MIB_PRECEDENCE contienne la liste des précédences, que le répertoire ../my contienne les fichiers *.my ASN.1, que mosy version SNMPv1 soit installé dans le répertoire $HOME/grenouille, et que mosy version SNMPv2 soit installé dans le répertoire $HOME/tinuviel; voici le script-shell permettant de générer automatiquement un fichier MIB compilé objects.defs :

```
#!/bin/sh

rm -f objects.defs
for i in `cat ../MIBS_PRECEDENCE`
do
        $HOME/grenouille/isode-install/usr/local/bin/mosy -1 -o ${i}.1 \
                                                        ../my/${i}.my
        if test ! -f ${i}.1
        then
                cp ../my/${i}.my ${i}.1
        fi
        $HOME/tinuviel/isode-install/usr/local/bin/mosy -o ${i}.1.defs \
                                                                ${i}.1
done

for i in `cat ../MIBS_PRECEDENCE`
do
        cat $i.1.defs >> objects.defs
done
```

Avec les mêmes suppositions que précédemment, voici le script shell permettant de générer un fichier MIB compilé objects.defs pour SNMPv2. Il servira pour gawksnmp v2, et surtout pour notre implémentation du module DCC-MIB, à l'aide des librairies ISODE pour SNMPv2. En effet, au début de notre implémentation, notre programme chargera le fichier objects.defs afin de pouvoir connaitre les objets qu'il aura à administrer.

```
#!/bin/sh

for i in `cat ../MIBS_PRECEDENCE`
do
        $HOME/grenouille/isode-install/usr/local/bin/mosy -o ${i}.defs \
                                                        ../my/${i}.my
done

rm -f objects.defs
touch objects.defs
for i in `cat ../MIBS_PRECEDENCE`
do
        cat ${i}.defs >> objects.defs
done
```

Maintenant, interrogeons par exemple la variable sysDescr, puis le contenu de la table ifEntry, et notemment ses champs ifIndex et ifMtu :

```
fenyo@mousson > setenv MIBDEFS ./objects.defs
fenyo@mousson > ~/bin/gawksnmp.V1 -v AGENT=ulysse -v PORT=161 -v \
  COMMUNITY=public 'BEGIN { print sysDescr; for (i in ifIndex) print \
  "ifIndex=", i, " - ifMtu=", ifMtu; }'
4BSD/ISODE SNMPv2
ifIndex= 1  - ifMtu= 1500
ifIndex= 2  - ifMtu= 1500
ifIndex= 3  - ifMtu= 1536
```

En réitérant la commande précédente avec l'option -S, gawksnmp fournit une analyse détaillée du contenu des paquets qu'il envoie ou reçoit, cf annexe 5. On voit dans l'exemple qui précède que gawksnmp est bien plus pratique que des outils tels que snmpget ou snmpset, qui n'utilisent pas de fichier de MIB compilé, et qui ne peuvent ainsi référencer les objets qu'avec leurs OID : la commande `snmpget 1.3.6.1.2.1.1.1.0` correspond à la commande `gawksnmp 'BEGIN { print sysDescr; }'`.

4. Une implémentation d'outil réseau avec SNMP

On vient de voir que gawksnmp est un moyen simple d'accéder à SNMP. Plus simple qu'avec les outils de base snmpget/snmpset, mais il y a quand même un certain travail avant d'arriver à un résultat avec gawksnmp - à commencer par l'apprentissage de gawk ! Pour fournir à tous l'accès à SNMP, j'ai créé une interface accessible via le protocole World Wide Web.

L'URL[32] est http://www.enst.fr/~fenyo/snmp-intro

Le système est constitué d'une première page Mosaic permettant un contrôle d'accès, afin de ne fournir le service qu'aux clients WWW de l'ENST. Les utilisateurs extérieurs se verront proposer un code d'accès. Le contrôle d'accès à ce service est complexe, mais nécessaire, les instances administratives du Centre de Calcul et du Département Informatique ayant demandé que ce service soit restreint à la seule utilisation depuis le réseau de l'ENST, pour des raisons de sécurité.

L'arborescence pour ce service dans le répertoire `public_html` est constituée de quatre sous-répertoires :

. snmp-intro ce répertoire contient une page HTML[33] pointant soit vers snmp pour les utilisateurs de l'ENST, soit vers snmp-secure pour les autres.

. scripts ce répertoire contient des scripts permettant d'interroger les variables mib : le script get-mib-value.sh permet d'interroger des ensembles de variables prédéfinis, alors que le script get-mib-value2.sh permet d'interroger une variable particulière; de plus, le script snmp-control.sh est utilisé par la page HTML du répertoire snmp-intro pour rediriger les utilisateurs en fonction de leur provenance.

. snmp ce répertoire contient la page d'interrogation réelle, il est muni d'un contrôle d'accès configuré pour le serveur Mosaic de l'école, afin qu'il ne serve cette page qu'à l'ENST.

. snmp-secure ce répertoire contient la page d'interrogation réelle, il est muni d'un contrôle d'accès configuré pour le serveur Mosaic de l'école, afin qu'il ne serve cette page qu'après avoir demandé un mot de passe.

La page réelle du service présente le réseau de l'école, et montre notemment les différents routeurs et sous réseaux. Sur cette carte, figurent explicitement les machines répondant au protocole SNMP. Deux *forms* HTML apparaissent en dessous. On peut pour chacun sélectionner un agent, une communauté et un port. De plus, pour le premier *form*, on peut sélectionner un ensemble de variables, qui correspondent à un script gawk particulier. J'ai référencé ici tous les scripts standards de gawk, mais rien n'interdit d'en créer d'autres. Le deuxième *form* permet de selectionner la variable de son choix dans une MIB, afin de l'interroger.

---

[32] Uniform Resource Locator
[33] HyperText Markup Language

En annexe 6, les différents scripts, fichiers d'accès et documents HTML constituant ce service, sont présentés.

5. Une implémentation d'un agent SNMP

Il était interessant de montrer comment construire un agent SNMP, et comment définir un module MIB conforme au modèle SNMPv2. Le choix s'est porté sur l'administration d'entités de gestion de mémoire partagé dans un système distribué, nommé DCC.

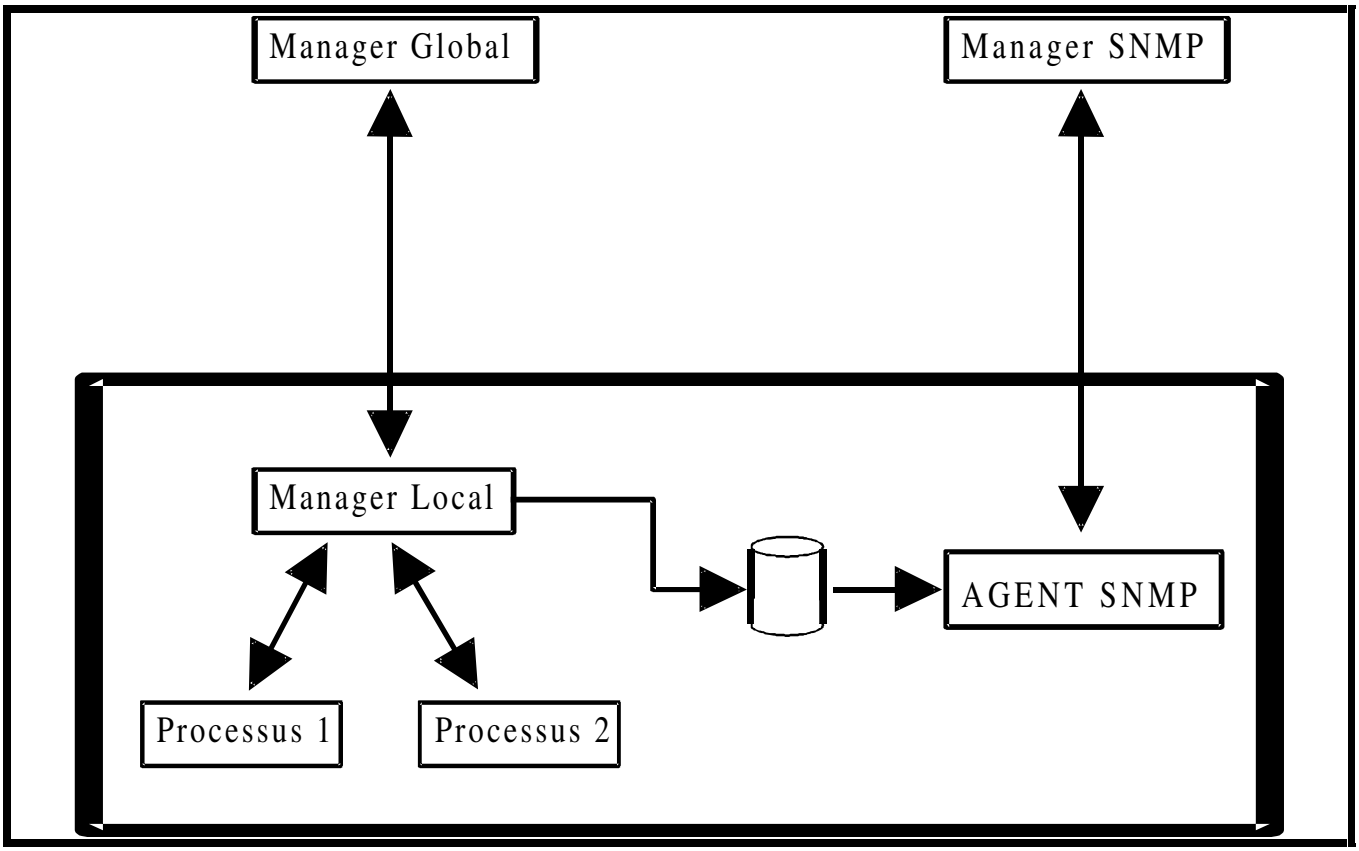
DCC est un compilateur C qui permet d'écrire des logiciels distribués, c'est-à-dire qui peuvent communiquer de manière transparente entre eux. Les développements sous DCC, à l'ENST, concernent l'intégration d'un système de mémoire partagée. Le projet s'appelle DCC-SHM et fait l'objet du mémoire de Samuel Tardieu.

DCC-SHM fonctionne de la manière suivante : à chaque groupes de processus destinés à communiquer via de la mémoire partagée, est associé un numéro de programme. Les processus d'un même programme peuvent indifférement tourner sur la même machine, ou sur plusieurs machines. Ils peuvent, par la même interface C, accéder à des variables de mémoire partagée. Pour obtenir ce résultat, il existe sur chaque machine, et pour chaque numéro de programme, un gestionnaire de mémoire partagée local, appelé local manager. Les différents local managers d'un même programme, donc sur des machines distinctes, communiquent entre eux via un gestionnaire global, le global manager.

Afin de pouvoir contrôler le fonctionnement du système distribué de mémoire partagée sous DCC, il est apparu interessant de fournir un moyen d'interaction avec les local managers de chaque machine. Samuel Tardieu et moi-même avons donc défini un ensemble de variables MIB permettant d'administrer et de contrôler les gestionnaires locaux.

J'ai ensuite implémenté, à l'aide des outils du package ISODE-8.0/SNMPv2, un agent SNMPv2 permettant de fournir à des managers SNMPv1/v2 des informations quand à l'état du local manager DCC-SHM. Le code de cette implémentation est en annexe 7.

La communication entre l'agent SNMP et le gestionnaire local de mémoire partagée s'est faite par l'intermédiaire du système de fichier local d'Unix. Le principe est d'associer à chaque variable MIB du module DCC-SHM un fichier dans le répertoire /tmp/.dcc-shm. Le gestionnaire de mémoire partagée écrit dans ces fichiers dès qu'un changement dans les variables partagées qu'il maintient se produit, et l'agent SNMP lit dans ses fichiers dès qu'une requête d'un manager lui parvient. Il est clair que la méthode choisie n'est pas la plus efficace possible, mais l'objet de l'implémentation de l'agent SNMP, autant que celui de l'implémentation du gestionnaire de mémoire partagée, n'est pas l'efficacité à tout prix, mais il s'agit plus ici de montrer par la pratique un moyen d'utilisation de SNMP dans le cadre de l'administration et du contrôle d'un système fournissant des services distribués transparents.



Interactions entre SNMP et DCC-SHM

En plus des variables concernant DCC-SHM, j'ai ajouté des variables permettant de contrôler la charge de la machine, afin d'augmenter la part du contrôle à distance, pour mieux gérer la distribution DCC. En effet, il peut être interessant pour un manager DCC-SHM de lancer un nouveau processus sur la machine la moins chargée disponible. A l'aide de SNMP, le manager peut connaitre différentes informations sur la charge de chaque machine.

Le principe de l'introduction de ASN.1 dans SNMP est de formaliser un méthode de description d'un module, ainsi que les conformances de l'implémentation aux spécifications. Voici le module DCC-MIB qui définit non seulement les objets administrés, mais aussi les déviations par rapport au modèle, autorisées dans les implémentations, ainsi que les capacités réelles de l'implémentation par rapport aux spécifications, comme cela est demandé à tout implémenteur de MIB dans SNMPv2.

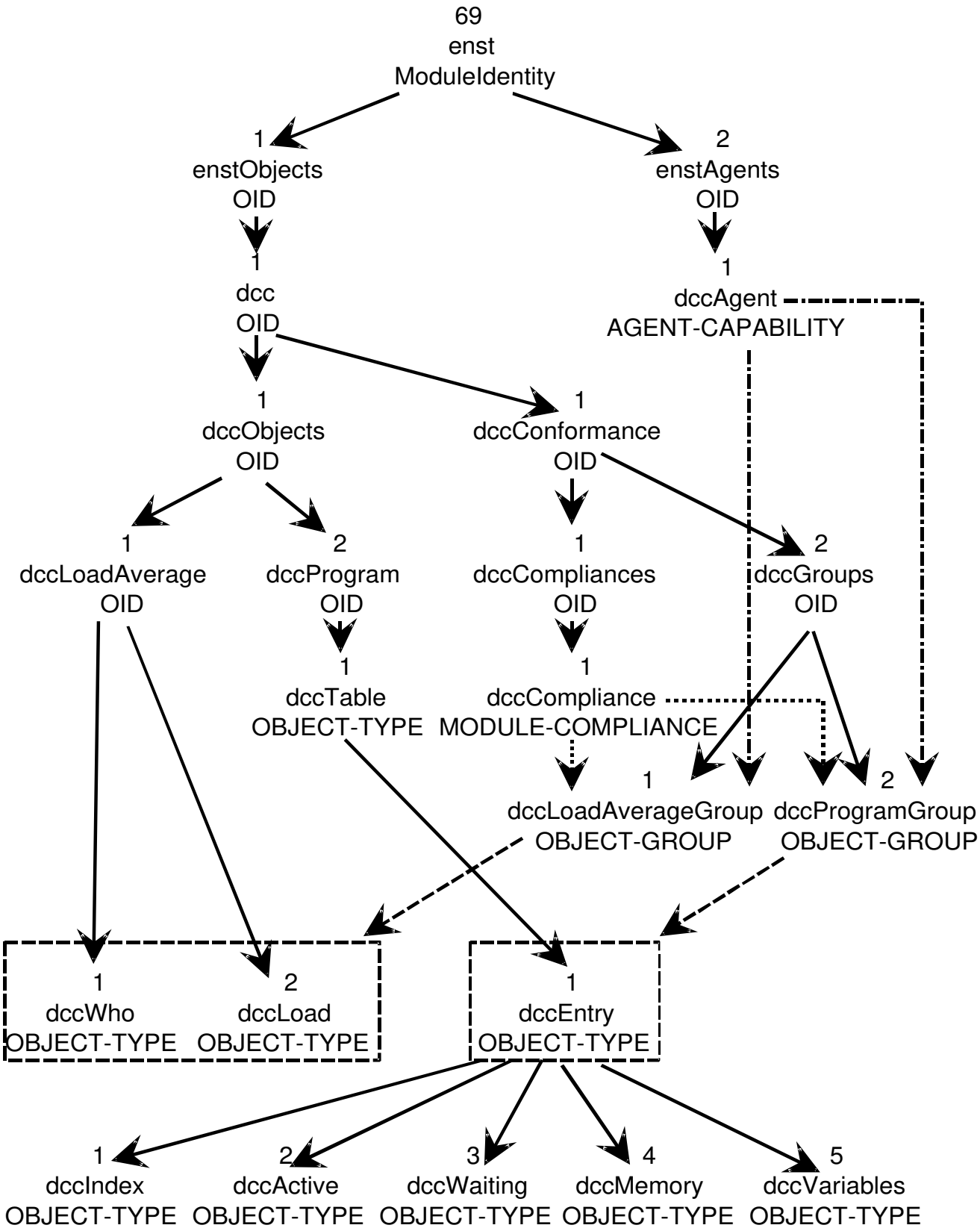
```
-- DCC MIB module

DCC-MIB DEFINITIONS ::= BEGIN

-- Title: DCC SHM MIB for SNMPv2
-- Date: July 3, 1994
-- By:          Alexandre Fenyo / ENST <fenyo@enst.fr>
-- For:         Samuel Tardieu / ENST <sam@enst.fr>


IMPORTS
        MODULE-IDENTITY, OBJECT-TYPE, enterprises
                FROM SNMPv2-SMI
        DisplayString, TEXTUAL-CONVENTION
```

```
                FROM SNMPv2-TC
        AGENT-CAPABILITIES, MODULE-COMPLIANCE, OBJECT-GROUP
                FROM SNMPv2-CONF;


-- Textual Convention

ProcessCount ::= TEXTUAL-CONVENTION
        STATUS current
        DESCRIPTION
                "A number of processes matching a specified condition."
        SYNTAX INTEGER (1..2147483647)


-- Module Identity

        enst MODULE-IDENTITY
                LAST-UPDATED "9407030000Z"
                ORGANIZATION "Ecole Nationale Superieure des Telecommunications de Paris"
        CONTACT-INFO "Alexandre Fenyo fenyo@enst.fr / Phone: +33 (1) 45.81.71.70"
        DESCRIPTION "The MIB module for DCC local memory manager."
        ::= { enterprises 69 }


-- main object identifiers

enstObjects OBJECT IDENTIFIER ::= { enst 1 }

enstAgents OBJECT IDENTIFIER ::= { enst 2 }

dcc OBJECT IDENTIFIER ::= { enstObjects 1 }


-- the dcc objects

dccObjects OBJECT IDENTIFIER ::= { dcc 1 }


-- the loadAverage group

dccLoadAverage OBJECT IDENTIFIER ::= { dccObjects 1 }

dccWho OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..255))
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
        "General informations about the users and load average."
        REFERENCE "See SunOS 4.x manual page about w."
        ::= { dccLoadAverage 1 }

dccLoad OBJECT-TYPE
        SYNTAX INTEGER (1..2147483647)
        UNITS ".001 * number of jobs in the run queue averaged over 1 minute"
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
```

```
                "Load average."
        REFERENCE "See SunOS 4.x manyal page about w."
        ::= { dccLoadAverage 2 }



-- the program group

dccProgram OBJECT IDENTIFIER ::= { dccObjects 2 }

dccTable OBJECT-TYPE
        SYNTAX SEQUENCE OF DccEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
                "The information table about the DCC programs on this entity."
        ::= { dccProgram 1 }

dccEntry OBJECT-TYPE
        SYNTAX DccEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
                "The program id."
        INDEX { dccIndex }
        ::= { dccTable 1 }

DccEntry ::=
        SEQUENCE {
                dccIndex INTEGER (1..2147483647),
                dccActive ProcessCount,
                dccWaiting ProcessCount,
                dccMemory INTEGER (1..2147483647),
                dccVariables DisplayString (SIZE (0..255))
        }

dccIndex OBJECT-TYPE
        SYNTAX INTEGER (1..2147483647)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "A unique identifier of the program."
        ::= { dccEntry 1 }

dccActive OBJECT-TYPE
        SYNTAX ProcessCount
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "Number of active processes running this program id."
        ::= { dccEntry 2 }

dccWaiting OBJECT-TYPE
        SYNTAX ProcessCount
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
```

```
                "Number of active processes, running this program id, currently waiting for a variable."
        ::= { dccEntry 3 }

dccMemory OBJECT-TYPE
        SYNTAX INTEGER (1..2147483647)
        UNITS "bytes"
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "Size of valid shared memory handled by the local memory manager."
        ::= { dccEntry 4 }

dccVariables OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..255))
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "Set of \n separated shared variable names known by the local memory manager."
        ::= { dccEntry 5 }


-- conformance information

dccConformance OBJECT IDENTIFIER ::= { dcc 2 }

dccCompliances OBJECT IDENTIFIER ::= { dccConformance 1 }

dccGroups OBJECT IDENTIFIER ::= { dccConformance 2 }


-- groups definitions

dccLoadAverageGroup OBJECT-GROUP
        OBJECTS { dccWho, dccLoad }
        STATUS current
        DESCRIPTION
                "A collection of objects providing basic informations about the load average."
        ::= { dccGroups 1 }

dccProgramGroup OBJECT-GROUP
        OBJECTS { dccTable }
        STATUS current
        DESCRIPTION
                "A collection of objects providing informations about the programs;
                currently, there is only one object in this group, but future developpements
                could lead to the creation of new objects in this group."
        ::= { dccGroups 2 }


-- compliance statements

dccCompliance MODULE-COMPLIANCE
        STATUS current
        DESCRIPTION
                "The compliance statement for SNMPv2 entities which implement the dcc MIB."
        MODULE
        MANDATORY-GROUPS { dccProgramGroup }
```

```
        GROUP dccLoadAverageGroup
                DESCRIPTION
                "The dccLoadAverage group is mandatory only for those SNMPv2 entities which
                also implement the Unix operating system."
        ::= { dccCompliances 1 }


-- agent capabilities statements

dccAgent AGENT-CAPABILITIES
        PRODUCT-RELEASE "DCC SHM with SMNPv2 Control Module release 1.0 for SunOS 4.1.x"
        STATUS current
        DESCRIPTION "DCC SHM SNMP for SunOS 4.1.x"

        SUPPORTS RFC1213-MIB
                INCLUDES { system, snmp }

        SUPPORTS DCC-MIB
                INCLUDES { dccLoadAverageGroup, dccProgramGroup }

        VARIATION dccLoad
                SYNTAX INTEGER (1..2147483647)
        DESCRIPTION "Information taken by means of a call to /usr/ucb/w, not inside kernel data."
        ::= { enstAgents 1 }


END
```



Légende : les flèches en pointillés montrent les références croisées
les rectangles en pointillés montrent les groupes d'objets

Ces définitions écrites dans le sous-ensemble ASN.1 pour SNMPv2 ont ensuite été traduites en définitions pour ASN.1 pour SNMPv1 à l'aide de mosy pour SNMPv2 - cf annexe

8 -, puis compilé pour SNMPv1 à l'aide de mosy pour SNMPv1 - cf annexe 9 -, afin de l'intégrer au fichier objects.defs utilisé par gawksnmp v1. Ce fichier a aussi été directement compilé à l'aide de mosy pour SNMPv2, pour être intégré au fichier objects.defs utilisé par les librairies ISODE-8.0/SNMPv2 lorsque l'agent SNMPv2 DCC-SHM est activé - cf annexe 10.

L'agent SNMP développé est bien sûr suceptible d'être lancé sur différentes machines. Etant construit à l'aide de librairies SNMPv2, il nécéssite un ensemble de fichiers de description définissant les parties et contextes par défaut, chargés dans une base de données par les librairies ISODE. Les fichiers de configuration des parties sont dépendants de la machine sur laquelle l'agent doit être activé, car SNMPv2 définit les OIDs des parties et contextes par défaut sur chaque agent à partir de l'adresse IP de cette machine. Cette méthode permet d'assurer l'unicité des OIDs ainsi générés : deux machines distinctes ne pourront pas avoir les mêmes parties et contextes par défaut. Pour simplifier la génération des fichiers de configuration par défaut, j'ai développé un script shell, ainsi que des fichiers de configuration génériques utilisés par le script shell afin de générer des fichiers de configuration utilisable.

Il y a cinq fichiers de configuration par machine : ${hostname}.access qui définit la table des accès, ${hostname}.context, qui définit les différents contextes, ${hostname}.public et ${hostname}.private qui définissent la table des parties, et enfin ${hostname}.view qui définit les vues.

Le script shell permettant de générer ces fichiers de manière automatisée suit :

```
# more mk_party
#!/bin/sh
# A.Fenyo 1994

if test $# = 0;
then
        echo "usage: $0 hostname"
        exit 1
fi

IP_ADDRESS=`/usr/local/bin/nsl $1`
HEXA_ADDRESS=`/usr/local/isode-8.0/etc/parties/dec2hex ${IP_ADDRESS}`

echo IP_ADDRESS=${IP_ADDRESS}

for name in access context private public view;
do
        echo "# Fichier genere automatiquement par mk_party (A.Fenyo 1994)"
> ${
1}.${name}
        echo "# IP_ADDRESS=${IP_ADDRESS}" >> ${1}.${name};
        sed -e "s/\\\${IP_ADDRESS}/${IP_ADDRESS}/g" < template.${name} | \
        sed -e "s/\\\${HEXA_ADDRESS}/${HEXA_ADDRESS}/g" >> ${1}.${name}
        if test $name = private
        then
                chown root.wheel ${1}.${name}
                chmod 640 ${1}.${name}
        else
                chmod 644 ${1}.${name}
        fi
        echo "${1}.${name} created"
done
```

Les fichiers génériques de configuration par défaut suivent :

```
1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.1 \
      1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.1 \
      1.3.6.1.6.3.3.1.4.${IP_ADDRESS}.1 \
      active notify

1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.1 \
      1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.1 \
      1.3.6.1.6.3.3.1.4.${IP_ADDRESS}.3 \
      active notify

1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.1 \
      1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.2 \
      1.3.6.1.6.3.3.1.4.${IP_ADDRESS}.1 \
      active get get-next get-bulk

1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.1 \
      1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.2 \
      1.3.6.1.6.3.3.1.4.${IP_ADDRESS}.3 \
      active get get-next get-bulk

1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.2 \
      1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.1 \
      1.3.6.1.6.3.3.1.4.${IP_ADDRESS}.1 \
      active response

1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.2 \
      1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.1 \
      1.3.6.1.6.3.3.1.4.${IP_ADDRESS}.3 \
      active response

1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.3 \
      1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.4 \
      1.3.6.1.6.3.3.1.4.${IP_ADDRESS}.2 \
      active get get-next set get-bulk

1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.3 \
      1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.4 \
      1.3.6.1.6.3.3.1.4.${IP_ADDRESS}.3 \
      active get get-next set get-bulk

1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.4 \
      1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.3 \
      1.3.6.1.6.3.3.1.4.${IP_ADDRESS}.2 \
      active response

1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.4 \
      1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.3 \
      1.3.6.1.6.3.3.1.4.${IP_ADDRESS}.3 \
      active response

1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.5 \
      1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.6 \
      1.3.6.1.6.3.3.1.4.${IP_ADDRESS}.2 \
      active get get-next set get-bulk

1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.6 \
      1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.5 \
      1.3.6.1.6.3.3.1.4.${IP_ADDRESS}.2 \
      active response
```

fichier template.access

```
A 1.3.6.1.6.3.3.1.4.${IP_ADDRESS}.1 "${IP_ADDRESS}-system" active \
      view 1 "" currentTime

A 1.3.6.1.6.3.3.1.4.${IP_ADDRESS}.2 "${IP_ADDRESS}-internet" active \
      view 2 "" currentTime
```

```
A 1.3.6.1.6.3.3.1.4.${IP_ADDRESS}.3 "${IP_ADDRESS}-unix" active \
      proxy 1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.7 \
      0.0 \
      0.0
```

fichier template.context

```
1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.1   active

1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.2   active

1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.3   active      "NIQsvZAYunmgQk1J"

1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.4   active      "0cCi6.lQhiJnZ1UW"

1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.5   active      "4sF4e5SU203cDTmP"
"elJGCnAPse8ZQyEg"

1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.6   active      "elfdRYaHelfdRYaH"
"56eLmE2m22c9FIlY"

1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.7   active      "unix"
```

fichier template.private

```
L 1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.1 \
      snmpUDPDomain 0x${HEXA_ADDRESS}:00:a1 8192 \
      noAuth 0 0 \
      noPriv

M 1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.2 \
      snmpUDPDomain 0x00:00:00:00:00:00 8192 \
      noAuth 0 0 \
      noPriv

L 1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.3 \
      snmpUDPDomain 0x${HEXA_ADDRESS}:00:a1 8192 \
      v2md5AuthProtocol 69274060 300 \
      noPriv

M 1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.4 \
      snmpUDPDomain 0x00:00:00:00:00:00 8192 \
      v2md5AuthProtocol 69274060 300 \
      noPriv

L 1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.5 \
      snmpUDPDomain 0x${HEXA_ADDRESS}:00:a1 8192 \
      v2md5AuthProtocol 69274062 300 \
      desPrivProtocol

M 1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.6 \
      snmpUDPDomain 0x00:00:00:00:00:00 8192 \
      v2md5AuthProtocol 69274062 300 \
      desPrivProtocol

P 1.3.6.1.6.3.3.1.3.${IP_ADDRESS}.7 \
      rfc1157Domain 0x${HEXA_ADDRESS}:64:a1 8192 \
      rfc1157noAuth 0 0 \
      noPriv
```

fichier template.public

```
1     active      included    internet

2     active      included    internet
```

fichier template.view

Le script shell précédent fait appel au programme dec2hex afin de transformer les adresses IP en notation hexadécimal exigée par ISODE :

```
#include <stdio.h>

int main(ac, av)
int ac;
char **av;
{
  int cnt, nb;
  int address[6];

  if (ac != 2) {
    printf("usage: %s [[[[[[ip1].ip2].ip3].ip4].ip5].ip6]\nexample: %s
137.194.160.1", av[0], av[0]);
    return 1;
  }

  nb = sscanf(av[1], "%d.%d.%d.%d.%d.%d", address, address+1, address+2,
address+3, address+4, address+5);
  for (cnt = 0; cnt < nb; cnt++) printf((cnt == (nb-1)) ? "%.2x\n" :
"%.2x:", address[cnt]);

  return 0;
}
```

6. Conclusion

Actuellement, l'Internet subit une phase de croissance extraordinaire. La demande des utilisateurs est de plus en plus importante. Tous les jours de nouveaux réseaux sont interconnectés. La demande des entreprises pour obtenir les services de l'Internet est énorme, la guerre fait rage entre les différents *Internet Providers*.

Toutes les sociétés nouvellement connectées n'ont pas forcément ni l'envie ni les moyens d'administrer les problèmes journaliers auquels leurs réseaux sont confrontés. Actuellement, le domaine d'administration des Internet Providers s'arrête à l'entrée des sites alimentés. Bientôt, les utilisateurs demanderont des solutions clef en main, c'est-à-dire la possibilité d'obtenir les services de l'Internet, par l'intermédiaires d'Internet Providers, qui devront aller administrer jusqu'au sein des réseaux des clients. L'administration va donc devenir de plus en plus complexe, les réseaux administrés par une même entité de plus en plus vastes, et l'on n'aura plus les moyens, comme on le fait actuellement à Télécom Paris, d'aller sur place pour résoudre les problèmes journaliers.

SNMPv2 propose une solution générale pour résoudre tous ces problèmes futurs : un protocole sécurisé, et surtout, tout un modèle d'administration permettant le suivi du réseau, la possibilité de générer des alarmes pour découvrir le plus tôt possible toute situation anormale, et éventuellement la mise en place de solutions lorsque les problèmes se manifestent.

SNMPv2 est encore peu et partiellement implémenté. Il est trop jeune pour qu'on puisse imaginer son avenir, mais il est sûr qu'un protocole tel que lui va devoir s'imposer pour que l'Internet puisse continuer à fournir la même qualité de service à laquelle nous avons été jusqu'à maintenant habitués.

7. Annexes

7.1. ANNEXE 1 : écrans xmib

1er écran xmib

2ième écran xmib

7.2. ANNEXE 2 : écran tkined

écran Tkined

7.3. ANNEXE 3 : carte du réseau de l'école

Carte du réseau de l'école

7.4. ANNEXE 4 : patck gawksnmpV1

```
<fenyo@mousson> diff -c main.c ~/tinuviel/isode-8.0/snmp/gawk/gawk-
2.13/main.c
*** main.c      Sun Jul  3 22:03:53 1994
--- /inf/mousson2/idl/fenyo/tinuviel/isode-8.0/snmp/gawk/gawk-2.13/main.c
Fri Mar 18 02:18:17 1994
***************
*** 534,542 ****
--- 534,547 ----
        NODE *it;
        NODE **lhs;

+       extern int snmp_portno;
+
        cp = strchr(arg, '=');
        if (cp != NULL) {
                *cp++ = '\0';
+
+               if (!strcmp(arg, "PORT")) snmp_portno = atoi(cp);
+
                /*
                 * Recent versions of nawk expand escapes inside
assignments.
                 * This makes sense, so we do it too.
```

7.5. ANNEXE 5 : paquets générés par gawksnmp

```
{
  version 0,
  community "public",
  data {
    get-request {
      request-id 1948690223,
      error-status 0,
      error-index 0,
      variable-bindings {
        VarBind {
          name sysDescr.0,
          value NULL
        }
      }
    }
  }
}
{
  version 0,
  community "public",
  data {
    get-response {
      request-id 1948690223,
      error-status 0,
      error-index 0,
      variable-bindings {
        VarBind {
          name sysDescr.0,
          value "4BSD/ISODE
SNMPv2"
        }
      }
    }
  }
}
4BSD/ISODE SNMPv2
{
  version 0,
  community "public",
  data {
    get-next-request {
      request-id 1948690224,
      error-status 0,
      error-index 0,
      variable-bindings {
        VarBind {
          name ifType,
          value NULL
        },
        VarBind {
          name ifInErrors,
          value NULL
        },
        VarBind {
          name ifOutErrors,
          value NULL
        },
        VarBind {
          name ifPhysAddress,
          value NULL
        },
        VarBind {
          name ifAdminStatus,
          value NULL
        },
        VarBind {
          name ifOperStatus,
          value NULL
        },
        VarBind {
          name ifInOctets,
          value NULL
        },
        VarBind {
          name ifOutOctets,
          value NULL
        },
        VarBind {
          name ifIndex,
          value NULL
        },
        VarBind {
          name ifLastChange,
          value NULL
        },
        VarBind {
          name ifInUcastPkts,
          value NULL
        },
        VarBind {
          name ifInNUcastPkts,
          value NULL
        },
        VarBind {
          name ifOutUcastPkts,
          value NULL
        },
        VarBind {
          name ifOutNUcastPkts,
          value NULL
        },
        VarBind {
          name ifOutQLen,
          value NULL
        },
        VarBind {
          name ifSpeed,
          value NULL
        },
        VarBind {
          name ifSpecific,
          value NULL
        },
        VarBind {
          name ifDescr,
          value NULL
        },
        VarBind {
          name ifMtu,
          value NULL
        },
        VarBind {
          name ifInDiscards,
          value NULL
        },
        VarBind {
          name ifOutDiscards,
          value NULL
        },
        VarBind {
          name ifInUnknownProtos,
          value NULL
        }
      }
    }
  }
}
{
  version 0,
  community "public",
  data {
    get-response {
      request-id 1948690224,
      error-status 0,
      error-index 0,
      variable-bindings {
        VarBind {
          name ifType.1,
          value 6
        },
        VarBind {
          name ifInErrors.1,
          value "\\"
        },
        VarBind {
          name ifOutErrors.1,
          value "W"
        },
        VarBind {
          name ifPhysAddress.1,
          value '08002010f8db'H
        },
        VarBind {
          name ifAdminStatus.1,
          value 1
        },
        VarBind {
          name ifOperStatus.1,
          value 1
        },
        VarBind {
          name ifInUcastPkts.1,
          value '494df0'H
        },
        VarBind {
          name ifOutUcastPkts.1,
          value '5ec583'H
        },
        VarBind {
          name ifIndex.1,
          value 1
        },
        VarBind {
          name ifInUcastPkts.1,
          value '494df0'H
        },
        VarBind {
          name ifInUcastPkts.1,
          value '494df0'H
        },
        VarBind {
          name ifInErrors.1,
          value "\\"
        },
        VarBind {
          name ifOutUcastPkts.1,
          value '5ec583'H
        },
        VarBind {
          name ifOutDiscards.1,
          value '038e'H
        },
        VarBind {
          name ifOutQLen.1,
          value '00'H
        },
        VarBind {
          name ifSpeed.1,
          value '00a00000'H
        },
        VarBind {
          name ifSpecific.1,
          value 0.0
        },
        VarBind {
          name ifDescr.1,
          value "le0"
        },
        VarBind {
          name ifMtu.1,
          value 1500
        },
        VarBind {
          name ifInErrors.1,
          value "\\"
        },
        VarBind {
          name ifOutDiscards.1,
          value '038e'H
        },
        VarBind {
```

```
          name ifOutUcastPkts.1,
          value '5ec583'H
        }
      }
    }
  }
}
ifIndex= 1  - ifMtu= 1500
{
  version 0,
  community "public",
  data {
    get-next-request {
      request-id 1948690226,
      error-status 0,
      error-index 0,
      variable-bindings {
        VarBind {
          name ifType.1,
          value NULL
        },
        VarBind {
          name ifInErrors.1,
          value NULL
        },
        VarBind {
          name ifOutErrors.1,
          value NULL
        },
        VarBind {
          name ifPhysAddress.1,
          value NULL
        },
        VarBind {
          name ifAdminStatus.1,
          value NULL
        },
        VarBind {
          name ifOperStatus.1,
          value NULL
        },
        VarBind {
          name ifIndex.1,
          value NULL
        },
        VarBind {
          name ifInUcastPkts.1,
          value NULL
        },
        VarBind {
          name ifOutUcastPkts.1,
          value NULL
        },
        VarBind {
          name ifOutQLen.1,
          value NULL
        },
        VarBind {
          name ifSpeed.1,
          value NULL
        },
        VarBind {
          name ifSpecific.1,
          value NULL
        },
        VarBind {
          name ifDescr.1,
          value NULL
        },
        VarBind {
          name ifMtu.1,
          value NULL
        },
        VarBind {
          name ifOutDiscards.1,
          value NULL
        }
      }
    }
  }
}
{
  version 0,
  community "public",
  data {
    get-response {
      request-id 1948690226,
      error-status 0,
      error-index 0,
      variable-bindings {
        VarBind {
          name ifType.2,
          value 6
        },
        VarBind {
          name ifInErrors.2,
          value '00'H
        },
        VarBind {
          name ifOutErrors.2,
          value '02'H
        },
        VarBind {
          name ifPhysAddress.2,
          value '08002010f8db'H
        },
        VarBind {
          name ifAdminStatus.2,
          value 1
        },
        VarBind {
          name ifOperStatus.2,
          value 1
        },
        VarBind {
          name ifIndex.2,
          value 2
        },
        VarBind {
          name ifInUcastPkts.2,
          value '4dc87d'H
        },
        VarBind {
          name ifOutUcastPkts.2,
          value '1bdf6c'H
        },
        VarBind {
          name ifOutQLen.2,
          value '00'H
        },
        VarBind {
          name ifSpeed.2,
          value '00a00000'H
        },
        VarBind {
          name ifSpecific.2,
          value 0.0
        },
        VarBind {
          name ifDescr.2,
          value "le1"
        },
        VarBind {
          name ifMtu.2,
          value 1500
        },
        VarBind {
          name ifOutDiscards.2,
          value '00'H
        }
      }
    }
  }
}
ifIndex= 2  - ifMtu= 1500
{
  version 0,
  community "public",
  data {
    get-next-request {
      request-id 1948690228,
      error-status 0,
      error-index 0,
      variable-bindings {
        VarBind {
          name ifType.2,
          value NULL
        },
        VarBind {
          name ifInErrors.2,
          value NULL
        },
        VarBind {
          name ifOutErrors.2,
          value NULL
        },
        VarBind {
          name ifPhysAddress.2,
          value NULL
        },
        VarBind {
          name ifAdminStatus.2,
          value NULL
        },
        VarBind {
          name ifOperStatus.2,
          value NULL
        },
        VarBind {
          name ifIndex.2,
          value NULL
        },
        VarBind {
          name ifInUcastPkts.2,
          value NULL
        },
        VarBind {
          name ifOutUcastPkts.2,
          value NULL
        },
        VarBind {
          name ifOutQLen.2,
          value NULL
        },
        VarBind {
          name ifSpeed.2,
          value NULL
        },
        VarBind {
          name ifSpecific.2,
          value NULL
        },
        VarBind {
          name ifDescr.2,
          value NULL
        },
        VarBind {
          name ifMtu.2,
          value NULL
        },
        VarBind {
          name ifOutDiscards.2,
          value NULL
        }
      }
    }
  }
}
{
  version 0,
  community "public",
  data {
    get-response {
      request-id 1948690228,
```

```
      error-status 0,
      error-index 0,
      variable-bindings {
        VarBind {
          name ifType.3,
          value 1
        },
        VarBind {
          name ifInErrors.3,
          value '00'H
        },
        VarBind {
          name ifOutErrors.3,
          value '00'H
        },
        VarBind {
          name ifPhysAddress.3,
          value '000000000000'H
        },
        VarBind {
          name ifAdminStatus.3,
          value 1
        },
        VarBind {
          name ifOperStatus.3,
          value 1
        },
        VarBind {
          name ifIndex.3,
          value 3
        },
        VarBind {
          name ifInUcastPkts.3,
          value '46e1f7'H
        },
        VarBind {
          name ifOutUcastPkts.3,
          value '46e1f7'H
        },
        VarBind {
          name ifOutQLen.3,
          value '00'H
        },
        VarBind {
          name ifSpeed.3,
          value '00'H
        },
        VarBind {
          name ifSpecific.3,
          value 0.0
        },
        VarBind {
          name ifDescr.3,
          value "lo0"
        },
        VarBind {
          name ifMtu.3,
          value 1536
        },
        VarBind {
          name ifOutDiscards.3,
          value '00'H
        }
      }
    }
  }
}
ifIndex= 3  - ifMtu= 1536
{
  version 0,
  community "public",
  data {
    get-next-request {
      request-id 1948690230,
      error-status 0,
      error-index 0,
      variable-bindings {
        VarBind {
          name ifType.3,
          value NULL
        },
        VarBind {
          name ifInErrors.3,
          value NULL
        },
        VarBind {
          name ifOutErrors.3,
          value NULL
        },
        VarBind {
          name ifPhysAddress.3,
          value NULL
        },
        VarBind {
          name ifAdminStatus.3,
          value NULL
        },
        VarBind {
          name ifOperStatus.3,
          value NULL
        },
        VarBind {
          name ifIndex.3,
          value NULL
        },
        VarBind {
          name ifInUcastPkts.3,
          value NULL
        },
        VarBind {
          name ifOutUcastPkts.3,
          value NULL
        },
        VarBind {
          name ifOutQLen.3,
          value NULL
        },
        VarBind {
          name ifSpeed.3,
          value NULL
        },
        VarBind {
          name ifSpecific.3,
          value NULL
        },
        VarBind {
          name ifDescr.3,
          value NULL
        },
        VarBind {
          name ifMtu.3,
          value NULL
        },
        VarBind {
          name ifOutDiscards.3,
          value NULL
        }
      }
    }
  }
}
{
  version 0,
  community "public",
  data {
    get-response {
      request-id 1948690230,
      error-status 0,
      error-index 0,
      variable-bindings {
        VarBind {
          name ifMtu.1,
          value 1500
        },
        VarBind {
          name ifOutUcastPkts.1,
          value '5ec583'H
        },
        VarBind {
          name ifOutQLen.1,
          value '00'H
        },
        VarBind {
          name ifAdminStatus.1,
          value 1
        },
        VarBind {
          name ifOperStatus.1,
          value 1
        },
        VarBind {
          name ifInUcastPkts.1,
          value '494df0'H
        },
        VarBind {
          name ifDescr.1,
          value "le0"
        },
        VarBind {
          name ifInErrors.1,
          value "\\"
        },
        VarBind {
          name ifOutDiscards.1,
          value '038e'H
        },
        VarBind {
          name ifSpecific.1,
          value 0.0
        },
        VarBind {
          name ifPhysAddress.1,
          value '08002010f8db'H
        },
        VarBind {
          name
atIfIndex.1.1.137.194.2.5,
          value 1
        },
        VarBind {
          name ifType.1,
          value 6
        },
        VarBind {
          name ifSpeed.1,
          value '00a00000'H
        },
        VarBind {
          name ifOutErrors.1,
          value "W"
        }
      }
    }
  }
}
```

7.6. ANNEXE 6 : W3 Network Manager

Fichier snmp-control.sh

```
#!/bin/sh

DEPT=UNKNOWN
case `echo $REMOTE_ADDR | sed 's/^137\.194\.\([0-9]*\)\.[0-9]*$/\1/'` in
        2 |   4 |   6                       ) DEPT=ENST;;
       16 |  18 |  20 | 22                  ) DEPT=ADM;;
       32 |  34 |  36 | 38                  ) DEPT=CAL;;
       64 |  66                         ) DEPT=COM;;
       96 |  98                         ) DEPT=ELEC;;
      128                                ) DEPT=IMA;;
      160 | 162 | 164 | 166 | 168          ) DEPT=INF;;
      192 | 194 | 196 | 198 | 200 | 202   ) DEPT=SIG;;
esac

if test `echo $REMOTE_ADDR | sed 's/^137\.194\..*$/OK/'` = "OK";
then
      if test $DEPT = "ELEC" || test $DEPT = "IMA"
      then
            echo "Though your WEB client is connected from within ENST,
this service is <Em>not available</Em> from"

            if test $DEPT = "ELEC"
            then
                  echo Electronic Department.
            fi
            if test $DEPT = "IMA"
            then
                  echo Image Department.
            fi

            echo "If you are an authorised user, please feel free to click
<A HREF="/~fenyo/snmp-secure">here</A> to access W3 Network Manager."
      else
            echo "Click <A HREF="/~fenyo/snmp">here</A> to access W3
Network Manager."
      fi
else
      echo "Click <A HREF="/~fenyo/snmp-secure">here</A> to access W3
Network Manager."
fi
```

Fichier get-mib-value.sh

```
#!/bin/sh

cat << FIN
Content-type: text/html

<TITLE>Network Manager</TITLE>
<H1>
<iMG ALIGN=MIDDLE SRC="/~fenyo/snmp/sunset.gif"> <iMG ALIGN=TOP
SRC="/~fenyo/snmp/title.gif">
</H1>
<HR>
<H1><iMG ALIGN=MIDDLE SRC="/~fenyo/snmp/graph05.gif"> SNMP Result</H1>
FIN

eval "`echo $QUERY_STRING | /users/fenyo/public_html/scripts/scan.yy | sed
's/ /-/g' | sed 's/&/ /g'"
```

```
echo "<PRE>"

case $option in
      System           ) /users/fenyo/bin/gawksnmp.V1 -v AGENT=$hostname -
v PORT=$portno -v COMMUNITY=$community -f
/users/fenyo/public_html/snmp/gawk.scripts/mib.system;;
      ARP         ) /users/fenyo/bin/gawksnmp.V1 -v AGENT=$hostname -v
PORT=$portno -v COMMUNITY=$community -f
/users/fenyo/public_html/snmp/gawk.scripts/mib.arp;;
      Connections ) /users/fenyo/bin/gawksnmp.V1 -v AGENT=$hostname -v
PORT=$portno -v COMMUNITY=$community -f
/users/fenyo/public_html/snmp/gawk.scripts/mib.connections;;
      EGP         ) /users/fenyo/bin/gawksnmp.V1 -v AGENT=$hostname -v
PORT=$portno -v COMMUNITY=$community -f
/users/fenyo/public_html/snmp/gawk.scripts/mib.egp;;
      FDDI        ) /users/fenyo/bin/gawksnmp.V1 -v AGENT=$hostname -v
PORT=$portno -v COMMUNITY=$community -f
/users/fenyo/public_html/snmp/gawk.scripts/mib.fddi;;
      "File-Systems"    ) /users/fenyo/bin/gawksnmp.V1 -v AGENT=$hostname -
v PORT=$portno -v COMMUNITY=$community -f
/users/fenyo/public_html/snmp/gawk.scripts/mib.fs;;
      Interfaces  ) /users/fenyo/bin/gawksnmp.V1 -v AGENT=$hostname -v
PORT=$portno -v COMMUNITY=$community -f
/users/fenyo/public_html/snmp/gawk.scripts/mib.interfaces;;
      Print       ) /users/fenyo/bin/gawksnmp.V1 -v AGENT=$hostname -v
PORT=$portno -v COMMUNITY=$community -f
/users/fenyo/public_html/snmp/gawk.scripts/mib.print;;
      Protocols   ) /users/fenyo/bin/gawksnmp.V1 -v AGENT=$hostname -v
PORT=$portno -v COMMUNITY=$community -f
/users/fenyo/public_html/snmp/gawk.scripts/mib.protocols;;
      Routes           ) /users/fenyo/bin/gawksnmp.V1 -v AGENT=$hostname -
v PORT=$portno -v COMMUNITY=$community -f
/users/fenyo/public_html/snmp/gawk.scripts/mib.routes;;
      Users       ) /users/fenyo/bin/gawksnmp.V1 -v AGENT=$hostname -v
PORT=$portno -v COMMUNITY=$community -f
/users/fenyo/public_html/snmp/gawk.scripts/mib.users;;
esac

echo "</PRE>"

cat << FIN
<HR>
<A HREF="http://www/~fenyo/snmp">
<iMG ALIGN=MIDDLE SRC="/~fenyo/snmp/arw01lt.gif">
Back to Network Manager home page</A>
<HR>
FIN

cat /users/fenyo/public_html/snmp/signature.html
```

Fichier get-mib-value2.sh

```
#!/bin/sh

cat << FIN
Content-type: text/html

<TITLE>Network Manager</TITLE>
<H1>
<iMG ALIGN=MIDDLE SRC="/~fenyo/snmp/sunset.gif"> <iMG ALIGN=TOP
SRC="/~fenyo/snmp/title.gif">
</H1>
<HR>
<H1><iMG ALIGN=MIDDLE SRC="/~fenyo/snmp/graph05.gif"> SNMP Result</H1>
FIN
```

```
eval "`echo $QUERY_STRING | /users/fenyo/public_html/scripts/scan.yy | sed
's/ /-/g' | sed 's/&/ /g'"

OID=`echo $OID | sed 's/\.\(.*\)$/["\1"]/'`

/users/fenyo/bin/gawksnmp.V1 -v AGENT=$hostname -v PORT=$portno -v
COMMUNITY=$community "BEGIN { print $OID; }"

cat << FIN
<HR>
<A HREF="http://www/~fenyo/snmp">
<iMG ALIGN=MIDDLE SRC="/~fenyo/snmp/arw01lt.gif">
Back to Network Manager home page</A>
<HR>
FIN

cat /users/fenyo/public_html/snmp/signature.html
```

scan.lex -> lex.yy.c -> scan.yy (pour parser les paramètres envoyés par le serveur WWW)

```
%%
\+ putchar(' ');
%[0-9a-fA-F][0-9a-fA-F] {
  int caractere;
  yytext[1]=tolower(yytext[1]);
  yytext[2]=tolower(yytext[2]);
  sscanf(yytext+1, "%x", &caractere);
  putchar(caractere);
  }
%%
```

Fichier snmp-intro/snmp-intro.html

```
<TITLE>SNMP Manager</TITLE>
<H1>
<iMG ALIGN=MIDDLE SRC="/~fenyo/snmp-intro/sunset.gif"> <iMG ALIGN=TOP
SRC="/~fenyo/snmp-intro/title.gif">
</H1>
<HR>

<H1><iMG ALIGN=MIDDLE SRC="/~fenyo/snmp-intro/misc09.gif"> WARNING !</H1>
For some political reasons, please note that this service is <Em>not
available</Em> from outside Télécom Paris. To request special access rights
to this service, please post a request to <Em>fenyo@enst.fr</Em>.
<P>

<!--#exec cmd="/users/fenyo/public_html/scripts/snmp-control.sh"-->

<HR>
<A HREF="http://www/subject.html">
<iMG ALIGN=MIDDLE SRC="/~fenyo/snmp-intro/arw01lt.gif">
Back to ENST by subjects</A>
<HR>
<!--#include virtual="/~fenyo/snmp/signature.html"-->
```

Fichier snmp/.htaccess

```
AuthUserFile /dev/null
AuthGroupFile /dev/null
AuthName AllowENST
AuthType Basic
```

```
<Limit GET>
order deny,allow
deny from all
allow from .enst.fr
</Limit>
```

Fichiers snmp/snmp.html et snmp-secure/snmp.html

```
<TITLE>SNMP Manager</TITLE>
<H1>
<iMG ALIGN=MIDDLE SRC="/~fenyo/snmp/sunset.gif"> <iMG ALIGN=TOP
SRC="/~fenyo/snmp/title.gif">
</H1>
<HR>
<H1><iMG ALIGN=MIDDLE SRC="/~fenyo/snmp/clip04.gif"> Technical
Informations</H1>
<DL>
<DT><i>Domain Name:</I>
<DD><B>enst.fr</B>
<DT><i>Internet Provider:</I>
<DD><B>Renater</B> (AS 1717)
<DT><i>Network Address:</I>
<DD><B>137.194.0.0</B>
<DD>89.c2.0.0
<DT><i>Broadcast Address:</I>
<DD><B>137.194.255.255</B>
<DD>89.c2.ff.ff
<DT><i>Net Mask:</I>
<DD><B>255.255.0.0</B>
<DD>ff.ff.00.00
<DT><i>Subnet Mask:</I>
<DD><B>255.255.254.0</B>
<DD>ff.ff.fe.00
</DL>
<HR>
<H1><A HREF="/~fenyo/snmp/enst-net.gif"><iMG ALIGN=MIDDLE
SRC="/~fenyo/snmp/map-us.gif"></A> Our Network Map</H1>
<iMG ALIGN=MIDDLE SRC="/~fenyo/snmp/enst-net.gif">
<HR>
<H1><A HREF="/~fenyo/snmp/legend.gif"><iMG ALIGN=MIDDLE
SRC="/~fenyo/snmp/misc09.gif"></A> Legend</H1>
<iMG ALIGN=MIDDLE SRC="/~fenyo/snmp/legend.gif">
<P>Red boxes contain SNMPv1 capable agents.
<HR>
<H1><IMG ALIGN=MIDDLE SRC="/~fenyo/snmp/graph05.gif">
Getting a predefined set of MIB values</H1>

<FORM ACTION="http://www.enst.fr/cgi-bin/snmp.sh" METHOD="GET">
<PRE>
Host Name  <INPUT TYPE="text" NAME="hostname" SIZE=20 MAXLENGTH="64"
VALUE="panoramix.enst.fr">
Port Number<INPUT TYPE="text" NAME="portno" SIZE=20 MAXLENGTH="64"
VALUE="161">
Community  <INPUT TYPE="text" NAME="community" SIZE=20 MAXLENGTH="64"
VALUE="public">
MIB Group <SELECT NAME="option">
<OPTION> ARP
<OPTION> Connections
<OPTION> EGP
<OPTION> FDDI
<OPTION> File Systems
<OPTION> Interfaces
<OPTION> Print
<OPTION> Protocols
<OPTION> Routes
```

```
<OPTION SELECTED> System
<OPTION> Users
</SELECT>
          <INPUT TYPE="submit" VALUE="Send request !">
</PRE>
</FORM>

<HR>
<H1><IMG ALIGN=MIDDLE SRC="/~fenyo/snmp/graph05.gif">
Getting a single MIB value</H1>

<FORM ACTION="http://www.enst.fr/cgi-bin/snmp2.sh" METHOD="GET">
<PRE>
Host Name  <INPUT TYPE="text" NAME="hostname" SIZE=20 MAXLENGTH="64"
VALUE="panoramix.enst.fr">
Port Number<INPUT TYPE="text" NAME="portno" SIZE=20 MAXLENGTH="64"
VALUE="161">
Community  <INPUT TYPE="text" NAME="community" SIZE=20 MAXLENGTH="64"
VALUE="public">
Object ID  <INPUT TYPE="text" NAME="OID" SIZE=20 MAXLENGTH="128"
VALUE="sysDescr.0">
          <INPUT TYPE="submit" VALUE="Send request !">
</PRE>
</FORM>

<HR>
<A HREF="http://www/subject.html">
<iMG ALIGN=MIDDLE SRC="/~fenyo/snmp/arw01lt.gif">
Back to ENST by subjects</A>
<HR>
<!--#include virtual="/~fenyo/snmp/signature.html"-->
```

Fichier snmp-secure/.htaccess

```
AuthUserFile /users/fenyo/.htpasswd
AuthGroupFile /dev/null
AuthName Network Manager
AuthType Basic

<Limit GET>
require user private
</Limit>
```

Fichier $HOME/.htpasswd

```
private:VR2zpohD.ysfQ
```

7.7. ANNEXE 7 : code de l'agent DCC SHM

```
/*
   AGENT SNMP POUR LE CONTROLE DCC SHM
   Auteur: Alexandre FENYO

   3 Juillet 1994
*/

#include <stdio.h>

#include "/users/fenyo/grenouille/isode-8.0/snmpV2/smux.h"
#include "/users/fenyo/grenouille/isode-8.0/snmpV2/objects.h"
#include "/users/fenyo/grenouille/isode-8.0/h/tailor.h"
```

```
#include "/users/fenyo/grenouille/isode-8.0/snmpV2/snmp-g.h"

#include <sys/types.h>
#include <sys/time.h>

extern char PY_pepy[];

#define error printf

int debug = 0;

static int snmp_fd = NOTOK;
static char *aParty = "initialPartyId.137.194.160.36.7";
static char *pParty = "/etc/parties/stheno";

static char *roots[] = { "ccitt", "iso", "joint-iso-ccitt" };

int dcc_get_integer(prg, name)
int prg;
char *name;
{
  char filename[256];
  char chaine[256];
  FILE *f;

  sprintf(filename, "/tmp/.shm/%d.%s", prg, name);

  if (NULL == (f = fopen(filename, "r"))) return -1;

  if (NULL == fgets(chaine, sizeof(chaine)-1, f)) return -1;

  fclose(f);
  return atoi(chaine);
}

int dcc_get_string(prg, name, return_value, size)
int prg;
char *name;
char *return_value;
int size;
{
  char filename[256];
  char chaine[256];
  FILE *f;
  int len;

  sprintf(filename, "/tmp/.shm/%d.%s", prg, name);

  if (NULL == (f = fopen(filename, "r"))) return -1;

  if (NULL == fgets(return_value, size, f)) return -1;

  fclose(f);

  len = strlen(return_value);
  if ((len > 0) && (return_value[len-1] == 10)) return_value[len-1] = 0;
  len = strlen(return_value);
  if ((len > 0) && (return_value[len-1] == 13)) return_value[len-1] = 0;
  len = strlen(return_value);
  if ((len > 0) && (return_value[len-1] == 10)) return_value[len-1] = 0;
  return 0;
}



void printoid(oid)
OIDentifier *oid;
```

```
{
  int i;

  for (i = 0; i < oid->oid_nelem; i++) printf((i != (oid->oid_nelem-1)) ?
"%d." : "%d\n", oid->oid_elements[i]);
}

OT      myname2obj (oid)
OID     oid;
{
    register int    i,
                    j;
    register unsigned *ip;
    register OID   nm;
    register OT    ot;

    if (oid == NULLOID
            || oid -> oid_nelem < 1
            || (i = (ip = oid -> oid_elements)[0])
                    >= (sizeof roots / sizeof roots[0])
            || (ot = text2obj (roots[i])) == NULL)
        return NULLOT;

    i = 0;
    while (ot) {
        if ((j = (nm = ot -> ot_name) -> oid_nelem) > oid -> oid_nelem)
            return NULLOT;

        if (bcmp ((char *) ip, (char *) (nm -> oid_elements + i),
                  (j - i) * sizeof *ip))
            ot = ot -> ot_sibling;
        else
            if (oid -> oid_nelem == j
                    || ot -> ot_children == NULLOT)
                break;
            else {
                ot = ot -> ot_children;
                ip = oid -> oid_elements + j, i = j;
            }
    }

    return ot;
}

OI      myname2inst (oid)
OID     oid;
{
    static object_instance ois;
    register OI oi = &ois;

    if ((oi -> oi_type = name2obj (oi -> oi_name = oid)) == NULLOT)
        return NULLOI;

    return oi;
}

int static_getfnx(oi, v, offset)
OI oi;
register struct type_SNMP_VarBind *v;
int offset;
{
static int lastq = -1;

int ifvar;
register OID oid = oi->oi_name;
register OT ot = oi->oi_type;

ifvar = (int) ot->ot_info;
```

```
switch (offset) {
 case type_SNMP_PDUs_get__request:
  if (oid->oid_nelem != ot->ot_name->oid_nelem+1 || oid->oid_elements[oid-
>oid_nelem-1] != 0) return int_SNMP_error__status_noSuchName;
  break;

  case type_SNMP_PDUs_get__next__request:
    if (oid->oid_nelem == ot->ot_name->oid_nelem) {
      OID new;
      if ((new = oid_extend(oid,1)) == NULLOID) return
int_SNMP_error__status_genErr;
      new->oid_elements[new->oid_nelem-1] = 0;
      if(v->name) free_SNMP_ObjectName(v->name);
      v->name = new;
    }
  else return NOTOK;
  break;

 default:
  return int_SNMP_error__status_genErr;
}

switch (ifvar) {
 case 0:
  {
    FILE *f;
    char chaine[512];

    f = popen("/usr/ucb/w", "r");

    fgets(chaine, sizeof(chaine), f);

    pclose(f);

    printf("chaine=%s\n", chaine);
    return o_string(oi, v, chaine, strlen(chaine)-1);
  }

 case 1:
  {
    FILE *f;
    char chaine[512];
    float charge;

    f = popen("/usr/ucb/w | /usr/ucb/head | /bin/awk '{print $10}' | tr , '
'", "r");

    fgets(chaine, sizeof(chaine), f);

    pclose(f);

    sscanf(chaine, "%f", &charge);

    return o_integer(oi, v, (int) (charge*100));
  }

 default:
  return int_SNMP_error__status_noSuchName;
}
}

int static_setfnx(oi, v, offset)
OI oi;
register struct type_SNMP_VarBind *v;
int offset;
{
printf("Should not be called\n");
}
```

```
struct mbstat {
  int champs1;
  int champs2;
} dccstat;

int table_getfnx(oi, v, offset)
OI oi;
register struct type_SNMP_VarBind *v;
{
  int ifnum, ifvar;
  struct mbstat *m = &dccstat;

  OID oid = oi -> oi_name;

  OT ot = oi -> oi_type;

  ifvar = (int) ot-> ot_info;

  switch (offset) {
  case type_SNMP_PDUs_get__request:
    if(oid -> oid_nelem != ot -> ot_name -> oid_nelem +1) return
int_SNMP_error__status_noSuchName;
    ifnum = oid -> oid_elements[oid -> oid_nelem -1];

    if (ifvar >= 16) return int_SNMP_error__status_noSuchName;
    break;

  case type_SNMP_PDUs_get__next__request:
    if (oid -> oid_nelem == ot -> ot_name -> oid_nelem) {
      OID new;

      ifnum = 1;

      if ((new = oid_extend(oid,1)) == NULLOID) return
int_SNMP_error__status_genErr;
      new->oid_elements[new -> oid_nelem -1] = ifnum;

      if (v->name) free_SNMP_ObjectName(v->name);
      v->name = new;
    }

    else {
      int i = ot->ot_name ->oid_nelem;
      ifnum = oid->oid_elements[i]+1;

      if (ifnum >= 16) return NOTOK;

      oid -> oid_elements[i]=ifnum;
      oid->oid_nelem=i+1;
    }
    break;

  default:
    return int_SNMP_error__status_genErr;
  }

if (ifnum > 15) return int_SNMP_error__status_noSuchName;

if (ifvar < 6) {
  if (ifvar == 5) {
    char chaine[512];
    if (!dcc_get_string(ifnum, "variables", chaine, 512)) return
o_string(oi, v, chaine, strlen(chaine));
    else return o_string(oi, v, "Non Existant Program", 20);
  }
  else {
    int res;
```

```
    switch (ifvar) {
    case 1:
      res = dcc_get_integer(ifnum, "active");
      break;

    case 2:
      res = dcc_get_integer(ifnum, "waiting");
      break;

    case 3:
      res = dcc_get_integer(ifnum, "memory");
      break;

    case 4:
      res = dcc_get_integer(ifnum, "variables");
      break;
    }

    return o_integer(oi,v,res);
  }
}
else return int_SNMP_error__status_noSuchName;
}

int table_setfnx(oi, v, offset)
{
printf("Should not be called\n");
}

void init_mib()
{
  register OT ot;

  if (ot = text2obj("dccWho")) ot->ot_getfnx = static_getfnx, ot->ot_setfnx
= static_setfnx, ot->ot_info = (caddr_t) 0;

  if (ot = text2obj("dccLoad")) ot->ot_getfnx = static_getfnx, ot-
>ot_setfnx = static_setfnx, ot->ot_info = (caddr_t) 1;

  if (ot = text2obj("dccIndex")) ot->ot_getfnx = table_getfnx, ot-
>ot_setfnx = table_setfnx, ot->ot_info = (caddr_t) 1;
  if (ot = text2obj("dccActive")) ot->ot_getfnx = table_getfnx, ot-
>ot_setfnx = table_setfnx, ot->ot_info = (caddr_t) 2;
  if (ot = text2obj("dccWaiting")) ot->ot_getfnx = table_getfnx, ot-
>ot_setfnx = table_setfnx, ot->ot_info = (caddr_t) 3;
  if (ot = text2obj("dccMemory")) ot->ot_getfnx = table_getfnx, ot-
>ot_setfnx = table_setfnx, ot->ot_info = (caddr_t) 4;
  if (ot = text2obj("dccVariables")) ot->ot_getfnx = table_getfnx, ot-
>ot_setfnx = table_setfnx, ot->ot_info = (caddr_t) 5;

  if ((snmp_fd = snmp_init("snmp.X11d", "private.1", aParty, pParty, 0, 1,
0)) == NOTOK)
    error("snmp_init: %s", PY_pepy);
  else switch (snmp_trap(NULLOID, int_SNMP_generic__trap_coldStart, 0,
(struct type_SNMP_VarBindList *) 0)) {
  case OK:
    break;
  case DONE:
    snmp_fd = NOTOK;
  case NOTOK:
  default:
    error("snmp_trap: %s", PY_pepy);
  }
}

void do_snmp(pdu, offset)
register struct type_SNMP_GetRequest__PDU *pdu;
```

```
int offset;
{
int idx, status;
object_instance ois;
register struct type_SNMP_VarBindList *vp;
IFP method;

idx = 0;

for (vp = pdu->variable__bindings; vp; vp = vp->next) {
  register OI oi;
  register OT ot;
  register struct type_SNMP_VarBind *v = vp->VarBind;

  idx++;

  if (offset == type_SNMP_PDUs_get__next__request) {
    if (((oi = name2inst(v->name)) == NULLOI)
        && (oi = next2inst(v->name)) == NULLOI) goto no_name;
    if ((ot = oi->oi_type) -> ot_getfnx == NULLIFP) goto get_next;
  }
  else {
    if ((oi = name2inst(v->name)) == NULLOI) goto no_name;
    ot = oi->oi_type;

    if ((offset == type_SNMP_PDUs_get__request ? ot->ot_getfnx : ot-
>ot_setfnx) == NULLIFP) {
    no_name:;
      pdu->error__status = int_SNMP_error__status_noSuchName;
      goto out;
    }
  }

 try_again:;
  switch (offset) {
  case type_SNMP_PDUs_get__request:
    if (!(method = ot->ot_getfnx)) goto no_name;
    break;

  case type_SNMP_PDUs_get__next__request:
    if (!(method = ot->ot_getfnx)) goto get_next;
  break;

 case type_SNMP_PDUs_set__request:
  if (!(method=ot->ot_setfnx)) goto no_name;
  break;

 default:
  goto no_name;
  }

switch(status = (*method)(oi, v, offset)) {
 case NOTOK:
 get_next:;
  oi = &ois;
   for (;;) {
     if ((ot = ot->ot_next) == NULLOT) {
       pdu->error__status = int_SNMP_error__status_noSuchName;
       goto out;
     }
     oi->oi_name = (oi->oi_type = ot) -> ot_name;
     if (ot->ot_getfnx) goto try_again;
   }

 case int_SNMP_error__status_noError:
   break;

 default:
```

```
   pdu->error__status = status;
   goto out;
 }
}

idx = 0;

 out:;
pdu->error__index = idx;

if (offset == type_SNMP_PDUs_set__request) {

  int cor = pdu->error__status ? type_SNMP_PDUs_rollback :
type_SNMP_PDUs_commit;
}

switch (snmp_response()) {
 case OK:
  break;

 case DONE:
  snmp_fd = NOTOK;

 case NOTOK:
 default:
  error("snmp_response: %s", PY_pepy);
  break;
}
}

void doit_snmp()
{
  struct type_SNMP_PDUs *event;

  switch (snmp_wait(&event, NOTOK)) {
  case OK:
    if (!event) return;
    break;

  case DONE:
    snmp_fd = NOTOK;

  case NOTOK:
  default:
    error("snmp_wait: &s", PY_pepy);
    return;
  }

  switch (event->offset) {
  case type_SNMP_PDUs_get__request:
  case type_SNMP_PDUs_get__next__request:
  case type_SNMP_PDUs_set__request:
    do_snmp(event->un.get__request, event->offset);
    break;

  default:
    error("badOperation: %d", event->offset);
    snmp_close();
    snmp_fd = NOTOK;
    break;
  }
}

void main_loop()
{

int action;
```

```
int nfds = 0;
fd_set ifds;
fd_set ofds;

FD_ZERO(&ifds);
FD_ZERO(&ofds);

for (;;) {
  int secs;
  fd_set rfds, wfds;

  secs = NOTOK;

  rfds = ifds;
  wfds = ofds;

  if (snmp_fd != NOTOK) {
    FD_SET(snmp_fd, &rfds);
    if (snmp_fd >= nfds) nfds = snmp_fd + 1;
  }

  if (xselect(nfds, &rfds, &wfds, NULLFD, secs) == NOTOK) {
    error("xselect failed");
  }

  if (snmp_fd != NOTOK && FD_ISSET(snmp_fd, &rfds)) doit_snmp();

  switch (action = *((integer *) *agentAction))
    {
    case AGENT_OTHER:
      break;

    case AGENT_COLDSTART:
    case AGENT_TERMINATE:
      error("%s...", action == AGENT_COLDSTART ? "initiating coldStart" :
"I'll be back");

      snmp_close();
      if (action == AGENT_TERMINATE) exit(0);
      exit(0);

      /* NOTREADCHED */

    case AGENT_WARMSTART:
      switch (snmp_trap(NULLOID, int_SNMP_generic__trap_warmStart, 0,
(struct type_SNMP_VarBindList *) 0)) {
      case OK:
      break;
      case DONE:
      snmp_fd = NOTOK;
      case NOTOK:
      default:
      error("snmp_trap: %s", PY_pepy);
      break;
      }
      goto reset_action;

    case AGENT_DEBUGON:
      goto reset_action;

    case AGENT_DEBUGOFF:
      goto reset_action;

    default:
      error("agentAction reset from %d", *((integer *) *agentAction));

    reset_action:;
```

```
      *((integer *) *agentAction) = AGENT_OTHER;
      break;
    }
}
}

main(ac, av)
int ac;
char **av;
{
if (readobjects("/users/fenyo/memoire/src5/program.defs") == NOTOK)
error("readobjects: %s", PY_pepy);

init_mib();

main_loop();
}
```

7.8. ANNEXE 8 : mosy -1

```
-- automatically generated by mosy 7.1 #366 (milhaud), do not edit!

DCC-MIB DEFINITIONS ::= BEGIN


IMPORTS
      DisplayString
            FROM RFC1213-MIB
      OBJECT-TYPE
            FROM RFC-1212
      enterprises
            FROM RFC1155-SMI;

ProcessCount ::=
    INTEGER

-- created from enst (9407030000Z)

enst OBJECT IDENTIFIER ::= { enterprises  69 }

enstObjects OBJECT IDENTIFIER ::= { enst  1 }

enstAgents OBJECT IDENTIFIER ::= { enst  2 }

dcc OBJECT IDENTIFIER ::= { enstObjects  1 }

dccObjects OBJECT IDENTIFIER ::= { dcc  1 }

dccLoadAverage OBJECT IDENTIFIER ::= { dccObjects  1 }

dccWho OBJECT-TYPE
    SYNTAX  DisplayString
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "General informations about the users and load average."
    REFERENCE
            "General informations about the users and load average."
    ::= { dccLoadAverage  1 }

dccLoad OBJECT-TYPE
    SYNTAX  INTEGER     -- UNITS .001 * number of jobs in the run queue
averaged over 1 minute

    ACCESS  read-write
```

```
    STATUS  mandatory
    DESCRIPTION
            "Load average."
    REFERENCE
            "Load average."
    ::= { dccLoadAverage  2 }

dccProgram OBJECT IDENTIFIER ::= { dccObjects  2 }

dccTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF DccEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "The information table about the DCC programs on this entity."
    ::= { dccProgram  1 }

dccEntry OBJECT-TYPE
    SYNTAX  DccEntry
    ACCESS  not-accessible
    STATUS  mandatory
    DESCRIPTION
            "The program id."
    INDEX   { dccIndex }
    ::= { dccTable  1 }

DccEntry ::=
    SEQUENCE {
        dccIndex
            INTEGER,

        dccActive
            ProcessCount,

        dccWaiting
            ProcessCount,

        dccMemory
            INTEGER,

        dccVariables
            DisplayString
    }

dccIndex OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "A unique identifier of the program."
    ::= { dccEntry  1 }

dccActive OBJECT-TYPE
    SYNTAX  ProcessCount
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "Number of active processes running this program id."
    ::= { dccEntry  2 }

dccWaiting OBJECT-TYPE
    SYNTAX  ProcessCount
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "Number of active processes, running this program id, currently
waiting for a variable."
    ::= { dccEntry  3 }
```

```
dccMemory OBJECT-TYPE
    SYNTAX  INTEGER     -- UNITS bytes

    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "Size of valid shared memory handled by the local memory
manager."
    ::= { dccEntry  4 }

dccVariables OBJECT-TYPE
    SYNTAX  DisplayString
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "Set of \n separated shared variable names known by the local
memory manager."
    ::= { dccEntry  5 }

dccConformance OBJECT IDENTIFIER ::= { dcc  2 }

dccCompliances OBJECT IDENTIFIER ::= { dccConformance  1 }

dccGroups OBJECT IDENTIFIER ::= { dccConformance  2 }

dccLoadAverageGroup OBJECT IDENTIFIER ::= { dccGroups  1 }

dccProgramGroup OBJECT IDENTIFIER ::= { dccGroups  2 }

dccCompliance OBJECT IDENTIFIER ::= { dccCompliances  1 }

dccAgent OBJECT IDENTIFIER ::= { enstAgents  1 }

END
```

7.9. ANNEXE 9 : compilation mosy pour SNMPv1

```
-- automatically generated by mosy 8.0 #1 (mousson), do not edit!

-- object definitions compiled from DCC-MIB

enst                 enterprises.69
enstObjects          enst.1
enstAgents           enst.2
dcc                  enstObjects.1
dccObjects           dcc.1
dccLoadAverage       dccObjects.1
dccProgram           dccObjects.2
dccConformance       dcc.2
dccCompliances       dccConformance.1
dccGroups            dccConformance.2
dccLoadAverageGroup  dccGroups.1
dccProgramGroup      dccGroups.2
dccCompliance        dccCompliances.1
dccAgent             enstAgents.1

dccWho          dccLoadAverage.1 DisplayString   read-only       mandatory
dccLoad         dccLoadAverage.2 INTEGER         read-write      mandatory
dccTable        dccProgram.1     Aggregate       not-accessible  mandatory
dccEntry        dccTable.1       Aggregate       not-accessible  mandatory
dccIndex        dccEntry.1       INTEGER         read-only       mandatory
dccActive       dccEntry.2       INTEGER         read-only       mandatory
dccWaiting      dccEntry.3       INTEGER         read-only       mandatory
dccMemory       dccEntry.4       INTEGER         read-only       mandatory
```

```
dccVariables    dccEntry.5       DisplayString   read-only       mandatory
```

7.10. ANNEXE 10 : compilation mosy pour SNMPv2

```
-- automatically generated by mosy 7.1 #366 (milhaud), do not edit!

-- object definitions compiled from DCC-MIB

enst                 enterprises.69

enstObjects          enst.1
enstAgents           enst.2
dcc                  enstObjects.1
dccObjects           dcc.1
dccLoadAverage       dccObjects.1
dccProgram           dccObjects.2
dccConformance       dcc.2
dccCompliances       dccConformance.1
dccGroups            dccConformance.2

%tc             ProcessCount     INTEGER          ""
dccWho          dccLoadAverage.1 DisplayString    read-only       current
%er             dccWho           0                255
dccLoad         dccLoadAverage.2 INTEGER          read-write      current
%er             dccLoad          1                2147483647
dccTable        dccProgram.1     Aggregate        not-accessible  current
dccEntry        dccTable.1       Aggregate        not-accessible  current
%ei             dccEntry         "dccIndex"
dccIndex        dccEntry.1       INTEGER          read-only       current
%er             dccIndex         1                2147483647
dccActive       dccEntry.2       ProcessCount     read-only       current
dccWaiting      dccEntry.3       ProcessCount     read-only       current
dccMemory       dccEntry.4       INTEGER          read-only       current
%er             dccMemory        1                2147483647
dccVariables    dccEntry.5       DisplayString    read-only       current
%er             dccVariables     0                255

dccLoadAverageGroup  dccGroups.1
dccProgramGroup      dccGroups.2

dccCompliance        dccCompliances.1

dccAgent             enstAgents.1
```

7.11. ANNEXE 11 : L'arbre de la MIB-II

L'arbre de la MIB-II

8. Table des matières

1. Préface 2
2. Analyse de SNMP 4
    2.1. Historique 4
        2.1.1. Simple Gateway Monitoring Protocol 4
        2.1.2. Simple Network Management Protocol 4
        2.1.3. Simple Network Management Protocol Security 6
        2.1.4. Simple Network Management Protocol version 2 6
    2.2. Limites dans SNMPv1 8
        2.2.1. La sécurité 9
        2.2.2. Perte de bande-passante 10
        2.2.3. Pas de communication entre managers 11
        2.2.4. Insuffisances dans les structures 11
        2.2.5. Limitations dans les opérations du protocole 11
        2.2.6. Limitation du nombre de codes d'erreurs 11
    2.3. SNMPv2 11
        2.3.1. Structure des informations gérées. 11
            2.3.1.1. Base de la MIB 12
            2.3.1.2. Définitions de types 13
            2.3.1.3. Macros SNMPv2 13
                2.3.1.3.1. Macro MODULE-IDENTITY 14
                2.3.1.3.2. Macro OBJECT-IDENTITY 15
                2.3.1.3.3. Macro OBJECT-TYPE 16
                2.3.1.3.4. Macro NOTIFICATION-TYPE 18
            2.3.1.4. Conventions textuelles 19
        2.3.2. Concepts d'administration 20
            2.3.2.1. Modèle administratif 21
                2.3.2.1.1. Mécanisme des parties 21
                2.3.2.1.2. Mécanisme des contextes 21
                2.3.2.1.3. Mécanisme des vues 22
                2.3.2.1.4. Mécanisme des contrôle d'accès 22
            2.3.2.2. La base de donnée d'administration - Party MIB 22
        2.3.3. Sécurité 25
        2.3.4. Opérations du protocole 26
            2.3.4.1. Les cinq opérations 27
            2.3.4.2. Les sept types de messages 27
            2.3.4.3. Scénario type 30
        2.3.5. Conclusion 31
3. Les logiciels domaine public 33
    3.1. Le logiciel xmib 33
    3.2. Le logiciel tkined 33
    3.3. Le logiciel gawksnmp 34
4. Une implémentation d'outil réseau avec SNMP 37
5. Une implémentation d'un agent SNMP 38
6. Conclusion 49
7. Annexes 50
    7.1. ANNEXE 1 : écrans xmib 50
    7.2. ANNEXE 2 : écran tkined 53

7.3. ANNEXE 3 : carte du réseau de l'école....55
7.4. ANNEXE 4 : patch gawksnmpV1 57
7.5. ANNEXE 5 : paquets générés par gawksnmp 57
7.6. ANNEXE 6 : W3 Network Manager 61
7.7. ANNEXE 7 : code de l'agent DCC SHM....65
7.8. ANNEXE 8 : mosy -1....74
7.9. ANNEXE 9 : compilation mosy pour SNMPv1 76
7.10. ANNEXE 10 : compilation mosy pour SNMPv2....77
7.11. ANNEXE 11 : L'arbre de la MIB-II....78
8. Table des matières 80