Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

Simple network management protocol et policy-based network management appliqué à un cas pratique: le réseau de l'Office national des pensions

des Touches, Vincent

Award date:
2001

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 24. Apr. 2024

FACULTÉS UNIVERSITAIRES NOTRE-DAME DE LA PAIX, NAMUR

INSTITUT D'INFORMATIQUE

RUE GRANDGAGNAGE, 21, B-5000 NAMUR (BELGIUM)

Simple Network Management Protocol

et

Policy-based Network Management

Vincent des Touches

Appliqué à un cas pratique :
Le réseau de l'Office National des Pensions

Mémoire présenté en vue de l'obtention du grade de
Licencié en Informatique

Année Académique 2000 - 2001

Résumé

Ce document traite de deux concepts utilisés pour la gestion des réseaux. Le plus ancien est celui basé sur le protocole *Simple Network Management Protocol* (SNMP) et le plus récent concerne la gestion des réseaux via des politiques. Le fonctionnement des trois versions de SNMP est décrit de manière assez approfondie et suivi par une description d'une amélioration majeure apportée à SNMP qu'est le *Remote Network Monitoring* (RMON). La gestion des réseaux via des politiques est aussi envisagée de manière théorique afin de mettre en évidence son fonctionnement et ses avantages. C'est sur base du réseau de l'Office National des Pensions qu'une réflexion a été faite sur l'applicabilité des deux concepts de gestion des réseaux décrits dans la partie théorique.

Mots clés : réseau, gestion, SNMP, politique, PBNM, COPS

Abstract

This document discuss about two networks management concepts. The older one is based on the *Simple Network Management Protocol* (SNMP) et the newest is the Policy-based Network Management (PBNM). The working of the three versions of SNMP is depicted in depth and followed by the description of the SNMP major improvement : *Remote Network Monitoring* (RMON). The Policy-based Network Management is also seen in a theoretical way to display his working and advantages. The network of the Office National des Pensions is the starting point for the practice study of this two concepts of network management.

Keywords : network, management, SNMP, policy, PBNM, COPS

Avant-propos

Avant d'entrer dans le vif du sujet, je voudrais, tout d'abord, remercier toute une série de personnes sans qui ce document n'aurait pas pu voir le jour.

En premier lieu, mes remerciements vont à Manoëlle (mon épouse) et à Martin, Thomas et Pauline (nos trois enfants) qui ont, pendant trois ans (deux années de cours du soir et une année consacrée au mémoire), aussi participé à leur manière à la licence en informatique à horaire décalé des Facultés Universitaires Notre-Dame de la Paix (Namur).

Dans le cadre des Facultés, je tiens à remercier M. Olivier Bonaventure (mon promoteur) pour ses conseils judicieux et ses cours orientés réseaux, ainsi que MM. Pierre Materne et Alain Nicolas pour les encouragements et l'aide qu'ils m'ont apportés.

Il me faut aussi remercier les différentes personnes qui, à l'Office National des Pensions, m'ont soutenu et épaulé. Il s'agit des membres présents et passés de l'équipe *Systems and Networks* dont Mr Luc Coppens qui en a la charge, et de MM. Michel Filleul et Louis Arnold responsables du centre informatique.

Je terminerais ces remerciements en ayant une pensée pour ma famille en général (au sens large) et à mes parents en particulier.

Tables des Matières

Introduction

Ce mémoire a été écrit afin d'obtenir le grade de licencié en informatique décerné par les Facultés Universitaires Notre-Dame de la Paix à Namur (FUNDP) et ce , dans le cadre de la licence en informatique à horaire décalé (LIHD). Sur base d'une proposition des FUNDP, le sujet du mémoire a été choisi en accord avec l'organisme au sein duquel je travaille. Le responsable du réseau de données de l'Office National des Pensions (ONP) accepta la proposition en même temps que le sujet du mémoire.

Il fallait cependant que le document final puisse aussi servir pour la gestion du réseau de l'ONP. En effet, le temps et le matériel qui étaient mis à ma disposition devaient être considérés comme un investissement : le mémoire devant servir de base de réflexion pour la mise en place d'une gestion du réseau de données de l'ONP. Le mémoire traitera donc de la gestion des réseaux avec en toile de fond, le réseau de l'ONP.

La gestion des réseaux étant un sujet fort vaste, je me suis dirigé vers une description du protocole de gestion le plus couramment utilisé dans les réseaux de données, le *Simple Network Management Protocol* (SNMP), mais aussi vers le nouveau concept de gestion des réseaux qui est en train de voir le jour, le *Policy-based Network Management* (PBNM) autrement dit la gestion des réseaux via des politiques.

La première partie du mémoire, appelée "étude théorique", sera donc constituée de différents éléments à propos des ces deux concepts de gestion des réseaux de données. Quant à la seconde partie du mémoire, intitulée "étude pratique", elle décrira les possibilités de mise en œuvre de ces deux conceptions théoriques sur le réseau de l'Office National des Pensions.

Le premier chapitre de l'étude théorique sera constitué de la description de différents concepts généraux liés à la gestion des réseaux. Il s'agit des différentes formes de gestion des réseaux, à savoir, la gestion des problèmes, la gestion des comptes, la gestion de la configuration, la gestion des performances et la gestion de la sécurité. Toujours dans ce premier chapitre, on retrouvera une description succincte de différents protocoles de gestion afin de situer le protocole SNMP parmi eux. Je présenterai donc brièvement le modèle de gestion OSI et le modèle de l'ITU-T. On retrouvera ici aussi des informations sur l'origine de SNMP ainsi que sur son évolution jusqu'à aujourd'hui.

C'est dans le deuxième chapitre que l'on retrouvera une description détaillée de chacune des versions de SNMP. Je commencerai par la première version de SNMP en en décrivant les principes généraux, les méthodes de transmission de l'information et les opérations d'échange des données. Ce sont les limites de cette version de SNMP qui clôtureront la description de la version 1. La version 2 de SNMP sera, quant à elle, décrite via les principes généraux, la transmission de l'information de gestion et les opérations d'échange de données qui y sont liés. Il sera aussi question de la coexistence de SNMPv1 et SNMPv2 au moyen d'un gestionnaire bilingue ou d'un agent de proximité.

La description de la troisième et dernière version de SNMP (SNMPv3) est un peu plus ardue que les deux versions précédentes. Je commencerai donc par une vue d'ensemble de cette version suivie de la description de son architecture. Je continuerai par des explications concernant le moteur SNMPv3, par la description des applications, par le format des messages ainsi que par un bref exposé sur les MIB qui y sont liées. Une des principales améliorations apportées par cette version concerne la problématique de la sécurité et du contrôle d'accès. Ces deux points seront mis en évidence par la description du modèle de sécurité (*User-based Security Model* - USM) et du modèle de contrôle d'accès (*View-based Access Control Model* - VACM) proposés par l'*Internet Engineering Task Force* (IETF).

Tandis que la première partie du chapitre 2 traitant de SNMP se clôturera par un point relatif à l'évolution de SNMP et par un résumé concernant les différentes versions de SNMP, la seconde partie de ce chapitre traitera du *Remote Network Monitoring* (RMON). Après l'énoncé de quelques généralités et une description de la MIB II, l'étude de RMON se déclinera en deux phases : une pour chacune des versions de RMON. La première version de RMON sera expliquée via ses concepts de base, les spécifications de la MIB RMON et aussi via les principes de filtres et d'alarmes. En ce qui concerne la deuxième version de RMON, je débuterai par une vue d'ensemble de cette version suivie par une description des différents groupes spécifiés dans la MIB RMON2 qui mettra un terme au chapitre 2 et donc, au premier des deux concepts de gestion des réseaux.

L'étude théorique du deuxième concept de gestion des réseaux, la gestion via des politiques, s'effectuera dans le chapitre 3. Si l'étude de SNMP et RMON a pu se faire sur base de documents "anciens" et de ce fait stables, cela n'a pas été le cas de l'étude de la gestion des réseaux via des politiques. La principale caractéristique des documents ayant servi à cette étude est qu'ils sont très récents, en attente d'une prise de

position des constructeurs ou non stable. Non stable dans le sens où ils sont encore sujets à modification du fait de leur statut *d'Internet-Draft* donné par l'IETF. Après une brève introduction, une partie de ce chapitre est consacrée aux concepts liés à cette méthode de gestion des réseaux. Je commencerai donc par une série de définitions de termes ayant trait à la gestion via des politiques, puis par les exigences et les principes de base. Dans cette partie traitant des concepts, un point plus important sera consacré à l'étude des différentes architectures proposées par les constructeurs ou par les organismes de standardisation. Je parlerai par conséquent des modèles *two-tiers* et *three-tiers*, du modèle multidomaine, du modèle proposé par l'IETF sur base de SNMP. Il y aura aussi un point consacré à la problématique liée à l'hétérogénéité et à l'homogénéité des réseaux.

La suite de cette étude servira à collecter des informations sur la syntaxe et la sémantique des politiques en faisant un état des lieux de la situation actuelle, en décrivant les deux types d'approches rencontrées et en montrant, sur base d'exemples, le principe des *Policy Information Base* (PIB).

Afin de mettre en place une gestion du réseau via des politiques, il est nécessaire de faire appel à des protocoles permettant de distribuer les informations relatives aux politiques entre les différents éléments jouant un rôle plus ou moins actif dans ce type de gestion. Il existe trois façons de le faire : en utilisant, soit le modèle *push*, soit le modèle *pull*, soit un modèle mixte reprenant les deux modèles précédents. Il sera nécessaire, dans le cadre du modèle *pull,* d'étudier le fonctionnement du protocole *Common Open Policy Service* (COPS). Ce chapitre se terminera par une réflexion sur le problème de stockage des informations de gestion lié à la gestion via des politiques.

La seconde partie du mémoire, comme mentionné plus haut, décrira les possibilités de mise en œuvre de ces deux conceptions théoriques sur le réseau de l'Office National des Pensions. Il était donc nécessaire dans un premier temps de décrire l'environnement réseau de l'ONP : c'est ce qui sera fait dans le chapitre 4 de ce document.

Cette description de l'environnement débutera par une présentation générale de l'Office National des Pensions (rôles, fonctions, ...) ainsi qu'une explication succincte des différents services qui en font partie. La description du réseau de l'ONP sera scindée en trois parties : le site central, les sites régionaux et le WAN reliant les différents sites. Une brève description des systèmes informatiques précédera les explications concernant les applications employées par les utilisateurs pour remplir les missions confiées à l'ONP.

Le chapitre 5 de ce document constitue le chapitre le plus important de la partie du mémoire consacrée à l'étude pratique. Le premier point est relatif à la mise en place de SNMP et de RMON. Sur base d'une stratégie de départ, il a fallut configurer les différents éléments actifs du réseau devant être gérés et configurer les stations de gestion.

Afin de mettre en place une gestion plus efficace en terme de qualité de service du réseau, une analyse plus précise de l'utilisation des ressources du réseau devra être effectuée. Cette analyse sera réalisée sur base d'informations provenant d'applications ou via un analyseur de protocoles permettant de mettre en évidence les protocoles utilisés, les hôtes générant du trafic, ainsi que les conversations entre hôtes du réseau.

La dernière partie de ce chapitre sera consacrée à la mise au point de quelques exemples de politiques qui pourraient être mise en œuvre à l'ONP si une gestion du réseau via des politiques était envisagée. Les exemples, au nombre de trois, sont basés sur les informations contenues dans le chapitre 4 et surtout sur le point précèdent consacré à l'analyse du réseau. Comme à l'ONP, il n'y a pas d'infrastructure permettant la mise en place d'une gestion via des politiques, l'implémentation des politiques imaginées dans ce chapitre ne fera pas partie du mémoire.

PARTIE 1 - Etude théorique

Introduction

Dans cette partie, nous allons étudier plus ou moins en détails les différentes possibilités (existantes ou en cours de développement et de standardisation) de gestion des réseaux. Mais il est d'abord nécessaire de mieux cerner ce qu'est la gestion des réseaux en obtenant des réponses à différentes questions simples et de base : que peut-on administrer dans un réseau ? qui peut administrer ? et surtout comment administrer ? C'est dans le chapitre 1 que l'on pourra répondre à ces questions en ayant un aperçu des différentes formes de gestion ainsi qu'un bref état des lieux sur les protocoles de gestion.

Sur base de cet état des lieux, le protocole *Simple Network Management Protocol* (SNMP) sera étudié plus en détails afin de mettre en évidence les différences entre les trois versions successives de ce protocole. Le chapitre 2 sera donc concerné, dans un premier temps, par cette étude, mais aussi par l'étude de *Remote Network Monitoring* (RMON) qui permet d'optimiser la gestion des réseaux via SNMP en ayant une vue plus complète de celui-ci. RMON connaît aussi deux versions qui seront décrites plus en détails.

Le dernier chapitre de cette partie théorique, le chapitre 3, est employé à la description d'une nouvelle philosophie de la gestion des réseaux. Il s'agit de *Policy-based Network Management* ou la gestion des réseaux en se basant sur le principe de politique. Une description des différentes tendances est entreprise ici en tenant compte de l'état d'avancement du développement et de la standardisation de ces nouveaux concepts.

Chapitre 1 : Concepts

La gestion des réseaux consiste à initialiser, surveiller et modifier les différents systèmes qui composent le réseau afin qu'ils puissent supporter les exigences des utilisateurs. Par exemple, permettre l'accès au réseau et tenir à l'œil l'échange des données entre les utilisateurs.

1.1 Les différentes formes de gestion

Afin de garantir le bon fonctionnement des différents systèmes, on subdivise la gestion des réseaux en plusieurs zones : les zones fonctionnelles de gestion OSI. Les renseignements ci-dessous sont retirés de deux sources bien définies : [sta1999] et [puj1997].

1.1.1 La gestion des problèmes (*fault management*)

Afin de garder opérationnel un réseau complexe, il faut le surveiller dans son entièreté mais aussi vérifier le bon fonctionnement de chacun des systèmes qui le composent. Quand un problème survient, il faut dans les plus brefs délais :

- Déterminer où se situe le problème exactement ;
- Isoler le reste du réseau afin qu'il continue à fonctionner sans les interférences dues à la panne ;
- Reconfigurer ou modifier le réseau pour minimiser l'impact dû au(x) composant(s) défectueux ;
- Réparer ou remplacer les composants dans le but de retrouver le réseau tel qu'il était avant le problème ;

Il est nécessaire de définir avec précision le concept de problème (*fault*) qui est à distinguer de celui des erreurs. Un problème est une situation anormale qui nécessite une action ponctuelle comme, par exemple, remplacer ou réparer un composant du réseau. Une erreur, quant à elle, est considérée comme un simple événement. Un problème survient souvent lors d'une panne d'un composant du réseau ou d'un nombre excessif d'erreurs.

La plupart des utilisateurs acceptent les coupures occasionnelles du réseau, mais ils s'attendent à recevoir une notification immédiate du problème et que celui-ci soit traité dans les plus brefs délais. Pour obtenir un haut niveau de fiabilité du réseau, il faut avoir des fonctions de détection et de diagnostic très rapides. On peut aussi minimiser l'impact des problèmes ou pannes en ayant un réseau à tolérance de pannes : les composants sensibles doivent être redondants. Et il en va de même pour la gestion des problèmes et des pannes. Les utilisateurs espèrent être informés des coupures prévues du réseau ou d'un service de celui-ci.

Il est aussi nécessaire, après correction, de s'assurer que le problème est bien résolu : il faut donc mettre en place un système de suivi du problème et un système de contrôle plus performant afin d'éviter que ce même problème ou type de problème ne se représente une nouvelle fois.

1.1.2 La gestion des comptes (*accounting management*)

Dans beaucoup de réseaux d'entreprise, une facturation interne concernant l'utilisation du réseau est effectuée sur base de différents critères : par division de l'entreprise, par centre de coût, par projet, ... Quoi qu'il en soit, le gestionnaire du réseau doit être capable de suivre l'utilisation du réseau par utilisateur ou type d'utilisateurs et ce pour un certain nombre de raisons :

- Un utilisateur final ou un groupe d'utilisateurs finaux peut faire un usage abusif de ses droits d'accès et ainsi surcharger le réseau aux dépends des autres utilisateurs ;
- Un utilisateur final peut mal utiliser le réseau et le gestionnaire du réseau peut ainsi le guider vers des choix plus favorables en terme, par exemple, de performances ;
- Le gestionnaire du réseau peut mieux planifier la croissance du réseau, s'il connaît en détails l'utilisation qu'en font les utilisateurs ;
- ...

Le gestionnaire du réseau doit être capable de mettre en place les différents systèmes qui lui permettront de rédiger de manière automatique des rapports concernant l'utilisation du réseau. Il est nécessaire de savoir quelle information sera stockée pour chaque nœud du réseau, à quelle fréquence elle sera transmise et aussi quel algorithme sera utilisé pour comptabiliser l'utilisation du réseau. De plus, ces informations ne seront manipulées que par un nombre limité de personnes qui en auront les droits.

1.1.3 La gestion de la configuration (*configuration management*)

L'initialisation du réseau, l'arrêt du réseau (dans sa totalité ou non) et l'ajout, la mise à jour et la maintenance des relations entre les différents composants du réseau et de leurs états sont les différentes fonctions attribuées à la gestion de la configuration.

Comme les composants actuels des réseaux peuvent avoir différentes fonctionnalités (par exemple : routeur et système final), il faut que le gestionnaire de la configuration, une fois qu'un choix sur l'utilisation d'un composant a été fait, définisse le software approprié ainsi que les attributs et les valeurs de ceux-ci qui seront pris en compte pour ce composant.

Les opérations de démarrage et d'arrêt du réseau sont sous la responsabilité du gestionnaire de configuration qui connaît la composition du réseau et des connectivités entre les composants. De plus, le gestionnaire doit être capable d'adapter la configuration du réseau en fonction des besoins des utilisateurs. La reconfiguration du réseau se fait bien souvent en réponse à une évaluation des performances ou de la sécurité, lors de l'expansion du réseau ou suite à une panne.

Les utilisateurs finaux ont besoin d'être informés sur l'état du réseau en général, sur l'état des composants, sur les changements apportés à la configuration, ... et ce afin de mener à bien leurs tâches quotidiennes.

1.1.4 La gestion des performances (*performance management*)

La gestion des performances comprend deux grandes catégories de fonctions :

- Les fonctions de surveillance sont utilisées pour suivre les activités sur le réseau ;
- Les fonctions de contrôle permettent à la gestion des performances de faire des ajustements afin de renforcer les performances du réseau.

Y a-t-il trop de trafic ? Quel est la capacité d'utilisation du réseau ? Le débit est-il acceptable ? Y a-t-il des problèmes de congestion ? Etc., ... sont des questions auxquelles le gestionnaire des performances est appelé à répondre. Pour ce faire, il devra se baser sur des associations entre des attributs et leurs valeurs respectives qui devront être significatifs du niveau d'utilisation du réseau. La surveillance d'un grand nombre de ressources fournira les informations nécessaires pour déterminer le niveau d'opération du réseau. La collecte et l'analyse de ces résultats donnent une valeur indicative qui permettra de détecter les modifications de l'utilisation du réseau et par conséquent de ses dégradations.

Les utilisateurs finaux doivent être avertis des temps de réponse minimaux et maximaux qu'ils peuvent attendre du réseau. Il en va de même pour la fiabilité des services réseaux fournis. Pour ce faire, le gestionnaire des performances doit connaître en détails les requêtes faites par les utilisateurs via le réseau.

C'est sur base des statistiques de performance que le gestionnaire du réseau pourra planifier, gérer et effectuer la maintenance du réseau. La détection de problèmes va amener le gestionnaire à appliquer des actions correctives qui peuvent prendre la forme de changements dans les tables de routage pour mieux redistribuer la charge du réseau au travers de zones réseau sous utilisées.

1.1.5 La gestion de la sécurité (*security management*)

La gestion des protections et des droits d'accès, la génération, la distribution et le stockage des clés de chiffrement, la maintenance et la distribution des mots de passe, des autorisations et des accès de contrôle font partie des fonctions attribuées à la gestion de la sécurité. Le contrôle et la surveillance des accès fait sur les ordinateurs ou sur tout autre nœud du réseau sont aussi attribués au gestionnaire de la sécurité. Les journaux, les audits de sécurité sont des outils fort utilisés pour gérer les problèmes de sécurité.

La gestion de la sécurité fournit des protections tant au niveau des ressources du réseau qu'à celui des informations détenues par les utilisateurs. Ces outils ne doivent être mis à la disposition que d'un nombre restreint de personnes et les utilisateurs finaux n'ont besoin que de connaître la politique générale de sécurité appliquée au sein du réseau.

1.2 Les protocoles de gestion des réseaux

1.2.1 Le modèle OSI, le modèle TMN, etc.

Actuellement, il existe plusieurs protocoles utilisés dans la gestion des réseaux. Leur utilisation dépend fortement du type de réseau dans lesquels ils sont intégrés et de leur niveau d'acceptation et de déploiement par l'industrie

des réseaux. Une brève présentation des protocoles OSI et TMN précédera une description plus détaillée du protocole SNMP qui est considéré comme le protocole de fait des réseaux de données.

1.2.1.1 Le modèle OSI

La gestion OSI [puj1997] prend comme principe de faire remonter dans un processus appelé SMAP (*System-Management Application Process*) par l'intermédiaire d'une entité d'application de la couche 7, appelée SMAE (*System Management Application Entity*), toutes les informations de gestion, et de les traiter à ce niveau. D'autres choix auraient pu être faits, comme une entité de gestion à chaque niveau de la hiérarchie qui aurait été capable de prendre les décisions de gestion de ce niveau.

Les informations de gestion sont mémorisées dans une base de données : *Management Information Base* (MIB). Cette MIB est d'une part remplie par les informations provenant des couches de protocoles à gérer, d'autre part elle est consultée par le processus de gestion SMAP. L'entité SMAE récupère les informations demandées par le SMAP via une interface dénommée SMI (*System-Management Interface*) (figure 1-1 a).

La gestion OSI comprend trois types d'activités :

- La gestion système (*System Management*) :
  la gestion système définit l'échange de l'ensemble des informations de gestion concernant les ressources (objets gérés) utilisées dans un système ouvert. Ces échanges se font au niveau 7 de l'architecture OSI entre entités d'application pour la gestion système (SMAE) ;
- La gestion de couche (*Layer Management*) :
  la gestion de couche définit les échanges d'informations concernant la gestion d'une couche (N) particulière. Ces informations ne concernent que les ressources propres à cette couche (mémoires tampons, connexions, etc.). Cette gestion de couche correspond à des protocoles spécifiques, utilisés uniquement pour la gestion.
- La gestion d'opération de couche (*Layer Operation*) :
  la gestion d'opération de couche couvre les échanges d'informations relatives à une instance de communication (une opération) dans une couche donnée. Cela englobe les données véhiculées par les protocoles de communication OSI.

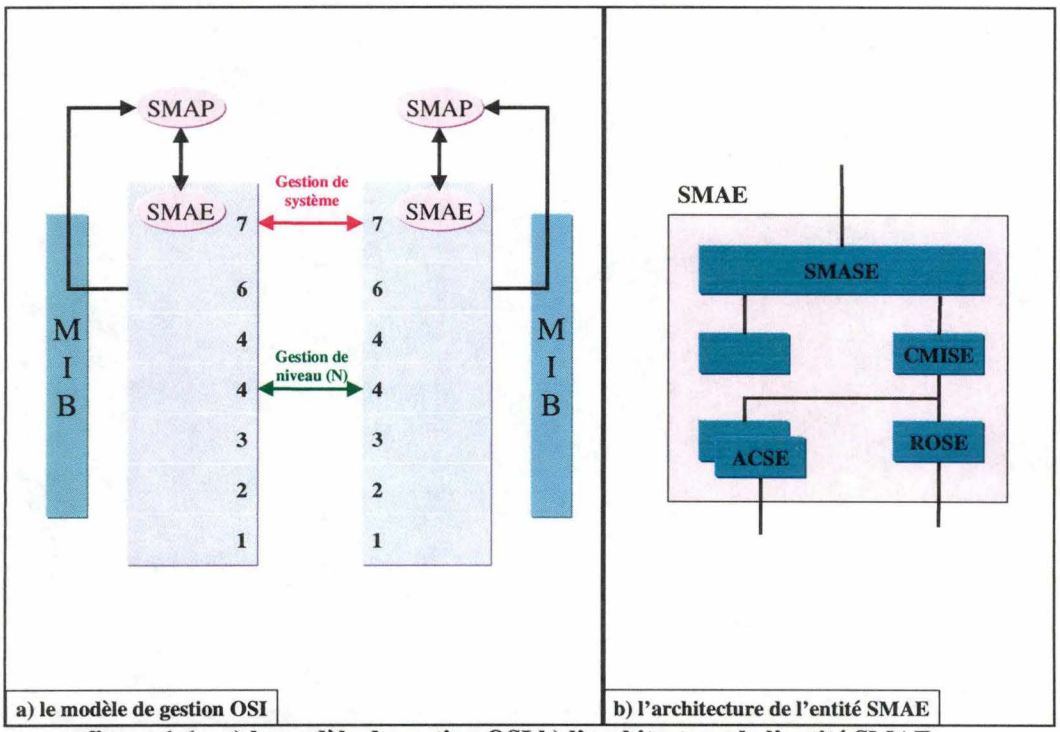

figure 1-1 : a) le modèle de gestion OSI b) l'architecture de l'entité SMAE

La gestion système est au cœur du modèle de gestion OSI [puj1997], c'est là que vont se prendre les décisions de gestion et que sont élaborées les demandes d'informations nécessaires pour réaliser cette gestion. Cette gestion système (figure 1-1 b) est effectuée par l'entité d'application SMAE qui regroupe en général quatre ASE (*Application service Element*) qui s'appuient chacun sur un protocole spécifique :

- Les services d'application de gestion système SMAS (*System-Management application Service*) qui gère la gestion système en s'appuyant sur le protocole MAP (*Management Application Protocol*) qui transporte des MAPDU;
- Les services communs à toutes les fonctions de gestion : CMIS (*Common Management Information Service*) qui utilise le protocole CMIP (*Common Management Information Protocol*) transportant des CMIPDU;
- Les services de contrôle d'association permis par ACSE (*Association Control Service Element*) qui utilise quatre primitives (*associate, release abort, p-abort*) qui sont réalisées par le protocole ACSE *Protocol* qui transporte des ACSE-PDU;
- Les services d'opération à distance permis par ROSE (*Remote Operations Service Element*) qui s'appuie sur cinq primitives (*invoke, result, error, reject-u, reject-p*) réalisées par le protocole ROSE *Protocol* qui transporte des ROSE-PDU.

En résumé, une entité d'application SMAE est constituée d'un élément de service de gestion de systèmes (SMASE), d'un élément de service d'informations de gestion commune (CMISE) et d'un élément de service de contrôle d'association (ACSE). Le SMASE définit la syntaxe et la sémantique de l'information de gestion transférée par des MAPDU. Les services de l'élément ACSE sont utilisés pour initialiser et terminer les associations. Le service de communication utilisé par SMASE peut-être fourni par CMISE ou tout autre ASE (Comme FTAM – *File transfert Access Management* – ou TP - *Transaction Processing*). L'utilisation de CMIS requiert la présence de ROSE, élément de service pour les opérations distribuées qui permet de véhiculer, de manière asynchrone, des échanges de type question/réponse entre sites distants.

Pour compléter cette architecture de gestion du modèle de référence OSI, il faut indiquer que le processus de gestion SMAP travaille sur des fonctions administratives qui ont été regroupées en cinq domaines fonctionnels : la configuration, les pannes, l'audit des performances, la sécurité et la comptabilité. Il faut aussi noter que le processus de gestion SMAP peut être soit un processus gérant, soit un processus agent. Les utilisateurs des processus SMAP que l'on appelle MIS-Users (*Management Information Service User*) peuvent donc être, soit des agents, soit des gestionnaires. Les rôles d'agent et de gestionnaire ne sont pas assignés définitivement, certains MIS-Users peuvent selon les opérations être agent ou gérant.

1.2.1.2 Le modèle de l'ITU-T

L'ITU-T (*International Telecommunication Union – Telecommunication standardization sector*) est un organisme de normalisation qui a décrit une architecture physique et fonctionnelle qui peut prendre en charge la gestion de tous les types de réseaux de télécommunication. Le *Telecommunication Management Network* (TMN) est une norme de l'ITU-T, applicable aux réseaux privés et publics, aux réseaux à commutations de circuits et de paquets et aux équipements associés. Même si l'architecture de TMN est conceptuellement définie pour être une base de travail qui couvre tous les types de réseaux, dans les faits TMN est plutôt orienté vers l'administration des réseaux de type circuits commutés que l'on rencontre dans les environnements de télécommunication.

Le TMN détermine une structure de fonctions, de protocoles et de messages que l'administrateur de réseaux peut sélectionner. Ces ensembles forment les spécifications d'un système TMN. En revanche, TMN ne spécifie pas le système d'administration de réseaux. Il ne donne aucune idée sur l'implantation du système et il ne spécifie pas la manière dont les fonctions TMN sont implantées. Seule est disponible une liste de fonctions qui peuvent être utilisées par l'administration de réseaux. De plus, TMN est applicable uniquement pour l'administration des ressources de communication, il ne l'est pas pour l'administration des applications [puj1997].

TMN identifie cinq catégories de fonctions de gestion. Celles-ci sont définies dans la recommandation X.700 : la gestion des fautes, la gestion comptable, la gestion de configuration, la gestion des performances et la gestion de la sécurité. Cette architecture propose un découpage en couches des fonctions de gestion , quatre grandes catégories ont été déterminées par l'ITU-T : gestion de l'entreprise, gestion des services, gestion du réseau et gestion des éléments.

Le premier niveau concerne les besoins de l'entreprise et de la gestion de l'entreprise au niveau global. Le niveau de gestion de service se préoccupe des points d'accès aux utilisateurs et de l'administration des services offerts aux utilisateurs. Ces services peuvent aussi s'adresser aux fournisseurs de services. Le niveau de gestion du réseau gère les éléments du réseau où le mot élément est pris au sens d'un ensemble d'éléments de base. Le dernier niveau gère cet ensemble d'éléments de base pris individuellement, comme les lignes, les multiplexeurs, les commutateurs, etc. [puj1997].

TMN offre une structure de réseaux définie pour permettre l'interconnexion de différents types de systèmes d'exploitation et des équipements de télécommunication regroupés en architectures hétérogènes. Ceci permet l'administration de différents réseaux et fournit un ensemble de normes à respecter par les constructeurs des

équipements de télécommunication. De façon conceptuelle, c'est un réseau indépendant qui interface les réseaux de télécommunication en différents points pour en recevoir les informations et pour contrôler les opérations.

TMN utilise les architectures normalisées existantes comme le modèle OSI ou le modèle de l'ITU-T pour ATM. Dans le cas du modèle OSI, on retrouve naturellement le modèle de gestion normalisé par l'ISO (CMIS/CMIP). Pour le modèle ITU-T, une adaptation de CMIS/CMIP est effectuée.

1.2.2 Le modèle IETF

1.2.2.1 Les origines de SNMP

Au départ, les hôtes connectés à ARPANET étaient placés dans un environnement composé de programmeurs système et de *designer* de protocoles réseaux. De ce fait, peu d'attention avait été donnée à la gestion du réseau.

A la fin des années 70, il n'y avait pas encore de protocole de gestion des réseaux à proprement parlé. Le seul outil utilisé était *Internet Control Message Protocol* (ICMP) qui fournit un moyen de transmettre des messages de contrôle à partir des routeurs ou d'un hôte vers un hôte destinataire afin d'avoir un *feedback* sur les problèmes liés à l'environnement de travail.
ICMP est accessible sur toutes les entités qui supporte IP. Ses fonctionnalités les plus courantes sont *echo/echo-reply* et timestamp/timestamp-reply. L'exemple le plus courant est *PING* (Packet Internet Groper).

Le nombre d'hôtes connectés à Internet a augmenté de manière très rapide. Cette croissance a eu des répercussions en taille et en complexité. Il a donc été nécessaire de mettre au point un protocole avec plus de fonctionnalités que le *PING* et facilement utilisable par quiconque.

Le point de départ fût le *Simple Gateway Management Protocol* (SGMP) sorti en 1987 (figure 1-2) qui était utilisé pour surveiller les passerelles de connexions (gateways). Le besoin d'un outil plus général a permis de mettre en avant 3 approches différentes :

- *High-Level Entity Management System* (HEMS) : qui était une généralisation du tout premier protocole de gestion des réseaux, le *Host Monitoring Protocol* (HMP) ;
- *Simple Network Management Protocol* (SNMP) : qui est une version avancée de SGMP ;
- *CMIP over TCP/IP* (CMOT) : qui était un essai d'étendre au maximum le protocole (*Common Management Information Protocol* (CMIP)), les services et la structure de la base de données qui était en cours de standardisation par l'ISO pour la gestion des réseaux.

En 1988, l'IAB (*Internet Architecture Board*) a approuvé le développement de SNMP pour le court terme et de CMOT pour le long terme car ils comptaient sur le fait que les entités TCP/IP migreraient vers les protocoles OSI. Pour faciliter cela, l'IAB a décidé que les deux protocoles devraient utiliser la même base de données des objets gérés (c'est-à-dire, avec peu de caractéristiques de base comme les types de variables et leurs attributs).

OSI est devenu de plus en plus compliqué et donc, pour garder SNMP simple, il a été décidé qu'il ne devrait pas suivre les concepts sophistiqués prévus pour CMOT.

1.2.2.2 L'évolution de SNMP

Avec le développement de SNMP, libéré des contraintes OSI, les progrès furent rapides et reflètent l'histoire de TCP/IP. Dés lors, SNMP se trouva disponible sur une large gamme d'équipement.

De plus SNMP devint rapidement le protocole de gestion standardisé de choix pour les utilisateurs. De même que TCP/IP a dépassé les prédictions sur sa durée de vie, SNMP semble rester sur le haut de la vague, tandis que le déploiement généralisé du modèle de gestion des réseaux OSI (CMOT) continue à être retardé. Il faut toutefois aussi mentionné que cet état de fait dépend des environnements car CMIP est utilisé dans les réseaux de type téléphone (mais pas dans les réseaux TCP/IP).

Des travaux sont en cours pour porter l'utilisation de SNMP sur OSI et d'autres suites de protocoles non TCP/IP. En plus de tout cela, les améliorations de SNMP ont été poursuivies dans différentes directions.

Il est probable que la plus importante des initiatives est le développement des capacités de gestion à distance pour SNMP : les spécifications du *Remote Monitoring* (RMON) définissent des ajouts à la base d'informations standards (MIB) de SNMP, ainsi que des fonctions qui exploitent la MIB RMON.

Tout en étant une MIB parmi d'autres, RMON donne au gestionnaire du réseau la capacité de surveiller les sous-réseaux dans leur entièreté, et non plus comme des appareils individuels sur le sous-réseau. Considéré par beaucoup comme une extension de SNMP, RMON a été rapidement et largement déployé.

D'autres extensions à la MIB SNMP ont été développées. Certaines sont indépendantes des constructeurs et concernent d'autres réseaux standardisés (FDDI, Token Ring, ...). Par contre, certaines sont spécifiques aux constructeurs et donc considérées comme des extensions privées de SNMP.

RMON, dont les principes seront développés au point 2.2, représente un peu une limite par rapport aux extensions possibles des fonctionnalités de SNMP. Cependant SNMP (au point 2.1.1) étant appliqué à des réseaux plus larges et plus sophistiqués, fait apparaître des défaillances. Certaines d'entre elles ont été prises en compte dans la version 2 de SNMP (au point 2.1.2). En plus des améliorations apportées à SNMP, il était au départ prévu d'étendre le cadre de travail en y incluant des mécanismes de sécurité et d'administration plus performants. Les RFC qui définissaient les premières spécifications à ce sujet parlaient de *Secure SNMP*.

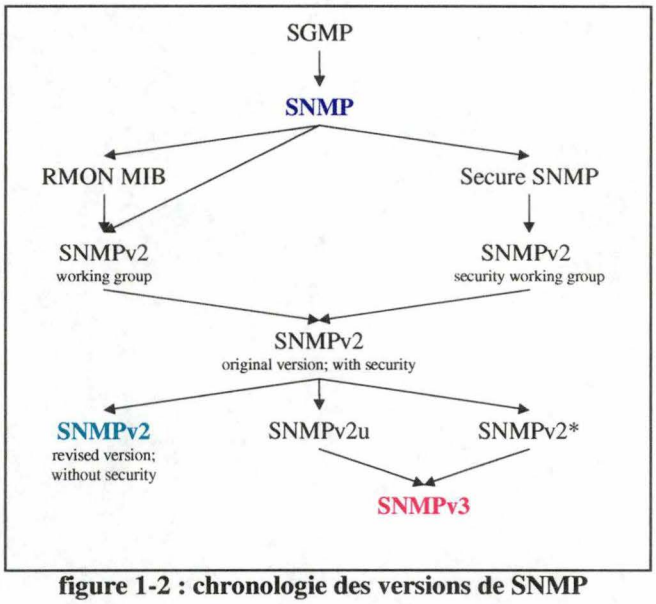

figure 1-2 : chronologie des versions de SNMP

La version suivante, SNMPv2p, qui était passé au stade de *proposed standard* par un groupe de l'IETF, modifiait le cadre de travail administratif basé sur le principe de communauté. Cette version dite *party-based* a aussi tenté d'inclure plus de sécurité. Mais l'industrie ne fût pas encline à la mettre en œuvre, car le nouveau cadre de travail administratif et de sécurité était trop complexe.

Ces désagréments, à propos des aspects administratif et sécurité, ont provoqué la division de SNMPv2 en plusieurs points de vue, dont les différences se situaient au niveau de l'implémentation de la sécurité :

- **SNMPv2u** : SNMPv2 avec la sécurité basée sur le nom des utilisateurs et des opérations de base du protocole snmpv2 ;
- **SNMPv2* (star)** : SNMPv2 avec la sécurité basée sur le nom des utilisateurs, mais avec l'ajout de caractéristiques provenant de la combinaison expérimentale des particularités les plus demandées de SNMPv2p et SNMPv2u ;
- **SNMPv2c** : SNMPv2 sans sécurité qui combine l'approche basée sur la communauté de SNMP avec les opérations de SNMPv2 sans les aspects sécurité de SNMPv2. Cette sous version est largement acceptée et implémentée par les grands produits de gestion.

La création de la version 3 de SNMP (au point 2.1.3), par un groupe mandaté par l'*Internet Engineering Steering Group* (IESG), est le résultat de l'unification des concepts et des mécanismes de SNMPv2u et SNMPv2* en un seul protocole avec un aspect sécurité plus fort. SNMPv3 est donc SNMPv2 avec plus de sécurité et d'administration.

Chapitre 2 : SNMP ou le modèle IETF

2.1 Simple Network Management Protocol

2.1.1 SNMP

2.1.1.1 Principes généraux

SMI, MIB, ASN-1, ...

SNMP est un protocole très spécialisé qui a été créé pour transférer des informations de gestion du réseau entre différentes entités réseau. C'est à dire entre un gestionnaire et des clients que l'on appelle aussi agents [tan1997] Par information, on entend toutes les données utilisées pour surveiller, configurer et contrôler l'état d'un élément du réseau.

SNMP utilise trois langages différents pour transmettre les informations de gestion [mur1998]:

- *Structure of Management Information* (SMI) spécifie le format utilisé pour définir les objets gérés qui seront accédés par le protocole SNMP ;
- *Abstract Syntax Notation One* (ASN-1) est utilisé pour définir le format des messages SNMP et des objets gérés en utilisant un format de description de données non ambigu ;
- *Basic Encoding Rules* (BER) est utilisé pour encoder les messages SNMP en un format valable pour la transmission au travers du réseau.

Comme pour tous les systèmes de gestion des réseaux, le fondement de la gestion d'un réseau basé sur TCP/IP est une base de données contenant l'information sur les éléments devant être gérés. Dans les deux environnements, OSI et TCP/IP, la base de données fait référence à la base d'informations de gestion (Management Information Base : MIB). Chaque ressource à gérer est représentée comme un objet. La MIB est une collection structurée de ce type d'objet.

La *Structure of Management Information* (SMI) définit le cadre général de travail qui permettra de définir et de construire une MIB. La SMI identifie quels types de données peuvent être utilisés dans une MIB et spécifie comment les ressources sont représentées et nommées. La philosophie derrière SMI est d'encourager la simplicité et l'extensibilité dans une MIB (les structures de données complexes, par exemple, ne sont pas supportées par SMI). Cette philosophie est différente du modèle OSI, qui lui, permet l'utilisation de structures de données complexes pour supporter de plus grandes fonctionnalités.

Dans le cadre de travail de SNMP, les informations de gestion sont représentées en utilisant ASN-1. Une MIB consiste en une collection d'objets organisés en groupe. Les objets contiennent des valeurs qui représentent les ressources gérées. La SMI définit donc les types de données de ASN-1 permis et les structures MIB permises. La plupart des objets sont scalaires et donc les types d'objets permis sont `integer`, `octet string`, `null` et `object identifier`. Les types `sequence` et `sequence-of` sont utilisés pour construire de simples tableaux à deux dimensions (il n'y a pas d'autres structures élaborées permises).

Chaque système (station de travail, serveur, routeur, bridge, etc.) dans un réseau ou inter réseau, maintient une MIB qui est le reflet de l'état des ressources gérées sur ce système. Une station de gestion du réseau peut alors surveiller les ressources en lisant les valeurs des objets contenus dans la MIB et peut ainsi contrôler les ressources en modifiant certaines valeurs.

La *Management Information Base* (MIB) renferme l'ensemble des variables gérées et définit :

- Un jeu de variables qui a trait à la fois au matériel et au logiciel ;
- Un jeu de points de tests et de contrôles qu'un système supportant la gestion SNMP est censé vérifier.

Ces variables sont définies sous formes de compteurs, de seuils, de répertoires de noms et d'adresses.

La MIB attribue des noms aux éléments selon une hiérarchie d'enregistrements définie par l'ISO pour la désignation des objets réseaux ; cela définit une structure dans laquelle les variables ayant une relation logique sont regroupées en table. Par exemple, les variables concernant la vitesse, le taux d'erreurs et les adresses des interfaces de communication sont placées dans une table "`interface`". Les objets de la MIB sont classés en une structure hiérarchisée de classes d'objets. Cette structure est aussi appelée *MIB tree* (figure 2-1).

Le premier niveau de classes d'objets de la MIB (MIB I) comprend les huit groupes d'objets suivants (qui ne sont pas toujours implémentés, et ce, en fonction du type d'équipements) :

- `system` : pour la gestion du nœud lui-même ;
- `interfaces` : pour les ports et interfaces réseaux ;
- `address translation` : traduction d'adresses IP ;
- `IP` : Internet Protocol
- `ICMP` : Internet Control Message Protocol ;
- `TCP` : Transport Control Protocol ;
- `UDP` : User Datagram Protocol ;
- `EGP` : Exterior Gateway Protocol.

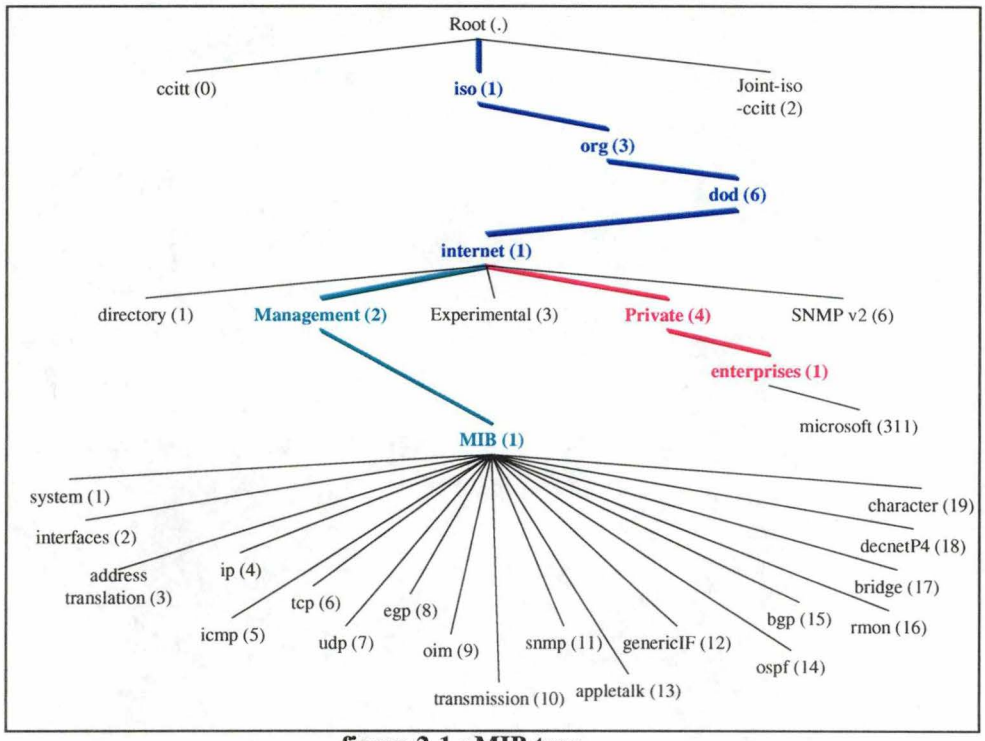

figure 2-1 : MIB tree

Cette structure propose un grand nombre d'entrées permettant de décrire la majorité des objets réseau, mais certaines sociétés ont défini leur propre MIB. Des entrées propres aux constructeurs ont d'ailleurs été prévues dans cette organisation. La MIB II comprend quant à elle, ce qui a été défini dans la MIB I en y ajoutant des attributs ou des objets ainsi que des groupes supplémentaires d'objets (par exemple : `transmission`, `SNMP`, `OSPF` et `RMON`). La MIB II gère environ 200 variables. (La MIB II est plus détaillée au point 2.2.2)

Identifiant et ordre lexicographique.

Chaque objet de la MIB a un identifiant unique qui est défini par la position de l'objet dans la structure en arbre de la MIB. L'identifiant d'un objet est donc une séquence d'entiers qui est le reflet d'une structure hiérarchique ou en arbre de la MIB.

Etant donné cette structure en arbre de la MIB, l'identifiant d'un objet peut-être retrouvé en traçant un chemin à partir de la racine (*Root*) jusqu'à l'objet. Comme les identifiants sont une séquence d'entiers, on parle alors d'un ordre lexicographique.

Cet ordre peut être généré lors du parcours de l'arbre des identifiants de la MIB (du fait qu'un nœud enfant d'un nœud parent est toujours représenté par un ordre numérique croissant). La seule restriction est que l'arbre doit être dessiné de manière telle que les branches en dessous de chaque nœud soient rangées en ordre croissant de gauche à droite. Avec cette convention, on utilise la méthode *Depth-First Search* de parcours d'un arbre (figure 2-2)

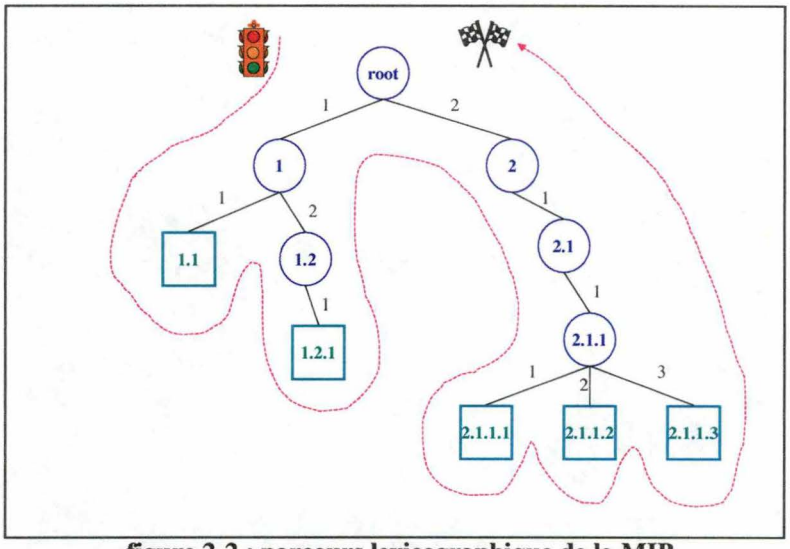

figure 2-2 : parcours lexicographique de la MIB

Une station de gestion du réseau peut ne pas connaître la constitution de la MIB qu'un agent lui fournit. Cette station de gestion peut dés lors avoir besoin de moyens de recherche et/ou d'accès des objets sans spécifier leur nom. Et donc l'utilisation de l'ordre lexicographique permet à une station de gestion de parcourir la structure d'une MIB donnée.

2.1.1.2 La transmission de l'information

Avec SNMP, des informations sont échangées entre une station de gestion et un agent sous forme d'un message SNMP. Chaque message comprend un numéro de version indiquant quelle version de SNMP est utilisée, un nom de communauté (*community name*) utilisé pour cet échange, et un des cinq *protocol data units* (PDU) (figure 2-3).

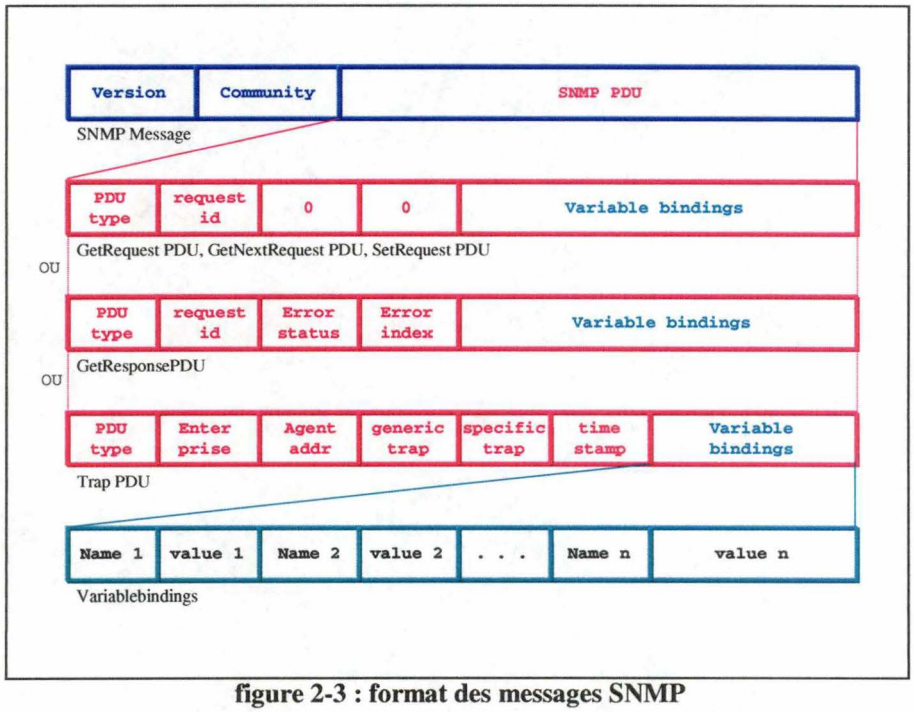

figure 2-3 : format des messages SNMP

Où chaque champs est défini comme suit :

`Version`	Version de SNMP
`Community`	Le nom de communauté qui agira comme un mot de passe pour l'authentification des messages SNMP entre un agent et une application SNMP
`Request-id`	Utilisé pour identifier une requête parmi toutes les autres requêtes
`Error-status`	Utilisé pour indiquer qu'une exception est survenue lors du traitement de la requête
`Error-index`	Si `error-status` est non nul, ce champ permet d'obtenir plus d'informations en indiquant par exemple la variable qui pose un problème
`Variablebindings`	Une liste de noms de variables et la valeur correspondante
`Enterprise`	Type d'objet qui a généré un *`trap`* (basé sur `SysObjetcID`)
`Agent-addr`	Adresse de l'objet qui généré un *`trap`*
`Generic-trap`	Valeurs des types de *`trap`* générique
`Specific-trap`	Code de *`trap`* spécifique
`Time-stamp`	Temps écoulé depuis la dernière (ré)initialisation de l'entité réseau et la génération du *`trap`* (contient la valeur de `SysUpTime`)

En principe, une entité SNMP effectue les actions suivantes pour transmettre une information à une autre entité SNMP :

1) Le PDU est construit sur base de la structure ASN-1 définie dans la RFC 1157 [RFC1157];
2) Ce PDU est alors passé à un service d'authentification, avec les adresses source et destination ainsi que le nom de communauté. Le service d'authentification effectue alors toutes les transformations nécessaires pour cet échange (par exemple cryptage ou l'ajout d'un code d'authentification) et renvoie le résultat ;
3) L'entité protocole construit le message sur base de ce résultat, de la version et du nom de communauté ;
4) Ce nouvel objet ASN-1 est encodé en utilisant le *Basic Encoding Rules* (BER) et est passé au service transport.

L'entité SNMP qui réceptionne un message SNMP, exécute normalement les actions suivantes :

1) Vérification syntaxique de base du message entrant et rejet de celui-ci s'il est erroné ;
2) Vérification du numéro de version et rejet du message entrant s'il y a une incohérence ;
3) L'entité protocole passe le nom de l'utilisateur, la partie PDU du message entrant, et les adresses sources et destinations au service d'authentification.
   Si l'authentification échoue, le service authentification le signale à l'entité protocole de SNMP qui rejette le message et génère un *trap*.
   Par contre si l'authentification réussit, le service d'authentification renvoie le PDU sous forme d'un objet ASN-1 ;
4) L'entité protocole exécute une vérification syntaxique de base du PDU et le rejette si nécessaire. Sinon, en utilisant l'information sur la communauté désignée, l'accès SNMP approprié est choisi et le PDU est traité.

En pratique, le service d'authentification utilise principalement le nom de communauté pour autoriser la réception du message entrant.

En définissant un nom de communauté, un agent limite l'accès à sa MIB à un groupe choisi de stations de gestion. En utilisant plus d'un nom de communauté, l'agent peut fournir différentes catégories d'accès à la MIB, en fonction par exemple de la station de gestion. Un profil de communauté est donc associé à chaque communauté définie par un agent. Le tableau ci-dessous reprend les relations entre la catégorie MIB `ACCESS` et le mode d'accès de SNMP :

MIB Catégorie ACCESS	Mode d'accès SNMP	
	READ-ONLY	READ-WRITE
Read-only	Disponible pour les opérations *get* et *trap*	
Read-write	Disponible pour les opérations *get* et *trap*	Disponible pour les opérations *get, set* et *trap*
Write-only	Disponible pour les opérations *get* et *trap* mais la valeur est spécifique à l'implémentation	Disponible pour les opérations *get, set* et *trap* mais la valeur est spécifique à l'implémentation pour les opérations *get* et *trap*.
Not-accessible	Non disponible	

2.1.1.3 Les opérations d'échange des données

a) GetRequest

C'est l'application de gestion du réseau qui génère le PDU *GetRequest* qui contient les informations suivantes :

- `PDU Type` : indique qu'il s'agit d'un *GetRequest ;*
- `request-id` : un numéro permettant d'identifier de manière univoque la requête : cette information est utile pour traiter les réponses et supprimer par exemple les doubles PDU qui sont générés par un service transport non fiable ;
- `variablebindings` : la liste des instances d'objets dont on demande la valeur.

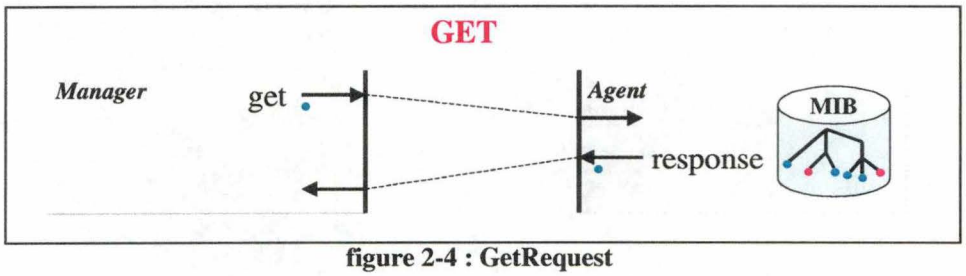

figure 2-4 : GetRequest

L'entité SNMP recevant un *GetRequest* répond par un *GetResponse* (figure 2-4) contenant le même `request-id`. L'opération *GetRequest* est atomique : si l'entité répondant est capable de fournir une valeur pour toutes les variables contenues dans la liste, alors le PDU *GetResponse* contiendra le champs `variablebindings` avec une valeur pour chaque variable. Par contre, si au moins une des valeurs ne peut être fournie, alors aucune valeur ne sera renvoyée.

Différentes erreurs peuvent survenir lors de l'opération *GetRequest* :

1) Soit un nom d'objet contenu dans la liste d'objets du champs `variablebindings` ne correspond pas avec un identifiant d'objet contenu dans la MIB, soit un nom d'objet est un nœud (et pas une feuille de l'arbre) et donc ne contient pas de valeur. Dans ces cas, l'entité répond avec un PDU *GetResponse* contenant `noSuchName` dans le champs `error-status` et dans le champs `error-index` le numéro indiquant la place de la variable (provoquant le problème) dans le champs `variablebindings` ;
2) L'entité qui répond peut donner une valeur pour toutes les variables contenues dans le champs `variablebindings`, mais la taille de la réponse contenue dans le PDU *GetResponse* dépasse la limite attendue. Dans ce cas, l'entité répond avec un PDU *GetResponse* contenant `tooBig` dans le champs `error-status` ;
3) Pour toute autre raison, l'entité qui répond peut ne pas être capable de fournir la valeur d'une instance d'un objet. Dans ce cas, l'entité répond avec un PDU *GetResponse* contenant `genErr` dans le champs `error-status` et dans le champs `error-index` le numéro indiquant la place de la variable (provoquant le problème) dans le champs `variablebindings` ;

Il ne faut pas oublier que SNMP permet seulement de retrouver les objets correspondant à une feuille de l'arbre MIB. Il n'est donc pas possible de retrouver une ligne entière d'une table en ne faisant référence qu'à l'identifiant de la première entrée de la table. La station de gestion doit, pour ce faire, demander la valeur de chacun des objets composant une ligne de la table.

b) Set Request

Le PDU *SetRequest* est envoyé par une entité SNMP (au nom d'une station de gestion du réseau). Il a le même schéma d'échange et le même format que le PDU *GetRequest* (figure 2-5)

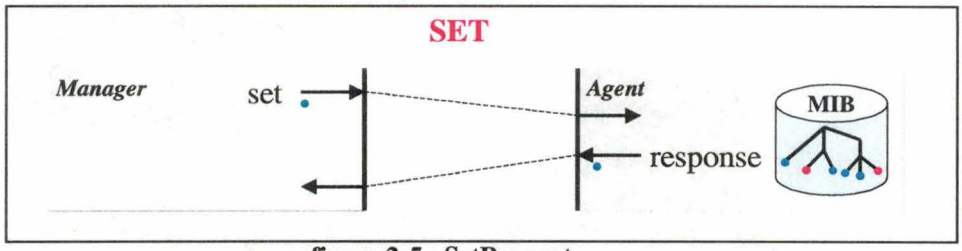

figure 2-5 : SetRequest

Cependant le *SetRequest* est employé pour écrire une valeur d'un objet et non pour la lire. Donc, le champs `variablebindings` est composé aussi bien des identifiants des objets que des valeurs qui devront être assignées à chacune des instances d'objets.

L'entité SNMP recevant un PDU *SetRequest* répond avec un PDU *GetResponse* qui contient le même `request-id`. L'opération *SetRequest* est atomique : soit toutes les variables de la liste sont mises à jour, soit aucune. Si l'entité répondante est capable de mettre à jour toutes les variables listées, elle répond par un *GetResponse* qui contient pour chaque variable la valeur correspondante fournie. Si au moins une des mises à jour ne peut être faite alors aucune n'est faite : les mêmes conditions d'erreurs que le *GetRequest* sont renvoyées (`noSuchName, tooBig, genErr`). Un autre type d'erreurs peut survenir (`badValue`) lorsque qu'il y a une incohérence entre la variable et la valeur qui lui est assignée : par exemple, quand le type, la longueur ou la valeur actuelle sont inconsistant.

c) GetNextRequest

Le PDU *GetNextRequest* est quasi identique au PDU *GetRequest* : il a le même schéma d'échange et le même format (figure 2-6).

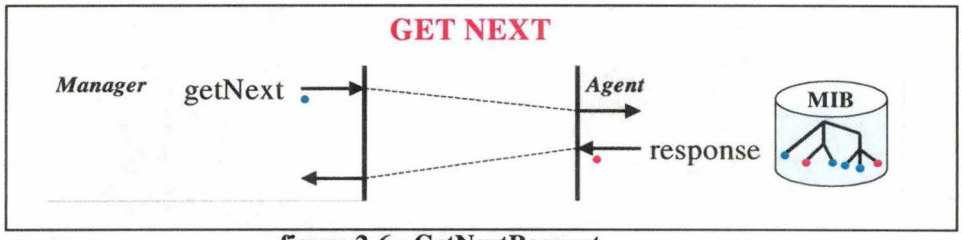

figure 2-6 : GetNextRequest

La seule différence est la suivante : dans *GetRequest* chaque variable du champs `variablebindings` fait référence à une instance d'objet dont la valeur est renvoyée, par contre dans un PDU *GetNextRequest*, l'entité qui répond renvoie la valeur de l'instance de l'objet suivant dans l'ordre lexicographique. *GetNextRequest* est aussi atomique (c'est tout ou rien).

Cette seule différence entre *Get* et *GetNext* permet donc à une station de gestion de découvrir dynamiquement la structure d'une MIB. Il est aussi possible de l'utiliser comme un mécanisme efficace pour lire une table dont on ne connaît pas les entrées.

d) Trap request

Le PDU *trap* est, lui, généré par la partie agent du protocole SNMP (figure 2-7). Il est utilisé pour fournir à la station de gestion une notification suite à un événement significatif. Son format est légèrement différent des autres PDU SNMP.

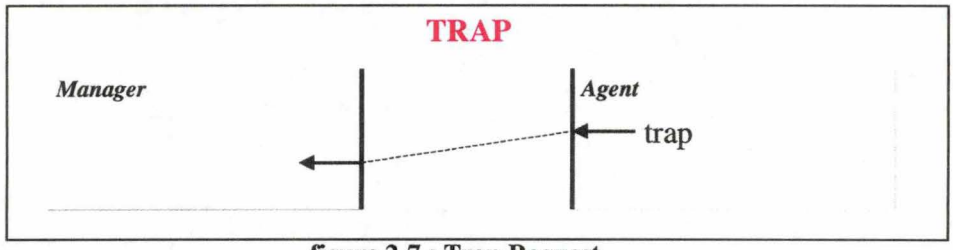

figure 2-7 : Trap Request

Le PDU *trap* contient les champs suivants :

- `PDU Type` : indique qu'il s'agit d'un PDU *trap* ;
- `Enterprise` : identification de l'entité qui a généré cet événement (*trap*). Sa valeur est celle de `sysObjectID` du groupe `System` de la MIB ;
- `Agent-addr` : l'adresse IP de l'agent qui a généré le *trap* ;
- `Generic-trap` : un des types de *trap* prédéfinis ;
- `Specific-trap` : un code qui indique plus spécifiquement la nature de l'événement ;
- `Time-stamp` : le temps qui sépare la dernière (ré)initialisation du système qui a émis le *trap* et l'émission du *trap* ;
- `variablebindings` : contient de l'information supplémentaire sur le *trap*. Cette information dépend spécifiquement de l'implémentation.

Le champ `generic-trap` peut prendre une des sept valeurs suivantes :

- `coldStart (0)` : l'entité SNMP se réinitialise elle-même. De cette manière il se peut que la configuration de l'agent ou que l'implémentation du protocole soit altérée. Cela se produit typiquement lors d'un redémarrage inattendu dû à un crash ou à un problème majeur ;
- `warmStart (1)` : l'entité SNMP se réinitialise elle-même. De cette manière, ni la configuration de l'agent, ni l'implémentation du protocole n'est altérée. Cela se produit typiquement lors d'une routine normale de redémarrage ;
- `linkDown (2)` : signale la perte d'une des lignes de communications de l'agent. Le premier élément du champs `variablebindings` est le nom et la valeur de l'instance `ifIndex` de l'interface référencée ;
- `linkUp (3)` : signale le rétablissement d'une des lignes de communication de l'agent. Le premier élément du champs `variablebindings` est le nom et la valeur de l'instance `ifIndex` de l'interface référencée ;
- `authentificationFailure (4)`: ceci signale que l'entité qui envoie a reçu un message qui n'a pas réussi la phase d'authentification ;
- `egpNeighborLoss (5)` : ceci signale qu'un lien avec l'EGP voisin d'où venait le message SNMP est marqué comme perdu ;
- `EnterpriseSpecific (6)` : signifie que l'entité émettrice reconnaît qu'un événement spécifique est survenu. Le champs `Specific-trap` contient alors le type de *trap.*

Contrairement aux PDU *GetRequest, SetRequest* et *GetNextRequest,* le PDU *trap* n'obtient pas de réponse de la station destinataire.

2.1.1.4 Les limites de SNMP

Les personnes qui se basent sur SNMP doivent être au courant de ses limites. Parmi celles-ci, voici les plus courantes :

1) SNMP ne peut pas convenir pour la gestion de très larges réseaux : car il y a une diminution des performances si on interroge (via Get, Set, ...) un grand nombre d'agents. De plus, cela dépend aussi des performances de la machine de gestion. Avec SNMP, on doit envoyer un paquet d'informations pour recevoir un autre. Ce type d'interrogation donnera alors un large volume de messages à traiter ainsi que des temps de réponse inacceptables.
2) SNMP n'est pas valable pour récupérer de grands volumes de données tel qu'une table de routage en entier.
3) Les *trap* SNMP ne sont pas confirmés. Comme on utilise UDP/IP, un agent ne peut pas être sûr qu'un message critique a bien été reçu par la station de gestion.
4) Le standard SNMP de base ne fournit qu'une authentification insignifiante. Donc SNMP de base est plus valable pour surveiller que pour contrôler.
5) SNMP ne supporte pas les ordres urgents. Le seul moyen pour déclencher un événement, c'est de le faire de manière indirecte, en modifiant la valeur d'un objet.
6) Le modèle MIB de SNMP est limité et ne supporte pas facilement les applications qui font des requêtes complexes basées sur la valeur ou sur le type des objets.
7) SNMP ne supporte pas les communications entre gestionnaires. Par exemple, il n'y a pas de mécanisme permettant à un système de gestion d'obtenir des informations sur des appareils gérés par une autre station de gestion.

Comme UDP est non fiable, il est possible qu'un message SNMP se perde. SNMP n'est pas conçu pour garantir le bon acheminement des messages. Il revient donc à la partie application de gérer la perte des messages :

- Dans le cas d'un *GetRequest* ou d'un *GetNextRequest,* la station de gestion peut se douter qu'il y a une perte de messages si elle ne reçoit pas de *GetResponse* pendant une période de temps donnée. Donc, la station de gestion peut renvoyer la demande une ou plusieurs fois, jusqu'à ce qu'elle reçoive une réponse ou qu'elle décide que l'agent n'est plus actif;
- Comme la réponse contient le même `request-id` que la demande il est facile d'identifier les messages dupliqués et de les rejeter;
- Dans le cas d'un *SetRequest*, il est possible de tester que la modification a bien été exécutée en questionnant l'agent avec un *GetRequest*. Si la réponse ne convient pas, il est alors toujours possible de réitérer le *SetRequest*;
- Par contre, comme il n'y a pas d'acquis lors d'un envoi d'un *trap*, il n'est pas facile de détecter la perte de ce type de messages. Un *trap* est alors plus utilisé pour envoyer une alarme lorsqu'un événement significatif survient. Il est nécessaire que la station de gestion interroge périodiquement l'agent pour en avoir un état fiable.
- ...

2.1.2 SNMP v2

2.1.2.1 Principes généraux de SNMPv2

Pour suppléer aux déficiences de SNMP une actualisation majeure du standard a été publiée, connue comme la version 2 de SNMP (SNMPv2) qui a étendu les fonctionnalités de SNMP et élargi ses capacités pour y inclure tant les réseaux basés sur le modèle OSI que sur le modèle TCP/IP.

SNMPv2 peut supporter aussi bien une stratégie de gestion de réseaux fortement centralisée, qu'une stratégie distribuée. Dés lors, certains systèmes peuvent opérer en tant que gestionnaire et en tant qu'agent. Dans le rôle d'un agent, un tel système va accepter les commandes d'un système de gestion supérieur. Ces commandes ont soit pour but l'accès à l'information stockée localement sur le gestionnaire intermédiaire, soit elles peuvent nécessiter l'intervention de celui-ci pour fournir une information résumée sur les agents qui dépendent de lui. De plus, un gestionnaire intermédiaire peut générer un événement (*trap*) pour informer le gestionnaire supérieur.

La SMI de la version 2 étend le SMI de SNMP dans différentes directions. La *macro* utilisée pour définir les types d'objets a été agrandie pour y inclure plusieurs nouveaux types de données et pour améliorer la documentation associée à un objet. Un changement notable est qu'une nouvelle convention a été fournie pour créer et effacer les rangées conceptuelles dans une table (convention inspirée de la MIB RMON mais plus élaborée).

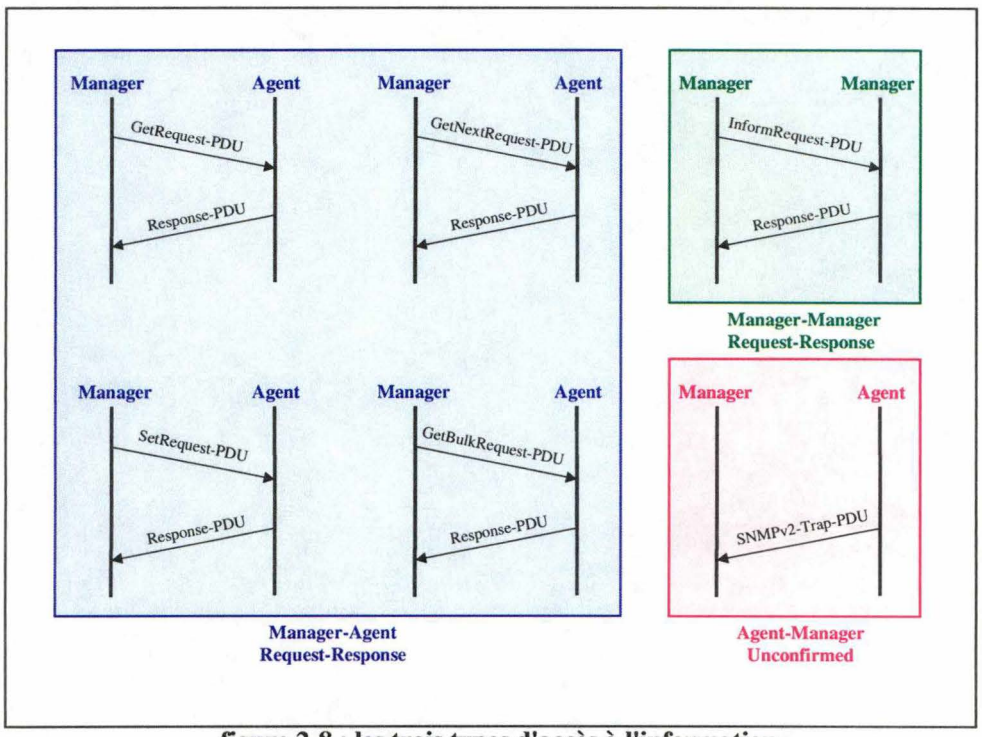

figure 2-8 : les trois types d'accès à l'information

La MIB de SNMPv2 contient les informations du trafic de base à propos des opérations du protocole SNMPv2 (ceci est analogue au groupe `snmp` de la MIB-II). Mais la MIB de SNMPv2 contient aussi des informations au sujet de la configuration d'un agent ou d'un gestionnaire SNMPv2.

Au sujet des opérations du protocole, c'est l'ajout de deux nouveaux PDU qui représente le changement le plus notable : le PDU `GetBulkRequest` et la PDU `InformRequest`.

Comme mentionné plus haut, SNMPv2 est une extension de SNMP. Les PDU sont donc aussi encapsulés dans un message dont le format fournit les fonctionnalités requises pour les caractéristiques de SNMPv2 : la forme et le sens de l'entête du message sont déterminés par un cadre administratif qui définit aussi bien la politique d'authentification que la politique d'intimité (*privacy*).

SNMPv2 accepte trois types d'accès à l'information de gestion (figure 2-8):

- *Manager-Agent Request-Response* : une entité, jouant le rôle de gestionnaire, envoie une demande à une autre entité, jouant le rôle d'un agent, qui répond à la demande. Ce type d'accès est utilisé pour obtenir ou modifier les informations de gestion associées à un appareil ;
- *Manager-Manager Request-Response* : une entité, jouant le rôle de gestionnaire, envoie une demande à une autre entité, jouant le rôle d'un autre gestionnaire, qui répond à la demande. Ce type d'accès est utilisé pour échanger de l'information entre entités de gestion ;
- *Agent-Manager unconfirmed* : une entité, jouant le rôle d'un agent envoie un message, appelé *trap* à une entité (gestionnaire) qui ne lui répond pas. Ce type d'accès est utilisé pour informer le gestionnaire qu'un événement exceptionnel est apparu suite à un changement de l'information de gestion associée à l'appareil géré.

2.1.2.2 La transmission de l'information

Comme pour SNMP, l'information est échangée en utilisant des messages SNMPv2 (figure 2-9)contenant des PDU. L'entête du message contient un nom de communauté qui peut être utilisé pour l'authentification, et un champ `version` qui contient la valeur 1 s'il s'agit de SNMPv2. Le format des messages SNMPv2 fait référence au format de SNMP car actuellement la sécurité fournie par la version 2 de SNMP est basée sur les concepts de communauté de SNMP.

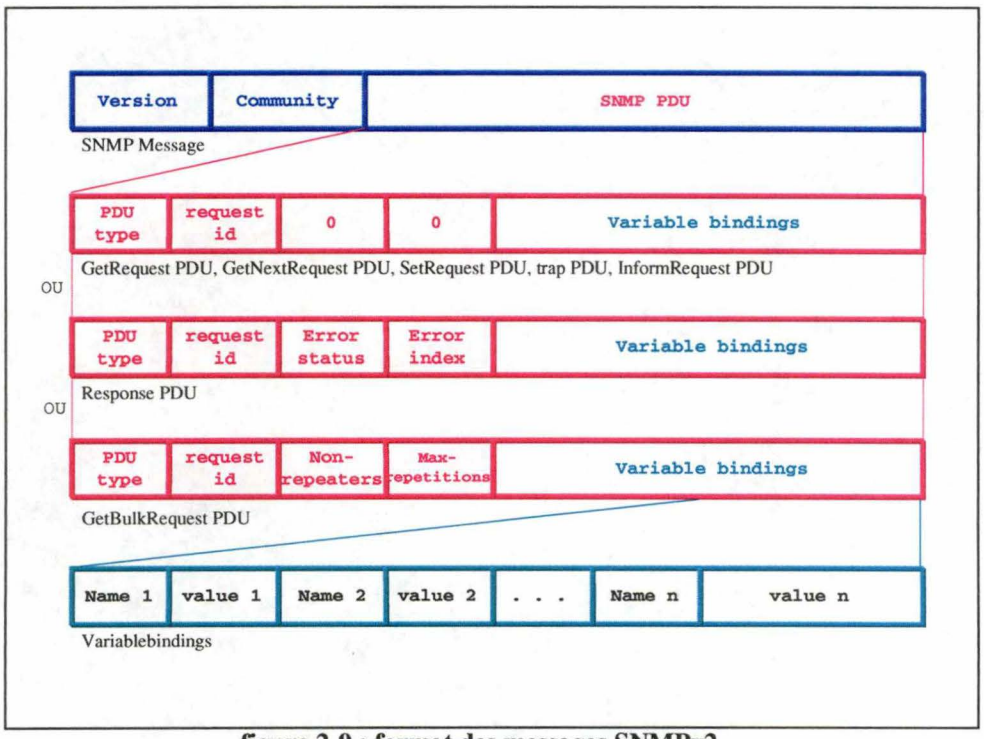

figure 2-9 : format des messages SNMPv2

Avant d'examiner plus en détail les différents PDU, voici un bref commentaire sur chacun des champs contenus dans les PDU de SNMPv2 :

<table>
<tr><td>Request-id</td><td>La valeur de ce champ dans PDU Response doit être égale à celle fournie par la requête (Request PDU). Le gestionnaire peut assigner un numéro unique à chacune des requêtes sortantes pour le même agent afin de pouvoir identifier l'origine des réponses</td></tr>
<tr><td>Error-status</td><td>Une valeur différente de zéro indique qu'une exception est survenue lors du traitement de la requête. Ci-dessous une table reprenant les différents types d'erreur appropriés pour chaque PDU
<table>
<tr><th>Type d'erreur</th><th>Get, GetNext, GetBulk</th><th>Set</th><th>Inform</th></tr>
<tr><td>NoError(0)</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>TooBig(1)</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>NoSuchName(2)</td><td></td><td></td><td></td></tr>
<tr><td>BadValue(3)</td><td></td><td></td><td></td></tr>
<tr><td>Readonly(4)</td><td></td><td></td><td></td></tr>
<tr><td>GenError(5)</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>NoAccess(6)</td><td></td><td>X</td><td></td></tr>
<tr><td>WrongType(7)</td><td></td><td>X</td><td></td></tr>
<tr><td>WrongLength(8)</td><td></td><td>X</td><td></td></tr>
<tr><td>WrongEncoding(9)</td><td></td><td>X</td><td></td></tr>
<tr><td>Wrongvalue(10)</td><td></td><td>X</td><td></td></tr>
<tr><td>NoCreation(11)</td><td></td><td>X</td><td></td></tr>
<tr><td>InconsistentValue(12)</td><td></td><td>X</td><td></td></tr>
<tr><td>Resourceunavailable(13)</td><td></td><td>X</td><td></td></tr>
<tr><td>CommitFailed(14)</td><td></td><td>X</td><td></td></tr>
<tr><td>UndoFailed(15)</td><td></td><td>X</td><td></td></tr>
<tr><td>AuthorizationError(16)</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>NotWritable(17)</td><td></td><td>X</td><td></td></tr>
<tr><td>InconsistentName(18)</td><td></td><td>X</td><td></td></tr>
</table>
Les erreurs (2), (3) et (4) ne sont utilisées que pour des raisons de coexistence avec SNMP.</td></tr>
<tr><td>Error-index</td><td>Quand le champ error-status est différent de zéro, la valeur contenue dans error-index identifie la variable dans variable-bindings qui a généré l'erreur.</td></tr>
<tr><td>Variable-bindings</td><td>Ce champ est constitué d'une séquence de paires, dont le premier élément est un identifiant d'objet. Le second élément de la paire est une des possibilités suivantes :
<table>
<tr><td>Value</td><td>La valeur associée à chacune des instances d'objets spécifiées dans une requête</td></tr>
<tr><td>UnSpecified</td><td>Une valeur nulle est utilisée par des requêtes de réparation</td></tr>
<tr><td>NoSuchObject</td><td>Indique que l'agent n'a pas implémenté l'objet référencé par cet identifiant</td></tr>
<tr><td>NoSuchInstance</td><td>Indique que cette instance d'objet n'existe pas pour cette opération</td></tr>
<tr><td>endOfMibView</td><td>Indique qu'un essai a été fait pour référencer un identifiant d'objet qui est au-delà de la fin de la MIB</td></tr>
</table></td></tr>
</table>

Lors de l'envoi d'un PDU d'une entité SNMPv2 vers une autre entité SNMPv2, les actions suivantes sont effectuées :

1) Le PDU est construit en utilisant la structure ASN.1 définie dans les spécifications du protocole ;
2) Le PDU est alors passé au service d'authentification, en même temps que les adresses transport source et destination, et que le nom de communauté. Le service d'authentification exécute alors toutes les transformations nécessaires pour l'échange (chiffrement, ajout d'un code d'authentification, etc.) et renvoie le résultat ;
3) L'entité protocole construit un message constitué du champ version, du nom de communauté et le résultat de l'étape deux ;
4) Ce nouvel objet ASN.1 est codé en utilisant BER, et est passé au service transport.

NB : (en pratique le chiffrement n'est pas utilisé).

A la réception du message, une entité SNMPv2 exécute les actions suivantes :

1) Contrôle syntaxique de base du message et rejet du message s'il y a un problème ;
2) Vérification du numéro de version et rejet du message s'il y a une discordance ;
3) L'entité protocole passe au service authentification les éléments suivants : le nom de l'utilisateur, la partie PDU du message, les adresses transport source et destination. Si l'authentification échoue le service authentification le signale à l'entité protocole de SNMPv2 qui génère alors un *trap* et écarte le message. Par contre si l'authentification réussit, le service authentification renvoie le PDU sous forme d'un objet ASN.1 correspondant aux spécifications du protocole ;
4) L'entité protocole fait un contrôle de syntaxe du PDU et rejette les PDU s'il y a un problème. Sinon, en utilisant le nom de communauté, la politique d'accès appropriée est choisie pour traiter le PDU.

NB : (en pratique le service authentification sert principalement pour vérifier que le nom de communauté permet la réception de message).

Ci-dessous un tableau reprenant les relations entre la valeur `MAX-ACCESS` de la MIB SNMPv2 et le mode d'accès du protocole.

	Mode d'accès SNMPv2	
MAX-ACCESS	`READ-ONLY`	`READ-WRITE`
Read-only	Disponible pour les opérations *get* et *trap*	
Read-write	Disponible pour les opérations *get* et *trap*	Disponible pour les opérations *get, set* et *trap*
Read-create	Disponible pour les opérations *get* et *trap*	Disponible pour les opérations *get, set, create* et *trap*
Accessible-for-notify	Disponible pour l'opération *trap*	
Not-accessible	Non disponible	

2.1.2.3 Les opérations d'échange des données

a) GetRequest

Le PDU *GetRequest* de SNMPv2 est identique à celui de SNMP du point de vue format et sémantique. La seule différence est la façon dont la réponse est créée : l'opération n'est pas atomique. La liste variable-bindings est préparée même si toutes les valeurs ne peuvent être fournies pour toutes les variables. Si une condition d'exception survient pour une variable, le nom de la variable est associé à une indication d'erreur plutôt qu'à une valeur. Le PDU *Response* SNMPv2 est construit de la manière suivante :

1) Si la variable n'a pas un préfixe (`OBJECT IDENTIFIER`) qui correspond exactement au préfixe de n'importe quelle variable accessible par la requête, alors son champ valeur est mis sur `noSuchObject` ;
2) Sinon, si le nom de la variable ne correspond pas au nom de n'importe quelle variable accessible par la requête, alors son champ valeur est mis `sur noSuchInstance` ;
3) Sinon le champ valeur est égal à la valeur de la variable désignée.

Si le traitement de la variable échoue pour toute autre raison, alors aucune valeur n'est renvoyée. A la place, un PDU *Response* est renvoyé avec un `error-status` ou un `genErr` et la valeur d'index de la variable problématique est placée dans le champ `error-index`.

Si la taille du message contenant le PDU *Response*, dépasse la taille limite locale ou la taille maximum autorisée pour la destination (l'expéditeur du message contenant le PDU *GetRequest*), alors le PDU *Response* est abandonné et un nouveau PDU *Response* est construit. Ce nouveau PDU a un `error-status` égal à `tooBig`, un `error-index` égal à zéro et un champ `variable-bindings` vide.

Avec SNMP, si au moins une des variables posait un problème, l'agent renvoyait un `error-status` égal à `noSuchName` ce qui obligeait le gestionnaire à détecter le problème et à recomposer la requête *GetRequest* en tenant compte du problème. La modification de l'opération *GetRequest* pour permettre les réponses partielles est une amélioration significative pour éliminer cet inconvénient.

b) SetRequest

Le PDU *SetRequest* de SNMPv2 est identique à celui de SNMP en ce qui concerne le format et la sémantique. La seule différence est la façon dont la réponse est manipulée.

Premièrement, l'agent qui répond détermine quelle serait la taille de la réponse compte tenu des informations reçues dans le PDU (`request-id`, `error-status`, `error-index` et `variable-`

bindings). Si cette taille dépasse la limite locale ou le maximum autorisé par l'expéditeur du PDU *SetRequest*, alors un PDU *Response* est construit avec un `error-status` égal à `tooBig`, un `error-index` égal à zéro et le champs `variablebindings` vide. Sinon, un PDU *Response* est construit et les champs contiennent les mêmes valeurs que celles reçues (à l'exception de celles qui ont eu un problème de validation)

Les variables sont, quant à elles, traitées en deux phases. Lors de la première phase, chaque paire de variables qui constitue une opération *set* individuelle, est validée. Si toutes les variables sont validées, alors chaque variable sera modifiée lors de la seconde phase. Celle-ci traitera donc chaque opération *set* de manière individuelle. Donc comme pour SNMP, l'opération *set* de SNMPv2 est atomique : soit toutes les variables sont modifiées soit aucune.

Lors de la phase de validation des variables, les actions suivantes sont exécutées dans cet ordre :

1. Si la variable n'est pas accessible, alors `error-status` est `noAccess` ;
2. S'il n'y a pas de variable qui partage le même identifiant d'objet que ceux contenus dans le champ `variable-bindings` et qu'il n'est pas possible de les créer ou de les modifier, alors `error-status` est `notWritable` ;
3. S'il y a une inconsistance entre le type de la valeur spécifiée et le type de la valeur requise, alors `error-status` est `wrongType` ;
4. S'il y a une inconsistance entre la longueur de la valeur spécifiée et la longueur de la valeur requise, alors `error-status` est `wrongLength` ;
5. Si la valeur contient un code ASN.1 qui est incompatible avec l'étiquette ASN.1 de ce champ, alors `error-status` est `wrongEncoding` ;
6. Si la valeur ne peut être, pour quelque raison que ce soit, assignée à une variable, alors `error-status` est `wrongValue` ;
7. Si la variable n'existe pas et ne pourra jamais être créée, alors `error-status` est `noCreation`;
8. Si la variable n'existe pas et ne pourra pas être créée avec les circonstances actuelles, alors `error-status` est `inconsistentName` ;
9. Si la variable existe et ne peut pas être modifiée alors `error-status` est `notWritable` ;
10. Si la valeur, sous d'autres circonstances, peut être assignée à la variable mais est actuellement inconsistante, alors `error-status` est `inconsistentValue` ;
11. Si l'assignation de la valeur à la variable nécessite l'allocation de ressources qui ne sont pas pour le moment disponibles, alors `error-status` est `resourceUnavaillable` ;
12. Si le traitement d'une des variables échoue pour une raison autre que celles mentionnées ci-dessus, alors, `error-status` est `genErr`;

Si une des conditions ci-dessus est rencontrée sur n'importe laquelle des variables, alors un PDU *Response* est généré avec le champ error-status rempli de manière appropriée et la valeur du champ `error-index` remplie avec l'index de la variable qui a provoqué le problème. De ce fait il est plus facile de déterminer rapidement la cause de l'échec d'une requête et donc de prendre les dispositions nécessaires pour régler le problème. Cela constitue une amélioration par rapport à SNMP.

S'il n'y a pas d'erreur de validation, un essai est fait pour mettre à jour les variables du PDU *SetRequest* : pour chaque variable de la liste, un nom de variable est créé si nécessaire et la valeur spécifiée lui est attribuée. Si une des assignations échoue, alors aucune assignation n'est faite et un PDU *Response* est généré avec le champ `error-status` égal à `commitFailed` et la valeur du champ `error-index` égal à l'index de la variable qui a échoué. S'il n'est pas possible d'annuler les assignations déjà exécutées, alors un PDU *Response* est généré avec le champ `error-status` égal à `undoFailed` et le champ `error-index` égal à zéro.

c) GetNextRequest

Le PDU *GetNextRequest* de SNMPv2 est identique au PDU *GetNextRequest* de SNMP d'un point de vue format et sémantique. La seule différence, comme pour le DPU *GetRequest* vient du fait que le *GetNextRequest* de SNMPv2 traite le plus de variables possibles.

Le PDU *Response* pour un *GetNextRequest* est construit en traitant chaque variable de la liste entrante selon les règles suivantes :

1) La variable déterminée (ciblée) est la suivante dans l'ordre lexicographique par rapport à la variable donnée (entrante). La paire résultante (du champ `variable-bindings`) est composée du nom et de la valeur de la variable déterminée ;

2) S'il n'y a pas de successeur dans l'ordre lexicographique, alors la paire résultante (du champ `variable-bindings`) est composée du nom de la variable entrante dont le champ valeur correspondant est égal à `endOfMibView`.

Si le traitement d'une des variables échoue pour n'importe quelle raison, ou si la réponse est trop grande (`error-status` égal à `tooBig`) alors les mêmes procédures que pour *GetRequest* sont appliquées.

d) Trap Request

Le PDU *trap* est généré et transmis par une entité SNMPv2 jouant le rôle d'un agent, quand un événement inhabituel survient. Ce PDU remplit le même rôle que celui de SNMP mais le format est différent : le PDU utilise le même format que les autres PDU de SNMPv2 à l'exception du PDU *GetBulkRequest*.

Le champ `variable-bindings` contient les paires nom d'objet/valeur suivantes :

- `sysUpTime.0`
- `snmpTrapOID.0` : partie du groupe `trap` de la MIB SNMPv2 ;
- Si la clause `OBJECTS` de la macro `NOTIFICATION-TYPE` est présente alors chaque variable et sa valeur associée sont copiées dans le champ `variable-bindings` ;
- Des variables supplémentaires peuvent être incluses par l'agent.

Comme pour le *trap* de SNMP, il n'y a pas de réponse au PDU *trap* de SNMPv2.

e) GetBulkRequest

Le PDU *GetBulkRequest*, qui est une des améliorations majeures de SNMPv2, a pour but de minimiser le nombre d'échanges nécessaire pour retrouver une grande quantité d'informations de gestion.

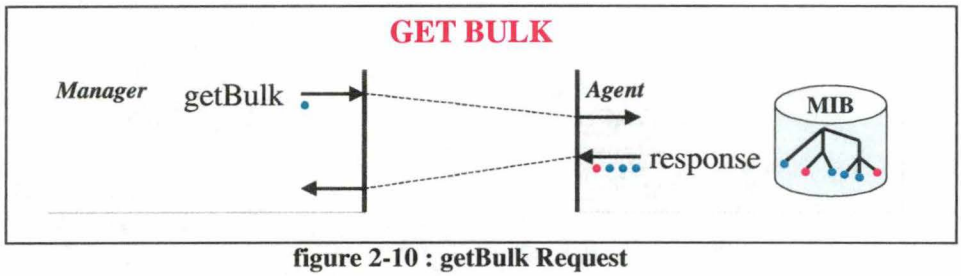

figure 2-10 : getBulk Request

Le PDU *GetBulkRequest* permet à un gestionnaire de demander une réponse aussi grande que possible en tenant compte de la taille maximale d'un message (figure 2-10).

L'opération *GetBulkRequest* utilise les mêmes principes de sélection que l'opération *GetNextRequest* : l'ordre lexicographique permet de déterminer la prochaine instance d'objet. La différence est que, pour *GetBulkRequest,* il est possible de spécifier que plusieurs successeurs (dans l'ordre lexicographique) seront sélectionnés.

Le PDU *GetBulkRequest* contient une liste de (N+R) variables dans le champ `variable-bindings`. Pour chaque N premières variables, la recherche se fait comme pour un *GetNextRequest* : pour chaque variable, la variable suivante dans l'ordre lexicographique et sa valeur sont renvoyées. S'il n'y a pas de successeur, la variable est renvoyée accompagnée de `endOfMibView` comme valeur correspondante. Pour les R variables suivantes, de multiples successeurs lexicographiques sont renvoyés.

Le PDU *GetBulkRequest* a deux champs que l'on ne retrouve pas dans les autres PDU : il s'agit des champs `non-repeaters` et `max-repetitions`. Le champ `non-repeaters` détermine le nombre de variables dans la liste `variable-bindings` pour lesquelles un seul successeur lexicographique doit être renvoyé. Le champ `max-repetitions` détermine le nombre de successeurs lexicographiques qui doivent être renvoyés par le reste des variables de la liste `variable-bindings`.

Pour chacune de ces variables restantes, les actions ci-dessous sont exécutées :

1) Obtenir la valeur du successeur lexicographique de la variable de départ ;
2) Obtenir la valeur du successeur lexicographique de l'instance d'objet retrouvée dans l'étape précédente ;
3) Obtenir la valeur du successeur lexicographique de l'instance d'objet retrouvée dans l'étape précédente ;
4) Répéter l'opération "max-repetitions" fois.

Si, lors d'un des traitements, il n'y a plus de successeurs lexicographiques, alors la valeur `endOfMibView` est renvoyée, associée avec le nom du denier successeur connu ou s'il n'y en a pas avec le nom de la variable contenu dans la requête de départ.

Les successeurs des R dernières variables sont retrouvés niveau par niveau, plutôt que de rechercher tous les successeurs de la première variable, puis ceux de la deuxième variable, etc. Le nombre maximum de paires contenues dans `variable-bindings` est égal à N + (M * R) où M représente `max-repetitions`. Un petit nombre de paires (probablement aucune) peut être généré pour une de ces trois raisons :

1) Si la taille du message contenant le PDU *Response* généré dépasse la limite locale ou la taille maximum autorisée par la partie qui a fait la requête, alors le PDU *Response* est généré avec un plus petit nombre de paires de variables dans le champ `variable-bindings`. Donc le PDU *Response* est construit en plaçant les paires variable/valeur jusqu'au moment où la taille maximum est atteinte ;
2) Si durant la phase de recherche des successeurs, toutes les paires variable/valeur suivantes ont une valeur de `endOfMibView`, alors le champ `variable-bindings` peut-être tronqué à partir de là ;
3) Si le traitement d'un PDU *GetBulkRequest* nécessite beaucoup plus de temps à l'agent qu'une requête normale, alors l'agent peut arrêter le processus après une ou plusieurs itération et ainsi ne renvoyer qu'une partie du résultat.

Si le traitement d'un nom de variable échoue pour toute autre raison que `endOfMibView`, alors aucune valeur n'est renvoyée. A la place, un PDU *Response* est renvoyé avec un `error-status` ou un `genErr` et la valeur d'index de la variable problématique est placée dans le champ `error-index`.

L'opération *GetBulkRequest* supprime une restriction majeure de SNMP, qui lui, est incapable de retrouver efficacement un large bloc de données. Cela permet aussi de déterminer facilement par une procédure *trial-and-error*, le nombre maximum de paires variable/valeur à mettre dans une requête. De même, si une réponse est trop grande, il suffit à l'agent de répondre avec le plus de paires possibles au lieu du message d'erreur tooBig. Donc, le gestionnaire ne doit retransmettre sa requête que pour les données manquantes : il ne doit pas scinder sa requête initiale en plus petites pour obtenir toutes les données.

f) **Inform Request**

Le PDU *InformRequest* est envoyé par une entité SNMPv2 jouant le rôle d'un gestionnaire (figure 2-11), via une application, à une autre entité SNMPv2 jouant aussi le rôle de gestionnaire pour fournir de l'information de gestion à l'application de cette dernière entité.
Le format du PDU *InformRequest* comprend un champ `variable-bindings` avec les mêmes éléments que ceux du PDU *trap* de SNMPv2.

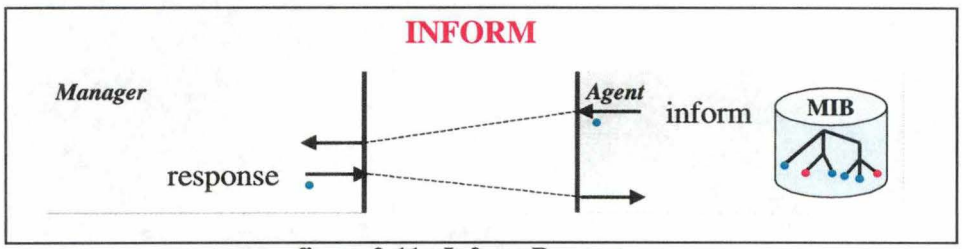

figure 2-11 : Inform Request

Quand une entité reçoit un PDU *InformRequest,* elle détermine d'abord quel serait la taille de la réponse compte tenu des informations reçues dans le PDU (`request-id`, `error-status`, `error-index` et `variable-bindings`). Si cette taille dépasse la limite locale ou le maximum autorisé par l'expéditeur du PDU *InformRequest*, alors un PDU *Response* est construit avec un `error-status` égal à `tooBig`, un `error-index` égal à zéro et le champs `variable-bindings` vide.

2.1.2.4 Coexistence de SNMP et SNMPv2

Comme SNMPv2 est dérivé de SNMP, les concepteurs de la version 2 ont voulu que le passage d'une version à l'autre se fasse en douceur. Pour accomplir cette évolution sur un réseau existant, le meilleur moyen est de mettre à niveau le système de gestion avec la version 2 de SNMP. Ainsi il est possible de faire coexister les gestionnaires SNMPv2, les agents SNMPv2 et les agents SNMP.

SNMPv2 spécifie quelques lignes directrices afin de réaliser cette coexistence de l'information de gestion (SMI) et des opérations de base des deux versions du protocole.

Les modules utilisant le SMI actuel de SNMP doivent encore être utilisés avec le protocole SNMPv2. Pour que les modules MIB de SNMP soient conformes au SMI de SNMPv2, il est nécessaire de faire certains changements. Ces changements qui concernent la définition des objets, des *trap*, de conformité et des capacités, ne sont pas nécessaires pour l'interopérabilité des 2 versions de SNMP.

Au niveau des opérations des protocoles SNMP et SNMPv2, la différence réside principalement dans les deux PDU *GetBulkRequest* et *InformRequest* et aussi dans la possibilité de fournir des résultats partiels.

Afin d'assurer la coexistence, deux possibilités sont offertes : l'utilisation d'un agent *proxy* ou celle d'un gestionnaire bilingue.

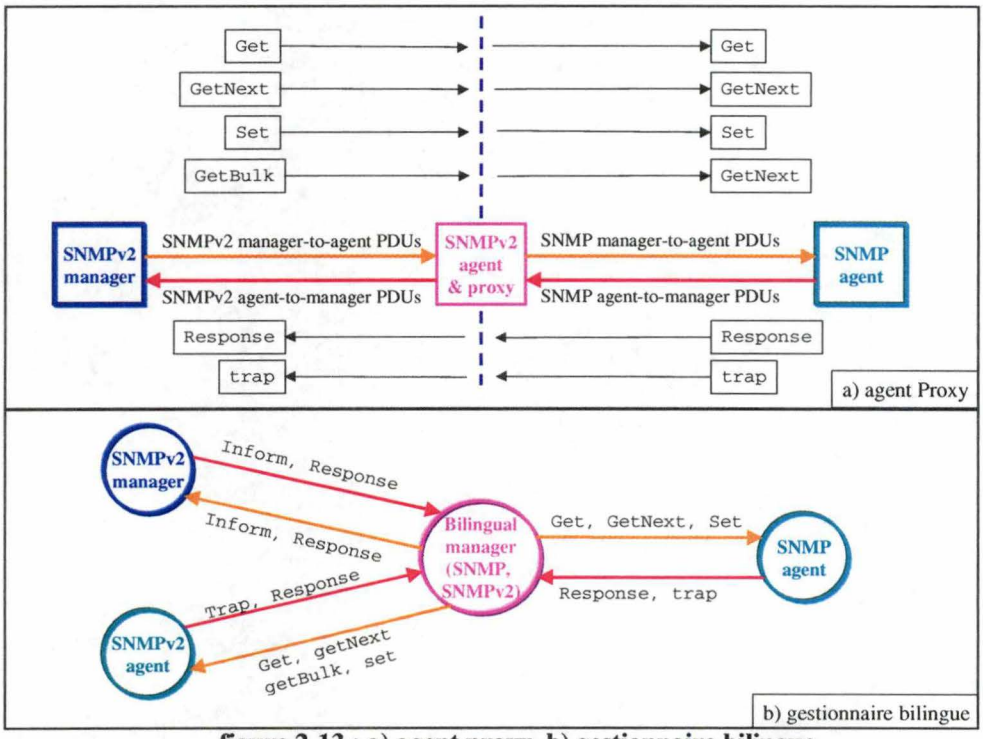

figure 2-13 : a) agent proxy, b) gestionnaire bilingue

Le moyen le plus simple est de laisser les agents SNMP en place et pour les atteindre avec un gestionnaire SNMPv2, on utilise un agent proxy. Un proxy est une entité jouant le rôle d'un agent SNMPv2 qui est configurée pour permettre les conversations entre les protocoles SNMP et SNMPv2. (figure 2-13 a).

L'agent proxy a besoin de transposer les messages d'une version à l'autre et vice versa :

- Les PDU SNMPv2 venant d'un gestionnaire SNMPv2 sont convertis en PDU SNMP pour être envoyés à un agent SNMP en suivant ces deux règles :
  1) Les PDU *GetRequest*, *GetNextRequest* et *SetRequest* sont transmis inchangés ;
  2) Un PDU *GetBulkRequest* est converti en un PDU *GetNextRequest* avec la même `liste variable-bindings`. Le seul effet de cette transposition est que seule la première ligne de la portion `max-repetitions` de `variable-bindings` sera renvoyée.

- Les PDU SNMP venant d'un agent SNMP sont convertis en PDU SNMPv2 pour être transmis à un gestionnaire SNMPv2 selon les règles suivantes :
  1) Un PDU *Response* est transmis inchangé. Les error-status *noSuchName*, *badValue* et *readOnly* ne sont pas utilisés avec SNMPv2. Cependant ces valeurs sont reconnues par un gestionnaire SNMPv2 afin d'interpréter au mieux les réponses d'un agent SNMP.
     Avec SNMP, si une réponse est trop grande, un PDU *Response* est renvoyé contenant le champ `variable-bindings`, alors qu'avec la version 2 du protocole, ce champ est vide. Donc, lors du passage par le proxy, celui-ci retire le champ `variable-bindings` avant de propager la réponse.

Un agent SNMPv2 ne générera jamais un PDU *Response* à un *GetBulkRequest* avec un `error-status` égal à `tooBig` ; car il renvoie autant de paires variable/valeur que possible. Néanmoins un agent proxy va transmettre une réponse à un *GetBulkRequest* avec un `error-status` égal à `tooBig`.

2) Un PDU *Trap* est converti en un PDU *Trap* de SNMPv2. Ceci est fait en ajoutant deux nouvelles paires dans le champ `variable-bindings` : `sysUpTime.0` et `SNMPv2-TrapOID.0`.

Un moyen alternatif pour accomplir cette coexistence consiste à employer un gestionnaire bilingue (figure 2-13 b) : c'est à dire une station de gestion qui "parle" avec les agents des deux versions du protocole. En fonction d'informations sur les agents, stockées localement, le gestionnaire utilise soit les PDU SNMP soit les PDU SNMPv2.

Cette double capacité du gestionnaire n'est visible qu'au niveau SNMPv2/SNMP. Les applications de gestion peuvent être écrites uniquement pour des agents SNMPv2 et la communication avec les agents SNMP se fera comme avec un proxy.

2.1.3 SNMPv3

Le document RFC 2571 [RFC1157] définit une architecture SNMP complète : il décrit les fonctionnalités majeures et les éléments de sécurité qui doivent être pris en compte lors de l'implémentation des agents ou des gestionnaires SNMPv3

2.1.3.1 Vue d'ensemble

Différents buts ont guidé le développement de cette architecture. Parmi ceux-ci on retiendra :

- Tenir compte du travail déjà réalisé et pour lequel des expériences et des consensus ont été dégagés ; l'architecture de SNMPv3 est fortement inspirée des versions SNMPv2u et SNMPv2* ;
- La nécessité de sécuriser le message *SetRequest* car le manque de sécurité est une des défaillances les plus importantes de SNMP et de SNMPv2 ;
- Modéliser une architecture qui :
  (a) Permettra l'implémentation sur un large spectre d'environnements opérationnels dont certains auront besoin de fonctionnalités minimales et peu coûteuses et d'autres devront supporter des caractéristiques supplémentaires pour gérer de plus larges réseaux ;
  (b) Permettra de faire avancer certaines parties de l'architecture au point de vue des standards, même si un consensus n'est pas atteint pour toutes les parties de l'architecture ;
  (c) S'adaptera aux modèles de sécurité alternatifs.
- Garder SNMP aussi simple que possible.

En se basant sur ces buts, différentes décisions ont été prises lors de l'élaboration des documents traitant de SNMPv3. On retrouve l'entièreté de ces décisions dans la RFC 2571 [RFC1157], mais ci-dessous on en retrouvera les points principaux :

- Architecture : Une architecture doit être définie ; elle identifie les limites conceptuelles entre les documents. Les sous-systèmes doivent aussi être déterminés pour définir les services abstraits fournis par une partie spécifique du cadre SNMP. Les interfaces des services abstraits, définis comme des primitives de services, spécifient des limites abstraites entre les documents. Il en va de même pour les services abstraits qui sont fournis par les sous-systèmes conceptuels du cadre SNMP.
- *Documents auto-indépendants (Self-contained documents)* : les éléments des procédures et les objets MIB (qui sont nécessaires pour le traitement d'une portion spécifique du cadre SNMP) doivent être définis dans le même document et si possible non référencés ailleurs. Ceci permet de concevoir et de documenter des parties consistantes et indépendantes de SNMP. Cela correspond à l'approche générale des modules MIB. Cette conception permet de modifier certaines parties sans avoir d'impact direct sur les autres parties et donc les différents documents peuvent suivre indépendamment le chemin de la standardisation.
- *Remote configuration* : le sous-système sécurité a besoin de changer fréquemment des données "secrètes" sur plusieurs entités SNMP. Pour faciliter ce déploiement dans de grands environnements, ces paramètres SNMP doivent pouvoir être modifiés à distance.
- *Controlled complexity* : les constructeurs de systèmes de gestion veulent que les ressources allouées à SNMP soient minimales. Cependant, il y a un besoin d'avoir des configurations plus complexes qui peuvent donc utiliser plus de ressources et fournir plus de fonctionnalités. Un compromis a été trouvé : permettre aux environnements complexes d'étendre logiquement les environnements simples.

- Menaces : le sous-système sécurité doit protéger les entités contre les principales menaces :
  - **Modification de l'information** : une entité peut altérer un message en transit, généré par une autre entité autorisée, dans le but d'effectuer des opérations de gestion non-autorisées comme la modification de la valeur des objets. Cela comprend aussi la modification des paramètres de gestion, de la configuration, des opérations et des comptes ;
  - **Mascarade** : une station de gestion qui n'a pas les autorisations sur une entité peut essayer de se faire passer pour une autre station de gestion qui elle, a les autorisations nécessaires ;
  - **Modification du flux de message** : SNMP est conçu pour travailler avec un protocole transport sans connexion. Donc, il y a une menace que les messages soient réordonnés, retardés ou rejoués avec pour effet de provoquer des opérations de gestion non-autorisées ;
  - **Révélation** : une entité peut observer les échanges entre un gestionnaire et un agent, et ensuite "prendre note" des valeurs des objets gérés ainsi que d'être averti d'événements significatifs.

SNMPv3 n'est pas sécurisé contre les deux types de menaces suivants :

1) Refus de Service (*Denial of service*) : un assaillant peut empêcher l'échange de données entre gestionnaire et agents. Cela peut se comprendre car il est difficile de distinguer une attaque de refus de service d'une panne au niveau du réseau et que de plus, un refus de service atteint tous les échanges de données, pas seulement ceux qui concernent SNMP ;
2) Analyse de trafic : un assaillant peut observer le modèle général du trafic entre les gestionnaires et les agents. Le modèle de trafic est souvent prévisible pour SNMP en général car les messages sont presque toujours générés par une ou plusieurs même stations de gestion.

Plusieurs RFC (2571 à 2575) [RFC2571-RFC2575] font référence collectivement à SNMPv3, mais cela est insuffisant pour définir une implémentation complète de SNMPv3. Le RFC 2571 [RFC2571]contient donc les informations sur les autres documents qui seront utilisés conjointement pour définir une implémentation complète de SNMPv3 et aussi pour servir de guide lors de la création de documents supplémentaires. Comme montré sur la figure 2-14, les documents sont classés en différentes catégories.

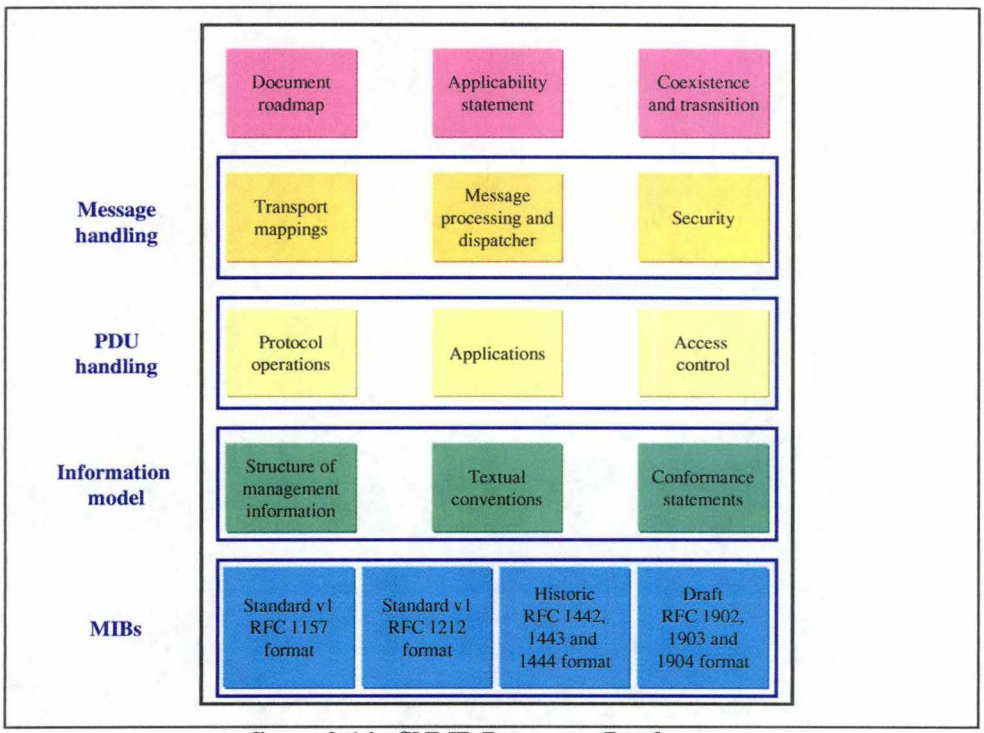

figure 2-14 : SNMP *Document Roadmap*

Les trois documents suivants sont planifiés pour une publication ultérieure :

- *Document Roadmap* : ce document contiendra les RFC en cours d'utilisation ainsi que les indications relatives à leurs relations ;
- *Applicability Statements* : ce document décrit quels sont les environnements appropriés pour certaines versions de SNMPv3 et ceux qui ne le sont pas ;
- *Coexistence and Transition* : ces documents fournissent une aide permettant la coexistence des implémentations des différentes versions de SNMP et la transition graduelle d'une version à l'autre. (Actuellement, il n'existe que le RFC 1908 [RFC1908] qui décrit la coexistence entre SNMP et SNMPv2).

Les autres documents sont repris dans quatre catégories différentes :

1) ***Message Handling*** (traitement des messages) : le message est le bloc de données le plus à l'extérieur traité par SNMP. C'est le message qui est transmis au travers du réseau au moyen des couches plus basses (comme UDP). A la réception d'un message, l'entête est traitée en premier lieu pour examiner plus en détails le PDU qui y est inclus, et à l'envoi du message, l'entête est créée pour le PDU sortant. L'entête contient les informations nécessaires à la sécurité (pour déterminer les opérations de sécurité à prendre en compte pour le message). Les documents sont, dans cette catégorie, subdivisés en trois parties :
   - *Transport Mappings* : ce document décrit la façon dont les messages SNMP sont transcrits dans la couche transport de différents protocoles ;
   - *Message Processing and Dispatcher* : ce document décrit le format des messages, mais il définit aussi un ou plusieurs modules MIB qui seront utiles pour le traitement des messages et pour mesurer le trafic à ce niveau ;
   - *Security* : un document "sécurité" décrit les fonctions de sécurité comme l'authentification et l'intimité/intrusion, et définit la sémantique des champs de l'entête ayant trait à la sécurité.

2) ***PDU Handling*** (traitement des PDU) : les documents de cette catégorie traite du traitement du PDU contenu dans le message. Ils sont subdivisés en trois groupes :
   - *Protocol Operations* : ce document décrit un jeu de formats de PDU correspondants aux fonctions spécifiques de gestion et définit les opérations du protocole en respectant le traitement des PDU ;
   - *Applications* : ce type de document définit les fonctions supportées au moyen des PDU SNMP. Dans ce contexte, une application est une série de fonctions bas niveau, comme les commandes *get* et *set,* les notifications et les opérations du proxy ;
   - *Access Control* : le contrôle d'accès est une fonction de sécurité exécutée au niveau du PDU. Un document de ce type définit les mécanismes qui permettent d'accéder ou non aux objets gérés.

3) ***Information Model*** : ces documents définissent la structure et la syntaxe des informations de gestion. Ils sont aussi subdivisés en 3 groupes :
   - *Structure of Management Information* : un document SMI définit la syntaxe utilisée lors de la construction de MIB et lors de la définition des objets gérés ainsi que les autres éléments de la base d'informations de gestion ;
   - *Textual Conventions* : les conventions textuelles sont utilisées pour améliorer la clarté de lecture d'une spécification MIB. Une convention définit un nouveau type syntaxique similaire à ceux définis dans SMI, mais avec une sémantique plus précise ;
   - *Conformance Statements* : ce document définit ce qu'est une implémentation minimale acceptable et fournit une note pour indiquer le niveau actuel de l'implémentation terminée.

4) ***MIB*** : les documents MIB décrivent les collections d'objets gérés qui contiennent les représentations de l'information de gestion.

2.1.3.2 Architecture

L'architecture SNMP consiste en une collection distribuée et interagissant d'entités SNMP. Chaque entité implémente une portion de SNMP et peut alors jouer le rôle d'un agent, d'un gestionnaire ou des deux.

Chaque entité SNMP consiste en une collection de modules qui interagissent les uns avec les autres pour fournir des services. Ces interactions peuvent être modélisées comme une série de primitives abstraites et de paramètres.

Chaque entité SNMP contient un simple moteur SNMP qui implémente les fonctions d'envoi et de réception des messages, d'authentification et chiffrement/déchiffrement des messages, et le contrôle d'accès aux objets gérés. Ces fonctions sont fournies comme services à une ou plusieurs applications qui sont configurées avec le moteur SNMP pour former une entité SNMP.

L'entité est constituée comme une collection de modules discrets qui fournissent certains avantages : le rôle d'une entité est déterminé par les modules dont elle est constituée. Un certain nombre de modules sont requis pour un agent SNMP, alors qu'un autre jeu de modules est nécessaire pour un gestionnaire. Cette conception en module permet aussi de définir des versions différentes de chaque module afin d'utiliser des versions alternatives ou améliorées sans devoir passer par une nouvelle version du standard. Ceci est facilité par la possibilité de coexistence de différentes versions et par les stratégies de transition.

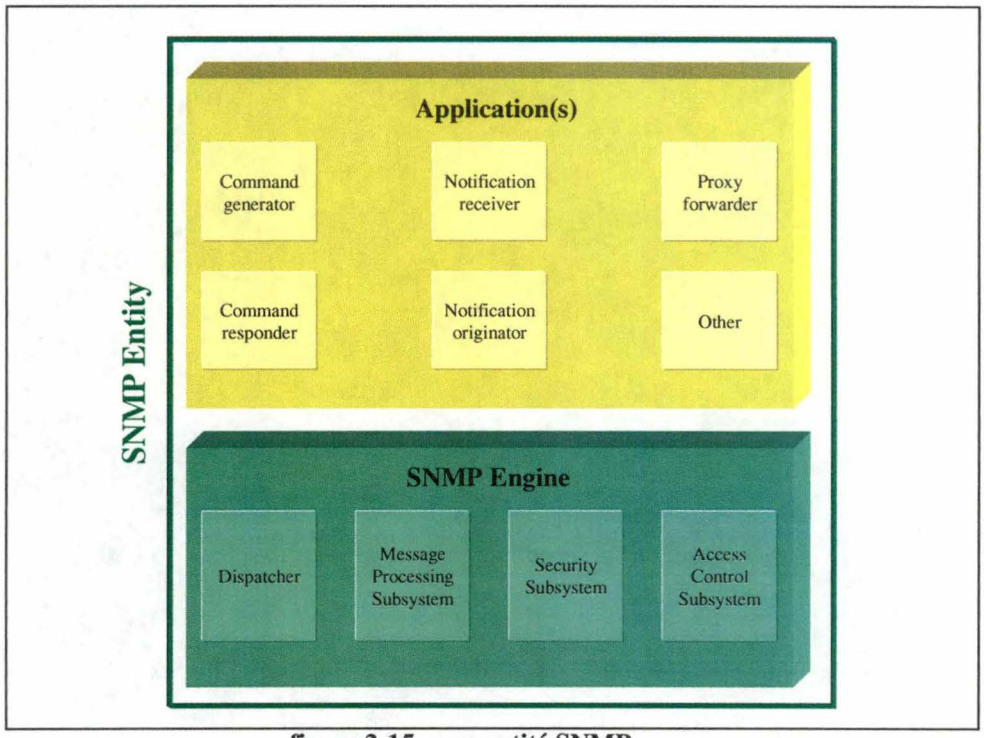

figure 2-15 : une entité SNMP

Voici une courte description de chacun des modules repris dans la figure 2-15 ci-dessus (une explication plus approfondie sera donnée dans les paragraphes 2.1.3.3 et 2.1.3.4).

- *Dispatcher* : permet au moteur SNMP de supporter les messages provenant des différentes versions de SNMP. Il est responsable de :
  - Accepter les PDU venant des applications pour les transmettre sur le réseau et donner les PDU entrant aux applications ;
  - Donner les PDU sortant au *Message Processing Subsystem* pour les inclure dans un message et passer les messages entrant au *Message Processing Subsystem* pour extraire les PDU ;
  - Recevoir et envoyer les messages SNMP sur le réseau.
- *Message Processing Subsystem* : responsable de la préparation des messages à envoyer et de l'extraction des données des messages entrant ;
- *Security Subsystem* : fournit les services de sécurité comme l'authentification et l'intimité des messages. Ce sous système contient potentiellement plusieurs modèles de sécurité ;
- *Access Control* : fournit une série de services d'autorisation qu'une application peut utiliser pour vérifier les droits d'accès. Le contrôle d'accès peut être demandé pour les requêtes de recherche ou de modification et pour les opérations de notifications générales ;
- *Command Generator* : lance les PDU *Get*, *GetNext*, *GetBulk* et/ou *Set* et traite la réponse à une requête qu'il a générée ;
- *Command Responder* : reçoit les PDU *Get*, *GetNext*, *GetBulk* et/ou *Set* destiné au système local. Cette application effectue les opérations appropriées en utilisant le contrôle d'accès, et génère la réponse qui sera envoyée à l'expéditeur de la requête ;
- *Notification Originator* : surveille un système pour détecter les événements ou les conditions particulières ; génère les PDU *Trap* et/ou *Inform*. Cette application doit avoir des mécanismes pour déterminer où envoyer les messages, quelle version de SNMP et quels paramètres utiliser lors de l'envoi d'un message ;
- *Notification Receiver* : écoute les messages de notification et génère une réponse quand le message reçu contient un PDU *Inform* ;
- *Proxy Forwarder* : transmet les messages SNMP ; l'implémentation d'un *Proxy Forwarder* est optionnelle.

Le gestionnaire SNMP traditionnel

Dans la terminologie SNMPv3, un gestionnaire SNMP traditionnel comprend trois catégories d'applications (figure 2-16):

- L'application *Command Generator* surveille et manipule les informations de gestion à distance sur les agents. Il peut utiliser les PDU *get*, *GetNext*, *GetBulk* et *Set* des versions SNMPv1 et SNMPv2 ;
- L'application *Notification Generator* qui envoie les messages asynchrones : le PDU *Inform* est utilisé à cet effet ;
- L'application *Notification Receiver* traite les messages asynchrones entrants. Il s'agit des PDU *Inform* et *Trap* (des versions 1 et 2 de SNMP). Dans le cas d'un PDU *Inform,* cette application répondra avec un PDU *Response*.

Un gestionnaire SNMP traditionnel est composé d'un moteur SNMP qui exécute différentes tâches via les sous-systèmes suivants (figure 2-16):

- Le ***Dispatcher*** est un simple gestionnaire de trafic. Pour les messages sortants, le *Dispatcher* réceptionne les PDU venant des applications. Il détermine le type de traitement des messages (dépendant de la version de SNMP) et passe le PDU au module de traitement approprié. Le *Message Processing Subsystem* lui renvoie un message contenant le PDU ainsi que l'entête du message. Le *Dispatcher* passe alors ce message à la couche transport pour le transmettre. Pour les messages entrants, le *Dispatcher* réceptionne le message venant de la couche transport. Ensuite il route chaque message vers le module de traitement approprié. Le *Message Processing Subsystem* lui renvoie le PDU contenu dans le message. Le *Dispatcher* passe alors le PDU à l'application adéquate.
- Le ***Message Processing Subsystem*** réceptionne les PDU sortants provenant du *Dispatcher* et les prépare pour la transmission en les insérant dans un message avec une entête et les renvoie au *Dispatcher*. Le *Message Processing Subsystem* réceptionne les messages entrants provenant du *Dispatcher*, traite chaque entête et renvoie les PDU au *Dispatcher*. Le *Message Processing Subsystem* contient soit un module pouvant traiter toutes les versions des messages SNMP, soit plusieurs modules traitant chacun les messages d'une seule version de SNMP.

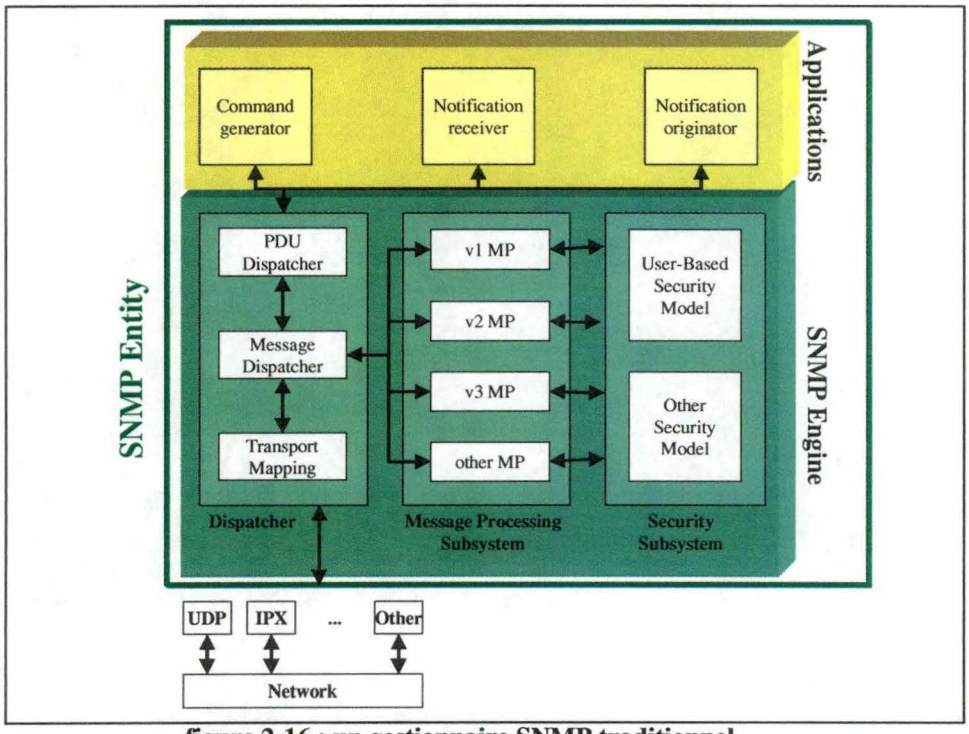

figure 2-16 : un gestionnaire SNMP traditionnel

- Les fonctions d'authentification et de chiffrements sont exécutées par le ***Security Subsystem***. Pour chaque message sortant, c'est le *Message Processing Subsystem* qui transmet le PDU au *Security Subsystem* qui selon les services demandés peut le chiffrer ou peut générer un code d'authentification et l'insérer dans l'entête du message. Une fois cela fait, tout est rendu au *Message Processing Subsystem*. Il en va de même pour les messages entrants qui sont aussi passés au *Security Subsystem* par le *Message Processing Subsystem*. Si cela est nécessaire, le Security Subsystem vérifie le code d'authentification et exécute le déchiffrement. Le résultat est renvoyé au *Message Processing Subsystem*. Un Security Subsystem peut implémenter distinctement ou non les

différents modèles de sécurité : le seul connu est le *User-based Security Model* (USM) pour SNMPv3.

L'agent SNMP traditionnel

Un agent SNMP traditionnel peut contenir trois types d'applications (figure 2-17):

- L'application *Command Responder* fournit l'accès aux informations de gestion en répondant aux requêtes entrantes (c'est à dire en retrouvant et/ou modifiant les valeurs des objets gérés) en générant un PDU *Response*.
- L'application *Notification Originator* est à l'origine des messages asynchrones en utilisant les PDU *Trap* des différentes versions de SNMP.
- L'application *Proxy Forwarder* fait suivre les messages entre différentes entités.

Le moteur SNMP (figure 2-17)pour un agent est identique au moteur SNMP pour un gestionnaire. La seule différence est qu'il contient un sous-système de contrôle d'accès (*Access Control Subsystem*) qui fournit les services d'authentification pour le contrôle d'accès (en lecture ou en écriture) aux objets gérés. Ces services sont exécutés sur base du contenu du PDU. Un *Access Control Subsystem* peut supporter distinctement un ou plusieurs modèles de contrôles d'accès. Le seul connu est le *View-based Access Control Model* (VACM) de SNMPv3.

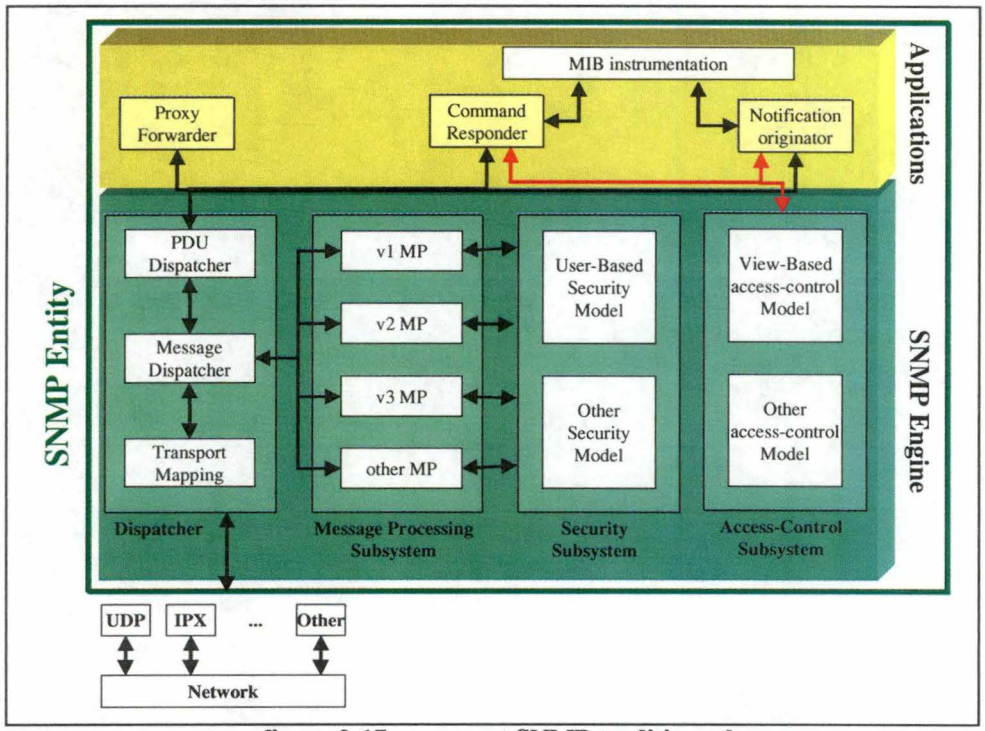

figure 2-17 : un agent SNMP traditionnel

Remarque : Le *Security Subsystem* s'occupe de l'intimité et de l'authentification et se situe au niveau des messages SNMP. L'*Access Control Subsystem* s'occupe lui de l'accès autorisé ou non à l'information de gestion et se situe au niveau des PDU SNMP.

2.1.3.3 Moteur SNMPv3

Les services entre les modules d'une entité SNMP sont définis en termes de primitives qui spécifient les fonctions à exécuter, et de paramètres utilisés pour passer les données et pour contrôler les informations. (Les figures 2-18 et 2-19 sont des exemples d'utilisation des primitives)

Les primitives du *Dispatcher*

Les primitives du *Dispatcher* définissent l'interface entre les applications SNMP et le *Dispatcher* :

- `sendPdu` : utilisé par le *Command Generator* pour envoyer un PDU de type *Request* ou *Notification* à une autre entité SNMP. Cette primitive renvoie un paramètre (`statusInformation`) qui contiendra la valeur `sendPduHandle` si le *Dispatcher* a réussi à préparer le message pour la transmission. Cette valeur unique associée au PDU permet d'identifier de façon univoque la réponse à un message.

L'application fournit tous les renseignements nécessaires pour le traitement, c'est à dire : le type de transport, l'adresse de destination, le modèle de traitement du message, le modèle de sécurité, le niveau de sécurité demandé, le contexte du PDU (le `contextEngineID` et le `contextName` pour l'information de gestion demandée) et bien entendu le PDU lui-même.

- `processResponsePdu` : utilisé par le *Dispatcher* pour passer un PDU *Response* à une application. Le paramètre `statusInformation` indique la réussite ou l'échec du PDU *Request* ou *Notification* envoyé précédemment. C'est le paramètre `sendPduHandle` qui permet d'identifier une réponse par rapport à une demande.
- `processPdu` : utilisé par le *Dispatcher* pour transmettre les PDU de type *Request* et *Notification* à une application. Le paramètre `stateReference` contient l'information relative à la sécurité, générée par le *Security Subsystem*. Lors du traitement d'un message entrant, le *Security Subsystem* peut créer une information nécessaire pour générer un message *Response* approprié. Cette information a besoin d'être renvoyée au *Security Subsystem* avec le message *Response* sortant. Le paramètre `pduVersion` indique la version du PDU.
- `returnResponsePdu` : utilisé par le *Command Responder* pour renvoyer un PDU *Response* en réponse à un PDU entrant de type *Request* ou *Notification*. Cette primitive contient plusieurs paramètres : une indication sur la taille maximale allouée ou PDU délivré ; un paramètre `stateReference` qui correspond à celle du `processPdu` ; et un paramètre `statusInformation` qui indique la réussite ou l'échec de l'opération demandée dans le PDU *Request* ou *Notification* entrant.
- `registerContextEngineID` : utilisé pour enregistrer la responsabilité d'un `contextEngineID` spécifique pour un `pduType` spécifique.
- `unregisterContextEngineID` : utilisé pour dé-enregistrer la responsabilité d'un `contextEngineID` spécifique pour un `pduType` spécifique.

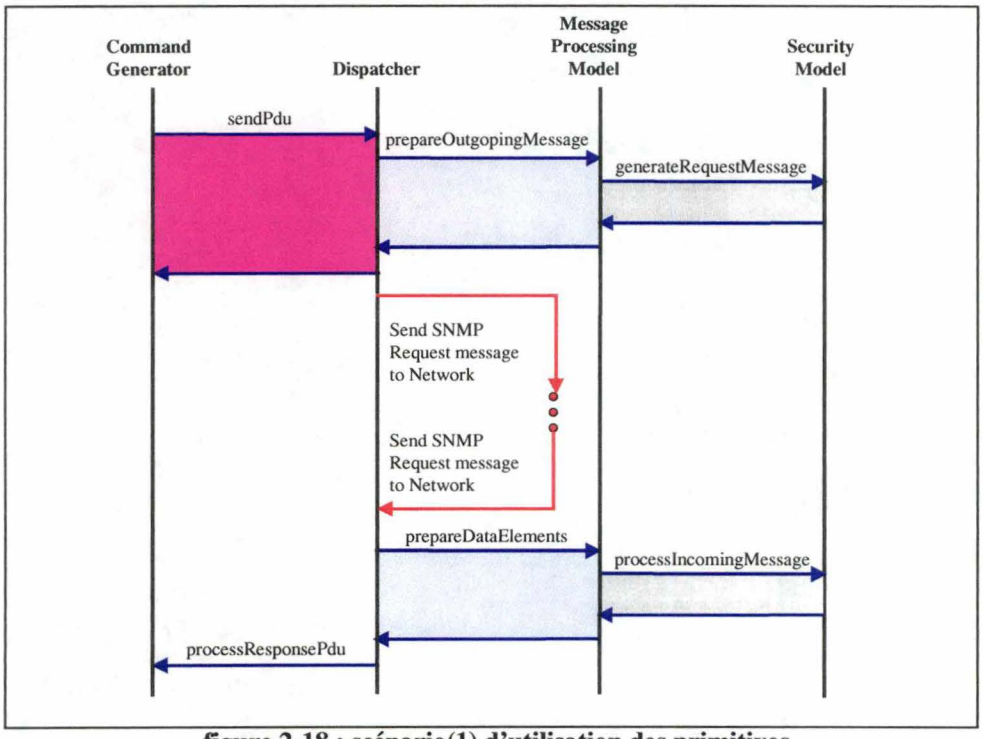

figure 2-18 : scénario(1) d'utilisation des primitives

Les primitives du *Message Processing Subsystem*

Les primitives de ce sous-système définissent l'interface entre le *Dispatcher* et le *Message Processing Subsystem*. Les trois primitives suivantes représentent les appels venant du *Dispatcher* vers le *Message Processing Subsystem* :

- `prepareOutgoingMessage` : utilisé pour demander la préparation de messages contenant un PDU SNMP de type *Request* ou *Notification*. Cette primitive renvoie une valeur (`statusInformation`) qui indique la réussite ou l'échec de la préparation du message. Le *Dispatcher* fournit le PDU comme paramètre en entrée et, si la préparation du message est réussie, il reçoit en retour un message, contenant le PDU, comme paramètre de sortie.
- `prepareDataElements` : utilisé pour extraire un PDU du message entrant. Le *Dispatcher* fournit un message en entrée et, si le traitement du message est réussi, il reçoit un PDU en sortie

avec une indication sur le type de PDU. Le paramètre stateReference contient une information qui sera utilisée lors d'une probable réponse.

- `prepareResponseMessage` : utilisé pour demander la préparation d'un message qui contiendra un PDU *Response* sortant, en réponse à un PDU de type *Request* ou *Notification*. Cette primitive renvoie une valeur (result), indiquant l'échec ou la réussite de la préparation du message. Le paramètre `stateReference` (en entrée) contient une information se trouvant dans la primitive `prepareDataElements` correspondante.

Les primitives du *Security Subsystem*

Les primitives du *Security Subsystem* définissent l'interface entre le *Message Processing Subsystem* et le *Security Subsystem*. Les trois primitives suivantes représentent les appels provenant du *Message Processing Subsystem* vers le *Security Subsystem* :

- `generateRequestMessage` : utilisé pour générer un message contenant un PDU sortant de type *Request* ou *Notification*. Cette primitive renvoie une valeur (`statusInformation`) qui indique la réussite ou l'échec de la génération du message. Le *Message Processing Subsystem* fournit une entête et un `scopedPDU` (`contextEngineID`, `contextName`, et le PDU) comme paramètre entrant, et reçoit en retour un message "préparé" (y compris, si nécessaire, l'authentification et le chiffrement) et les paramètres de sécurité associés comme paramètres de sortie.
- `processIncomingMessage` : utilisé pour fournir le traitement au niveau sécurité des messages entrants. Cette primitive renvoie une valeur (`statusInformation`) indiquant la réussite ou l'échec du traitement de la sécurité. Le *Message Processing Subsystem* fournit le message reçu comme paramètre entrant et, si le traitement s'est bien déroulé, il reçoit un PDU en sortie avec une indication sur le type de PDU. Le paramètre `securityStateReference` sortant contient de l'information utile pour une réponse probable.
- `generateResponseMessage` : utilisé pour générer un message contenant un PDU *Response* sortant, en réponse à un PDU entrant de type Request ou Notification. Cette primitive renvoie une valeur (`statusInformation`) indiquant la réussite ou l'échec de la génération du message. Le *Message Processing Subsystem* fournit une entête et un `scopedPDU` comme paramètre entrant, et reçoit en retour un message "préparé" (y compris, si nécessaire, l'authentification et le chiffrement) et les paramètres de sécurité associés comme paramètres de sortie.

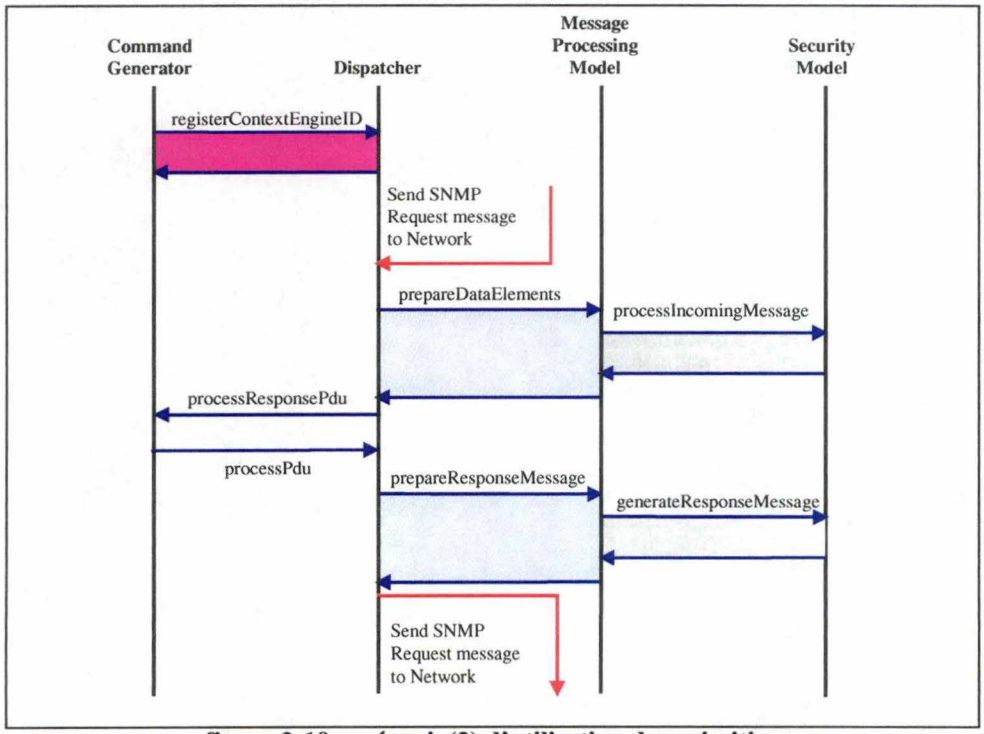

figure 2-19 : scénario(2) d'utilisation des primitives

Les primitives de l'Access *Control Subsystem*

L'application *Access Control Subsystem* fournit une simple primitive (`isAccessAllowed`) pour définir l'interface entre les applications SNMP et le sous-système de contrôle d'accès. Les paramètres de cette primitive sont tous des paramètres entrants venant de l'application de l'entité appelante pour

l'application *Access Control Subsystem* de l'entité appelée. L'application spécifie qui veut avoir un accès, le modèle de sécurité utilisé, le niveau de sécurité voulu, le type d'opération à exécuter, le `contextName` et l'identifiant de l'objet ciblé. La primitive renvoie une valeur pour signaler si l'accès est autorisé ou non.

2.1.3.4 Les applications

Command Generator application

L'application *Command Generator* utilise les primitives `sendPdu` et `processResponsePdu` du *Dispatcher*. C'est lui qui délivre chaque PDU *Response* à la bonne application *Command Generator* qui alors exécute les étapes suivantes :

1. L'application *Command Generator* examine les paramètres contenus dans la primitive `processResponsePdu`. Elle compare les différents paramètres reçus (`messageProcessingModel`, `securityModel`, `securityName`, `pduVersion`, `contextEngineID` et `contextName`) avec ceux contenus dans la requête correspondante. S'ils ne correspondent pas tous, la réponse est rejetée. Si `statusInformation` indique que la requête a échoué, alors une action dépendante de l'implémentation est prise (renvoi de la requête ou informer le gestionnaire supérieur).
2. L'application *Command Generator* examine le contenu du PDU *Response* et en extrait le `pduType`, le `request-id`, l'`error-status`, l'`error-index` et le champs `variable-bindings`. Si la valeur de `request-id` ne correspond pas à celle de la requête d'origine, alors la réponse est rejetée.
3. Si les étapes 1. et 2. Réussissent, alors l'application *Command Generator* prend une action qui dépend de l'implémentation.

Command Responder application

L'application *Command Responder* utilise quatre primitives du *Dispatcher* (`registerContextEngineID`, `unregisterContextEngineID`, `processPdu` et `returnResponsePdu`) et une primitive du sous-système de sécurité (`isAccessAllowed`). C'est via la primitive `registerContextEngineID` qu'une application *Command Responder* peut s'associer avec un moteur SNMP dans le but de traiter un certain type de PDU.

Le *Dispatcher* délivre chaque PDU de type *Request* entrant à la bonne application *Command Responder* en utilisant la primitive `processPdu`. L'application *Command Responder* exécute ensuite les étapes suivantes :

1. L'application *Command Responder* examine le contenu du PDU afin de vérifier la correspondance entre le type d'opération (*operation type*) du PDU avec un type d'opération précédemment enregistré pour cette application.
2. L'application *Command Responder* détermine si l'accès est permis pour l'opération demandée. C'est dans ce but que la primitive `isAccessAllowed` est appelée. Le paramètre `securityModel` indique quel modèle de sécurité sera utilisé par l'*Access Control Subsystem* qui contrôle si le demandeur a la permission de faire cette opération sur cet objet.
3. Si l'accès est permis, l'application *Command Responder* exécute alors l'opération de gestion et prépare un PDU *Response*. Si l'accès est refusé, l'application *Command Responder* prépare un PDU *Response* approprié pour signaler l'échec.
4. L'application *Command Responder* appelle le *Dispatcher* avec la primitive returnResponsePdu pour envoyer le PDU *Response*.

Notification Generator application

L'application *Notification Generator* suit les mêmes procédures générales utilisées pour l'application *Command Generator*. Si un PDU *Inform* doit être envoyé, les primitives `sendPdu` et `processResponsePdu` sont utilisées de la même manière que pour l'application *Command Generator*. Si un PDU *Trap* doit être envoyé, seule la primitive `sendPdu` est utilisée.

Notification Receiver application

L'application *Notification Receiver* suit une partie des procédures générales utilisées pour l'application *Command Responder*. Le *Notification Receiver* doit d'abord être enregistré comme receveur du PDU *Inform* et/ou *Trap*. Les deux PDU sont reçus au moyen de la primitive `processPdu`. Pour un PDU *Inform* la primitive `returnResponsePdu` est utilisée pour répondre.

Proxy Forwarder application

L'application *Proxy Forwarder* utilise des primitives du *Dispatcher* pour faire suivre les messages SNMP. L'application *Proxy Forwarder* s'occupe de quatre types de message de base :

- Les messages contenant un PDU venant d'une application *Command Generator* : l'application *Proxy Forwarder* détermine le moteur SNMP cible ou le moteur SNMP le plus proche ou en aval de la cible et lui envoie le PDU de type *Request* approprié.
- Les messages contenant un PDU venant d'une application *Notification Originator* : l'application *Proxy Forwarder* détermine quel moteur SNMP doit recevoir les notifications et lui envoie les PDU.
- Les messages contenant un PDU *Response* : l'application *Proxy Forwarder* compare le PDU *Response* avec un des PDU qu'il a précédemment fait transiter, et si cela correspond, il envoie le PDU *Response* à qui de droit.
- Les messages contenant un PDU *Report* : les PDU *Report* sont des PDU SNMPv3 de moteur à moteur. L'application *Proxy Forwarder* compare le PDU *Report* avec un des PDU qu'il a précédemment fait transiter, et si cela correspond, il envoie le PDU *Report* à l'expéditeur de la requête ou de la notification d'origine.

2.1.3.5 Format du message SNMPv3

Comme pour les autres versions de SNMP, l'information pour SNMPv3 est échangée entre des entités SNMP au moyen de messages (figure 2-20). Chaque message contient une entête et un PDU.

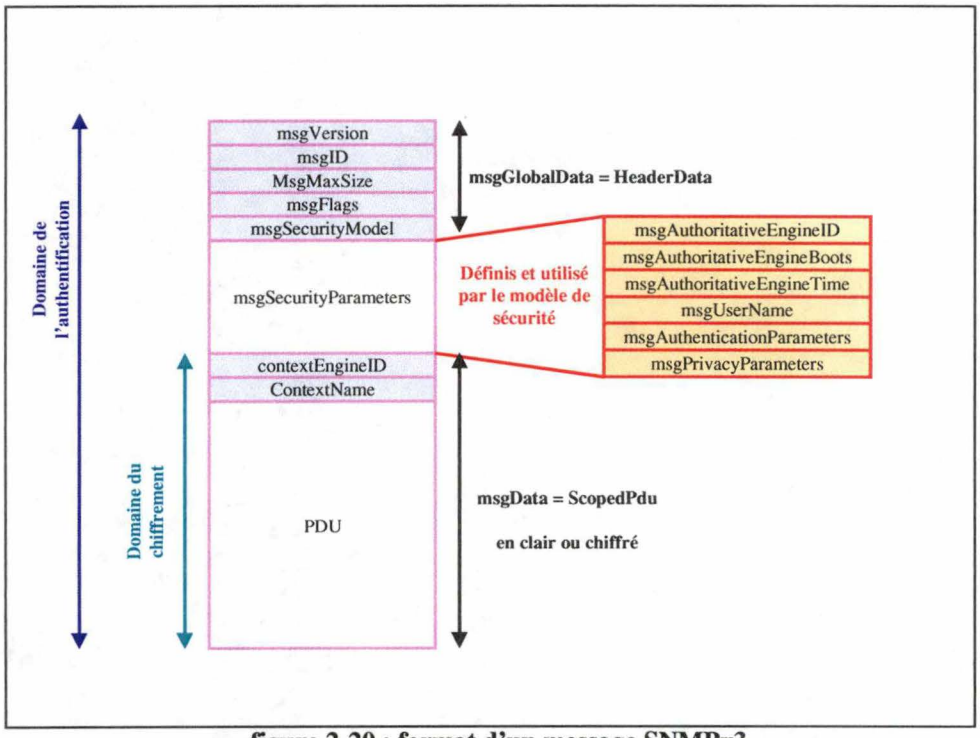

figure 2-20 : format d'un message SNMPv3

Les différents champs du message, qui sont créés et traités par le *Message Processing Subsystem* (à l'exception du PDU et du champ `msgSecurityParameters`) sont définis de cette façon :

- `msgVersion` : est égal à 3 pour SNMPv3
- `msgID` : un identifiant unique utilisé entre deux entités SNMP pour coordonner les messages *Request* et *Response* et pour coordonner le traitement des messages par les différents sous-systèmes de l'architecture de la version 3 de SNMP. Sa valeur étant comprise entre 0 et 2^{31}-1.
- `msgMaxSize` : comprend la taille maximum en octet d'un message supporté par l'émetteur du message (de 484 à 2^{31}-1). Cela représente la taille maximale du segment que l'émetteur peut accepter d'un autre moteur SNMP.
- `msgFlags` : un octet dont les trois bits les moins significatifs contiennent trois indicateurs :
  - `reportableFlag` : quand il est égal à 1 (message de type *Request* ou *Inform*), un PDU *Report* doit être renvoyé à l'émetteur si les conditions permettent la génération d'un *Report*. Quand il est égal à 0, un PDU *Report* peut ne pas être renvoyé.
  - `privFlag` : envoyé par l'émetteur pour indiquer le niveau de sécurité appliqué au message. S'il est égal à 1, alors le chiffrement est appliqué.

- `authFlag` : envoyé par l'émetteur pour indiquer le niveau de sécurité appliqué au message. S'il est égal à 1, l'authentification est appliquée.
  Toutes les combinaisons sont possibles sauf `privFlag` = 1 ET `authFlag` = 0 car le chiffrement sans l'authentification n'est pas permis.

- `msgSecurityModel` : un identifiant (entre 0 et 2^{31}-1) qui indique quel modèle de sécurité a été utilisé par l'émetteur pour préparer le message et aussi quel modèle de sécurité doit être employé par le receveur pour traiter le message. Certaines valeurs sont réservées : 1 = SNMPv1, 2 = SNMPv2c et 3 pour USM (User-based Security Model).
- `msgSecurityParameters` : un octet qui contient des paramètres générés par le sous-système sécurité de l'entité SNMP émettrice et traités par le sous-système sécurité de l'entité SNMP destination. Le contenu de ce champ n'est ni traité par le *Message Processing Subsystem* ni par le *Dispatcher*.
- `contextEngineID` : un octet qui identifie de manière unique une entité SNMP. Pour les messages entrants, ce champ détermine vers quelle application le PDU sera envoyé pour être traité. Pour les messages sortants, cette valeur est fournie par l'application générant une requête pour un message devant être envoyé.
- `contextName` : un octet qui permet d'identifier de manière unique un contexte particulier dans le domaine des contextes associés à un moteur SNMP.
- `data` : un PDU. Le modèle de traitement des messages SNMPv3 spécifie que ce doit être un PDU SNMPv2.

2.1.3.6 MIB

La RFC 2573 [RFC2573]définit trois MIB pour supporter les applications SNMPv3 :

Management Target MIB

La MIB *Management Target* contient des objets pour définir les cibles de gestion et comprend un groupe unique d'objets : le `snmpTargetObjects`. Une cible de gestion est une entité de gestion qui peut-être le destinataire des messages SNMP envoyés par un générateur de notification ou par un *Proxy Forwarder*.

Ces applications ont besoin de mécanismes pour déterminer où et comment envoyer un message. Pour ce faire, deux sortes d'information sont nécessaires :

- L'information sur la destination : constituée d'un identifiant permettant de déterminer la destination (par exemple l'adresse de la couche transport) ;
- Les paramètres des messages SNMP : composés du modèle de traitement de message, du modèle de sécurité, du niveau de sécurité, etc.

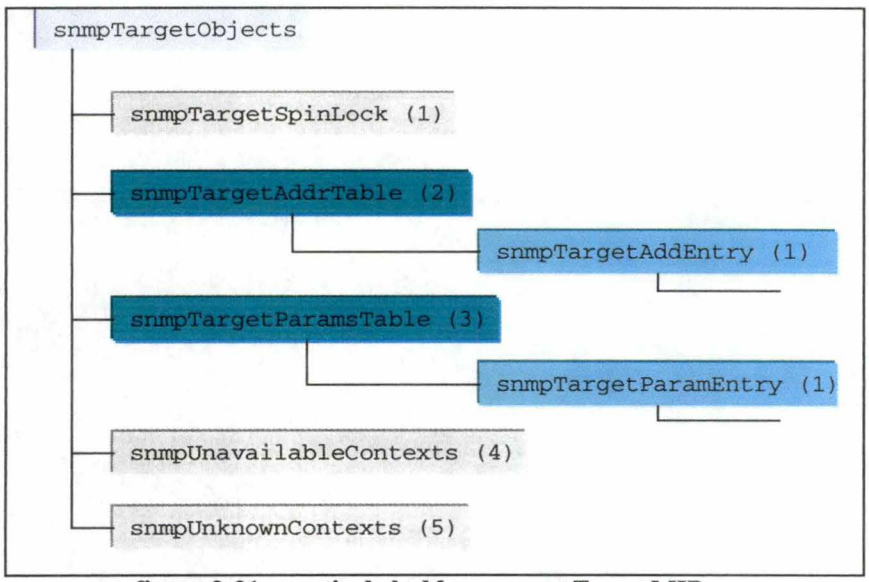

figure 2-21 : partie de la *Management Target* MIB

Le groupe `snmpTargetObjects`, contient une table pour chacune de ces deux informations (figure 2-21). Des relations N-N existent entre elles car plusieurs paramètres de messages SNMP valent pour une

seule destination et plusieurs destinations peuvent être atteintes par différents messages SNMP avec les mêmes paramètres.

Notification MIB

La MIB *Notification* contient les objets pour la configuration à distance des paramètres utilisés par une entité SNMP pour générer les notifications. Cette MIB ne comprend qu'un seul groupe, `snmpNotifyObjects`, qui comprend les 3 tables suivantes (figure 2-22):

- `snmpNotifyTable` : chaque entrée de cette table fait appel à une ou plusieurs entrées dans `snmpTargetAddrTable` à utiliser lors de la génération des notifications ;
- `snmpNotifyFilterProfileTable` : permet d'accroître `snmpTargetAddrTable` en y associant une série de filtres pour une cible de gestion particulière ;
- `snmpNotifyFilterTable` : définit les filtres utilisés pour limiter le nombre de notifications générées pour une cible de gestion particulière.

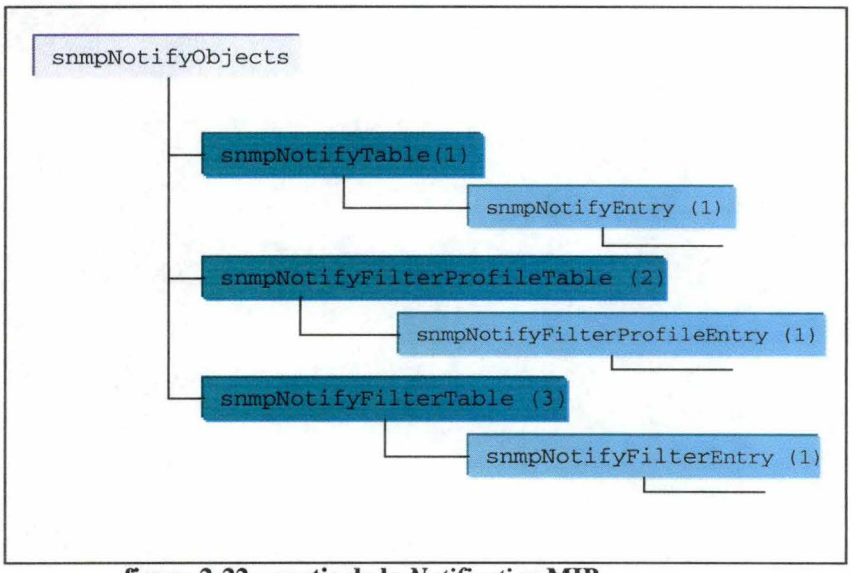

figure 2-22 : partie de la *Notification* MIB

Proxy MIB

La MIB *Proxy* contient les objets utilisés pour la configuration à distance des paramètres qui sont utilisés par une entité SNMP pour les opérations de type *Proxy Forwarding*. Cette MIB comprend un groupe unique, `snmpProxyObjects`, qui consiste en une seule table `snmpProxyTable`. Chaque entrée de la table contient un jeu de paramètres de translation qui seront employés par le *Proxy Forwarder*.

Une table `snmpProxyTable` correctement configurée, devrait permettre à un Proxy Forwarder d'accepter les messages entrants, de déterminer pour chaque message le ou les cibles, et de construire sur cette base les paramètres de sécurité pour les messages sortants.

2.1.3.7 Security Model

Actuellement, le seul modèle de sécurité est celui définit dans la RFC 2574 [RFC2574] : il s'agit de *User-based Security Model* (USM) pour SNMPv3. Ce modèle répond aux spécifications suivantes :

- Authentification : assure l'authentification de l'intégrité et l'origine des données. Le code d'authentification HMAC (avec ses fonctions de *hash* MD5 ou SHA-1) assure l'authentification ;
- *TimeLess* : protection contre le retard d'un message ou contre le fait qu'un message soit rejoué plus tard ;
- Intrusion : protège contre la divulgation des données contenues dans "la charge utile" du message. Le mode de chiffrement DES est utilisé pour le chiffrement ;
- Format des messages : définit le format du champ `msgSecurityParameters`, qui supporte les trois fonctions précédentes ;
- Découverte : définit les procédures avec lesquelles un moteur SNMP obtient les infos sur un autre moteur SNMP ;
- Gestion de clés : définit les procédures pour la génération, la mise à jour et l'utilisation de clés.

Dans toute transmission de message, une des deux entités (émetteur ou receveur) est désignée comme un moteur SNMP autoritaire selon les deux règles suivantes :

- Quand un message contient une charge nécessitant une réponse, alors le receveur d'un tel message est considéré comme autoritaire. Il s'agit des messages contenant un PDU *Get*, *GetNext*, *GetBulk*, *Set* ou *Inform*.
- Quand un message contient une charge ne nécessitant pas de réponse, alors l'émetteur de ce message est considéré comme autoritaire. On parle dans ce cas des messages contenant un PDU *SNMPv2-Trap*, *Response* ou *Report*.

Donc, pour les messages envoyés via un *Command Generator* et pour les messages *Inform* provenant d'un *Notification Originator,* le receveur est autoritaire. Pour les messages envoyés via un *Command Responder* ou pour un message *Trap* provenant d'un *Notification Originator,* l'émetteur est autoritaire. Cette désignation poursuit deux buts :

1) La ponctualité d'un message est déterminée grâce à une horloge maintenue par le moteur autoritaire. Quand un moteur autoritaire envoie un message (*Trap*, *Report* ou *Response*), celui-ci contient la valeur actuelle de son horloge pour que le receveur non autoritaire puisse se synchroniser dessus. Quand un moteur non autoritaire envoie un message (*Get*, *GetNext*, *GetBulk*, *Set* ou *Inform*), celui-ci contient sa valeur estimée de l'horloge de la destination, permettant ainsi à la destination d'estimer la ponctualité du message.
2) Une procédure de localisation de clé permet à un maître d'être propriétaire de clés stockées sur différents moteurs. Ces clés sont localisées au niveau du moteur autoritaire car, de cette façon, le maître n'est responsable que d'une seule clé et on écarte le risque de stocker de multiples copies de la même clé si on a un réseau distribué.

La RFC 2574 [RFC2574]définit une structure, `UsmSecurityParameters`, qui spécifie le format interne du champ `msgSecurityParameters` du message SNMPv3. Quand un message sortant est donné à l'USM par le *Message Processing Subsystem*, l'USM remplit le champ `msgSecurityParameters` (figure 2-20). Et quand un message entrant est donné à l'USM via le *Message Processing Subsystem*, l'USM traite les valeurs contenues dans le champ `msgSecurityParameters` qui est constitué des éléments suivants :

- `msgAuthoritativeEngineID` : le `snmpEngineID` du moteur SNMP autoritaire impliqué dans l'échange de ce message. Donc cette valeur fait référence à la source pour un message *Trap*, *Response* ou *Report* et à la destination pour un message *Get*, *GetNext*, *GetBulk*, *Set* ou *Inform*.
- `msgAuthoritativeEngineBoots` : le `snmpEngineBoots` du moteur SNMP autoritaire impliqué dans l'échange de ce message. Cette valeur est un entier entre 0 et 2^{31}-1 et représente le nombre de fois qu'un moteur SNMP a été réinitialisé ou s'est réinitialisé depuis sa configuration initiale.
- `msgAuthoritativeEngineTime` : le `snmpEngineTime` du moteur SNMP autoritaire impliqué dans l'échange de ce message. Cette valeur est un entier entre 0 et 2^{31}-1 et représente le nombre de secondes depuis la dernière incrémentation de `snmpEngineBoots`. Chaque moteur SNMP autoritaire est responsable de l'incrémentation de son propre compteur. Un moteur non autoritaire est responsable de l'incrémentation de sa propre notion de `snmpEngineTime` pour chaque moteur autoritaire distant avec qui il communique.
- `msgUserName` : l'utilisateur au nom duquel le message est échangé.
- `msgAuthenticationParameters` : nul si l'authentification n'est pas utilisée, sinon l'information sert de paramètres d'authentification. Pour USM, la valeur équivaut à HMAC.
- `msgPrivacyParameters` : nul si on n'utilise pas le principe de confidentialité. Pour la version actuelle de USM, ce paramètre est utilisé comme valeur initiale de l'algorithme DES CBC.

USM a besoin d'utiliser un processus de découverte pour obtenir suffisamment d'informations à propos des autres moteurs SNMP dans le but de communiquer avec eux. En particulier, un moteur SNMP non autoritaire doit connaître le `snmpEngineID` du moteur autoritaire avant de communiquer. Cela se fait au moyen d'un message *Request* particulier dont la réponse est le `snmpEngineID` du moteur autoritaire. De plus si l'authentification est nécessaire, il faut établir une synchronisation au niveau du temps et ce via un message *Request* authentifié particulier dont la réponse (message *Report*) sera composée de `snmpEngineBoots` et de `snmpEngineTime` du moteur autoritaire.

Pour pouvoir utiliser les services de confidentialité et d'authentification de SNMPv3, il est nécessaire de partager un clé d'authentification et une clé de confidentialité avant toute communication entre un moteur non autoritaire et un moteur autoritaire. Ces clés permettent à un utilisateur d'un moteur non autoritaire (un système de gestion) d'employer l'authentification et la confidentialité avec un système autoritaire distant géré par l'utilisateur.

2.1.3.8 Access Control Model

Le modèle de contrôle d'accès *View-based Access Control Model* (VACM) détermine comment l'accès à un objet géré d'une MIB locale par un acteur distant est accepté. VACM utilise une MIB qui définit la politique de contrôle d'accès pour cet agent et la rend accessible pour la configuration à distance.

La RFC 2575 [RFC2575] définit VACM à l'aide des cinq éléments suivants :

1) **Groupe** :
   Un groupe est défini comme une série de 0 à n-tuples <`securityModel`, `securityName`> sur base desquels les objets de gestion SNMP peuvent être accédés. Un nom de groupe unique est associé à chaque groupe. Le concept de groupe est fort utile pour associer des gestionnaires et des droits d'accès.

2) **Niveau de sécurité** :
   Les droits d'accès pour un groupe peuvent être différents en fonction du niveau de sécurité spécifié dans le message qui contient la requête. Par exemple, un agent peut permettre l'accès en lecture seule pour un message non authentifié et en écriture pour les messages authentifiés. De même, pour certains objets sensibles, l'agent peut demander que la requête et la réponse se fassent sous le principe de confidentialité.
3) **Contexte** :
   Un contexte MIB est une sous partie des instances d'objets d'une MIB locale. Le contexte est un moyen utile de regrouper les objets dans des collections avec des politiques d'accès différentes.
   Quand une station de gestion interagit avec un agent afin d'accéder à l'information de gestion de l'agent, alors l'interaction a lieu entre un gestionnaire principal et le moteur SNMP de l'agent. Les privilèges de contrôle d'accès sont formalisés dans une vue MIB et s'appliquent à ce gestionnaire principal et à ce contexte.
   Un contexte a les caractéristiques de base suivantes :
   - Une entité SNMP, définie de façon univoque par un `contextEngineID`, peut maintenir plus d'un contexte ;
   - Un objet ou une instance d'objet peut apparaître dans plus d'un contexte ;
   - Quand de multiples contextes existent, pour identifier une instance d'objet individuelle, ses `contextName` et `contextEngineID` doivent être identifiés en plus de son type d'objet et de son instance.
4) **Vues MIB** :
   Bien souvent, on voudrait restreindre l'accès à une partie des objets gérés d'un agent par un groupe particulier. Afin d'aboutir à cet objectif, l'accès à un contexte se fait au moyen de vues MIB qui définissent un ensemble spécifique d'objets gérés. VACM utilise une technique puissante et flexible pour définir des vues MIB : elles sont basées sur le concept de vue de sous-arbres et de vues "familiales". La vue MIB est définie en terme de collection (ou de famille) de sous-arbres. Chaque sous-arbre pouvant être inclus ou exclus d'une vue MIB.
   SNMPv3 ajoute le concept de sous-arbre qui est composé d'un nœud de la hiérarchie MIB (cfr. SMI) et de tous ses subordonnés. Plus formellement, un sous-arbre peut-être défini comme un ensemble d'objets et d'instances d'objet qui ont un préfixe ASN.1 OBJECT IDENTFIER commun à leur nom. Donc, le plus long préfixe commun à toutes les instances du sous-arbre est l'identifiant d'objet du nœud parent du sous-arbre.
   Trois vues MIB sont associées à chaque entrée de la table `vacmAccessTable`, une par type d'accès : lecture, écriture et notification. Chaque vue MIB est constituée d'un ensemble de vues de sous-arbres qui sont soit inclus soit exclus de la vue. Il est possible d'utiliser un masque afin de définir plus finement les contrôles d'accès requis.
5) **Politique d'accès** :
   VACM permet de configurer un moteur SNMP dans le but de renforcer un ensemble particulier de droits d'accès. Les accès sont déterminés à l'aide des facteurs suivants :
   - Sur base du maître faisant la requête. VACM rend possible à un agent de permettre des privilèges d'accès différents en fonction de l'utilisateur. Par exemple, un gestionnaire responsable de la configuration du réseau peut avoir une large autorité pour modifier la MIB locale, alors qu'un gestionnaire chargé de la surveillance du réseau ne doit avoir accès qu'en lecture et sur une partie de la MIB en écriture. Les utilisateurs sont assignés à des groupes et la politique d'accès est spécifiée en fonction des groupes.
   - Sur base du niveau de sécurité de la requête contenue dans le message SNMP communiqué. Souvent, un agent demandera l'utilisation de l'authentification pour les messages contenants un PDU *Set* (considérée comme une opération en écriture).
   - Sur base du modèle de sécurité utilisé pour le traitement de la requête. Si plusieurs modèles de sécurité sont implémentés sur l'agent, l'agent doit être configuré pour fournir

différents niveaux d'accès aux requêtes contenues dans des messages traités par différents modèles de sécurité. Par exemple, certains éléments seraient accessibles si le message contenant la requête est passé par USM (lors de sa création), et inaccessibles si le modèle de sécurité est celui de SNMPv1.

- Sur base du contexte MIB de la demande.
- Sur base de l'instance d'objet pour laquelle un accès est demandé. Certains objets contiennent des informations plus sensibles ou plus critiques que d'autres et donc la politique d'accès peut dépendre de l'instance d'objet spécifique demandée.
- Sur base du type d'accès demandé (lecture, écriture ou notification). Ces trois opérations sont des opérations de gestion différentes pour lesquelles une politique d'accès par type d'opérations est nécessaire.

Les différents concepts qui constituent VACM semblent être une définition complexe du contrôle d'accès. Le but de ces concepts est de clarifier les relations impliquées dans l'accès à l'information de gestion et de minimiser le stockage et les traitements requis par l'agent. Pour comprendre ces motivations on ne doit pas oublier que pour SNMPv1 et SNMPv2c, c'est le nom de communauté qui est utilisé pour représenter les informations en rapport avec la sécurité. Ces informations permettaient de connaître :

- L'identité de l'entité demanderesse (station de gestion) ;
- L'identité de l'entité de traitement (un agent ou un proxy) ;
- Localisation de l'information de gestion à accéder (agent ou proxy) ;
- L'information d'authentification ;
- L'information de contrôle d'accès (l'autorisation de traiter ou non une opération) ;
- L'information sur les vues MIB.

En rassemblant tous ces concepts dans une variable unique, on perd en fonctionnalité et en flexibilité. VACM fournit un jeu similaire d'informations relatives à la sécurité dont la représentation est la suivante :

Elément d'information	**Source de l'information**
Entité demanderesse	`MsgUserName` et `msgSecurityModel` des messages entrants ; consigné dans `groupName` par VACM
Entité de traitement	`contextEngineID` dans les messages entrants
Localisation de l'info de gestion	`contextName` dans les messages entrants
Information d'authentification	`msgAuthenticationParameters` dans les messages entrants. Pas directement utilisé par VACM ; les messages requérant une authentification sont seulement acceptés par un agent SNMP si le message est authentifié par USM
Information de contrôle d'accès	Entrée dans la table vacmAccessTable, qui dépend du `groupName`, du `contextName`, du `securityModel` et de `security-Level` ; le dernier élément est obtenu à partir de msgFlags des messages (requête) entrants
Information sur le vues MIB	Vues contenues dans la table `vacmViewTreeFamilyTable` qui est déterminée par le type d'opération (lecture, écriture, notification) indiqué par le PDU contenu dans les messages entrants

Ceci constitue une amélioration substantielle de SNMPv1 car cela permet de séparer différents concepts afin d'assigner les valeurs séparément les unes des autres.

2.1.4 Evolution de SNMP

Un groupe de travail est mis sur pied depuis janvier 2001 afin de revoir le protocole SNMP dans le but de le perfectionner. Les améliorations, qui portent essentiellement sur l'utilité et l'efficacité du protocole, doivent correspondre à l'architecture définie dans la RFC 2571 [RFC2571]. L'accent de ce groupe de travail est mis sur les améliorations facilement définissables et implémentables qui seront rapidement acceptées et déployées. De nouvelles opérations du protocole sont dans le domaine du possible, si elles font partie des améliorations acceptables.

Les différents points suivants font partie du travail initial de ce groupe de travail :

- Un document définissant un mécanisme par lequel les capacités d'une entité SNMP peuvent être déterminées. Ce document doit aussi définir les spécifications d'interopérabilité du protocole SNMP quand des extensions sont présentes ou non ;

- Un document définissant un mécanisme pour retrouver, créer et effacer efficacement des lignes dans une table ;
- Un document définissant un mécanisme à utiliser pour effacer les instances des objets gérés de toute une partie d'un sous-arbre. Par exemple, pour effacer toutes les informations concernant un utilisateur particulier de SNMP ;
- Un document définissant un mécanisme de compression des identifiants d'objets pour en retirer le plus d'informations redondantes possibles dans les données utiles de SNMP ;
- Un document définissant un mécanisme pour le transfert en masse des données SNMP.

Un calendrier précis des révisions est proposé et devrait permettre de conclure dans le courant du mois d'octobre 2001.

2.1.5 Résumé

Le protocole SNMP est comme son nom l'indique un protocole simple de gestion du réseau : un gestionnaire communicant avec des agents selon un schéma de type question réponse pour obtenir ou modifier des informations. La première version de ce protocole permet donc à un gestionnaire d'accéder en lecture ou en écriture à des informations sur les objets gérés qui sont stockées sur les agents. Ces informations sont décrites dans une MIB selon une structure bien définie : la SMI. Les opérations utilisées par la version 1 sont : *Get*, *GetNext*, *Set*. En plus de cela, l'agent peut envoyer des alertes (*Trap*) au gestionnaire en fonction d'événement bien précis.

Cette première version de SNMP a aussi quelques lacunes comme la difficulté de gérer de grands réseaux, un système d'authentification peu fiable, les alertes qui ne sont pas confirmées, etc. La confidentialité de SNMPv1 est assurée par un nom de communauté : le gestionnaire et les agents doivent appartenir à la même communauté pour échanger des informations de gestion.

La version 2 de SNMP quant à elle fut mise en place avec pour but de répondre aux lacunes de SNMPv1. Certaines de ces lacunes ont en effet été supprimées par l'ajout de nouvelles opérations (*GetBulk* et *Inform*) qui ont permis par exemple de gérer de plus grands réseaux. Dans cette deuxième version, il n'a par contre pas été possible d'obtenir un accord sur le problème de la sécurité, de la confidentialité et de l'authentification. La version 2 de SNMP étant actuellement basée sur les noms de communauté.

Les RFC 2571 à 2575 [RFC2571-RFC2575]définissent une nouvelle architecture pour SNMP qui doit satisfaire aux implémentations existantes de SNMPv1 et SNMPv2c, fournir un cadre de travail pour SNMPv3, et fournir un cadre de travail pour les révisions futures et les améliorations des éléments composant SNMP. Ces documents définissent aussi SNMPv3.

L'architecture SNMP définit cinq applications génériques pour générer et recevoir les PDU SNMP. Il s'agit de *Command Generator*, *Command Responder*, *Notification Originator*, *Notification Receiver* et *Proxy Forwarder*. L'architecture définit aussi un moteur SNMP qui est responsable de la préparation des messages à partir des PDU pour les transmettre, de l'extraction des PDU des messages entrants pour les donner aux applications, et du traitement, au niveau de la sécurité, des messages entrants et sortants. Le moteur SNMP est constitué de quatre composants : *Dispatcher*, *Message Processing Subsystem*, *Security Subsystem* et *Access Control Subsystem*.

SNMPv3 définit deux aspects de la sécurité. Le *User-based Security Model* fournit les services d'authentification et garanti l'intimité des informations (chiffrement) et si situe au niveau du message. Le *View-based Access Control Model* détermine quel est le type d'accès accordé à l'initiateur d'une requête pour atteindre un objet particulier et y exécuter une action particulière (cela si situe au niveau du PDU).

Parmi les trois versions de SNMP, il semble que les différents éléments du réseau qui doivent être gérés dans la partie pratique ne supportent que la première version (SNMPv1). Cela limitera fortement la mise en œuvre de SNMP comme on pourra le voir dans la partie de l'étude pratique qui y est consacrée.

2.2 Remote Network Monitoring

2.2.1 Généralités

La modification la plus importante apportée à la base SNMP (SMI, MIB, SNMP) est l'ajout des spécifications concernant la surveillance d'un réseau distant : *Remote Network Monitoring* (RMON). Ces spécifications sont reprises dans les quatre RFC suivantes : 1513[RFC1513], 1757[RFC1757], 2021 [RFC2021] et 2074 [RFC2074]. Elles définissent une MIB de surveillance à distance, en supplément à la MIB-II, qui permet au gestionnaire du réseau d'avoir des informations vitales sur les réseaux distants.

RMON est en fait une simple spécification MIB, qui ne modifie rien au protocole SNMP existant mais qui fournit une extension significative à SNMP.

2.2.2 MIB II

2.2.2.1 Principes de base

La RFC 2011 [RFC2011] (RFC 1213 *obsolete*) définit la seconde version de la base d'information de gestion (MIB II). La première version (MIB I) était définie dans la RFC 1156 [RFC1156]. La MIB II est une extension de la MIB I et comprend donc quelques groupes et objets additionnels qui ont été ajoutés par les concepteurs de la MIB II en suivant des critères particuliers (Cfr. RFC 1213) :

1. Un objet doit être essentiel, soit pour la gestion des problèmes, soit pour la gestion de la configuration ;
2. Seuls les objets de contrôle qui, lorsqu'on les modifie, ne peuvent occasionner que des dommages limités. Ceci est le reflet du faible niveau de sécurité des protocoles de gestion des réseaux actuels ;
3. Les objets doivent être utiles et utilisables actuellement ;
4. Quand MIB I a été développé, le nombre d'objets a été intentionnellement limité à 100 afin de permettre au vendeur de développer pleinement leur outil. Avec MIB II, cette limite est supprimée pour donner une base technologique étendue à la MIB ;
5. Pour éviter des objets redondants, aucun objet ne peut être dérivé d'autres objets de la MIB ;
6. Les objets spécifiques sont exclus (par exemple des objets particuliers pour BSD Unix) ;
7. Les développeurs donnent leur accord pour implémenter les outils en partant du principe "un compteur par section critique".

2.2.2.2 Les groupes de la MIB II

Depuis que la MIB II contient seulement des objets désignés comme essentiels, aucun des objets n'est optionnel. Le groupe `mib-2` est subdivisé en différents groupes.

`system` *Group*

Le groupe `system` fournit des informations générales à propos du système géré. La plupart des objets de ce groupe parlent d'eux-mêmes tandis que d'autres nécessitent une explication :

- L'objet `sysServices` est interprété comme un code de 7 bits. Chaque bit correspondant à une des couches de l'architecture TCP/IP ou OSI avec le bit le moins significatif pour la couche 1. Cela permet de savoir à quelle couche particulière le système fournit un service.
- L'objet `sysUpTime` indique la quantité de temps écoulée depuis la dernière réinitialisation du système. Ce compteur permet d'avoir une idée, dans le temps, sur les modifications des autres objets de la MIB.

`interface` *Group*

Le groupe `interface` contient des informations génériques sur les interfaces physiques de l'entité, y compris les informations relatives à la configuration et les statistiques des événements se produisant pour chaque interface. L'implémentation de ce groupe est obligatoire pour tous les systèmes.

Le groupe `interface` contient des informations de base qui sont utiles comme point de départ pour toutes les fonctions de gestion de réseaux comme la surveillance des performances et les contrôles des erreurs. On peut ainsi détecter les problèmes de congestion (en mesurant le nombre total d'octets entrant et sortant d'un système). Et si la congestion est détectée, d'autres objets du groupe peuvent servir pour déterminer l'activité d'un protocole en particulier afin de trouver le responsable de la congestion.

`at` *Group* (address translation)

Le groupe `address translation` est constitué d'une seule table. Chaque ligne de la table correspondant à une des interfaces physiques du système. La ligne fournissant une translation entre une adresse réseau et une adresse physique. L'adresse physique dépend de la nature du sous réseau : si l'interface correspond à une interface LAN alors l'adresse physique sera une adresse MAC et l'adresse réseau sera une adresse IP.

Ce groupe n'est plus utilisé dans la MIB II et il n'y est inclus que pour des raisons de compatibilité avec les nœuds MIB I. Cette dévalorisation est due à deux besoins bien précis : la nécessité de supporter des nœuds multi-protocoles et la nécessité d'une translation bidirectionnelle.

Dans la MIB II, les tables d'adresses sont situées dans les groupes de protocoles (par couche réseau) appropriés. Chaque groupe étant constitué d'une ou de deux tables permettant une translation bidirectionnelle. L'utilisation de deux tables permettant quant à elle une implémentation plus aisée (mais il est possible que les deux tables ne soient en fait que deux vues différentes de la même structure de données).

`ip` *Group*

Le groupe `ip` comprend les informations pertinentes sur l'implémentation et les opérations IP d'un nœud. Depuis que IP est implémenté aussi bien dans les systèmes finaux que dans les systèmes intermédiaires, tous les objets de ce groupe ne sont pas pertinents pour tous les systèmes : les objets "inutiles" ont alors une valeur nulle.

Le groupe `ip` contient certains compteurs de base à propos du trafic entrant et sortant de la couche IP. Trois tables y sont inclues :

- La table `ipAddrTable` contient des informations relatives à chaque adresse IP assignée à cette entité. Une ligne par adresse IP avec, entre autres, le numéro d'interface physique : cela est utile lors du contrôle de la configuration du réseau du point de vue IP.
- La table `ipRouteTable` contient des informations utiles pour le routage Internet. Cette information est relativement générale et peut être extraite des protocoles spécifiques aux tables de routage (RIP, OSPF, ...). Une entrée dans la table par route connue par l'entité. Ce genre d'information est aussi utile pour le contrôle de la configuration et si les droits le permettent, il est alors possible de modifier le routage.
- La table `ipNetToMediaTable` est une table de translation d'adresses entre les adresses physiques et les adresses IP. Il y a une entrée dans la table par interface qui n'utilise pas de techniques algorithmiques de translation.

En plus de ces trois tables, il existe toute une série d'objets scalaires qui sont utiles pour la surveillance des performances et des erreurs.

La RFC 1354 [RFC1354]a pour but de rendre les tables de routage plus flexibles en "remplaçant" la table `ipRouteTable` par la table `ipForwardTable` car le problème spécifique de `ipRouteTable` est quelle n'est indexée que sur l'adresse IP de destination. La table `ipForwardTable` contient les informations nécessaires pour transmettre les données reçues vers une autre destination au cas où la destination primaire pose un problème.

`icmp` *Group*

Le protocole ICMP, définit dans la RFC 792 [RFC0792], est une partie de la suite du protocole TCP/IP. C'est un compagnon de IP, car tous les systèmes implémentant IP doivent aussi implémenter ICMP. ICMP fournit un moyen de transférer des messages d'un routeur ou d'un hôte vers un hôte. Il est utile pour obtenir un feed-back à propos des problèmes de l'environnement de communication (quand un datagramme ne peut pas arriver à destination, quand un routeur n'a pas de mémoire tampon suffisante pour transférer un datagramme, etc.)
Le groupe `icmp` contient des informations significatives pour l'implémentation de ICMP et pour les opérations ICMP sur un nœud. Le groupe est constitué uniquement de compteurs des différents types de messages ICMP envoyés et reçus.

`tcp` *Group*

Le groupe `tcp` contient des informations significatives pour l'implémentation de TCP et pour les opérations TCP sur un nœud. La seule table de ce groupe est la table `tcpConnTable` qui traite du temps de retransmission. En effet, quand une entité TCP transmet un segment, elle s'attend à recevoir un

acquis. Si le segment est perdu ou endommagé, ou si l'acquis correspondant est perdu ou endommagé, l'entité TCP émettrice va, après un certain temps, retransmettre le segment. Le temps de retransmission est calculé selon un algorithme précis dont les paramètres sont stockés dans la table.

`udp` *Group*

Le groupe `udp` contient des informations significatives pour l'implémentation de UDP et pour les opérations UDP sur un nœud. En plus des informations concernant les datagrammes envoyés et reçus, le groupe `udp` comprend la table `udpTable`. Cette table contient l'adresse IP et le port UDP de toutes les entités UDP finales qui sont en communication avec l'entité locale.

`egp` *Group*

Le groupe `egp` contient des informations pertinentes concernant l'implémentation et les opérations de l'*External Gateway Protocol* (EGP). En plus des informations relatives aux messages EGP envoyés et reçus, le groupe `egp` comprend la table `egpNeighTable`. Cette table contient des informations sur chaque passerelle voisine connue par l'entité.

`dot3` *Group* (transmission)

Ce groupe a été voulu pour contenir des objets fournissant des détails sous-jacents au moyen de transmission pour chaque interface d'un système. En fait, ce n'est pas du tout un groupe, mais simplement un nœud en dessous duquel différents groupes spécifiques à une interface sont localisés.

Tandis que le groupe interface contient des informations générales qui s'appliquent à toutes les interfaces, ces MIB spécifiques à une interface contiennent des informations relatives à un type de sous-réseau spécifique (Par exemple : l'Ethernet MIB).

`snmp` *Group*

En ce qui concerne le groupe snmp, il a été étudié beaucoup plus en détails précédemment dans ce document.

2.2.3 RMON

Avec la MIB-II, le gestionnaire du réseau pouvait obtenir de l'information locale à propos d'appareils individuels. Si on considère un LAN avec un certain nombre d'appareils comprenant un agent SNMP, le gestionnaire peut facilement connaître le montant du trafic entrant et sortant sur chaque appareil mais pas le montant des données échangées sur le réseau en entier.

2.2.3.1 Concepts de base

De façon générale, les capacités de RMON fournissent un moyen efficace pour surveiller le comportement d'un large sous-réseau tout en diminuant les communications entre les agents et les stations de gestion. La RFC 1757 [RFC1757] met en évidence les différents buts de conception suivants :

- Opérations en différé : il peut être nécessaire de limiter ou d'arrêter les interrogations continuelles (*polling*) d'un moniteur par un gestionnaire réseau. Cela permet d'épargner sur les coûts de communication (surtout s'il s'agit de lignes à bas débit). En général, le moniteur doit collecter les informations concernant la configuration, les performances et les pannes continuellement même si la station de gestion ne fait pas d'interrogation. Le moniteur continue simplement d'accumuler les statistiques qui pourront être accédées plus tard par le gestionnaire. Le moniteur doit aussi essayer d'avertir la station de gestion quand un événement exceptionnel surgit ;
- Surveillance proactive : si le moniteur a des ressources suffisantes et si la mise en pratique n'est pas perturbante, le surveillant (*monitor*) peut continuellement exécuter des diagnostiques et enregistrer les performances du réseau. Si une panne survient quelque part, le surveillant doit être capable d'en avertir la station de gestion et de lui fournir les informations utiles pour établir un diagnostique ;
- Détection de problème et rapport : la surveillance préemptive nécessite une sonde réseau active (*probe*) et donc la consommation de ressources réseau pour contrôler les erreurs et les conditions d'exception. Une alternative est que le surveillant puisse reconnaître certaines conditions d'erreurs ou autres passivement sur base du trafic qu'il observe. Le moniteur peut être configuré pour contrôler continuellement certaines conditions. Si une d'elles survient, le surveillant peut enregistrer la condition dans un journal et essayer d'en avertir la station de gestion ;
- Données à valeur ajoutée : le moniteur peut effectuer des analyses spécifiques sur les données récoltées sur son sous-réseau, en enlevant cette responsabilité à la station de gestion (par exemple, déterminer quel est l'hôte qui a généré le plus de trafic). Ce type d'information au niveau du sous-

réseau ne serait accessible à la station de gestion que si elle était connectée directement au sous-réseau ;

- Gestionnaire multiple : une configuration réseau composée de plusieurs sous-réseaux peut avoir plus d'une station de gestion. Cette situation permet d'avoir plus de fiabilité et permet aussi de répartir les capacités de gestion entre différentes unités dans une même organisation. Le moniteur peut être configuré pour traiter avec plus d'une station de gestion.

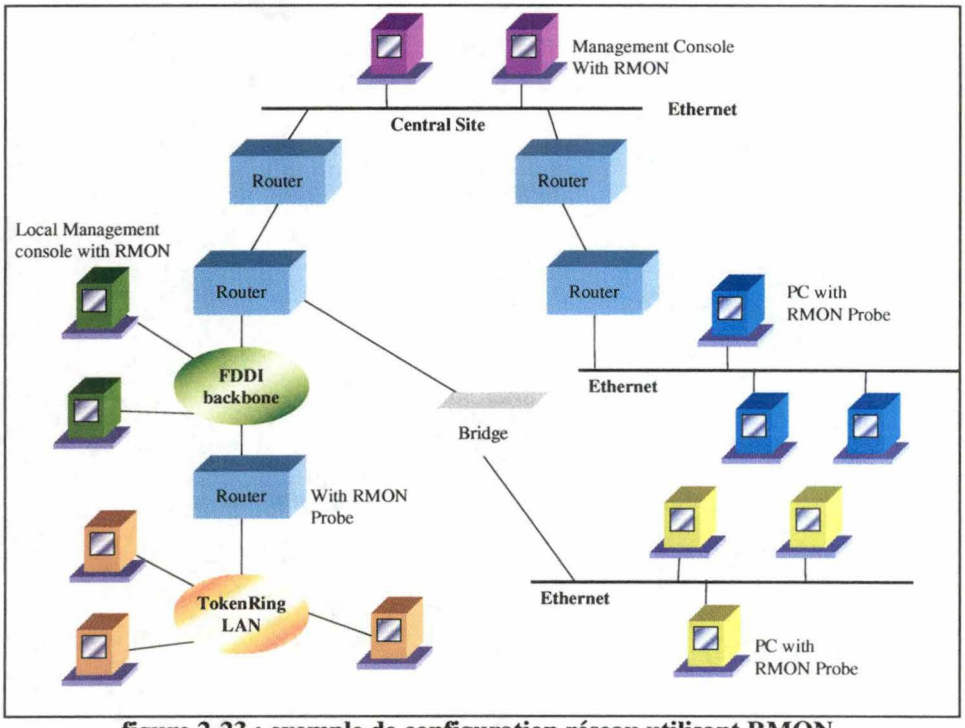

figure 2-23 : exemple de configuration réseau utilisant RMON

Tous les buts ne sont pas toujours atteints par tous les moniteurs réseau mais les spécifications de RMON fournissent les bases pour les supporter tous.

Un système qui implémente la MIB RMON est appelé RMON *Probe*. La sonde a un agent qui n'est pas différent de celui de SNMP. Il contient aussi une entité de traitement propre qui exécute les fonctionnalités de RMON. Cette entité est capable de lire et d'écrire dans la MIB RMON locale en réponse à une action de gestion. L'entité peut aussi exécuter une série de fonctions propres à RMON. Un moniteur distant peut être implémenté, soit comme un appareil dédicacé à part entière, soit comme une fonction disponible sur un système dont une partie des ressources est spécialement allouée pour la surveillance.

Afin de gérer efficacement un moniteur distant, la MIB RMON contient des éléments qui supportent le contrôle à partir d'une station de gestion. Ces éléments sont divisés en deux catégories : configuration et demande d'action.

Configuration

Un moniteur distant a besoin d'être configuré pour collecter des données. La MIB est organisée en un nombre de groupes qui contiennent une ou plusieurs tables de contrôle et une ou plusieurs tables de données. Une table de contrôle est de type *Read/Write* et contient les paramètres décrivant les données de la table de données de type *Read-Only*.

Les paramètres sont mis en place en ajoutant ou en modifiant une ligne dans la table de contrôle. Les informations sont récoltées en accord avec les paramètres de la table de contrôle, et sont enregistrées dans les tables de données correspondantes. Les fonctions exécutées par un moniteur sont donc définies en terme de lignes d'une table.

Pour modifier un paramètre dans une table de contrôle, il est d'abord nécessaire d'invalider la ligne existante dans la table de contrôle. Cela aura pour effet d'effacer la ligne concernée mais aussi d'effacer toutes les lignes concernées dans les tables de données. Ce même mécanisme est utilisé pour désactiver la récolte d'un type de données. Les ressources associées à cette collection sont aussi libérées.

Demande d'action

SNMP ne fournit pas de mécanisme pour faire exécuter une commande à un agent. SNMP ne sait que lire des valeurs d'objet ou écrire une valeur d'objet d'un MIB. Mais il est possible d'utiliser l'opération *Set* pour activer une commande : il s'agit du processus appelé *action invocation* (demande d'action).

Un objet peut-être utilisé pour représenter une commande : il suffit que l'objet soit mis à une valeur précise pour déclencher une action. La MIB RMON contient un certain nombre d'objets de ce type.

Gestionnaire multiple

Une sonde RMON peut-être assujettie par plusieurs stations de gestion. Comme les accès concurrentiels à une ressource sont permis, il peut y avoir des conflits ou des résultats non-désirés. Dans le cas d'une sonde RMON partagée, différentes difficultés peuvent survenir

- Les accès concurrents aux ressources peuvent surpasser les capacités du surveillant pour fournir ces ressources ;
- Une station de gestion peut accaparer les ressources du moniteur pour une longue période empêchant ainsi une autre station de gestion d'exécuter des fonctions de gestion ;
- Les ressources peuvent être allouées à une station de gestion qui peut tomber en panne et donc ne pas relâcher les ressources qu'il utilisait.

Afin de résoudre ces problèmes, de simples éléments ont été inclus dans la MIB RMON. Chaque table de contrôle est composée d'une colonne qui permet d'identifier le propriétaire d'une ligne particulière de la table et des fonctions qui y sont associées. Cette étiquette peut être utilisée de ces différentes façons :

- Une station de gestion peut reconnaître les ressources qu'elle utilise ou non ;
- Un opérateur réseau peut identifier la station de gestion à qui appartient une ressource ou une fonction particulière afin de négocier, si besoin est, leur utilisation ;
- Un opérateur réseau peut avoir l'autorité de libérer unilatéralement les ressources utilisées par un autre gestionnaire ;
- Si une station de gestion a dû être réinitialisée, elle peut reconnaître les ressources qu'elle avait utilisées avant et les libérer.

Les spécifications RMON suggèrent que cette étiquette contienne l'adresse IP, le nom de la station de gestion, le nom du gestionnaire réseau, le lieu ou le numéro de téléphone. Ce concept d'étiquette ne joue pas le rôle d'un mot de passe ou de mécanisme de contrôle d'accès. Le contrôle d'accès est celui de SNMP : basé sur le principe du nom de communauté. Donc une table de contrôle qui a un accès *Read/Write* est disponible en lecture et en écriture par toutes les stations de gestions pour qui les tables sont visibles (MIB RMON).

Une série de fonctions est mise en place par défaut lors de l'initialisation d'un moniteur réseau. Les lignes de contrôle qui définissent ces fonctions sont la propriété du moniteur. Par convention, chaque étiquette de propriété est initialisée avec la valeur `monitor`. Les ressources associées à ses fonctions sont gérées par le moniteur lui-même. Une station de gestion peut utiliser ces fonctions en lecture seule et ne peut donc pas les modifier ou les effacer.

2.2.3.2 RMON MIB

Une grande partie des spécifications de RMON est dévolue à la définition de la MIB RMON qui est maintenant incluse dans la MIB-II comme une branche à part entière dont l'identifiant est le 16.

Chaque groupe (figure 2-24)est utilisé pour enregistrer des données et des statistiques dérivées à partir de ces données récoltées à partir du moniteur à propos du ou des sous-réseaux qui y sont connectés. Tous les groupes de la MIB RMON sont optionnels. Cependant il existe trois dépendances :

- Le groupe `alarm` nécessite l'implémentation du groupe `event` ;
- Le groupe `hostTopN` nécessite l'implémentation du groupe `host` ;
- Le groupe `capture` nécessite l'implémentation du groupe `filter`.

`statistics` *Group*

La RFC 1757 [RFC1757] définit une table unique pour ce groupe, avec une entrée pour chaque interface surveillée. Les statistiques sont sous formes de compteurs qui démarrent à zéro dés qu'une entrée correcte dans la table est créée. La table `EtherStatsTable` collecte une série de compteurs pour chacun des sous-réseaux : le nombre d'octets, de paquets d'erreurs, de trames, etc. Le groupe `statistics` fournit des informations utiles sur la charge et sur la santé globale d'un sous-réseau car les erreurs, les CRC, les collisions, et les paquets sur et sous dimensionnés sont comptabilisés.

Cette table est utilisée pour les interfaces `Ethernet`. Une table pour les statistiques d'un sous-réseau `Tokenring` a été ajoutée. Les extensions futures de la MIB incluront d'autres types de LAN comme `FDDI`.

A titre d'exemple, voici quelques compteurs du groupe `statistic` :

- `etherStatsOctets` : le nombre d'octets de données reçus (y compris les mauvais paquets);
- `eherStatsPkts` : le nombre de paquets reçus y compris les mauvais paquets, les paquets broadcast et multicast;
- `etherStatsUndersizePkts` : le nombre de paquets reçus qui sont bien formés mais qui ont une longueur inférieure à 64 octets;
- `etherStatsCollisions` : la meilleure estimation du nombre total de collisions;
- `etherStatsPkts256to511Octets` : le nombre de paquets reçus (y compris les mauvais paquets) qui ont une longueur comprise entre 256 et 511 octets ;
- ...

Pour le groupe `statistics`, les fonctions des tables de contrôle et de données sont combinées. Les seuls objets de type *Read/Write* sont `etherStatsDataSource`, `etherStatsOwner` et `etherStatsStatus`. Donc, une station de gestion peut demander au moniteur de collecter les statistiques, qui sont les mêmes pour toutes les interfaces, sur une ou plusieurs interface `Ethernet`.

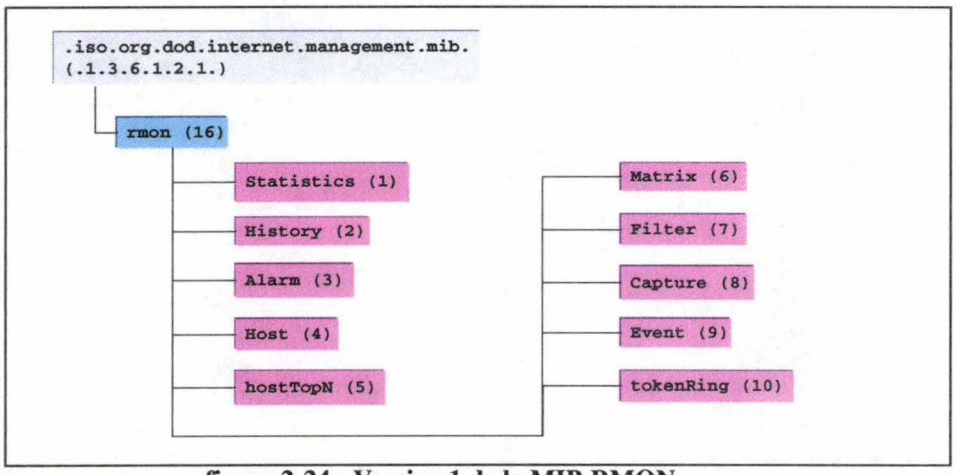

figure 2-24 : Version 1 de la MIB RMON

`history` *Group*

Le groupe `history` est utilisé pour définir des fonctions d'échantillonnage pour une ou plusieurs interfaces Ethernet. La RFC 1757 [RFC1157] définit deux tables `historyControlTable` et `etherHistoryTable`. Chaque ligne de la table `historyControlTable` définit un jeu d'échantillon à récolter à un intervalle donné pour une interface particulière. Chaque échantillon est enregistré dans la table `etherHistoryTable`.

Le surveillant ou la station de gestion peut définir un nouvel historique qui est unique en terme d'interface et d'intervalle d'échantillonnage. On peut donc avoir, pour un sous-réseau donné, plus d'un processus d'échantillonnage qui doivent tous avoir un intervalle différent. Chaque interface possède un nombre fini d'enregistrements pour y stocker un historique. C'est donc une zone tampon où on écrase les enregistrements les plus vieux si nécessaire.

Une période d'échantillonnage courte permet de détecter les changements soudains du trafic et les périodes longues mettent en évidence le comportement stable du trafic de l'interface à surveiller.

`host` *Group*

Le groupe `host` est utilisé pour récolter des statistiques à propos d'hôtes spécifiques du LAN. Le surveillant repère les nouveaux hôtes en observant les adresses MAC source et destination contenues dans les paquets. Pour chaque hôte spécifié au surveillant, une série de statistiques est maintenue. Une table de contrôle `hostControlTable` détermine pour quelle adresse cette fonction est exécutée, et deux tables de données, `hostTable` et `hostTimeTable`, servent à enregistrer les données récoltées.

Idéalement le moniteur devrait être capable de maintenir des statistiques pour tous les hôtes découverts d'un sous-réseau. Cependant, si le moniteur est à court de ressources, il peut décider d'effacer des données si nécessaire. De la même manière que pour la table de données de groupe `history`, ce sont les données les plus anciennes qui sont écrasées lorsque de nouvelles données sont enregistrées.

Chaque ligne de la table `hostTable` contient des statistiques à propos d'un hôte spécifique. Cette table est indexée sur l'adresse MAC de l'hôte et sur l'interface sur laquelle elle est connectée. La table `hostTimeTable` contient exactement les mêmes informations, ligne par ligne, que la table `hostTable` mais indexée sur l'ordre de création.

A titre d'exemple, voici quelques compteurs du groupe `host` :

- `hostInPkts` : le nombre de bons paquets transmis à cette adresse;
- `hostOutOctets` : le nombre d'octets transmis par cette adresse, y compris les mauvais paquets;
- `hostOutErrors` : le nombre de mauvais paquets transmis par cette adresse;
- `hostOutMulticastPkts` : le nombre de bons paquets multicast transmis par cette adresse;
- ...

Les deux tables de données contenant les mêmes informations organisées de deux manières différentes, pourraient être regroupées en une seule table tant que la vue logique présentée à la station de gestion est le reflet des deux tables.

`hostTopN` *Group*

Le groupe `hostTopN` est utilisé pour maintenir des statistiques à propos d'une série d'hôtes d'un sous-réseau qui sont les premiers d'une liste basée sur quelques paramètres. Par exemple, une liste peut-être composée des 10 hôtes qui ont transmis le plus de données un jour particulier.

Les statistiques générées pour ce groupe sont dérivées des données du groupe `host`. La série de statistiques pour un objet du groupe `host` sur une interface d'un sous-réseau, récoltée pendant un intervalle d'échantillonnage est appelé un *Report*.

Le groupe `hostTopN` est composé d'une table de contrôle `hostTopNControlTable` et d'une table de données `hostTopNTable`. La station de gestion crée une ligne dans la table de contrôle pour spécifier un nouveau *Report*. Cette entrée permet d'avertir le moniteur de mesurer la différence entre les valeurs de début et de fin d'une variable du groupe `host` particulière durant une période d'échantillonnage.

`matrix` *Group*

Le groupe `matrix` est utilisé pour enregistrer des informations à propos du trafic entre des paires d'hôtes d'un sous-réseau. Ces informations sont stockées sous forme d'une matrice. Cette méthode d'organisation est utile pour retrouver de l'information spécifique sur une paire donnée.

Le groupe matrix est constitué de trois tables : une table de contrôle (`matrixControlTable`) et deux tables de données (`matrixSDTable` et `matrixDSTable`). La table `matrixSDTable` est utilisée pour stocker les statistiques sur le trafic d'un hôte source particulier vers un certain nombre d'hôtes destination. Cette table est indexée sur l'index de la matrice, sur l'adresse source et ensuite sur l'adresse destination. La table `matrixDSTable` contient les mêmes informations mais indexées sur l'index de la matrice, puis sur l'adresse destination et ensuite sur l'adresse source.

Chaque ligne de la table de contrôle identifie un sous réseau unique. Les tables de données contiennent deux lignes pour chaque paire d'hôtes de ce sous-réseau qui ont échangé récemment des informations. Une des deux lignes témoigne du trafic dans une direction et l'autre ligne dans le sens opposé. Donc la station de gestion peut facilement indexer les informations pour mettre en évidence le trafic entre un hôte et les autres hôtes et vice versa.

Les deux tables de données contenant les mêmes informations organisées de deux manières différentes, on pourrait les regrouper en une seule table tant que la vue logique présentée à la station de gestion est le reflet des deux tables.

Si les limites d'une table sont atteintes, alors ce sont les données les plus anciennes qui sont écrasées au profit des nouvelles.

`Tokenring` *extensions to RMON*

La RFC 1513 [RFC1513]définit des extensions à la MIB RMON pour gérer les réseaux Token Ring 802.5. La plupart des groupes d'objets de RMON sont appropriés pour plusieurs types de réseau. Cependant, les objets de type compteur des groupes `statistics` et `history` sont dépendants du type de media. RMON lui-même définit des objets spécifiquement pour les sous réseaux `Ethernet`, alors que la RFC 1513 définit de nouvelles tables d'objets pour les sous-réseau Token Ring.

Ces extensions de la MIB RMON se situent à deux niveaux bien précis :

- Un groupe `tokenring` est ajouté à la MIB RMON et est constitué de quatre tables :
  - Token Ring Ring Station Group : ce groupe contient les statistiques et des informations d'état associées à chaque station sur l'anneau local. En plus, ce groupe fournit des informations sur l'anneau qui est surveillé.
  - Token Ring Ring Station Order Group : ce groupe fournit des informations sur l'ordre des stations sur l'anneau local surveillé.
  - Token Ring Ring Station Configuration Group : ce groupe gère les stations *Token Ring* via des moyens actifs. Chaque station sur un anneau surveillé peut être enlevée ou avoir sa configuration modifiée par téléchargement.
  - Token Ring Ring Source Routing Group : ce groupe contient des statistiques d'utilisation dérivées des informations de routage optionnelles contenues dans les paquets *Token Ring*.
- De nouvelles tables sont ajoutées aux groupes `statistics` et `history` :
  Les statistiques spécifiques au sous-réseau Token Ring sont répertoriées dans deux tables : la table *Token Ring Mac-Layer Statistics* (`tokenRingMLStatsTable`) et la table *Token Ring Promiscuous Statistics* (`tokenRingPStatsTable`). A la différence du protocole Ethernet CSMA/CD 802.3 qui met en place seulement la transmission des paquets de données au niveau de la couche MAC, le protocole MAC de *Token Ring* comprend une série de paquets de contrôle pour gérer le réseau *Token Ring*. Donc un jeu de statistiques est également fait sur les paquets de contrôle.
  Au niveau du groupe history deux nouvelles tables sont aussi ajoutées : la table *Token Ring Mac-Layer History* (`tokenRingMLHistoryTable`) récolte les informations sur les objets de type compteur qui sont définis dans la table `tokenRingMLStatsTable` ; et la table *Token Ring Promiscuous History* (`tokenRingPHistoryTable`) récolte les informations sur les objets de type compteur qui sont définis dans la table `tokenRingPStatsTable`.

2.2.3.3 Alarmes et filtres

Les autres groupes RMON, qui sont en rapport direct avec les statistiques du trafic, traitent des alarmes, du filtrage et de la capture des paquets.

`alarm` *Group*

Le groupe `alarm` est utilisé pour définir un jeu de seuils pour tester les performances du réseau. Si un seuil est atteint, dans le sens voulu, une alarme est générée et envoyée à la console centrale.

Le groupe `alarm` n'est constitué que d'une seule table (`alarmTable`). Chaque entrée dans la table spécifie une valeur particulière à surveiller, un intervalle d'échantillonnage et les paramètres de seuil. L'entrée unique dans la table pour cette variable utilisant cet intervalle contient la valeur la plus récente obtenue en fin d'échantillon. L'ancienne valeur est donc perdue au profit de la nouvelle.

Lors de la définition d'une nouvelle alarme, par le moniteur ou la station de gestion, deux seuils sont introduits :

- Un seuil maximum qui générera une alarme si la valeur de la variable de l'échantillon actuel dépasse ou est égale au seuil maximum et que la valeur précédente de cette même variable soit inférieure au seuil maximum.
- Un seuil minimum qui générera une alarme si la valeur de la variable de l'échantillon actuel passe sous ou est égale au seuil minimum et que la valeur précédente de cette même variable soit supérieure au seuil minimum.

Le groupe `alarm` définit aussi un mécanisme conçu pour ne pas tenir compte des alarmes mineures générées à maintes reprises. Ce mécanisme est différent pour les deux seuils fixés pour une même variable. Ce mécanisme par lequel les petites fluctuations ne sont pas prises en compte pour la génération d'alarmes est connu dans les spécifications RMON par le terme *hysteresis mechanism*. Si une valeur fluctue de trop et passe successivement au-dessus du seuil maximum et en dessous du seuil

minimum, il est bon de revoir l'intervalle d'échantillonnage : en effet on obtient plus de précision si on échantillonne plus souvent.

`filter` *Group*

Le groupe `filter` fournit un moyen par lequel une station de gestion peut informer un moniteur de n'observer que des paquets bien précis sur une interface particulière. La base de ce groupe est constituée de deux types de filtre différents :

- Les filtres de données (*data filter*) : qui permettent au moniteur de masquer les paquets observés sur base d'un modèle (au niveau des bits) ;
- Les filtres d'états (*status filter*) : qui permettent au moniteur de masquer les paquets observés sur base de leur état (par exemple, valid, CRC error, ...).

Ces filtres peuvent être combinés en utilisant les opérations AND et OR pour former un test complexe à appliquer aux paquets entrants. Les paquets qui passent le test avec succès constituent un *channel* et ils sont comptabilisés. De plus le *channel* peut être configuré pour générer un événement quand un paquet passe au travers du *channel* en mode actif. De plus, les paquets passant au travers du *channel* peuvent être capturés si le mécanisme correspondant se trouvant dans le groupe `capture` est défini.

Le groupe `filter` est constitué de deux tables de contrôle. Chaque ligne de la table `channelTable` définit un seul *channel*. Une ou plusieurs lignes de la table `filterTable`, qui définissent les filtres, sont associées à un *channel*.

Packet `capture` *Group*

Le groupe `Packet capture` peut être utilisé pour créer une zone tampon pour capturer les paquets d'un des *channel* défini dans le groupe `filter`. Il est constitué de deux tables : la table `bufferControlTable` qui définit en détails les fonctions de capture et la table `captureBufferTable` qui enregistre les paquets capturés.

Chaque ligne de la table `bufferControlTable` définit une zone tampon qui est utilisée pour capturer et stocker les paquets passant au travers d'un *channel*. La table de données `captureBufferTable` contient une ligne par paquet capturé.

`Event` *Group*

Le groupe `event` permet de définir des événements. Un événement est déclenché par une condition située ailleurs dans la MIB, et un événement peut déclencher une action définie quelque part dans la MIB. Un événement peut provoquer l'enregistrement d'informations dans un journal et aussi l'envoi d'un message de type *trap*.

De même, il est possible qu'un événement défini dans ce groupe déclenche une activité en relation avec un autre groupe : par exemple, un événement peut activer ou désactiver un *channel*.

Le groupe `event` est constitué d'une table de contrôle (`eventTable`) et d'une table de données (`logTable`). La table de contrôle contient les définitions des événements. Chaque ligne de la table contient les paramètres décrivant un événement devant être généré sous certaines conditions. Si un événement doit être enregistré, des entrées seront créées dans la table de données.

Les conditions pour qu'un événement se produise sont définies dans les autres groupes RMON. Le groupe `alarm` peut définir des événements pour les limites maximales et minimales et ces événements font référence à un index dans la table `eventTable`. De même, le groupe `filter` peut faire référence à un événement qui se produira quand un paquet est capturé.

2.2.3.4 Résumé

La MIB RMON est un ajout important au cadre de travail de SNMP, car elle définit une série d'objets gérés qui sont utiles pour les fonctions de surveillance à distance. Ces fonctions sont aussi définies dans la MIB RMON. La force de RMON est qu'il est compatible avec le cadre de travail SNMP actuel : il ne nécessite aucune amélioration du protocole.

Le terme "surveillance à distance" (*Remote Monitoring*) fait référence à l'utilisation d'un agent connecté au réseau *broadcast* pour récolter des statistiques concernant le trafic de ce réseau. Un agent n'est responsable que

de l'information de gestion qui lui est propre. Sans une fonction de surveillance à distance, il est difficile, si pas impossible, pour un gestionnaire d'élaborer un profile de l'activité sur un sous réseau en entier.

En plus des capacités à donner des statistiques sur le trafic du réseau, RMON fournit des éléments capables de définir des événements et des alarmes sur base du filtrage ou de la capture de paquets.

2.2.4 RMON2

2.2.4.1 Vue d'ensemble

En 1994, des travaux ont commencé pour étendre les spécifications de la MIB RMON en y incluant la surveillance du trafic au niveau des protocoles supérieurs à la couche MAC. Ces travaux, référencés comme RMON2, sont documentés dans deux RFC.

RMON2 décode les paquets des couches 3 à 7 du modèle OSI. Ceci permet de mettre en évidence deux importantes implications :

1) une sonde RMON peut surveiller le trafic sur base des protocoles de la couche réseau et des adresses y compris l'*Internet Protocol* (IP). Cela donne la possibilité à la sonde de regarder plus loin que le segment LAN auquel elle est connectée et de voir le trafic entrant sur le LAN via un routeur ;
2) parce que la sonde RMON peut décoder et surveiller le trafic au niveau des applications, comme le courrier électronique (SMTP), le transfert des fichiers (FTP, TFTP), les protocoles liés à Internet (HTTP, HTTPS,...), la sonde peut enregistrer le trafic, généré par une application, venant ou à destination d'un hôte particulier.

Avec RMON2, la sonde RMON a maintenant les capacités de voir au-dessus de la couche MAC en lisant les entêtes des protocoles de la couche réseau. Ceci permet à la sonde d'analyser le trafic passant au travers du routeur pour déterminer la source et la destination finale d'un paquet. Le gestionnaire du réseau peut donc répondre à des questions plus précises concernant l'utilisation du réseau. Par exemple :

- S'il y a une charge excessive du LAN provenant d'un routeur, quel est le réseau ou l'hôte qui est à l'origine de ce trafic massif entrant ?
- Si un routeur est surchargé par une grande quantité de trafic sortant, quel est l'hôte local responsable de ce trafic et qui en est le destinataire ?
- S'il y a une grande charge de trafic de passage (venant d'un routeur et à destination d'un autre routeur), quels sont les hôtes ou les réseaux responsables de ce trafic massif ?

Une sonde RMON2 peut aussi décoder les protocoles supérieurs aux protocoles de la couche réseau. Donc, une sonde RMON2 est capable de lire les entêtes TCP ou les entêtes des applications. Cela permet au gestionnaire de surveiller le trafic plus en détail. Pour RMON2, tous les protocoles supérieurs à la couche réseau sont considérés comme des protocoles de la couche application. Avec RMON2, les applications de gestion de réseau peuvent être implémentées pour générer des graphiques et des tableaux représentant la charge du réseau par protocole ou par application.

Par rapport à RMON, RMON2 apporte deux nouveaux éléments qui améliorent la puissance et la flexibilité de RMON :

- La SMI de SNMPv2 indique explicitement qu'il est possible d'utiliser un objet ne faisant pas partie de la table comme index de cette même table. Ceci permet de lier plus précisément les tables de contrôles et les tables de données.
- Lors du polling, seules les valeurs d'objets qui ont été modifiées sont renvoyées à la station de gestion.

2.2.4.2 RMON2 MIB

La MIB RMON2 est simplement une extension de la MIB RMON d'origine. Cette extension ajoute neuf nouveaux groupes (numérotés de 11 à 19) (figure 2-25):

`Protocol Directory` *Group*

Le groupe `protocol directory` permet de répertorier les protocoles utilisés par les couches supérieures à la couche MAC. Sur chaque réseau, une grande quantité de protocoles peut être utilisée, certains protocoles sont standardisés ou au moins connus, tandis que d'autres sont considérés comme des protocoles propriétaires développés expressément pour une application particulière.

Comme la plupart des objets de la MIB RMON2 traitent de la surveillance des activités de ces protocoles, il était nécessaire de regrouper les informations sur ces protocoles en un point central. Le groupe `protocol directory` fournit, au gestionnaire RMON, un moyen d'apprendre quels sont les protocoles décodés par une sonde RMON2 particulière.

Le groupe est composé d'une table avec une entrée pour chaque protocole que la sonde RMON doit être capable de décoder et de comptabiliser. Cette table contient une indication permettant de savoir quand la dernière mise à jour de la table a été effectuée.

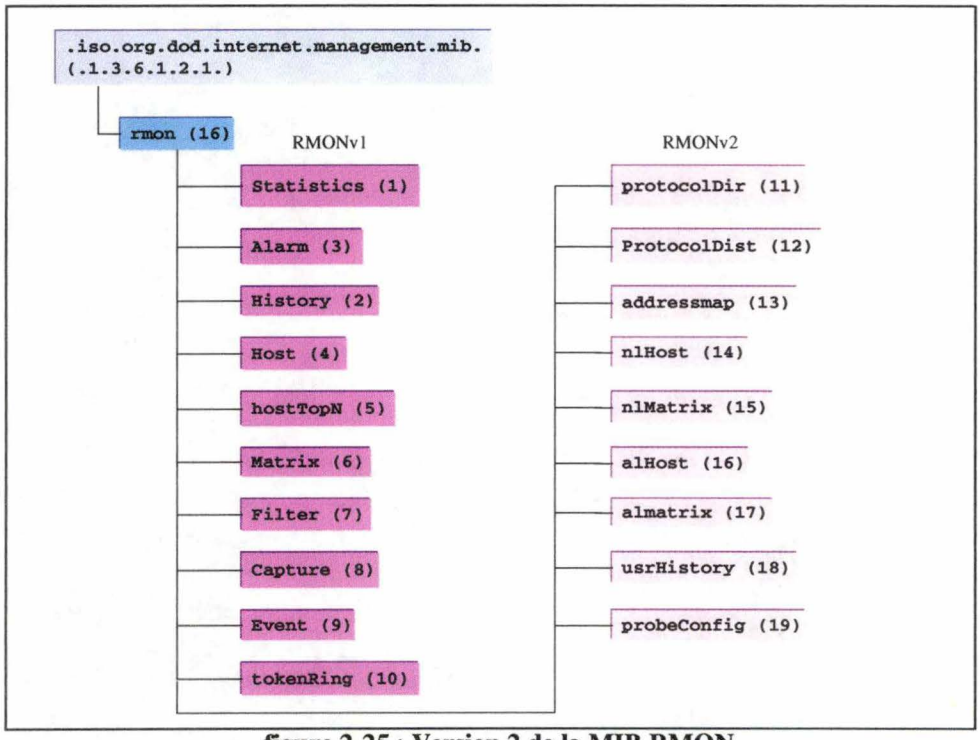

figure 2-25 : Version 2 de la MIB RMON

`Protocol Distribution` *Group*

Le groupe `protocol distribution` résume le nombre d'octets et de paquets qui ont été envoyés pour chacun des protocoles supportés. Il est constitué de deux tables : la table de contrôle `protocolDistControlTable` qui vérifie la récolte des statistiques de base pour chacun des protocoles, et la table de données `protocolDistStatsTable` pour y enregistrer les données.

Chaque ligne de la table `protocolDistControlTable` fait référence à une seule interface réseau pour cette sonde et contrôle un nombre de ligne de la table de données `protocolDistStatsTable` (une ligne par protocole reconnu sur cette interface).

`Address Map` *Group*

Le groupe `address map` fait correspondre chaque adresse réseau (IP ou autre) à une adresse MAC spécifique et à un port particulier sur l'appareil réseau concerné. Ceci est bénéfique pour les applications de découverte des hôtes d'un réseau et pour les applications de topologie réseau qui peuvent ainsi retrouver facilement le chemin parcouru par un type de trafic particulier.

Le groupe est constitué de :

- trois objets scalaires :
  1) `addressMapInserts` : le nombre de fois qu'une correspondance d'adresse a été insérée dans la table de données ;
  2) `addressMapDeletes` : le nombre de fois qu'une correspondance d'adresse a été effacée de la table de données ;
  3) `addressMapMaxDesiredEntries` : le nombre maximum voulu d'entrées dans la table `addressMapTable`. Si la valeur est –1, la sonde peut en créer autant que nécessaire.
- une table de contrôle (`addressMapControlTable`) ;
- une table de données (`addressMapTable`).

La structure des tables de contrôles et de données de ce groupe n'est pas identique aux autres groupes RMON où une entrée dans la table de contrôle crée et contrôle un certain nombre de lignes de la table de données. A la place, il y a une unique table de données qui contient les entrées établissant la correspondance entre une adresse réseau et une adresse MAC. Il y a une entrée dans la table de contrôles par interface (par sous-réseau connecté) et cette entrée permet la recherche d'adresses et l'enregistrement des correspondances d'adresses dans la table de données. Il est à noter que la table de données n'est pas indexée par une ligne de la table de contrôles, ce qui permet à une application de gestion de réseaux de repérer facilement les entrées doubles.

Il ne doit y avoir qu'une seule adresse MAC pour une adresse réseau spécifique (seulement une entrée dans la table de données) et il peut y avoir plusieurs adresses réseau pour une adresse MAC (plusieurs entrées dans la table de données sont possibles).

Ce groupe peut-être utilisé pour détecter les adresses IP doubles, car le système de gestion du réseau peut exécuter cette fonction en interprétant les informations de correspondance d'adresses contenues dans la table de données.

RMON2 Host *Group*

Deux groupes RMON2 traitent les statistiques récoltées sur base des hôtes : le groupe `network-layer Host` et le groupe `application-layer Host`. Les deux groupes sont constitués d'une table de données qui est contrôlée par une table de contrôle du groupe `network-layer Host`.

Network-layer Host *Group*

Le groupe `network-layer Host` permet aux utilisateurs de décoder les paquets sur base de leur adresse réseau. Ceci permet à un gestionnaire réseau de voir au-delà du routeur et ce jusqu'aux hôtes connectés.

Cette table récolte les mêmes statistiques que celles du groupe `host` de la version 1 de RMON. La seule différence vient du fait que dans ce cas ci, les statistiques se font sur base des adresses réseau et non plus sur base des adresses MAC.

Le groupe est constitué d'une table de contrôle (`nlHostControlTable`) et d'une table de données (`nlHostTable`). Des entrées seront créées dans la table de données pour tous les protocoles réseau du groupe `protocol directory` qui sont supportés. La sonde ajoute des entrées dans cette table pour toutes les adresses, source ou destination, contenues dans tous les paquets qui n'ont pas d'erreur au niveau de leur adresse MAC.

Le but de la table `nlHostTable` est de récolter des statistiques de base sur le trafic entrant et sortant de chaque hôte découvert sur base des adresses réseau. Lorsqu'une nouvelle ligne est introduite dans la table `nlhostcontroltable`, le moniteur commence à apprendre les adresses réseau de l'interface correspondante. Chaque fois qu'une adresse réseau est découverte sur cette interface, une ligne est ajoutée dans la table `nlHostTable` et on ajoute 1 à la valeur actuelle de `nlHostControlNlInserts`.

Voici quelques compteurs que l'on retrouve dans le groupe `network-layer host` :

- `nlHostInPackets` : le nombre de paquets corrects transmis à cette adresse depuis qu'elle est présente dans la table;
- `nlHostOutOctets` : le nombre d'octets transmis par cette adresse depuis qu'elle est présente dans la table, sans compter les octets des paquets contenant des erreurs;
- `nlHostOutMacNonUnicatsPkts` : le nombre de paquets transmis par cette adresse qui ont pour destination des adresses MAC broadcast ou multicast;
- ...

Depuis qu'il est possible pour un hôte unique d'avoir plusieurs adresses réseau, il peut y avoir plus d'entrées dans cette table que d'hôtes connus.

Application-layer Host *Group*

La table `nlHostControlTable` contrôle aussi la table `alHostTable` du groupe `application-layer host`. Il y a une entrée dans la table `alHostTable` par protocole de la couche application découvert pour chaque adresse réseau connue.

La table alHostTable créera les entrées pour tous les protocoles de la couche application supportés (c'est à dire ceux présents dans la table du groupe protocol directory). La sonde ajoute les entrées dans cette table pour toutes les adresses source et destination contenues dans les paquets qui n'ont pas d'erreurs MAC.

Sur base d'une adresse réseau donnée, des protocoles réseau et applications observés sur une interface particulière pendant un certain temps, les statistiques du trafic pour un protocole de la couche application peuvent être lues.

Voici quelques compteurs du groupe application-layer host :

- alHostOutPackets : le nombre de paquets corrects d'un type de protocole donné qui ont été transmis à partir par cette adresse depuis qu'elle est présente dans la table ;
- alHostInOctets : le nombre d'octets d'un protocole donné transmis à cette adresse depuis qu'elle est présente dans la table, sans compter les octets contenus dans les trames erronées ;
- ...

La table alHostTable permet à un utilisateur de tracer le trafic entrant et sortant d'un hôte sur base des protocoles de la couche application.

RMON2 Matrix *Group*

Deux groupes RMON2 traitent les statistiques récoltées à propos de paires d'hôtes : le groupe network-layer matrix et le groupe application-layer matrix. Les tables de données de ce groupe récoltent des statistiques similaires à celles récoltées dans le groupe matrix et hostTopN de la version 1 de RMON. Cependant, les groupes RMON1 récoltent des statistiques sur base des adresses MAC tandis que les groupes RMON2 récoltent des statistiques sur base des adresses réseau et des protocoles application.

Network-layer Matrix *Group*

Le groupe network-layer matrix est constitué de cinq tables : deux tables de contrôle et trois tables de données. Une des tables de contrôle (nlMatrixControlTable) et ses deux tables de données (nlMatrixSDTable et nlMatrixDSTable) traitent les statistiques récoltées (sous forme de matrices), tandis que l'autre table de contrôle (nlMatrixTopNControlTable) et la table de données restante (nlMatrixTopNTable) traitent les statistiques topN récoltées.

La table nlMatrixSDTable est utilisée pour enregistrer des statistiques sur le trafic entre une adresse réseau source particulière et un nombre de destinations. Cette table créera une entrée par protocole réseau supporté. La table nlMatrixDSTable contient les mêmes informations mais indexées d'abord sur l'adresse destination et puis sur l'adresse source.

Chaque ligne de la table nlMatrixControlTable identifie un sous-réseau unique. La table nlMatrixSDTable contient deux lignes pour chaque paire d'hôtes sur ce sous-réseau qui ont échangés récemment des informations : une des lignes concerne le trafic dans un sens entre une paire d'hôtes, et l'autre ligne concerne le trafic en sens inverse. Les mêmes lignes sont reprises dans la table nlMatrixDSTable. Une station de gestion peut donc indexer les informations concernant le trafic provenant d'un hôte ou à destination d'un hôte et ce en utilisant une des deux tables de données.

Quand le moniteur détecte un échange qui implique une nouvelle paire d'hôtes, il crée une nouvelle ligne dans chacune des deux tables. Si la limite d'une table est dépassée, le moniteur effacera des lignes de la table.

Comme les deux tables contiennent les mêmes informations organisées de deux façons différentes, il est possible d'implémenter une seule table tant que la vue logique présentée à la station de gestion est le reflet des deux tables.

Le groupe network-layer matrix contient aussi des statistiques qui traitent des statistiques topN dont la philosophie est différente de celle de la version 1 de RMON. Pour RMON1 le groupe hostTopN maintient des statistiques pour des hôtes d'un sous-réseau classés sur base de quelques paramètres pour un seul hôte. Pour RMON2, les tables contenant les

statistiques sont classées sur base de quelques paramètres aussi mais sur base du trafic d'une paire d'hôtes. Cela permet de mettre en valeur le trafic entre deux hôtes sur base de valeurs précises.

La table `nlMatrixTopNTable` est indexée par le champ `nlMatrixTopNControlIndex` qui est un élément de la table de contrôle et par `nlMatrixTopNIndex` qui fait partie de la table de données. Le premier index définit une série de lignes qui constitue un rapport. Le deuxième index est utilisé pour classer les N premières paires source-destination en terme de quantité de trafic observé (en paquets ou en octets).

Au début, la station de gestion crée une ligne dans la table de contrôle pour spécifier un nouveau rapport. Cette entrée incite le moniteur à mesurer la différence entre les valeurs de début et de fin d'un échantillonnage d'un hôte particulier. Lorsque la période d'échantillonnage est finie, le moniteur calcule les résultats finaux et crée n lignes de données pour y mettre les n premiers résultats.

Chaque nouveau rapport écrase les données précédentes, c'est pourquoi il est utile qu'un système de gestion de réseaux demande fréquemment les valeurs enregistrées dans ces tables, afin d'avoir une vue plus complète sur le trafic.

`Application-layer Matrix` *Group*

Le groupe `application-layer matrix` est constitué de trois tables de données et d'une table de contrôle. Deux des tables de données traitent les statistiques récoltées (sous forme de matrice), tandis que la table de données restante traite les statistiques `topN`.

Les deux tables de données traitent les statistiques source-destination basées sur les protocoles application. La table `alMatrixSDTable` est utilisée pour enregistrer les statistiques du trafic venant d'une adresse de la couche application vers un certain nombre de destinations. Cette table créera des entrées pour tous les protocoles supportés.

L'indexation de cette table se fait sur base de six objets provenant de trois tables différentes. La table `alMatrixDSTable` contient les mêmes infos que la table `alMatrixSDTable` mais d'abord indexée sur l'adresse destination et ensuite sur l'adresse source.

Les tables de contrôle et de données reprenant les statistiques topN de la couche application ont les mêmes structures que les tables `topN` concernées par les statistiques de la couche réseau. Une différence existe : la définition du taux d'utilisation de trafic est légèrement différente car le nombre de paquets et d'octets concernés ne tient pas compte du trafic généré par les protocoles "enfants".

`User History Collection` *Group*

Le groupe user history collection demande régulièrement des statistiques ou des variables particulières et enregistre les données sur base des paramètres définis par l'utilisateur. Avec cette nouvelle caractéristique, le gestionnaire du réseau peut obtenir des historiques sur n'importe quel compteur entretenu par une sonde RMON. Les spécifications RMON1 ne permettaient que les historiques sur une série prédéfinie de statistiques.

La structure de ce groupe est la plus complexe de tous les groupes RMON. Elle est constituée de tables hiérarchisées sur trois niveaux. Au sommet de la structure, on retrouve une table de contrôle (`usrHistoryControlTable`) qui spécifie en détails la fonction d'échantillonnage. Sous cette première table on retrouve une ou plusieurs instances de la table `usrHistoryObjectTable`, chacune définissant les variables à échantillonner. Sous chacune de ces instances, on retrouve une ou plusieurs instances de la table `usrHistoryTable` qui sont utilisées pour enregistrer les données spécifiées.

Le moniteur ou la station de gestion peut définir un nouvel historique qui est unique en terme d'objets et d'intervalles d'échantillonnage. Un index unique est associé par le moniteur à chaque ligne de la table `usrHistoryControlTable`. Associé à chaque historique, on retrouve une instance de la table `usrHistoryObjectTable` qui spécifie les objets à échantillonner et une série d'instances de la

table `usrHistoryTable` dont chaque ligne retient les statistiques récoltées pendant une période d'intervalle pour un objet.

Pour chaque intervalle d'échantillonnage, le moniteur ajoute une nouvelle ligne à chaque instance de `usrHistoryTable`. Le nombre de lignes pour un historique étant limité, on peut considérer ces lignes comme un tampon circulaire : c'est la ligne la plus ancienne qui est effacée au profit de la nouvelle ligne de données. Si pour une raison ou une autre, on doit modifier à la baisse le nombre de lignes pour un historique, ce sont aussi les lignes les plus anciennes qui seraient effacées.

`Probe Configuration` *Group*

Le groupe probe configuration est conçu pour améliorer l'interopérabilité entre les sondes RMON et les gestionnaires en définissant un jeu standard de paramètres de configuration pour les sondes. Cela permet à différentes applications RMON de configurer à distance n'importe quelle sonde RMON.

Ce groupe est constitué de plusieurs objets scalaires et de quatre tables dont voici un descriptif:

- `serialConfigTable` : contient une ligne pour chaque interface série présente sur la sonde ;
- `netConfigTable` : contient une ligne pour chaque interface réseau présente sur la sonde ;
- `trapDestTable` : définit les adresses de destination pour les *trap* générés sur cet appareil. Plusieurs adresses de destination peuvent être associées à un nom de communauté ;
- `serialConnectionTable` : enregistre les paramètres nécessaires à l'initiation d'une connexion SLIP vers une station de gestion.

Extensions to RMON1 for RMON2 Devices

Les spécifications de RMON2 contiennent la définition d'un certain nombre d'objets qui sont ajoutés aux tables définies dans la version 1 de RMON. Il s'agit des objets suivants :

- Un objet de type `createTime` est ajouté dans toutes les tables de contrôles ;
- Un objet de type `droppedFrames` est ajouté dans un certain nombre de tables (inclus comme un objet filtre dans toutes les définitions de filtres) ;
- L'objet `filterProtocolDirLocalIndex` est ajouté dans la table `filterTable`. Cet objet permet d'écarter les paquets qui ne correspondent pas à un protocole et à son filtre associé. Cette opération se fait en enlevant les entêtes successives, en commençant par l'entête de la couche MAC, pour exécuter l'opération de filtrage au niveau d'une couche supérieure (IP, etc.).

2.2.4.3 Résumé

RMON2 étend les possibilités de la MIB RMON d'origine pour y inclure les protocoles supérieurs à la couche MAC. L'ajout des protocoles de la couche réseau, comme IP, permet à la sonde de surveiller le trafic entre routeurs connectés au réseau local. La sonde peut alors surveiller les sources et les destinations du trafic hors réseau local arrivant ou sortant via un routeur. L'ajout de protocoles des couches supérieures, comme la couche application, permet à la sonde de fournir des statistiques détaillées sur le trafic au niveau des applications.

Les différents éléments qui devront être pris en compte par la gestion du réseau dans la partie pratique de ce document supportent les versions 1 et 2 de RMON. Il faudra seulement prendre en considération la mise en place de RMON et percevoir les avantages que l'on pourra en tirer dans le cadre réel de la partie pratique.

Remarques :

Actuellement le volume de données à traiter par les sondes RMON est déjà fort important. La centralisation de ces mêmes données vers un gestionnaire est aussi une source de problèmes. En effet, plus les capacités des réseaux augmentent (taille, vitesse, ...) plus les données à traiter sont nombreuses. Cette constatation vaut aussi pour les analyseurs de protocoles.

A titre d'exemple et pour mettre un point d'interrogation sur la viabilité de RMON dans sa version actuelle on peut se pencher sur l'arrivée de réseau 10 Gigabit Ethernet. Pour parvenir à s'adapter au volume de trafic 10G Ethernet des compteurs sur 64 bits sont indispensables [WP-eanm]. Les compteurs 32 bits actuellement mis en œuvre dans un grand nombre d'outils sont incapables de compter jusqu'à des niveaux suffisamment élevés. Sur base d'un paquet moyen circulant sur Internet (256k), il faudra dans ce cas à peine 3,7 secondes pour qu'un compteur d'octets 32 bits atteigne sa limite et moins de 16 secondes pour que le compteur de paquets atteigne la sienne.

Chapitre 3 : Policy-based Network Management

3.1 Introduction

Depuis des années, l'industrie relative aux réseaux IP a développé différents moyens pour gérer les réseaux. Les premiers essais ont fourni des mécanismes et des protocoles permettant de gérer et de configurer de manière individuelle, les différents appareils composant le réseau, plutôt que de permettre une gestion du réseau dans son entièreté.

De plus, les frais généraux inhérents à la gestion des réseaux ont augmenté, pas seulement à cause de l'augmentation de la taille des réseaux, mais aussi à cause de l'augmentation des capacités et des services disponibles sur les différents appareils composant un réseau. Dans un effort de relativiser la complexité galopante de la gestion des réseaux, l'industrie des réseaux a commencer à reconnaître le besoin de déplacer le centre d'intérêt des protocoles de transport et des mécanismes de configuration vers la gestion de l'information et sa distribution. Le concept de gestion des réseaux basée sur une politique bien précise en découle directement.

La gestion des réseaux basée sur une politique (*Policy-based Network Management* (PNM)) essaie d'apporter des perfectionnements aux services disponibles sur Internet en configurant intelligemment les appareils réseaux (comme les *switch* et les routeurs) et leur software en utilisant des règles bien définies. Ces règles peuvent être développées localement ou elles peuvent être récupérées à partir d'une base de données centrale via des serveurs de règles (policy) en utilisant des protocoles comme *Simple Network Management Protocol* (SNMP), *Common Open Policy Service* (COPS), DIAMETER, *LightWeight Directory Access Protocol* (LDAP). Le principe des politiques s'applique, entre autre, à l'amélioration des qualités de service (QoS) fournies.

3.2 Concepts

3.2.1 Définitions

Implémenter des politiques dans les réseaux de données n'est pas un concept totalement nouveau [ste1999]. Les *firewall* et les *switch* qui ont des possibilités de filtrage de paquets, implémentent en effet des politiques. Les routeurs et les serveurs de noms implémentent aussi des politiques. Configurer des routes spécifiques par défaut et établir des routes statiques pour des circonstances spéciales sont des exemples d'implémentations simples de politiques. Retirer des entrées dans les serveurs de noms est une autre technique d'implémentation des politiques. Il est donc nécessaire de définir un peu mieux le terme "politique". Dans la littérature, on en retrouve plusieurs définitions :

- Une politique (policy) est un terme générique utilisé pour décrire une régulation des accès aux ressources du réseau et aux services en se basant sur un ou des critères administratifs [raj1999];
- Une politique est un jeu de règles qui décrivent les actions à prendre lorsque certaines conditions sont rencontrées [bor];
- Une combinaison de règles et de services où les règles définissent les critères d'accès aux ressources et les critères d'utilisation pour gérer la bande passante disponible pour un trafic particulier. Une politique impose un nombre de conditions qui doivent être rencontrées avant qu'une action précise soit prise ;
- Un jeu de règles pour administrer, gérer et contrôler les accès aux ressources du réseau [ID-pt2001] ;
- Un but défini, une ligne de conduite ou une méthode d'action pour indiquer et déterminer les décisions présentes et futures. Les politiques sont implémentées ou exécutées dans un contexte particulier [ID-pt2001] ;
- ...

Il est utile, par la même occasion, de définir d'autres termes employés dans le cadre du *Policy-based Network Management* :

- Un domaine : une collection d'éléments et de services administré d'une façon coordonnée [ID-pt2001]
- Un domaine administratif : une collection d'éléments du réseau sous le même contrôle administratif et groupés ensemble dans un but administratif. Il est généralement géré par une seule entité. Dans le cadre de l'amélioration des QoS, un domaine du réseau fait référence aux domaines qui partagent des politiques QoS communes ;
- Un domaine "politique" : une partie du réseau assujettie à une politique. Une politique s'applique à un domaine et les domaines politiques ne se chevauchent pas ;
- Un domaine "politique" : une collection d'éléments et de services, et/ou une portion de l'Internet gérés d'un façon coordonnée par un jeu de politique commun et cohérent [ID-pt2001] ;
- Un *repository* : est considéré comme un endroit où l'on stocke des données ;

- Un mécanisme : Une opération ou un algorithme spécifique qui est implémenté dans un nœud dans le but de réaliser un ou plusieurs *Per-Hop-Behaviour*.

Dans une certaine mesure, la mise en œuvre de politiques pour la gestion des réseaux pourra s'effectuer sous les différentes formes de gestion d'un réseau :

- Gestion des problèmes : définition de règles permettant de prendre des actions en fonction des problèmes. Ces actions peuvent être diverses, allant du simple contact des personnes responsables à la modification automatique de la configuration du réseau ou d'une partie du réseau ;
- Gestion des comptes : définition de règles permettant la génération automatique de rapports sur l'utilisation du réseau, la fréquence des rapports, ... ainsi que la définition des algorithmes permettant de comptabiliser (et donc de facturer) l'utilisation du réseau ;
- Gestion de la configuration : définition des règles permettant d'établir des statistiques et le calcul des performances dans un but de planification à plus long terme du réseau ;
- Gestion des performances : définition des règles concernant la politique de sécurité du réseau, des méthodes d'audit et de contrôle d'accès, ... ;
- Gestion de la sécurité : définition des règles de configuration sur base d'accords commerciaux et sur base de l'état actuel du réseau, ...

La gestion de la configuration est un des points forts de la gestion via les politiques, car celle-ci permet, de manière pro active et dynamique, de garantir des qualités de services. La gestion de la configuration dépend fortement des autres formes de gestion : par exemple, si un problème survient, alors la configuration du réseau sera probablement modifiée (au moins durant la phase de réparation).

3.2.2 Exigences

Pour implémenter des politiques, dans un réseau de données, il est nécessaire d'avoir une infrastructure focalisée sur la distribution et la gestion des informations décrivant l'état et la configuration des entités de tout le réseau. Une telle infrastructure permettrait aux applications tournant un peu partout d'accéder aux données dont elles ont besoin.

De plus, le cadre de travail permettant la mise en place du concept de politique dans un réseau, ayant pour but de gérer les configurations, doit contenir les éléments suivants [lew2000]:

- une méthode pour définir les domaines d'un réseau ;
- une méthode pour définir les politiques ;
- une méthode pour relier les politiques à un domaine ;
- un gestionnaire de politiques pour surveiller les objets, exécuter les politiques et décider au cas où un conflit apparaît entre les politiques.

3.2.3 Principes de base

Internet est en train de passer d'un modèle de service de type *best-effort* (sans aucune garantie) vers un modèle capable de fournir des services prévisibles et à différents niveaux afin d'assurer des qualités de services (QoS). A titre d'exemple, on peut citer un fournisseur d'accès qui désire fournir un traitement préférentiel pour le trafic web orienté transaction en temps réel, ou un ISP (*Internet Service Provider*) qui veut s'assurer que le trafic *Voice over Ip* est bien assigné à une classe de service assurant une garantie contre la perte et une garantie de délai. De la même manière, mais en compliquant, on peut considérer un administrateur d'un réseau Intranet ayant des capacités RSVP qui désire restreindre les réservations provenant d'une certaine source durant la journée et aussi de limiter la bande passante maximum allouée à ce type de flux.

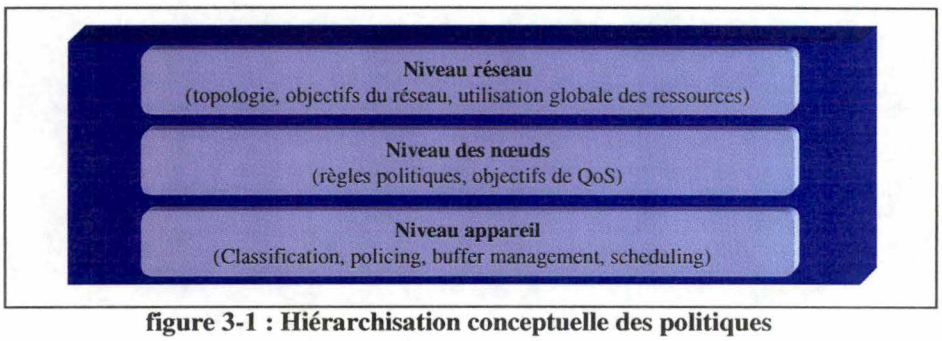

figure 3-1 : Hiérarchisation conceptuelle des politiques

Ces exemples montrent que l'utilité des QoS dépend fortement des mécanismes administratifs permettant de réguler l'accès aux ressources en se basant sur des catégories comme les utilisateurs, les hôtes, les applications, les comptes, etc.

De manière conceptuelle, une politique peut-être exprimée à différents niveaux (figure 3-1). Du point de vue du réseau, dans son entièreté, une politique est une perspective intuitive et de haut niveau de la topologie, de la connectivité, des objectifs de performance de bout en bout et de l'état dynamique du réseau [raj1999]. Cette politique du réseau est composée de plusieurs politiques des nœuds en fonction des objectifs et des spécificités des appareils du réseau. A ce niveau, les règles de politique sont perçues comme des injonctions atomiques au travers desquelles les multiples nœuds du réseau sont contrôlés. Ces différentes injonctions doivent pouvoir être traduites en instructions spécifiques aux appareils du réseau.

Dans le cadre d'une politique ayant pour but de garantir des QoS, il est nécessaire, pour déterminer cette politique, de connaître ou de décrire clairement l'utilisation du réseau; c'est à dire :

- Les points finaux ou terminaux des communications (les adresses IP source et destination, les adresses MAC source et destination, les adresses des DNS ou encore l'identifiant du domaine BGP, etc.) ;
- La route ou le chemin de transmission : le traitement des paquets et la disponibilité des ressources dépendent de la route suivie par le flux des paquets de données ;
- Les utilisateurs ou les groupes d'utilisateurs qui jouent un rôle important pour la détermination de leur accès au réseau ;
- L'information sur les applications : les caractéristiques des applications générant du trafic ont une influence sur la qualité et sur la nature des ressources allouées au trafic (par exemple les applications en temps réel) ;
- Les caractéristiques dynamiques du réseau : il est souvent utile de tenir compte de la disponibilité des ressources du réseau ou de leur manque, lorsque l'on permet ou refuse l'utilisation de ressources par un flux ou un groupe particulier ;
- Le moment de la journée : il est aussi souvent nécessaire d'adapter les politiques par rapport à un moment de la journée (ou a une date particulière). Ces politiques ne sont donc plus universelles, elles ne sont activées qu'en accord avec leur période de validité ;

Même s'il est aisé de récupérer de telles informations, il faut encore les transmettre en dehors du trafic, ou bien encore les utiliser pour coordonner les différents hôtes du réseau.

3.2.4 Les architectures

Les différentes architectures trouvées dans la littérature sont liées à l'étendue du champ d'action des politiques du réseau (sur un domaine unique, inter domaine, ...), mais aussi sur base des organismes de standardisation et de l'industrie qui soutiennent une architecture.

3.2.4.1 Le modèle three-tiers

L'architecture *three-tiers* est composée d'éléments dont la portée est en relation avec une politique de réseau limitée à un domaine administratif unique. C'est à dire à une portion du réseau qui est administrée et gérée en utilisant un jeu commun de définitions de politiques [raj1999].

Le modèle (figure 3-2 a) est donc composé de :

- *Policy Enforcement Point* (PEP) : le PEP est un composant en contact avec les paquets de données et il est responsable de l'application et de l'exécution des actions en rapport avec la politique. Le PEP est un composant opérationnel capable d'actions telles que le filtrage, le marquage de paquets, l'application de limites de bande passante, le *shaping*, la gestion des ressources, etc.
- *Policy Decision Point* (PDP) : le PDP est le composant responsable de la détermination des actions à entreprendre et des paquets ciblés par les actions. Le PDP interprète les règles de politique applicables à un ou plusieurs PEP sur base des informations contenues dans les paquets de données ou de signalisation et sur l'état actuel du réseau. Un PEP peut aussi demander à un PDP de prendre des décisions sur base d'occurrences d'un événement spécifique (comme l'arrivée de nouvelles demandes de réservation de ressources) ;
- Le *repository* est l'endroit où sont stockés les politiques définies pour le domaine. Il peut être localisé en un site physique unique ou peut-être répliqué sur plusieurs entités. Il peut être sous forme de base de données, d'un fichier plat, d'un serveur administratif ou d'un serveur d'annuaire.

Les fonctions de gestion de politiques sont responsables de la coordination entre les données introduites par un administrateur et les informations provenant d'autres politiques (interne ou externe), mais aussi de la traduction de ces informations en termes réseaux compréhensibles [wp-pna].

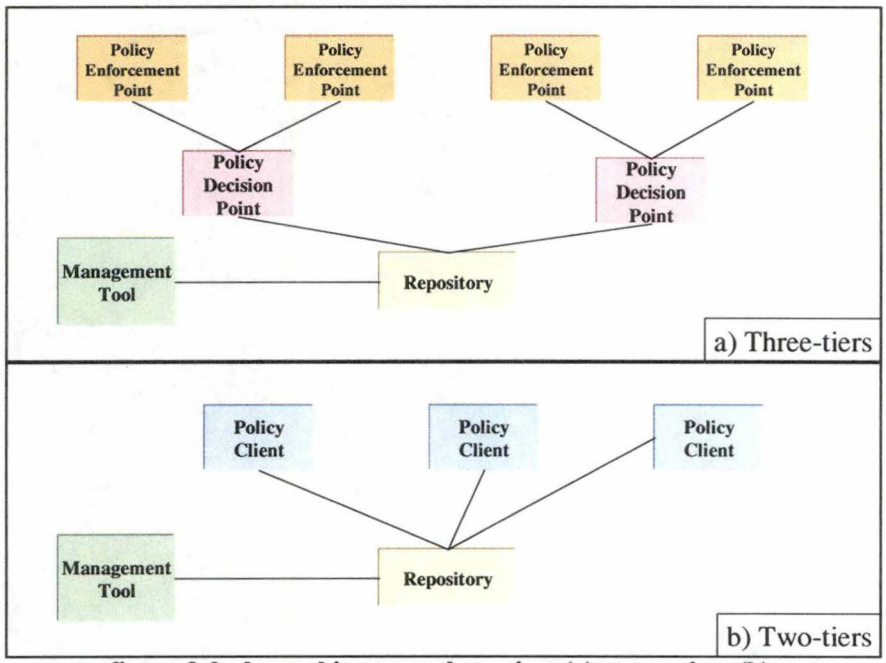

figure 3-2 : les architectures three-tiers (a) et two-tiers (b)

Les politiques sont stockées dans le *repository* au moyen d'un outil de gestion qui peut aussi fournir des fonctions de contrôle et de validation des politiques (bien formées, consistantes mutuellement, ...)

3.2.4.2 Le modèle two-tiers

Le modèle *two-tiers* est une simplification du modèle *three-tiers*. Le modèle *two-tiers* vient du fait que certains composants du réseau peuvent être assez puissants et flexibles pour prendre leurs propres décisions de politique en plus de les mettre en application. Le PEP et le PDP sont combinés en une seule entité [raj1999] (figure 3-2 b).

Il est possible, dans un réseau, d'avoir une combinaison des modèles *two-tiers* et *three-tiers*. Certains éléments du réseau ayant alors besoin de ressources supplémentaires font appel à un PDP intermédiaire pour établir leurs décisions politiques.

3.2.4.3 Le modèle de multidomaine

Dans un réseau multidomaine, il est irréaliste de concevoir un seul point de contrôle administratif pour tout le réseau. Une solution alternative est d'émuler le modèle de routage BGP interdomaine en se reposant sur les communications bilatérales, entre deux domaines indépendants, pour échanger les informations sur la politique. Ce mécanisme bilatéral permettant la mise en œuvre des QoS entre différents domaines est appelé *Bandwidth Broker* (BB) [raj1999].

Chaque domaine est placé sous le contrôle d'un BB. Les domaines adjacents négocient dans le but de déterminer la nature et l'étendue du trafic qui veut passer au travers de leurs limites communes. Par conséquent, chaque domaine décrit son niveau de service demandé à son BB voisin. Celui-ci fournit une décision d'admission basée sur la disponibilité de ses ressources, sur les arrangements financiers bilatéraux et sur un jeu de politiques administratives. Cette décision est appliquée en surveillant les flux entrants dans chaque domaine.

On considère, par exemple, un *Service Level Agreement* (SLA) entre deux domaines voisins qui spécifie un service "premium" garantissant la distribution des paquets d'un trafic donné en réservant les ressources nécessaires à ce trafic. Pour supporter un tel service "premium", les BB de ces domaines arrivent à un *Traffic Control Agreement* (TCA) en regardant les réservations de ressources pour ce trafic spécifique.

Un SLA est un élément du niveau réseau de la hiérarchie conceptuelle (figure 3-1) tandis que le TCA est un élément du niveau des nœuds. Dans le contexte d'interdomaine, les routeurs frontières qui sont à la limite entre deux domaines jouent un rôle dans la mise en application du SLA entre les deux domaines.

Un BB est conceptuellement une entité de gestion de politiques localisée dans un domaine permettant les politiques. Il négocie les SLA avec les domaines voisins et assure la conformité entre le domaine géré et les SLA contractées. Un BB a donc trois tâches distinctes :

- Négociation des SLA avec les BB des domaines voisins;
- Traduction des SLA en un ou plusieurs TCA pour les appareils en bordure du domaine;
- Distribution des TCA aux routeurs bordant le domaine administré en se servant d'un des protocoles existants.

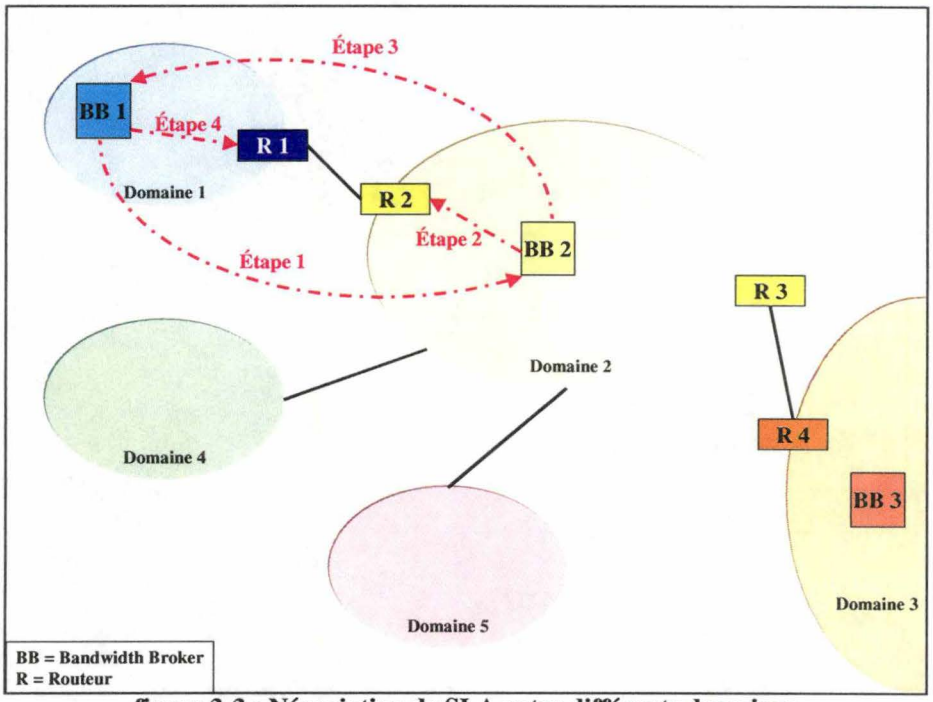

figure 3-3 : Négociation de SLA entre différents domaines

Afin d'illustrer ce principe, considérons, à titre d'exemple, la figure 3-3 : le domaine 1 représente un intranet, le domaine 2 un ISP local et le domaine 3 un ISP avec un *backbone* plus important. Si on suppose que les besoins du domaine 1 vers le domaine 3 sont satisfaits par un trafic *premium* de 64 Kb/s, alors les opérations suivantes sont exécutées :

- Le BB1 apprend intérieurement qu'une SLA de 64 Kb/s est demandée. Ceci peut être généré aussi bien par les demandes de réservations que par la détection d'un flux de paquets de 64 Kb/s par le routeur R1 ;
- Le BB1 demande le SLA au BB2 (étape 1) ;
- Le BB2 exécute les contrôles d'admission en se basant sur les ressources disponibles et sur les accords commerciaux permettant de telles requêtes ;
- Si la requête est admise, le BB2 envoie un TCA, dérivé du SLA demandé, à R2 qui est son routeur frontière (étape 2) et répond positivement au BB1 (étape 3) ;
- Ce TCA modélise le trafic devant être transféré du domaine 1 via R2 ;
- Un TCA similaire est envoyé par BB1 à son routeur frontière (étape 4) lui donnant les instructions pour permettre à un tel trafic de sortir du domaine 1 en direction du domaine 2.

Si les ressources ne sont pas disponibles, ou si la requête n'est pas supportée par les accords commerciaux ou les politiques de domaines, le BB2 peut décider de rejeter la requête provenant de BB1.

Il en va de même en ce qui concerne les relations entre le BB2 et le BB3 : BB2 peut demander plus de ressources au BB3 afin d'inclure la nouvelle demande de trafic au SLA existant entre BB2 et BB3. Dans ce cas, la réponse de BB2 à BB1 est conditionnée par la réponse que le BB2 recevrait du BB3 et ainsi de suite s'il y avait d'autres domaines à traverser.

Les routeurs frontières administrés par les BB exécutent différentes tâches afin d'implémenter les services différentiés. Ils peuvent classer les paquets en fonction d'un filtre multi-attributs et marquer avec un code DiffServ tous les paquets sortants qui y correspondent. Pour s'assurer que les classes de services ne soient surpeuplées, ils ont besoin de façonner le trafic sur base des profils donnés de trafic.

Dans l'exemple ci-dessus [raj1999], les routeurs frontières contrôlent la conformité du domaine avec le TCA en mesurant les paquets sortants et en comparant leur flux avec le TCA (le flux peut être modélisé, par exemple par un seau percé). Les paquets en excès sont soit écartés soit marqués comme des paquets de basse priorité dans le but de respecter le SLA.

Similairement, le trafic entrant qui ne correspond pas aux accords est soit écarté soit re-marqué par le routeur d'entrée dans le but de protéger le réseau d'une inondation causée par le trafic de haute priorité provenant des domaines voisins.

Les paramètres pour le marquage, le façonnage (*shaping*), le comptage (*metering*), la mise à l'écart et le re-marquage sont des éléments du niveau configuration (appareil) du modèle hiérarchique conceptuel (figure 3-1).

Le rôle des BB est seulement de négocier entre les domaines et non pas de fournir un service intra domaine. Par exemple, le domaine 1 peut implémenter IntServ, tandis que les domaines 2 et 3 peuvent implémenter DiffServ et pas IntServ. A partir de là, les routeurs frontières agissent comme des points de traduction entre IntServ et DiffServ (R1 est le traducteur et doit donc avoir des capacités IntServ et DiffServ).

Cependant, ce concept de *Bandwidth Brokerage* apporte son lot de questions intéressantes telles que:

- Comment un BB calcule-t-il le montant des ressources nécessaires pour un certain type de service étant donné ses propres paramètres liés à sa topologie? Si un domaine accepte trop de paquets de haute priorité, le niveau de service de cette classe risque de se dégrader. De même, si tous les liens RSVP sont alloués, alors les nouvelles connexions risquent de se faire rejeter (IntServ). D'un autre côté, si la portion de trafic de haute priorité entrant dans le réseau est continuellement restreinte alors le SLA sera aussi rejeté et les profits ne seront pas maximisés;
- Que se passe-t-il avec l'agrégation des SLA ? et de leurs décompositions en TCA ? ces deux actions sont-elles nécessaires dans les BB afin de transmettre le trafic contractuel? Dans notre exemple, le petit ISP local octroie trois SLA de service "premium" à trois organisations auxquelles il vend ses services (domaine 1, 4 et 5). L'ISP local est susceptible de transmettre une portion du trafic à un ISP global qui a un accès au *backbone* Internet (domaine 3). En conséquence, l'ISP local doit demander à l'ISP global d'ajouter (d'agréger) ce trafic dans le SLA qui permet à l'ISP local d'injecter et de transmettre le trafic futur des organisations. Sans un tel accord de transmission l'ISP local ne peut garantir ses SLA, car le trafic vers le domaine 3 risque d'être écarté ou re-marqué par les entrées du réseau de l'ISP Global.

  Le calcul de telles agrégations est actuellement une question ouverte à la recherche. Il faut remarquer que ce calcul est affecté de manière significative par cette portion de trafic qui doit être transmis au domaine 3 et est opposé au trafic destinés aux hôtes de l'ISP local du domaine 2.
- Une autre question concerne les flux multicast. Dans ce cas, les ressources demandées par un flux sont difficiles à estimer à cause de la duplication des paquets aux branches d'un arbre multicast. Opposé à un trafic *unicast*, un flot de trafic multicast envoyé entre des domaines voisins peut devenir un arbre énorme qui impose alors une charge élevée sur le réseau ;
- Plusieurs autres questions doivent être comprises et standardisées avant de voir un large déploiement de réseaux basés sur des politiques englobant le trafic *unicast* et multicast, et couvrant les scénarios inter et intra domaine. Les protocoles actuels pour les communications entre les *Bandwidth Broker* ne sont pas encore spécifiés, mais COPS, SNMP, LDAP, DIAMETER et RADIUS sont les protocoles mis à notre disposition.

Le *Bandwidth Broker* permet à différents domaines de négocier les politiques. Un ISP, par exemple, peut négocier dynamiquement différents SLA et ainsi que la bande passante qui y est attachée avec un client donné. Ou bien un fournisseur de service peut facturer différemment la bande passante en fonction de la demande. Pour faire cela, le gestionnaire de politiques devra avoir la capacité, en utilisant les protocoles standards de base, de communiquer avec les serveurs de politiques distants dans le but de négocier les politiques, et aussi avec les points locaux chargés de la mise en œuvre des politiques pour déterminer l'état du réseau [wp-pna]. Il semblerait que les *Bandwidth Broker* ne sont pas encore utilisés et que seuls des prototypes existent.

3.2.4.4 Le modèle de l'IETF

L'*IETF SNMP Configuration Working Group* (gestion de la configuration avec SNMP) [bor] recherche comment l'*Internet Standard Management* (SNMP) peut-être utilisé pour la gestion des configurations basée sur une politique.

La RFC 2571 [RFC2571] définit qu'un système de gestion SNMP contient plusieurs agents, au moins une station de gestion (ou gestionnaire) et un protocole de gestion pour transmettre les informations entre les différentes

entités de SNMP. Ajouter des capacités permettant des politiques à cette architecture existante semble assez manifeste. Un couple de modules MIB doit donc être ajouté à l'agent : le module MIB `Policy Management` et un module spécifique au domaine (par exemple : *DiffServ Policy MIB* [ID-dspmib2001]) (figure 3-4).

Les relations 1 et 2 de la figure 3-4 sont nouvelles pour l'architecture SNMP existante, elles représentent les nouvelles interactions de gestion basée sur une politique. La relation 3 représente la gestion traditionnelle de la configuration au niveau des appareils avec SNMP. Les relations 4 et 5 sont expliquées plus loin dans ce document.

D'un point de vue de la configuration via une politique, la station de gestion exécute les tâches suivantes :

- Distribuer les politiques à tous les appareils gérés (les agents) ;
- Surveiller l'utilisation et l'état d'exécution des politiques sur les appareils au travers du réseau.

Comme les politiques sont exécutées sur les appareils gérés, il est donc facile d'inspecter les caractéristiques des objets et il est possible d'exécuter les opérations décrites dans les politiques.

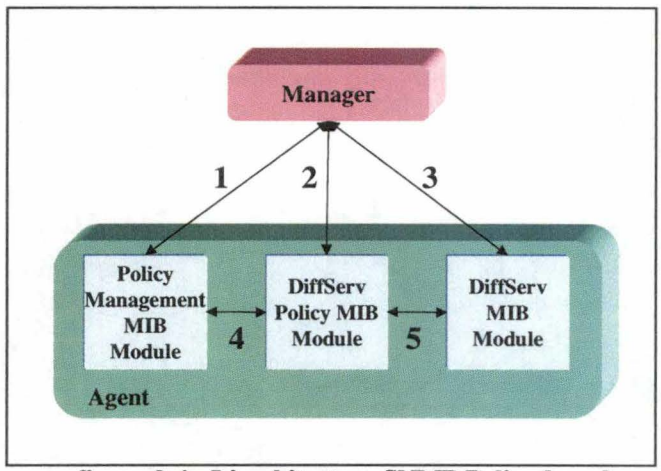

figure 3-4 : L'architecture SNMP Policy-based

Policy Management MIB Module

Le module MIB `Policy Management` contient [bor]:

- L'information sur les filtres à appliquer dans le but de sélectionner les éléments sur lesquels la politique sera mise en œuvre ;
- L'information sur les rôles existants et leurs associations avec les instances spécifiques ;
- L'information sur les horaires de la politique (quand l'appliquer et combien de temps ?) ;
- L'information sur la façon d'appliquer les paramètres des mécanismes spécifiques de la politique au niveau local (ces paramètres sont repris dans les modules MIB spécifiques au mécanisme).

Ce module MIB contient des objets qui sont compatibles avec la version 2 de SMI [bor] et l'information qui est contenue est organisée en table de la manière suivante :

- La table `pmPolicyTable` décrit une paire filtre/action qui sont décrits selon le langage d'expression de politiques. L'attribut `calendar` est un pointeur vers une entrée de la table `schedTable` (horaire) de la MIB `Scheduling`. L'attribut `description` contient une explication compréhensible et lisible de cette politique ;
- Les tables `pmRoleESTable` et `pmRoleSETable` permettent une visualisation rapide des éléments par rapport aux rôles et vice versa ;
- La table `pmCapabilitiesTable` contient une description des capacités inhérentes du système.

DiffServ Policy MIB Module

Le SNMP *Configuration Working Group* a été chargé d'élaborer un module MIB exemple qui convertit les mécanismes et les méthodes indépendantes des appareils en un niveau plus proche des instances spécifiques et des appareils[bor]. Le domaine qui a été choisi concerne les services différenciés (*Differentiated Services*) qui fournissent la capacité de manipuler séparément différentes classes de trafic. Les classes sont créées en employant la valeur DSCP (*Differentiated Services Code Point*) d'un paquet IP. Sur un routeur de type *Differentiated Services,* les classes de trafic peuvent recevoir un traitement différent appelé *Per-Hop-Behaviour* (PHB).

Actuellement deux PHB sont développés par le DiffServ Working Group :

- *Expedited Forwarding* (EF): une classe recevra la bande passante qui atteint ou excède un taux configuré;
- *Assured Forwarding* (AF): chaque classe est supposée avoir une allocation de bande passante minimum.

Cependant, les PHB ne spécifient pas les mécanismes pour atteindre les performances spécifiées. Différentes méthodes de gestion des files d'attentes (*queues*), de classification, différentes procédures de mesure et de marquage peuvent être utilisées pour implémenter le principe de PHB voulu. Pour modéliser ces structures, le module MIB DiffServ contient des tables de classification, de marquage, de mesure, d'utilisation de queues ainsi que des tables d'actions.

Le module MIB `DiffServ` agit comme un patron pour le module MIB spécifique au domaine (figure 3-5): les objets de ce module décrivent une configuration possible d'un sous-système DiffServ à un niveau conceptuel plus élevé que le module MIB `DiffServ`, en fournissant les valeurs par défaut qui satisfont différents PHB.

Le module MIB `DiffServ Policy` est conçu pour interopérer avec la MIB `Policy Management` et le module MIB `DiffServ` dans une architecture intégrée comprenant la gestion au niveau du réseau dans son sens large (*Policy-based*) et la gestion du réseau au niveau des appareils. Ce module est utilisé intentionnellement au dessus du module MIB `DiffServ` (qui agit au niveau des appareils) pour créer une interface vers le module MIB *Policy Management* (qui agit au niveau du réseau dans son ensemble).

Le module MIB `DiffServ Policy` agit comme une interface entre les définitions de politiques de haut niveau du réseau en général (qui affecte la configuration des sous-système DiffServ) et les informations spécifiques aux instances décrites dans le module MIB `DiffServ`.

Exemple

Considérons la politique suivante : "un utilisateur lambda reçoit un service "premium" pour le trafic *web* tant que ce trafic ne dépasse pas une limite précise. Le trafic excédentaire étant traité en *best-effort*". Pour réaliser cela, différentes étapes (figure 3-5) doivent être exécutées pour configurer les appareils pour répondre à cette politique :

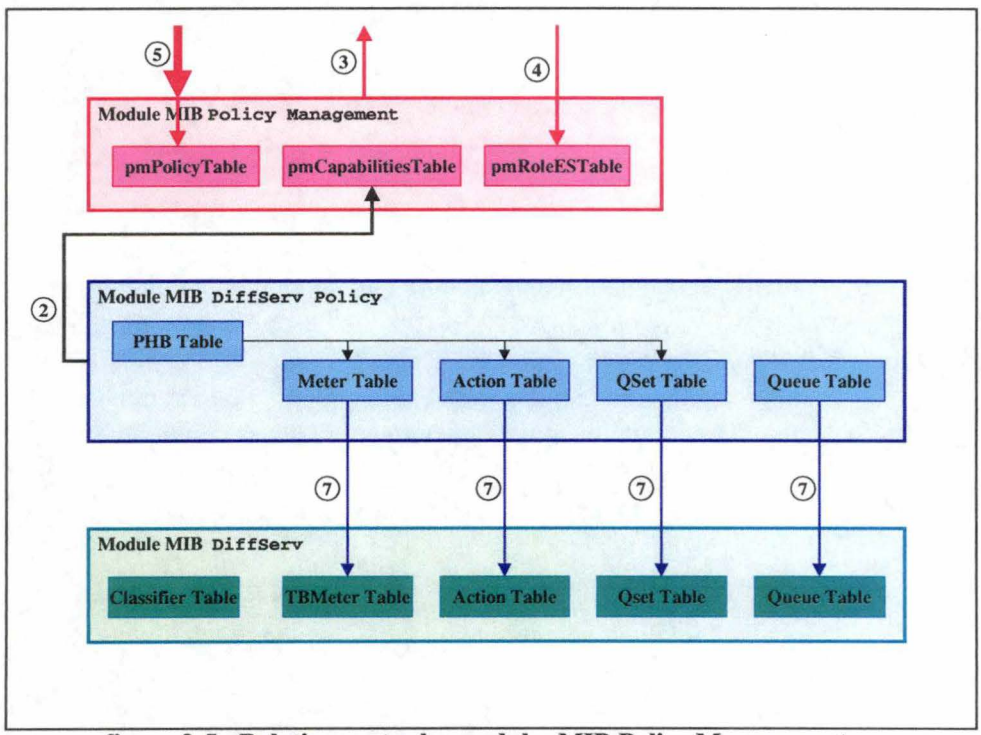

figure 3-5 : Relations entre les modules MIB Policy Management, DiffServ Policy et DiffServ [bor]

1) Du côté du gestionnaire, les utilisateurs (humains ou logiciels) définissent les règles politiques (filtres et actions) ainsi que toute autre information nécessaire pour l'application des politiques sur les appareils (rôles et horaires). Les utilisateurs définissent aussi les informations spécifiques aux appareils et à leurs mécanismes. Toutes ces informations seront envoyées au module MIB `Policy Management` et à un module MIB spécifique au domaine. Pour notre exemple, le filtre sera constitué de l'adresse IP de

destination, du port source (du niveau transport) ainsi que d'un taux qui doit être inférieur à, par exemple, 100 Kb/s; et l'action sera d'assurer le service "premium" (à l'aide *d'Expedited Forwarding*).

2) Le module `DiffServ Policy` qui est capable d'exécuter les mécanismes dépendants de la configuration se fait inscrire dans la table `pmCapabilitiesTable` du module MIB `Policy Management`. Il y indiquera ses capacités à exécuter *Expedited Forwarding*.
3) L'agent informe le gestionnaire sur ses capacités. Dans ce but, le message SNMP *Inform* serait utilisé comme un message de confirmation (ACK). L'agent sait donc qu'un gestionnaire est au courant de ses capacités. Ceci est surtout utile dans le cas de gestionnaires multiples.
4) Ensuite, le gestionnaire remplit la table `pmRoleESTable` du module MIB `Policy Management` et associe les rôles avec les éléments d'un appareil. Dans notre exemple, une association de rôle pourrait être que le service "premium" va être implémenté au moyen de DiffServ Expedited Forwarding avec des paramètres spécifiques et que l'utilisateur se connectera au réseau via l'interface A.
5) Le gestionnaire envoie les politiques aux appareils gérés. Cela implique, dans la plupart des cas, de remplir les valeurs de la table, `pmPolicyTable` du module MIB `Policy Management`. Les expressions indépendantes du domaine et des mécanismes sont chargées dans le module MIB `Policy Management` et l'information spécifique est chargée dans le module MIB particulier au domaine.
6) Les appareils évaluent les politiques (filtres et actions) dans le but de déterminer quels sont les éléments sur lesquels appliquer les politiques et quand les appliquer.
7) Le module MIB `DiffServ Policy` place les valeurs appropriées directement ou via le module MIB `DiffServ`. Dans notre exemple, le module `DiffServ Policy` remplira les tables du module `DiffServ` pour implémenter *Expedited Forwarding* pour l'utilisateur Lambda sur les interfaces entrantes spécifiées par les rôles de la table `pmRoleESTable`.
8) Finalement, il est nécessaire de surveiller l'utilisation des politiques et leurs états, et de vérifier les résultats des politiques afin de peaufiner les politiques si nécessaire.

3.2.4.5 Les architectures homogènes et hétérogènes

Indépendamment de l'infrastructure du réseau, la mise en place de politiques peut pendre des mois pour les implémenter et les raffiner [jud2001]. En plus de cette situation, qui peut être intimidante, la plupart des constructeurs réseaux qui offrent des solutions de gestion de réseaux via des politiques fournissent aussi des logiciels et des systèmes ayant comme fonction d'aider à mettre en place ses mêmes solutions.

Choisir un produit de gestion via des politiques pose un autre problème : la plupart des solutions proposées par les constructeurs sont optimisées pour leur ligne de produit. Néanmoins il faut signaler que la plupart des constructeurs fournissent les mêmes services de base tel que la prioritisation du trafic. D'un autre côté il existe aussi des vendeurs de logiciels qui sont indépendants des constructeurs.

La sélection d'une solution de gestion du réseau via des politiques, dépend de multiples variables. La principale considération est de savoir si une solution particulière pourra être mise en place sur votre infrastructure réseau. Si le réseau est homogène la meilleure solution est peut être de choisir la solution proposée par votre vendeur. Si cette solution fournit moins de fonctionnalités que celles nécessaires ou que le réseau est composé de matériels hétérogènes, il est sans doute plus intéressant d'opter pour une solution indépendante des constructeurs.

3.3 Syntaxe et sémantique

3.3.1 Situation actuelle

Les travaux initiaux de standardisation de l'IETF étaient focalisés [ste1999] sur la syntaxe et la sémantique des expressions de politiques. Par essence, une politique est une déclaration de la forme : `IF <condition> THEN <action>`. En pratique [jud2001], la gestion du réseau via des politiques se définit elle-même comme une série de règles qui définissent les relations entre une série de conditions, les réponses potentielles à ces conditions et les appareils qui implémenteront les réponses.

L'IETF *Policy FrameWork Working Group* a développé un modèle de données pour stocker les expressions de politiques. Le modèle de données définit les relations et les associations entre les conditions et les actions appartenant à une politique donnée et fournit, pour les regroupements, des jeux de politiques. Ce modèle de données ne tient pas compte d'un modèle particulier de stockage de données. Cependant, le groupe de travail a établi une transposition entre son modèle de données générique et un schéma d'annuaire pour un *repository*.

Ce schéma d'annuaire, désigné sous le nom de *Core Policy Schema* (CPS) [ste1999] formalise comment les relations et les associations décrites dans le modèle de données peuvent être effectivement représentées dans un annuaire. Le résultat du développement du *Core Policy Schema* est une compréhension commune sur comment

organiser le stockage des politiques dans un annuaire. Cependant, plus de travail est demandé pour atteindre l'interopérabilité entre les équipements réseaux et les politiques.

En utilisant le CPS comme il est, une application peut chercher les conditions et les actions associées à une politique, mais l'application peut ne pas avoir assez de perspicacité pour analyser le contenu des conditions et des actions. Le CPS, dans son état actuel, permet aux vendeurs d'organiser les politiques dans un format commun, mais les conditions et les actions elles-mêmes restent des données propriétaires stockées dans une structure standard.

Le CPS fournit une structure pour organiser le stockage des expressions de politiques, mais ne fournit pas encore de définitions pour les opérandes et les opérateurs. Ces expressions, ou règles, prennent différentes formes possibles. Elles peuvent gérer les caractéristiques des tables de routage, l'espace d'allocation des adresses IP, les critères de balancement de charge, les *Service Level Agreement* (SLA) et l'accès aux services comme les qualités de service (QoS) et les réseaux privés virtuels (VPN).

Une règle de politique peut limiter le nombre d'appels *Voice over IP* concurrentiels dans une zone particulière de l'organisation. Une autre règle peut bloquer l'accès à une partie du réseau quand un certain nombre de fichiers (d'un répertoire confidentiel pour l'entreprise) ont été copiés pendant un laps de temps spécifié. Les opérateurs dans une politique font référence aux utilisateurs, aux groupes, aux interfaces, aux queues, aux délais et aux statistiques d'un type de paquet particulier (taux, longueur de queue, champ, ...).

Dans chacun de ces exemples, les expressions de politique doivent être capable de faire référence aux opérandes de la politique de façon consistante et non ambiguë. Ces opérandes constituent une interface abrégée du logiciel ou des éléments matériels avec lesquels la politique souhaite interagir. Donc, par exemple, une règle de politique devrait être définie pour garantir un délai maximum, même si la topologie du réseau change en ajustant la bande passante allouée au service quand une augmentation d'utilisation est détectée. Cette règle doit surveiller la longueur d'une queue particulière et modifier le *scheduler* pour augmenter le taux de sortie de la queue quand la longueur de celle-ci augmente d'une façon particulière. Dans cet exemple, la règle se base sur un opérande pour représenter la longueur de la queue, et sur un autre opérande qui peut modifier l'allocation de la bande passante pour cette queue.

Une autre spécification des opérandes est qu'ils doivent être non ambigus. Jusqu'à aujourd'hui, l'industrie avait défini beaucoup d'interfaces décrivant les éléments comme les utilisateurs ou les filtres de paquets. Cependant, il est commun de trouver des définitions multiples pour un élément (la définition d'un utilisateur dans un système en est un bon exemple). Avec l'arrivée de la gestion des réseaux basée sur une politique, il y a une demande croissante d'unification des modèles afin de représenter les composants communs des appareils réseaux. Il y a un nombre de groupes de standardisation qui recherchent comment appliquer ce modèle commun aux politiques.

Le précurseur en ce domaine est le *Distributed Management Task Force* (DMTF) qui travaille sur la modélisation d'une variété de concepts différents incluant les utilisateurs, les appareils physiques, les systèmes logiques et les services [ste1999]. Ces modèles sont référencés collectivement sous l'appellation *Common Information Model* (CIM). En parallèle au développement du CIM, un groupe s'est formé pour construire un modèle pour les équipements réseaux. Le résultat de l'activité de ce groupe, appelé *Directory-Enabled Networks* (DEN), est intégré dans le schéma CIM 2.2 du DMTF.

La version suivante de CIM (2.3) contient le schéma pour les utilisateurs, les groupes, les réseaux virtuels (VLAN), le protocoles de routage, les qualités de service (QoS), la sécurité, etc. Initialement, la motivation primaire pour le développement d'un modèle commun d'information était dû à la puissance des divers produits d'annuaires du marché. Comme CIM est applicable pour beaucoup d'autres applications, l'utilisation des propriétés uniques de CIM permet de l'appliquer à la configuration des appareils et aux politiques. Les expressions des politiques peuvent donc être construites pour manipuler les caractéristiques de haut niveau et de bas niveau des appareils du réseau, des services et des protocoles. On peut aussi facilement en faire la transposition avec une variété de protocoles spécifiques tels que SNMP, COPS, RADIUS, DIAMETER.

Avec des opérandes décrivant les queues, les interfaces, les compteurs et les moteurs de traitement des protocoles, on peut développer les moyens nécessaires à la manipulation des appareils de manière consistante. Ces opérandes représentent une interface de programmation dans les appareils réseaux. Si on veut examiner la longueur actuelle d'une queue spécifique ou ajuster le transfert d'un type de trafic particulier, on a alors besoin d'interagir avec les composants appropriés d'un appareil. De même, l'aspect grammatical d'une politique nous permet de spécifier les circonstances pour changer ces interfaces et en quoi elles pourraient être changées.

3.3.2 Deux approches

L'industrie n'ayant pas réussi à trouver une solution à l'interopérabilité des appareils sous le niveau des protocoles, deux approches ont alors été entreprises [ste1999]. Prenant une approche *top-down*, l'industrie a perçu la nécessité de combler le fossé séparant les expressions de politiques de haut niveau et les paramètres bas niveau de la configuration spécifiques aux appareils. Dans bon nombre de groupes de travail de l'IETF, les grands vendeurs d'équipement réseau proposent d'utiliser la *Policy Information Base* (PIB) et le protocole COPS pour programmer les politiques dans les appareils.

Initialement introduit dans le IETF *Resource Allocation Protocol* (RAP) *Working Group*, la PIB est un peu plus qu'une variante de la MIB. La PIB représente une structure de données transférées plutôt qu'une structure de données stockées. Cependant la PIB (comme la MIB) fait peu pour améliorer l'interopérabilité. La PIB offre trop d'opportunités pour diverger des standards. Quand l'industrie développe de nouveaux appareils avec de nouvelles capacités, elle développe de nouvelles structures PIB pour les supporter. Dans un court laps de temps, la prolifération des structures PIB uniques menace plus l'interopérabilité que ce qu'ont pu faire les extensions MIB propriétaires. De plus, la PIB représente une structure utilisée pour changer les propriétés mais ne représente pas une structure qui peut être utilisée pour retrouver les propriétés.

La seconde approche est du type *bottom-up*. Au lieu de définir des interfaces de configuration standards comme la PIB, l'industrie doit se concentrer sur le développement de standards des composants de base d'une politique. L'utilisation des techniques de modélisation CIM (du DMTF) unifie la sémantique des politiques et interdit l'explosion du nombre de définitions et de services.

3.3.3 La Policy Information Base (PIB)

La politique à appliquer à une interface peut dépendre de plusieurs facteurs comme les caractéristiques immuables de l'interface (*Ethernet* ou *Frame Relay*), de l'état de l'interface (*half* ou *full duplex*) ou de la configuration des utilisateurs (direction, marketing, ...). Au lieu de spécifier les politiques explicitement sur chaque interface de tous les appareils du réseau, on spécifie les politiques en termes de fonctionnalités d'interface.

Pour décrire ces fonctionnalités on utilise le concept de rôles. Un rôle est une simple chaîne de caractère (une phrase) qui est associée à une interface. Une interface donnée peut avoir plusieurs rôles simultanément. Les *Provisionning Classes* (PRC) sont des regroupements de rôles ordonnés de manière lexicographique. Les instances d'une PRC ne sont appliquées que si et seulement si le jeu de rôles correspond au jeu de rôles de l'interface.

Un rôle fournit donc un moyen d'unir les politiques aux interfaces sans avoir besoin d'identifier explicitement d'une manière consistante, les interfaces de tous les appareils réseaux. De plus, si la même politique est utilisée pour plusieurs interfaces, on a besoin de ne l'envoyer qu'une fois.

A titre d'exemple, on suppose que l'on a un appareil composé de 3 interfaces ayant les rôles suivants :

```
InterFace1 (IF1) : production
InterFace2 (IF2) : production
InterFace3 (IF3) : direction
```

Et que l'on a un PDP avec deux politiques :

`P1` : les paquets venant du département production (rôle `production`) ont un DSCP égal à 5

`P2` : les paquets venant de la direction (rôle `direction`) ont un DSCP égal à 6

Pour obtenir une politique, le PEP rapporte au PDP les rôles attribués à ses interfaces. En réponse le PDP envoie la politique `P1` associée au rôle `production` et envoie la politique `P2` associée au rôle `direction`.

Maintenant, supposons qu'une personne du service production soit promue à la direction et que l'administrateur du système ajoute le rôle direction à `InterFace2`. Le PEP indique alors au PDP qu'il a trois rôles (`production`, `direction`, et `production+direction`). En réponse le PDP envoie une troisième politique associée à ce rôle mixte. La façon dont le PDP détermine la politique pour ce nouveau rôle est sous la responsabilité du PDP. Il peut le faire de manière algorithmique ou sur base de règles :

- Une règle peut spécifier que la politique direction est préférentielle à la politique production ;
- On peut mettre en place une nouvelle politique :

  `P3` : les paquets venant de la production+direction (rôle `direction` et rôle `production`) ont un DSCP égal à 7.

Le PDP est donc requis pour déterminer qu'elle est la politique à appliquer à ce nouveau rôle combiné et pour envoyer une troisième politique au PEP pour ce rôle production+direction (même si cette politique est la même qu'une déjà chargée). Le PEP n'est, quant à lui, pas requis pour construire une politique pour un nouveau rôle à partir des politiques existantes.

L'industrie et les organismes de standardisation sont actuellement en train de créer de nouvelles PIB. A titre d'exemple, deux PIB sont décrites sommairement ci-dessous. Il s'agit des PIB *DiffServ QoS* et *IP Traffic Engineering*.

3.3.3.1 Aperçu de la PIB DiffServ QoS

La PIB DiffServ QoS est constituée d'un module contenant les PRC de base pour mettre en place la politique *Differentiated Service*, les queues, les classificateurs, les compteurs, ... et contient aussi les capacités des PRC qui permettent au PEP de spécifier ses propres caractéristiques au PDP [ID-dsqospib2001]. Le module contient 2 groupes :

- ***QoS Capabilities Group*** **:**
  Ce groupe est constitué de PRC pour indiquer au PDP les types d'interfaces supportées sur le PEP en terme de capacité QoS et de PRC que le PDP peut installer dans le but de configurer ces interfaces (queues, classificateurs, compteurs, *scheduler*, ...) pour arriver à la politique voulue.
  Ce groupe décrit les capacités en terme de types d'interfaces et prend les configurations en terme de types d'interfaces et de combinaisons de rôles. Il ne traite pas directement avec les interfaces individuelles de l'appareil.
- ***QoS Policy Group :***
  Ce groupe contient la configuration des éléments fonctionnels qui comprennent la politique QoS qui s'applique à une interface et les paramètres spécifiques qui décrivent ses éléments. Ce groupe contient les classificateurs, les compteurs, les actions, les *droppers* ("compte-gouttes"), les queues et les *schedulers*. Ce groupe contient aussi la PRC qui associe les chemins d'accès aux éléments avec la combinaison de rôle.

3.3.3.2 Aperçu de la PIB IP Traffic Engineering

Le renforcement dynamique d'une politique d'ingénierie du trafic IP [ID-iptepib2001] est basée sur l'activation des protocoles de routage inter et intra domaine qui ont les capacités de prendre en compte les informations en relation avec l'ingénierie du trafic pour le calcul et la sélection de routes en accord (le plus parfait possible) avec les spécifications QoS qui ont été contractuellement définies entre les clients et les fournisseurs.

Ce type d'information est composé par défaut des valeurs *metric* qui vont être une sorte de reflet de la politique d'ingénierie du trafic IP, avec comme résultat le renforcement de cette politique de manière à ce que les clients et les fournisseurs puissent vérifier à tout moment que le service IP est approvisionné avec les niveaux de service et de qualité appropriés.

Donc, la PIB *IP Traffic Engineering* vise principalement :

- Le stockage et la maintenance des informations de configuration qui seront utilisées par les routeurs pour calculer et sélectionner les routes qui correspondront à une collection de spécifications QoS, comme le délai de transit maximum ou la variation maximum de délai entre les paquets ;
- Le stockage et la maintenance des informations relatives aux routes qui ont été installées dans les *Forwarding Information Bases* (FIB) des routeurs permettant aux fournisseurs de services d'avoir une connaissance permanente du réseau.

Dans ce but, la PIB *IP Traffic Engineering* est actuellement organisée selon les PRC suivantes :

- La table ***OSPF Traffic Engineering Metrics*** : cette classe représente la collection des données d'approvisionnement, qui refléteront une politique de routage intra domaine spécifique, en terme d'assignation de valeurs de *Traffic Engineering Metrics ;*
- La table ***BGP Traffic Engineering*** : cette classe vise à décrire les routes BGP qui ont été annoncées par les routeurs avec leurs QoS associées.
- La table ***IP Traffic engineering Route*** : cette classe décrit les informations relatives aux routes qui ont été mises en place par les routeurs dans leur FIB, en accord avec la politique de *Traffic Engineering* qu'ils doivent renforcer.

3.4 Protocoles utilisés

Quelque soit l'architecture utilisée, on a besoin de protocoles standards pour échanger l'information entre les différentes parties. On doit donc utiliser un jeu de protocoles standards pour améliorer l'interopérabilité de produits commerciaux et pour assurer, au problème de politiques, un environnement ouvert.

3.4.1 Pushing Configuration (SNMP & CLI)

Depuis le début, l'industrie des réseaux de données a appliqué le modèle *push* (figure 3-7) pour résoudre le problème de la gestion des réseaux. Dans ce modèle les appareils jouent un rôle relativement passif. En utilisant les mécanismes conventionnels (SNMP et CLI), les applications de gestion de réseau implémentent les politiques du réseau en poussant (pushing) les informations de configuration dans les appareils de manière individuelle [ste1999].

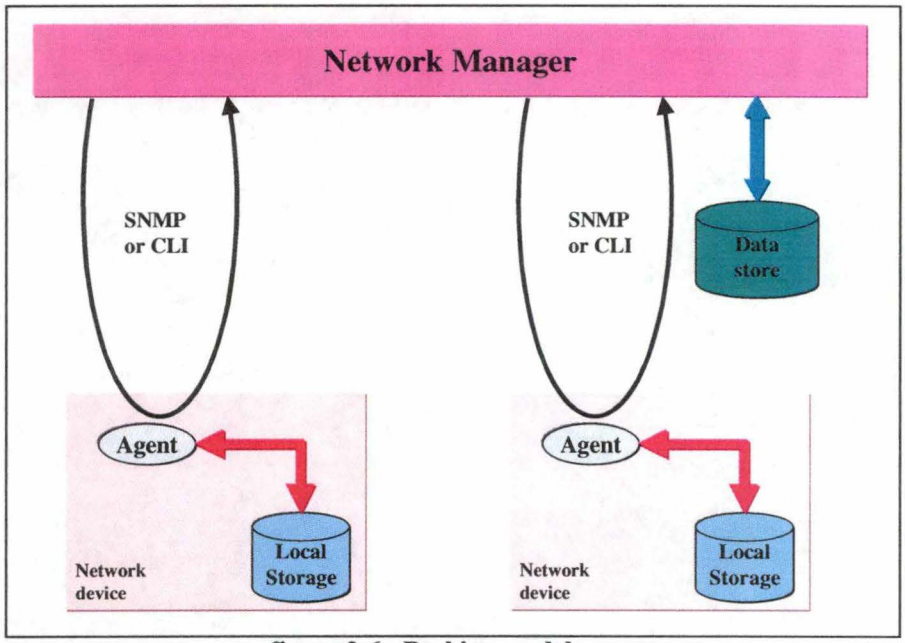

figure 3-6 : Pushing model

Bien que les applications de gestion du réseau puissent utiliser les *trap* SNMP (qui amorcent les communications à partir des appareils en réponse à des événements codés), le modèle demande au gestionnaire (humain ou système) de débuter toutes les activités de configuration. Dans le cadre du modèle *push*, quand un appareil démarre, celui-ci commence à opérer sur base des paramètres par défaut pré-programmés ou alors il se met en mode inactif. Déclenché par un administrateur réseau, une application de gestion du réseau pousse les nouveaux paramètres dans l'appareil. Enregistré sur une mémoire non volatile dans chaque appareil, les nouveaux paramètres téléchargés deviennent les valeurs par défaut (cette opération nécessite peut-être un redémarrage de l'appareil).

Le modèle *push* ne convient pas vraiment car les administrateurs réseaux doivent alors s'impliquer fortement. Les changements dans la configuration des appareils nécessitent que les changements au niveau des données se fassent via le réseau. En pratique, s'assurer que la reconfiguration de tous les appareils s'est bien déroulée est difficile. Faire les changements appareil par appareil est souvent source d'erreurs, même si l'inventaire de ces appareils est correct. On peut se retrouver facilement face à des inconsistances entre les données stockées et celles se trouvant localement (individuellement).

Le modèle *push* se base fortement sur SNMP et CLI, tend à nécessiter des systèmes de gestion propriétaires, et réduit souvent les capacités de gestion à un sous-système commun aux différents appareils. En réponse aux problèmes rencontrés par le modèle *push*, l'industrie tend à préconiser un transfert vers le modèle *pull*.

3.4.2 Pulling Configuration (COPS & LDAP)

3.4.2.1 Le protocole COPS

Le protocole *Common Open Policy Service* (COPS) [RFC2748] est un protocole simple de question réponse qui peut-être utilisé pour échanger les informations des politiques entre un serveur de politiques (*Policy Decision Point* – PDP) et ses clients (*Policy Enforcement Point* – PEP). Un routeur RSVP qui doit exécuter des contrôles

d'admission sur base de politiques d'utilisation de RSVP est un exemple de client politique. Il est nécessaire qu'un serveur de politiques soit présent dans chaque domaine administratif contrôlé.

Un des objectifs [RFC2748]de ce protocole de contrôle de politiques est de commencer avec une conception simple mais extensible dont les caractéristiques sont :

1) Le protocole emploie un modèle client/serveur où le PEP envoie les requêtes, les mises à jour et les suppressions au PDP et où le PDP renvoie les décisions au PEP ;
2) Le protocole utilise TCP comme protocole de transport pour l'échange fiable des messages entre les clients et le serveur, donc aucun mécanisme supplémentaire n'est nécessaire pour assurer la fiabilité de la communication entre les clients et le serveur ;
3) Le protocole est extensible car il est conçu pour tenir compte des objets auto-identifiants et qu'il peut supporter des informations provenant de divers clients spécifiques sans modification du protocole COPS lui-même. Le protocole a été créé pour l'administration générale, la configuration et l'amélioration des politiques ;
4) Le protocole COPS fournit des messages au niveau sécurité pour l'authentification, la reprise des protections et l'intégrité des messages. Le protocole COPS peut aussi réutiliser des protocoles comme IPSEC pour authentifier et sécuriser le canal entre le PEP et le PDP ;
5) Le protocole est déclaratif dans deux aspects principaux :
   - Les requêtes du client PEP sont "installées" ou retenues par le PDP distant jusqu'à ce qu'elles soient effacées par le PEP ;
   - Le serveur peut répondre aux nouvelles requêtes différemment à cause de l'état question/réponse qui a été retenu précédemment et à qui elles sont liées.
6) De plus, le protocole COPS est indicatif car il permet au serveur de pousser les informations de configuration vers le client et ensuite permet à ce même serveur de retirer un tel état du client quand il n'est plus applicable.

Il est acquis que chaque client participant à la politique est fonctionnellement cohérent avec un PEP. Le PEP peut communiquer avec un serveur de politique (appelé PDP distant) pour obtenir les décisions politiques ou les directives. Le PEP est responsable de l'établissement d'une connexion TCP persistante vers un PDP. Le PEP utilise cette connexion pour envoyer les requêtes et recevoir les décisions venant du PDP distant. Les communications entre le PEP et le PDP se font principalement sous la forme d'échanges de questions/réponses déclaratives. Cependant, le PDP peut occasionnellement envoyer des décisions non sollicitées au PEP pour forcer certains changements.

Le PEP a aussi la capacité d'avertir le PDP du bon traitement des décisions provenant du PDP (dans un but de comptabilisation et de surveillance). Le PEP est responsable de la notification au PDP quand un état de demande a changé sur le PEP. Et enfin, le PEP est responsable de la suppression de tous les états qui ne sont plus applicables à cause d'événements survenus sur le client où à cause de décisions provenant du serveur.

Quand le PEP envoie une demande de configuration, il s'attend à recevoir en continu des unités de données de configuration via des messages de décisions applicables comme demande de configuration. Quand une de ces unités est bien installée sur le PEP, celui-ci doit envoyer un message (rapport) au PDP pour en confirmer l'installation. Le serveur peut alors mettre à jour ou effacer l'information de configuration via un nouveau message de décision. Quand le PDP envoie une décision pour supprimer les données de configuration du PEP, celui-ci effacera la configuration spécifiée et enverra un message de confirmation au PDP.

Le protocole de politiques est conçu pour communiquer des objets auto-identifiants (*self-identifying*) qui contiennent les données nécessaires pour identifier l'état de la requête, pour établir le contexte d'une requête, pour transmettre les décisions politiques, pour rapporter les erreurs, pour fournir les messages d'intégrité et pour transférer les informations spécifiques du client.

Pour distinguer les différentes sortes de client, le type du client est identifié dans chaque message. Différents types de client peuvent avoir leurs données spécifiques et peuvent donc avoir besoin de décisions particulières. Il est prévu que chaque nouveau type de client soit accompagné d'une spécification sur son utilisation et sur son interaction avec le protocole de politique.

Le contexte de chaque requête correspond au type d'événement qui l'a déclenché. Le protocole COPS identifie trois types d'événements : l'arrivée d'un message entrant, l'allocation de ressources locales et le transfert d'un message sortant. Ces événements nécessitent que différentes décisions soient prises. Le contenu d'un message COPS question/décision dépend du contexte. Un quatrième type de demande est utile pour les clients qui veulent recevoir les informations de configuration du PDP.

Le PEP peut aussi avoir les capacités de décider de sa propre politique locale via son *Local Policy Decision Point* (LPDP), cependant, le PDP reste la seule autorité de décision. Cela implique que le LPDP est assujetti au PDP distant et doit donc lui transférer ses informations de décisions. Le PDP doit alors donner l'accès aux informations qui seront utiles pour l'établissement de la décision finale. Pour faciliter cette fonctionnalité, le PEP doit envoyer ses décisions locales au PDP distant. Et le PEP doit se conformer aux décisions du PDP et ce, de manière absolue.

La tolérance de panne est requise pour ce protocole. Cela est particulièrement dû au fait qu'il est associé à la gestion de la sécurité et des services des appareils réseaux distribués. La tolérance de panne peut être effectuée en ayant le PEP et le PDP qui vérifient constamment, l'un l'autre, leur connexion via des messages fictifs (*Keep-Alive*). Quand une panne est détectée, le PEP doit essayer de se reconnecter au PDP distant ou à un PDP alternatif (*backup*). Une fois que la connexion a été rétablie, il est demandé au PEP de notifier au PDP toutes les suppressions d'état ou les nouveaux événements survenus après la coupure de connexion. De plus, le PDP distant peut demander tous les états internes au PEP pour se synchroniser. Après la coupure et avant que la connexion ne soit complètement fonctionnelle, la coupure de service peut être minimisée si les zones tampon du PEP ont auparavant communiqué les décisions et s'il les utilise pendant un certain temps.

3.4.2.2 Le fonctionnement

Le modèle *pull* (figure 3-7) modifie le problème de stockage en déplaçant les données de configuration se trouvant individuellement dans les appareils vers un *repository* central [ste1999]. Le *repository* de choix actuellement est un annuaire accessible via LDAP. Le stockage des configurations centralisé demande alors que les appareils réseaux jouent un rôle plus actif dans les activités de gestion.

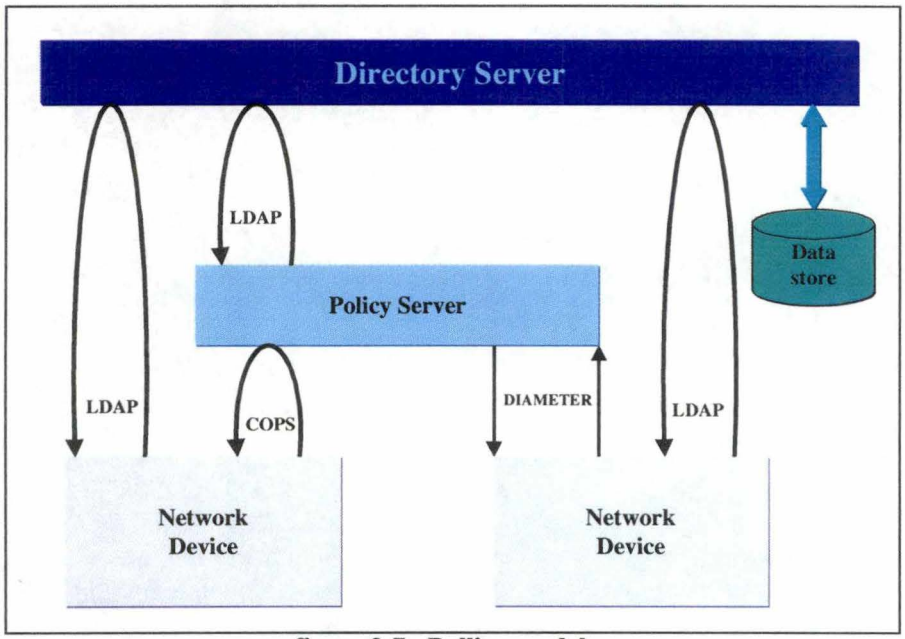

figure 3-7 : Pulling model

Avec les données de configuration centralisées, les appareils individuels deviennent responsables pour retrouver et intercepter leurs propres informations de configuration. Les appareils maintiennent une copie locale des informations de configuration dans une mémoire non volatile, mais la responsabilité est passée d'une entité externe à l'appareil lui-même. Dans le modèle *pull*, les administrateurs réseaux encodent les informations de configuration communes et celles spécifiques aux appareils dans un annuaire, permettant ainsi à n'importe quel appareil réseau de retrouver les informations de configuration.

Bien que la configuration spécifique aux instances nécessite quelques duplications de structure, les administrateurs n'encodent qu'une fois les informations de configuration partagées. Donc, si on a qu'une copie de la majorité des données de configuration, on limite alors significativement les erreurs de configuration.

En plus des informations de configuration, l'industrie des réseaux de données espère stocker les expressions des politiques opérationnelles dans un annuaire. Ils espèrent ainsi que les utilisateurs expriment les politiques à un niveau sémantique au delà de ce que les appareils réseaux peuvent interpréter directement. Pour compenser cela, la plupart des vendeurs pensent développer des serveurs de politiques pour aider leurs appareils à implémenter les politiques.

Les serveurs de politiques retrouveront les expressions de politiques dans un annuaire, interpréteront ces expressions et programmeront les appareils réseaux en utilisant des protocoles comme COPS, RADIUS et DIAMETER. En plus, les vendeurs continueraient à utiliser une MIB SNMP propriétaire ou leur CLI pour programmer les appareils pour ce qui pourrait être une longue période de transition.

3.4.3 Modèle mixte

Le modèle mixte est composé des modèles *pushing* et *Pulling*.(figure 3-8) Dans le cadre d'une gestion centralisée il est utile de pouvoir gérer le réseau dans son intégralité, c'est à dire en faisant abstraction des capacités des différents appareils du réseau. Il est fort probable qu'à un moment ou à un autre le réseau soit composé d'éléments ne supportant que le protocole SNMP (et les MIB) et d'éléments supportant la gestion via des politiques. Il est donc intéressant que les gestionnaire puissent compter sur les deux modèles existants pour gérer leur réseau.

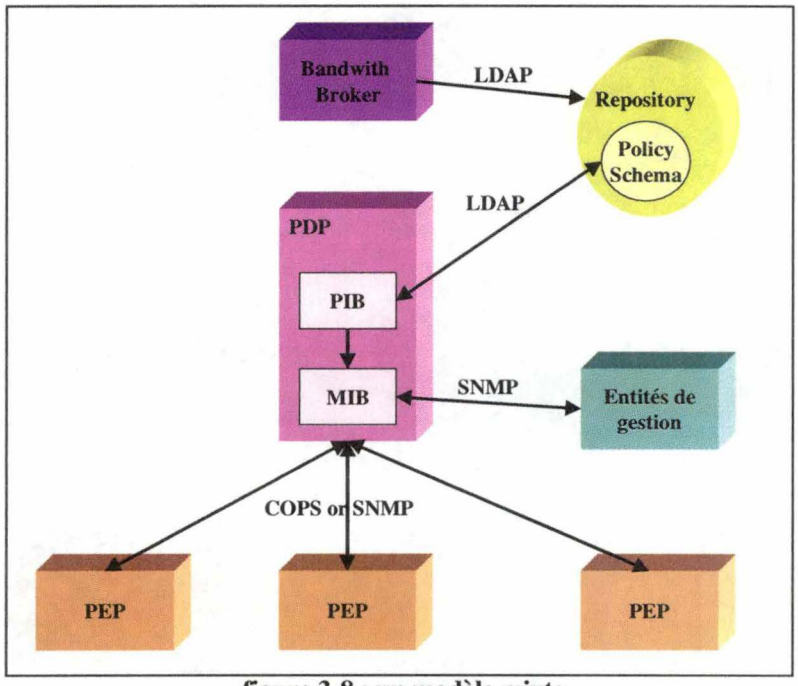

figure 3-8 : un modèle mixte

Dans une autre mesure, la mixité des deux modèles peut être envisagée dans le cas où les politiques de gestion se basent sur des informations provenant d'appareils ne supportant pas la gestion via des politiques. Il suffit d'imaginer que la condition d'une politique fasse référence à une valeur contenue dans une MIB pour modifier la configuration d'un élément du réseau qui supporte la gestion via des politiques.

Le PDP peut aussi jouer un rôle de traducteur entre les deux modèles de configuration. Une même politique pourrait donc être mise en œuvre selon l'un ou l'autre modèle en fonction des capacités des différents éléments du réseau ciblé par cette politique (figure 3-8). Le PDP utiliserait soit le protocole COPS soit le protocole SNMP pour modifier la configuration des éléments ciblés.

3.5 Problèmes de stockage

De façon croissante, les vendeurs se dirigent vers les annuaires et LDAP comme moyens pour le stockage et la distribution des informations réseaux. Les annuaires et LDAP sont décrits comme des applications puissantes du réseau, mais ils représentent seulement une solution à une partie du problème à résoudre par la gestion des réseaux basée sur les politiques [ste1999]. Les annuaires ont des avantages qui font qu'ils sont de bons outils pour le stockage de certains types de données, mais ils ont aussi des inconvénients qui font qu'ils sont un choix faible pour stocker les données générales du réseau.

Les annuaires fournissent un moyen de stockage d'information où la disponibilité est la première concernée. Les carnets d'adresses, les listes téléphoniques et la liste des serveurs d'impression sont des exemples d'informations qui ont un réel besoin de disponibilité. Les annuaires fournissent un moyen pratique pour stocker les données qui peuvent être disposées dans une hiérarchie. Le diagramme d'une organisation est un bon exemple d'informations faites pour être stockées dans un annuaire. Généralement, les informations, qui sont lues plus souvent qu'elles ne sont mises à jour, conviennent parfaitement pour être stockées dans un annuaire.

Les annuaires ne sont pas équivalents aux bases de données relationnelles ou aux autres systèmes de stockage de données. Néanmoins, l'IETF a commencé un travail pour définir des extensions à LDAP qui supporteraient les transactions de bases. Si une application a besoin de faire des changements sur différents nœuds dans l'annuaire, il n'y a pas de mécanismes prévus pour s'assurer que les mises à jour dépendantes sont atomiques. En plus, les annuaires ne fournissent pas de mécanismes pour renforcer l'intégrité des données. Les applications doivent accepter la responsabilité de la maintenance des liens entre les différents sous arbres de l'annuaire. Une application ayant un mauvais comportement peut facilement violer l'intégrité d'un modèle de données particulier, et ce car l'annuaire lui-même n'a pas conscience des relations entre les éléments de données qu'il contient. Une autre absence est remarquée : si une application a besoin d'être avertie de changements d'entrées dans l'annuaire quand ils surviennent, les développeurs doivent construire des mécanismes externes pour fournir des avis de changements aux applications.

Pris ensemble, les avantages et les inconvénients des annuaires limitent leurs utilisations dans le développement d'une infrastructure dans le cadre des politiques. Les données relativement statiques trouveront facilement une place dans un annuaire, mais les données qui ont des relations assez complexes, qui sont volatiles ou périssables auront besoin de mécanismes de stockage et de distribution que ne peut leur fournir un annuaire. La gestion des réseaux basée sur les politiques devenant alors plus complexe qu'une simple liste de règles stockées dans un *repository* directement accessible.

3.6 Conclusion

Définir une approche de gestion du réseau via des politiques pour gérer des QoS peut être complexe et exigeant en temps. Théoriquement, les règles devraient être développée pour toutes les ressources appartenant au réseau géré. Mais en pratique, seules certaines liaisons et/ou applications stratégiques sont traitées avec une haute priorité, tandis que le reste est maintenu à un niveau de type *best-effort*.

Les personnes chargées de mettre en place une solution de gestion du réseau sur base de politiques doivent comprendre comment la solution choisie et les appareils à gérer peuvent utiliser les protocoles mis à leur disposition. Cela est d'autant plus vrai dans un environnement hétérogène où il est difficile et plus complexe d'obtenir un niveau d'abstraction suffisant pour mettre en place une gestion cohérente du réseau via des politiques. En effet cela dépend fortement des capacités des différents appareils et l'on sait que les constructeurs n'évoluent ou ne développent pas tous en même temps les mêmes capacités ou fonctionnalités.

Les qualités de services permettent la différentiation du trafic (en flux individuels ou agrégés) et nécessitent la définition de politiques et des améliorations pour gérer les flux qui auront un service de qualité et ceux qui auront un service de base. Les réseaux existent généralement comme support des entreprises qui négocient des *Service Level Agreements* (SLA) avec leur fournisseur d'accès. Ces SLA définissent la disponibilité et les performances dans le but d'obtenir des garanties attribuées à un service de qualité. Les SLA relèvent essentiellement les exigences des politiques pour le fournisseur et le client. L'un comme l'autre seront punis si leur service ou trafic ne convient pas.

Pour implémenter les politiques sur le réseau, un cadre de travail et une architecture ont été définis pour identifier les fonctionnalités requises, comment les responsabilités sont distribuées, ainsi que les protocoles nécessaires aux entités du réseau pour mettre en œuvre les politiques. Les détails de bas niveau, comme les méthodes d'encodage des politiques, le stockage et la récupération des informations ont aussi été étudiées. Ainsi il est possible de traduire les SLA en des représentations de politiques (LDAP ou PIB) auxquelles les éléments du réseau et les stations de gestion accèdent.

Cependant, il faut aussi tenir compte de la mise en place des politiques dans le cadre de QoS de bout en bout, que ce soit en utilisant des protocoles spécifiques au QoS, RSVP ou DiffServ. Il existe pour cela différents types de politiques pouvant être implémentés, différents types de services (SLA) pouvant être supportés et différentes conceptions réseau pour y arriver.

D'autres éléments sont aussi à prendre en compte, comme la sécurité sans laquelle le concept de politique ne serait pas fiable, mais cela fait plutôt partie de l'accès aux données qui sont centralisées. Cette centralisation pose aussi un problème car les répertoires et les réseaux, en général, grandissent rapidement.

PARTIE 2 - Etude pratique

Introduction

Cette deuxième partie, comme son nom l'indique, traitera de la mise en pratique, sur un cas concret (le réseau de l'Office National des Pensions (ONP)) de différents concepts vu dans la partie théorique. Mais avant, il est nécessaire, pour mieux cerner la problématique et pour définir ce que l'on va devoir gérer, de décrire l'environnement sur lequel une gestion devra être appliquée.

C'est dans le chapitre 4 que cette description aura lieu. Elle sera constituée d'informations sur l'organisme qu'est l'ONP, sur le réseau de données qui y est installé, sur les systèmes informatiques utilisés ainsi que sur les applications mises à la disposition des utilisateurs pour effectuer la mission de l'ONP.

Le chapitre 5 est, quant à lui, constitué de trois subdivisions : la première concernera la mise en place de *Simple Network Management Protocol* (SNMP) et de *Remote Network Monitoring* (RMON) à l'ONP sur les différents éléments du réseau qui nous intéressent ; la seconde tentera une analyse du réseau du trafic à partir des moyens mis à notre disposition ; tandis que la troisième sera composée d'une réflexion pratique (appliquée à l'ONP) de la mise en place du concept de *Policy-based Network Management*.

Chapitre 4 : L'environnement

4.1 Présentation générale de l'entreprise

Les données reprises ci-dessous, concernant l'environnement de travail pour la partie pratique du mémoire, sont générales et permettent d'avoir un aperçu de la problématique liée à la gestion du réseau de l'Office National des Pensions en tenant compte de ses spécificités.

4.1.1 L'Office National des Pensions

L'Office National de Pensions (ONP) est un organisme parastatal chargé de plusieurs missions :

- Contrôler auprès de l'ex-CGER le bon fonctionnement de la collecte des données relatives à la carrière des travailleurs salariés ;
- Instruire et attribuer les pensions de retraite et de survie des travailleurs salariés, les allocations de chauffage, les rentes de vieillesse et de veuve ainsi que le revenu garanti aux personnes âgées ;
- Gérer les droits attribués et payer les prestations des régimes des travailleurs salariés et indépendants, le revenu garanti aux personnes âgées, les allocations à certains handicapés et le supplément préretraite agriculture.

Pour information, en 2000, l'ONP a dépensé 575,93 milliards de francs pour les pensions de retraite et de survie, 9,57 milliards de francs pour le revenu garanti aux personnes âgées et 1,93 milliards de francs pour les allocations aux handicapés.

L'ONP emploie environs 2.389 membres répartis dans une administration centrale et dans quinze bureaux régionaux.

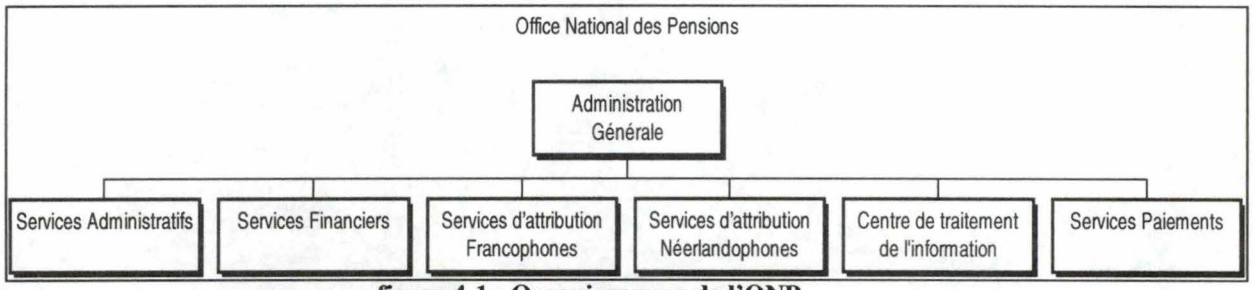

figure 4-1 : Organigramme de l'ONP

4.1.2 Les services de l'ONP

L'Office National des Pensions a, comme la plupart des entreprises, une structure interne lui permettant de remplir les tâches qui lui sont attribuées. Cette structure se retrouve dans l'organigramme (figure 4-1) dont les différents éléments sont décrits ci-dessous :
L'administration générale est chargée entre autre de la coordination au sein de l'ONP, des problèmes juridiques, ...

Les services administratifs comprennent le service du personnel et le service formation ; les immeubles, l'économat et l'imprimerie ; la gérance, le courrier et la téléphonie ; et le service traduction.

Les services financiers sont chargés principalement de gérer le "porte-monnaie" de la "famille" ONP. Ils sont composés des services suivants : budget, comptabilité, gestion des fonds, statistiques, actuariat et études techniques.

Les services d'attribution francophones sont chargés de l'attribution de la pension de retraite et de survie et du revenu garanti aux personnes âgées aux bénéficiaires qui habitent la partie francophone du pays (y compris ceux qui ont effectués des prestations à l'étranger).

Les services d'attribution néerlandophone sont chargés de l'attribution de la pension de retraite et de survie et du revenu garanti aux personnes âgées aux bénéficiaires qui habitent la partie néerlandophone du pays (y compris ceux qui ont effectués des prestations à l'étranger).

Le centre de traitement de l'information (CTI) est en quelque sorte un *service provider* pour tous les autres services de l'ONP.

La direction du CTI a pour mission principale de coordonner les différentes activités de l'équipe informatique. Celle-ci est composée de deux grands blocs : le bloc développement et le bloc support. Parallèlement on retrouvera deux autres branches : L'informatique décentralisée et le service "Etudes juridiques – Budget – Secrétariat Technique – Service de documentation Informatique" plus simplement appelé SDI.

Les groupes de développement sont chargés de l'implémentation des applications pour les services de paiement et d'attribution et ce sur le système Datapoint ou sur le système Siemens. Un des groupes est chargé plus particulièrement du développement d'applications client/serveur et des applications sur la plate-forme Escala de Bull.

Les services paiements sont chargés d'effectuer et de vérifier le paiement des pensions aux personnes résidant tant en Belgique qu'à l'étranger ainsi que de maintenir les données nécessaires au paiement.

4.2 Le réseau

4.2.1 Le site central

Le cœur du réseau du site central de l'ONP est constitué de quatre Smart Switch Router 8600 de Cabletron (SSR1 à 4 sur la figure 4-2). Ces SSR 8600 sont reliés entre eux par de la fibre optique (Gigabit Ethernet). Ces routeurs permettent la définition de différents réseaux virtuels (VLAN) :

- VLAN 2 : contenant une grande partie des PC et des imprimantes mais aussi des serveurs de fichiers et d'impression ;
- VLAN 3 : contenant l'autre grande partie des PC et des imprimantes, et d'autres serveurs de fichiers et d'impression ;
- VLAN 4 : contenant principalement des serveurs d'applications et d'infrastructures de Windows NT (Domain Controller, DHCP Server, WINS Server, SMS Server, Workflow Server, Mail Server, ...)
- VLAN 5 : comprend les systèmes UNIX (base de données signalétiques, base de données des dossiers scannés, ...) et les serveurs "imageries" utiles pour l'insertion de nouveaux documents scannés et pour la consultation des dossiers.
- VLAN 6 : permet la connexion vers les bureaux régionaux de l'ONP via le réseau BiLAN de Belgacom ;
- VLAN 7 : contenant les serveurs Siemens (données paiement) et Datapoint (données attribution).

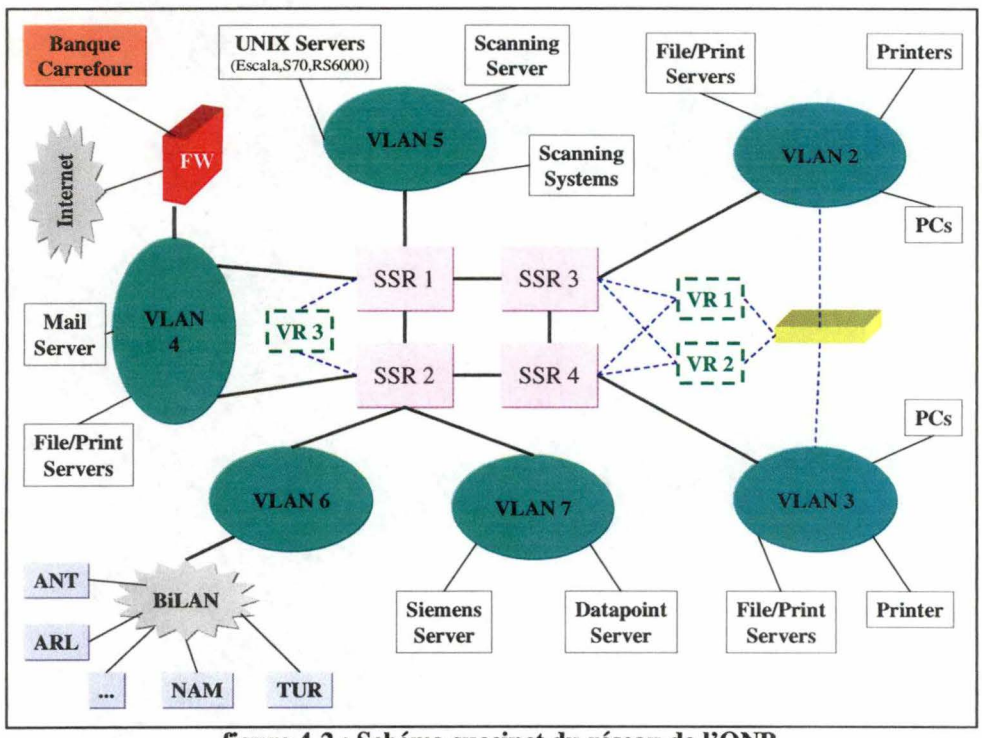

figure 4-2 : Schéma succinct du réseau de l'ONP

Afin de connecter les hôtes terminaux à ce cœur du réseau, une série de Smart Switch Router 2000 de Cabletron (figure 4-3) ont d'abord été mis en place, qui bien qu'étant plus petits que les SSR 8600 en ont les mêmes fonctionnalités. Ces SSR 2000 sont placés aux étages N et ce sont des fibres optiques (Gigabit Ethernet) qui relient les SSR8600 aux SSR2000.

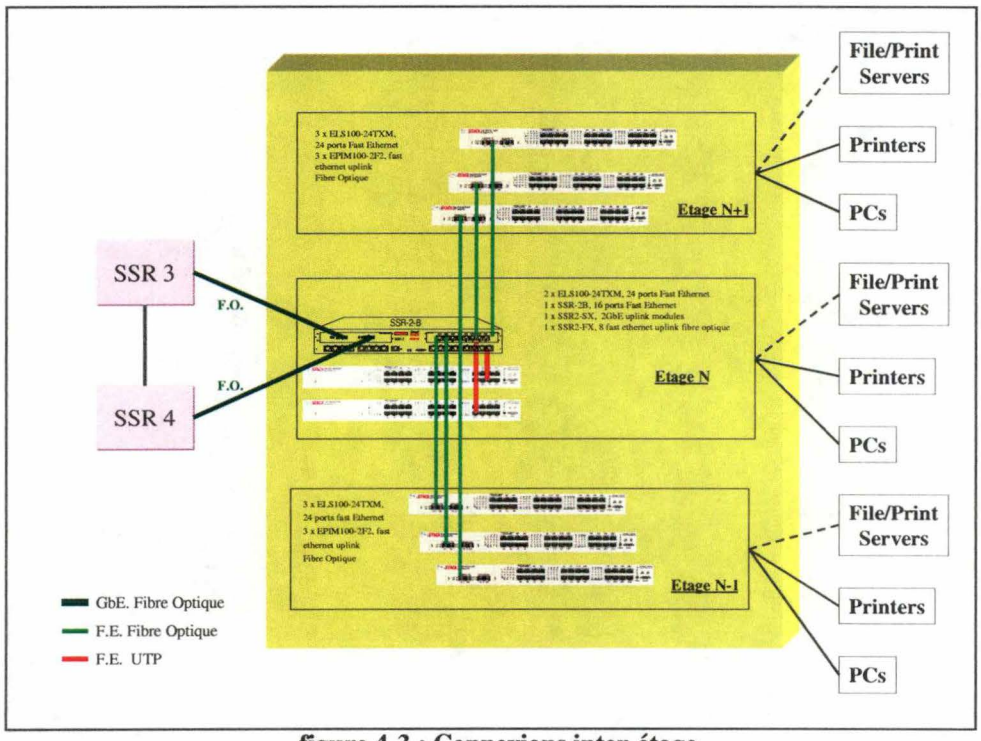

figure 4-3 : Connexions inter-étage

Ensuite, on retrouve une série de *switch Smart Stack* (Fast Ethernet) (figure 4-3)sur lesquels sont connectés les PCs, les imprimantes, les serveurs et les autres hôtes du réseau. Deux de ces switch Fast Ethernet, sont placés à l'étage N (un pour le VLAN 2 et un pour le VLAN3) et 2 ou 3 sont placés selon les besoins aux étages N+1 et N-1 qui desservent, l'un le VLAN 2, l'autre le VLAN 3. Ce système permet d'obtenir une répartition des hôtes sur les VLAN 2 et 3 et ce par trois étages.

Entre les étages N et N-1, et N et N+1, ce sont des fibres Optiques Fast Ethernet (100 Megabit) qui sont utilisées. Dans la plupart des cas les hôtes sont connectés aux *switch* Fast Ethernet via des câbles UTP Fast Ethernet 100 Mb en full Duplex.

La situation décrite ci-dessus est bien sûr une solution idéale, et il existe bien entendu des cas particuliers qui ont nécessité une connexion spécifique.

Deux des trois routeurs virtuels créés sont constitués des SSR 3 et 4, et sont définis comme étant les membres d'un cluster, chacun d'entre eux fonctionnant comme backup de l'autre. Si un Smart Switch Router ou l'une de ses connexions LAN ne fonctionne plus, l'autre Smart Switch Router reprend la main et re-route le trafic. Le troisième routeur virtuel permet d'accéder de manière redondante aux serveurs du VLAN 4 mais nécessite une intervention manuelle en cas de problème. En effet, ce routeur virtuel étant composé des SSR 1 et 2 et le VLAN 4 étant seulement présent dans les deux routeurs, cela signifie qu'en cas de défaillance d'un SSR, les connexions du VLAN 4 doivent être déplacées manuellement sur le routeur resté actif.

Remarques : les extensions Token Ring de RMON seront peut-être utiles pour l'Office National des Pensions, car il existe un petit réseau Token Ring qui relie entre eux les différents points de contrôle d'accès aux bâtiments et les différentes bornes de pointage. Ce réseau Token Ring est connecté au réseau Ethernet via une passerelle. Cette interconnexion sert principalement pour le calcul des pointages (heures supplémentaires, ...) effectué par le service du personnel et sert aussi pour le contrôle des accès par le service de la gérance.

4.2.2 Les sites régionaux

L'infrastructure informatique (figure 4-4) mise à la disposition des bureaux régionaux de l'ONP sert principalement à calculer les montants des pensions. Cette infrastructure est composée d'un serveur Datapoint contenant les données de la carrière des pensionnés, d'un serveur Windows NT contenant les formulaires utilisés pour les demandes d'informations complémentaires et aussi les données internes utiles au traitement des dossiers d'attribution de pensions (*file and print Server*), de plusieurs PC et imprimantes.

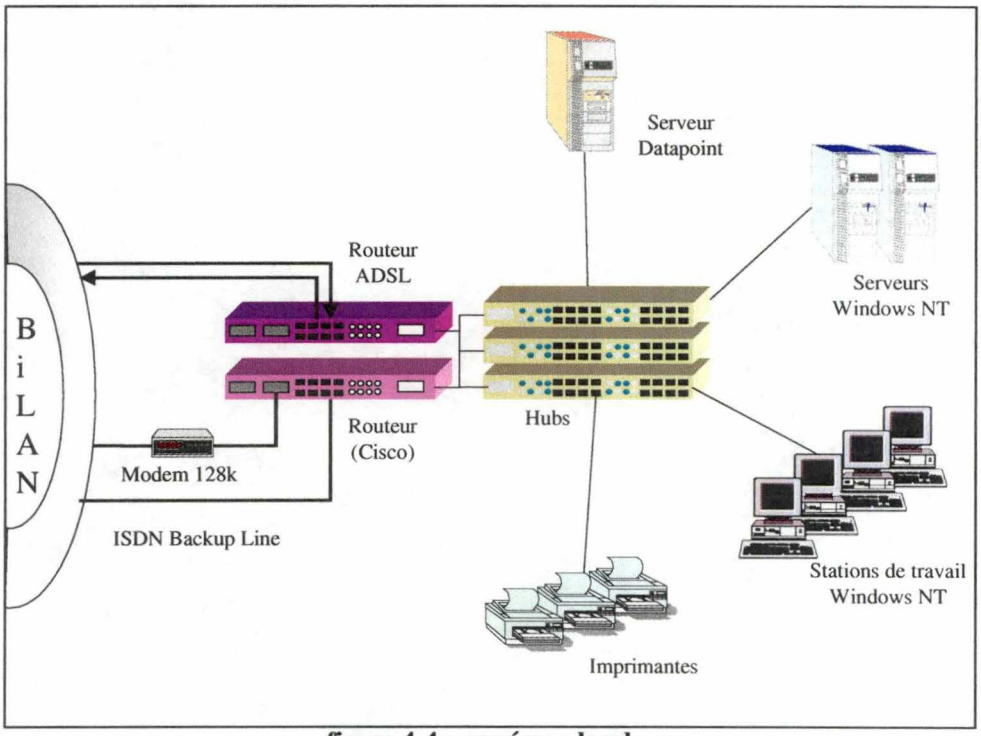

figure 4-4 : un réseau local

Les bureaux régionaux sont connectés au site central de l'ONP via le réseau BiLAN de Belgacom au moyen de :

1) Une ligne ADSL pour la consultation des dossiers attribution et paiement scannés ;
2) Une ligne PSTN (128k) pour les autres applications. Cette ligne est dédoublée par une ligne ISDN en cas de panne de la ligne PSTN.

Les éléments actifs du réseau local sont des routeurs Cisco et des *hub* 3Com.

4.2.3 Le WAN

Le réseau WAN (figure 4-5) permettant de relier le site central de l'Office National des Pensions situé à Bruxelles et ses treize bureaux régionaux répartis sur tout le territoire belge, est constitué de connexions vers le réseau BiLAN de Belgacom. Différentes modifications ont été apportées dernièrement.

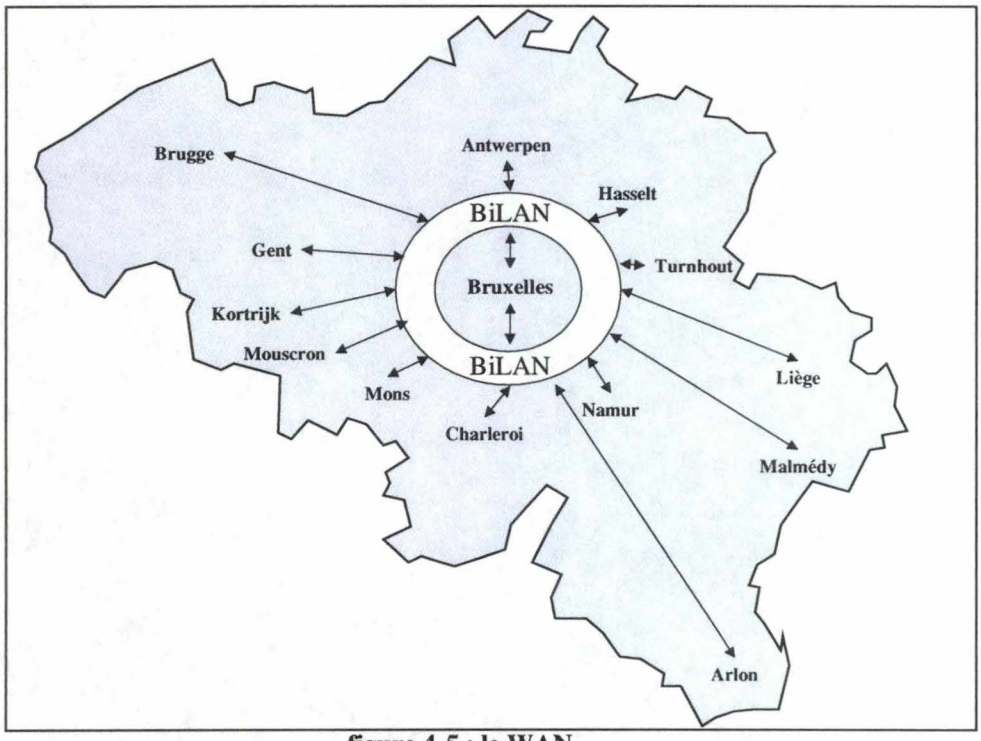

figure 4-5 : le WAN

Faute de fibre optique présente aux alentours du site bruxellois de l'ONP, la connexion entre ce site et le point d'entrée de BiLAN se fait via une antenne parabolique et un faisceau hertzien de 34 Mbit. Les bureaux régionaux sont connectés à BiLAN soit via une ligne PSTN de 128 K (doublée d'une ligne ISDN en cas de panne), soit via une connexion ADSL (256Kb/1Gb). Ce choix se fait sur base des ports TCP des paquets transitant entre le bureau régional et le site central; et seuls ceux correspondant à l'application Capucine transitent via ADSL.

La gestion de notre trafic transitant via BiLAN est totalement prise en charge par Belgacom : nous ne pouvons pas agir directement sur les composants du réseau. Il ne s'agit pas ici que de problèmes de configuration, mais même les notifications de pannes sur le réseau BiLAN (du moins dans sa partie nous intéresse) ne viennent pas jusqu'à nous.

4.3 Les systèmes

L'Office National des Pensions possède différents systèmes informatiques que l'on peut classer en deux catégories : la première comprenant les systèmes plus anciens est composée des systèmes Siemens BS2000 et Datapoint ; la seconde est composée des systèmes plus récemment acquis et est composée de plusieurs serveurs UNIX et de plusieurs serveurs Windows NT. Cette cohabitation est historique et provient du fait que théoriquement une migration doit être faite des anciens systèmes vers les nouveaux.

- **Unix**

Les systèmes Unix composés de cluster Escala de Bull, de S70 et de RS600 de IBM, contiennent les applications de type *backoffice*, les bases de données signalétiques, gèrent le stockage des dossiers scannés. Ces serveurs s'occupent aussi de la gestion des sauvegardes et de la gestion des impressions de masse.

- **Windows NT Server**

Les serveurs Windows NT servent de serveurs de fichiers et d'impressions pour les applications bureautiques et assimilées. Certains serveurs Windows NT, faisant partie de la chaîne de scanning, contiennent les bases de données reprenant les informations sur les dossiers scannés. D'autres serveurs NT servent de serveur de messagerie, de distribution de software, de serveur Intranet,...

- **Siemens**

De façon générale le mainframe Siemens contient les informations relatives au paiement des pensions. Les informations se trouvant sur ce mainframe seront normalement et prochainement transférées dans des bases de données se trouvant sur les serveurs Unix.

- **Datapoint**

Les serveurs Datapoint contiennent les informations relatives à l'attribution des pensions : le calcul du montant de la pension en fonction de la législation actuelle. Les serveurs Datapoint sont situés sur tous les sites de l'ONP. Les données stockées sur le système central permettent d'établir le dossier, de demander les données relatives à la carrière à l'ex-CGER, et de le transmettre au bureau régional concerné. Les serveurs situés dans les bureaux régionaux contiennent alors les informations complémentaires permettant de calculer le montant exact de la pension et d'en avertir les ayants droits. Il est également prévu de transférer les applications et les données contenues sur ces serveurs vers des serveurs Unix ou Windows NT.

4.4 Les applications

4.4.1 Description

a) **Terminal Emulation**
Il existe à l'ONP plusieurs produits permettant une émulation de terminal. Cette différenciation dépend des machines ciblées par cette émulation :

- **Siemens** : ce sont les données paiement qui sont consultées par les utilisateurs centraux et régionaux. L'administration de ce système se fait également par émulation de terminal (ou à partir de consoles du système) ;
- **Datapoint** : ce sont les données attribution (carrières, ...) qui sont consultées par les utilisateurs centraux et régionaux. L'administration de ce système se fait également par émulation de terminal (ou à partir de consoles du système) ;
- **Unix** : l'émulation de terminal à ce niveau est, dans sa totalité utilisée, par les administrateurs des systèmes Unix ;

b) **Thymon**
Cette application est utilisée pour accéder aux données signalétique relatives aux personnes mentionnées dans les dossiers de l'ONP. Cette application permet aussi de suivre l'état d'avancement d'un dossier et sa localisation.

c) **Capucine**
Ce sont les dossiers scannés (attribution et paiement) qui sont consultables à partir de cette application. Les utilisateurs centraux et régionaux qui traitent les dossiers de pensions doivent pouvoir accéder à ces données. Cette application est importante du point de vue du volume de données transférées

d) **Applications Internet**
Sous ce nom générique sont reprises les différentes applications WEB qui ont pour but d'obtenir des informations provenant d'Internet (en dehors de l'ONP) :

- Browsing : consultation de page Web via le protocole HTTP ou HTTPS ;
- File Transfert : récupération de fichiers (pilotes d'imprimantes, de carte réseau, etc.) ;
- L'envoi de messages électroniques.

e) **Applications bureautiques et assimilées**
Sous cette appellation, on retrouve les accès aux serveurs de fichiers, aux serveurs d'impressions et les applications se trouvant sur les serveurs de fichiers :

- Contrôle d'accès et de pointages ;
- Gestion du personnel ;
- Gestion d'inventaires ;
- Gestion des problèmes (helpdesk) ;
- Gestion des hypothèques ;
- ...

f) **Applications "backoffice"**
Cette dénomination ne fait pas allusion au *package* de Microsoft, mais reprend les applications servant à demander, à récupérer, à traiter et à stocker les informations de masse (provenant de la Banque Carrefour de la Sécurité Sociale, du Registre National, de l'ex-CGER, etc.) afin de les mettre à la disposition des applications "métiers" de l'ONP (accessible aux utilisateurs). On reprend aussi les systèmes de sauvegarde, les systèmes de distribution de software, ...

Il existe à l'ONP, à coté des applications principales, toute une série d'applications intra services qui n'ont que peu d'influence sur le réseau au vu de la masse de données échangées par les applications citées ci-dessous. Je ne tiens donc pas compte de ces applications.

4.4.2 Les applications et le réseau

a) Terminal Emulation (figure 4-6 a)

Le trafic provenant des émulations de terminal a comme source les hôtes des VLAN 2 et 3 (site central) et du VLAN 6 (les bureaux régionaux) et a comme destination les systèmes Siemens et Datapoint situés dans le VLAN7. Il faut aussi remarquer que l'émulation de terminal se fait :

- Pour Siemens, du site central ou des bureaux régionaux vers le site central ;
- Pour Datapoint, en plus des connexions site central ou bureaux régionaux vers le site central, il existe aussi des connexions au sein du bureau régional vers le serveur Datapoint local.

b) Thymon (figure 4-6 b)

Le trafic généré par l'application Thymon peut-être schématisé par un flux provenant des VLAN 2 et 3 (site central) et du VLAN 6 (les bureaux régionaux) à destination des serveurs Unix se trouvant dans le VLAN 5.

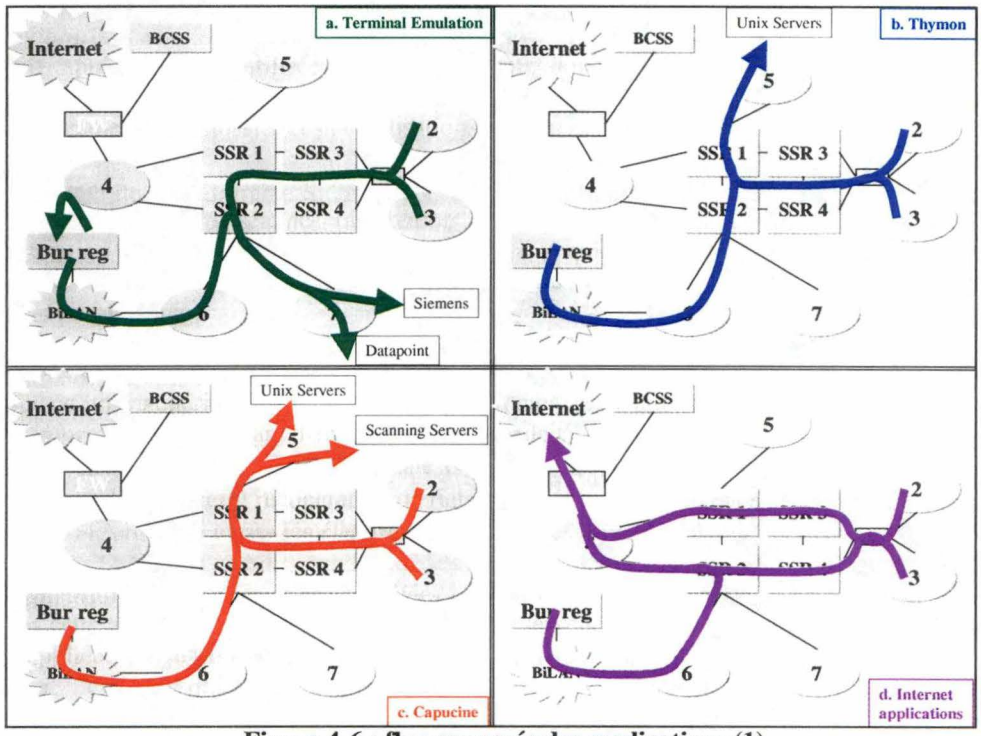

Figure 4-6 : flux supposés des applications (1)

c) Capucine (figure 4-6 c)

L'application Capucine génère du trafic à partir des VLAN 2 et 3 (site central) et VLAN 6 (bureaux régionaux) vers le VLAN 5 où se trouvent les serveurs Unix et les serveurs servant à "l'imagerie". Cette application fonctionne en deux temps :

- Recherche de la référence du dossier numérisé dans une base de données se trouvant sur un serveur Unix ;
- Recherche du dossier numérisé proprement dit sur les unités de stockage des serveurs de l'imagerie.

d) Applications Internet (figure 4-6 d)

Ce sont les clients des VLAN 2 et 3 qui génèrent le plus de trafic vers le réseau Internet, car seuls les utilisateurs du site central ont accès aux sites Web et FTP. Les utilisateurs du VLAN 6 (bureaux régionaux) n'ont accès qu'à la messagerie électronique sur Internet. Le Proxy Server et le Firewall sont en quelque sorte le goulot d'étranglement des connexions de type Internet.

e) **Applications bureautiques et assimilées** (figure 4-7 e)
Ces applications fonctionnent presque en vase clos : les serveurs de fichiers et d'impressions sont situés dans les mêmes VLAN que les stations et les imprimantes. Mais il n'est pas possible de limiter l'accès à un seul VLAN : les utilisateurs du VLAN 2 accèdent aussi aux serveurs de fichiers et d'imprimantes, ainsi qu'aux d'imprimantes se situant dans le VLAN 3 ; et vice versa.
Quelques petites applications traversent le cœur du réseau (les quatre SSR 8600) mais le trafic est semble-t-il de faible quantité.

f) **Applications "backoffice"** (figure 4-7 f)
Les applications *backoffice* génèrent quant à elle des flux totalement différents que ceux entre les utilisateurs et les applications de base (celles qui permettent au utilisateurs de faire leur travail quotidien). Ces flux d'informations se passent entre les gros systèmes informatiques : Escala, Datapoint, Siemens et les serveurs Windows NT.

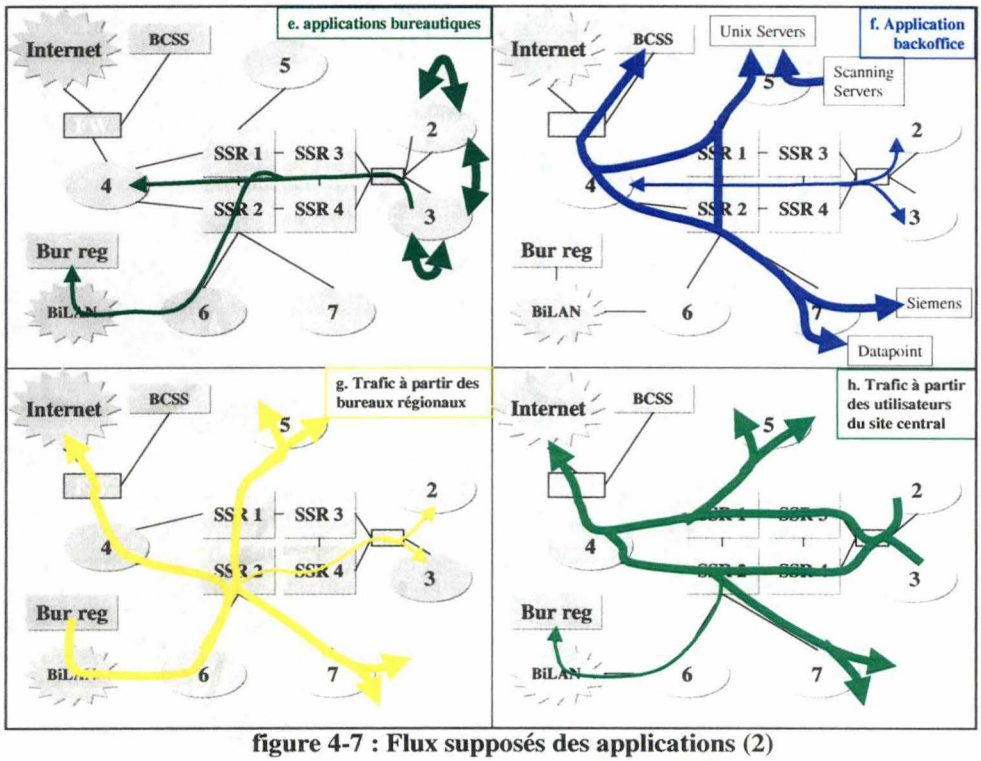

figure 4-7 : Flux supposés des applications (2)

De manière générale, on peut dire que le trafic (figure 4-7 g) en provenance des utilisateurs des bureaux régionaux (VLAN6), a comme destinations principales les serveurs Siemens et Datapoint (VLAN7), les serveurs Unix et d'imagerie (VLAN5), et les serveurs de fichiers et *backoffice* ainsi que l'accès à Internet (VLAN4). Il existe bien entendu un trafic local pour chaque bureau régional.

Pour les utilisateurs de la Tour du Midi, les flux (figure 4-7 h) les plus importants proviennent des VLAN 2 et 3 (PC, imprimantes et serveurs de fichiers et d'impression) et sont à destination des serveurs Siemens et Datapoint (VLAN7), des serveurs Unix et d'imagerie (VLAN5), et des serveurs de fichiers et *backoffice* ainsi que l'accès à Internet (VLAN4).

4.4.3 Le réseau et ses points faibles

On peut facilement, sur base des informations décrites ci-dessus, mettre en évidence deux zones génératrices de flux de données au travers du réseau de l'ONP (figure 4-8). Sur base de ces mêmes informations on peut mettre le doigt sur les point "faibles" du réseau (rond rouge) ; c'est à dire les points d'entrée dans le cœur du réseau constitué par les quatre SSR 8600. Certains de ces points faibles sont dédoublés (afin de garantir les connexions) au moyen de routeurs virtuels.

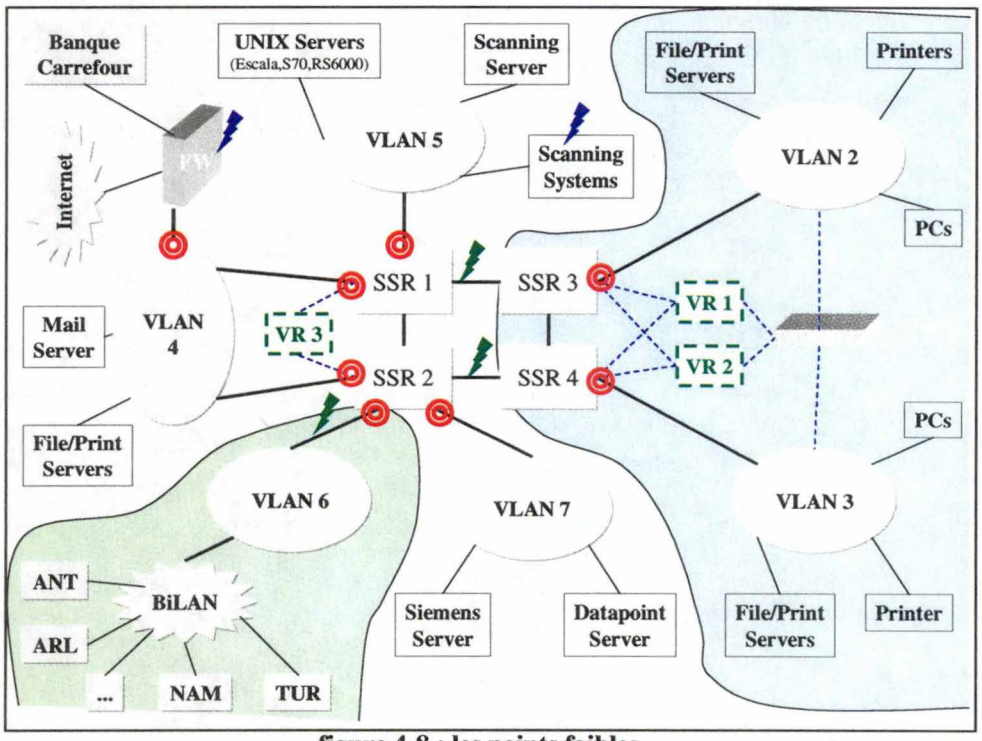

figure 4-8 : les points faibles

Pour compléter et affiner, sur base de valeurs chiffrées, l'analyse du réseau de l'ONP, il serait intéressant de mettre en place un moyen permettant l'analyse des flux à certains endroits du réseau. Dans le cadre de ce mémoire, une proposition est faite pour analyser le trafic entre les SSR3 et 1, et les SSR4 et 2. Ceci permettrait de catégoriser les différents flux provenant des utilisateurs de VLAN2 et 3.

Une deuxième proposition vise à analyser les flux entre le VLAN 6 et le SSR2. De cette manière, on pourrait obtenir une vue plus précises, des données échangées entre le site central et les bureaux régionaux. Ces précisions pouvant servir à élaborer le nouveau cahier des charges pour l'amélioration des connexions WAN.

D'autres moyens existent pour obtenir une idée approximative de l'utilisation des ressources du réseau. On peut en effet, sur base des fichiers journaux du Proxy Server, générer un rapport sur les connexions vers Internet. De même, sur base des journaux de l'imagerie, on peut se faire une idée de l'utilisation des ressources "imagerie" durant un certain temps.

4.6 Remarques

L'hétérogénéité des systèmes informatiques (serveurs, mainframe, élément réseau et autres hôtes) de l'Office National des Pensions se muera en une homogénéité toute relative lorsque la migration des anciens systèmes (Siemens et Datapoint) aura été réalisée. En effet, dans un but de faciliter la gestion des systèmes informatiques et du réseau, l'ONP a décidé, par exemple, que les nouvelles plates-formes seraient exclusivement Windows NT ou UNIX, que le protocole réseau standard serait la suite TCP/IP (Ethernet), etc.

Plus les systèmes seront homogènes, moins il faudra de compétences différentes pour la gestion de ces systèmes. Cela permettra entre autres d'obtenir un niveau d'abstraction suffisant pour mettre en place une gestion basée sur des politiques. Cette gestion se fera alors en fonction de groupes de systèmes ayant le même niveau d'abstraction.

Chapitre 5 : La gestion du réseau

5.1 Introduction

Actuellement à l'ONP, il n'y a pas de gestion structurée du réseau. Il n'est pas possible, par exemple, de quantifier le trafic généré par une application ou par un groupe d'utilisateurs. Néanmoins, un projet de gestion comprenant entre autre la gestion du réseau est en train de voir le jour. Ce travail et ce chapitre en particulier servira de base de réflexion pour la mise en place de la gestion du réseau à l'ONP.

Ce chapitre est constitué de trois parties distinctes : la première consiste à mettre en place SNMP et RMON sur différents appareils et systèmes de l'ONP ; la deuxième partie est basée sur l'analyse du trafic (à partir des applications ou d'un analyseur de protocoles) ; tandis que la troisième partie reprend différents exemples de politiques que l'on pourrait mettre en œuvre à l'ONP.

5.2 Mise en place de SNMP et de RMON

5.2.1 Stratégie

Pour mettre en place une stratégie de gestion du réseau de l'ONP via SNMP et RMON, on doit tenir compte de différents aspects comme :

- Les versions de SNMP et de RMON supportées par les différents éléments du réseau [annexe D] ainsi que les MIB supportées ;
- Les responsabilités, et donc des connaissances des membres du groupe *Systems&networks* chargés de la mise en place, de la configuration et de la maintenance des appareils réseau (hub, routeurs, switch, modem, ...), des serveurs Windows NT et Datapoint, et des imprimantes ;
- Les contrats passés avec les fournisseurs concernant le matériel, les logiciels et les services

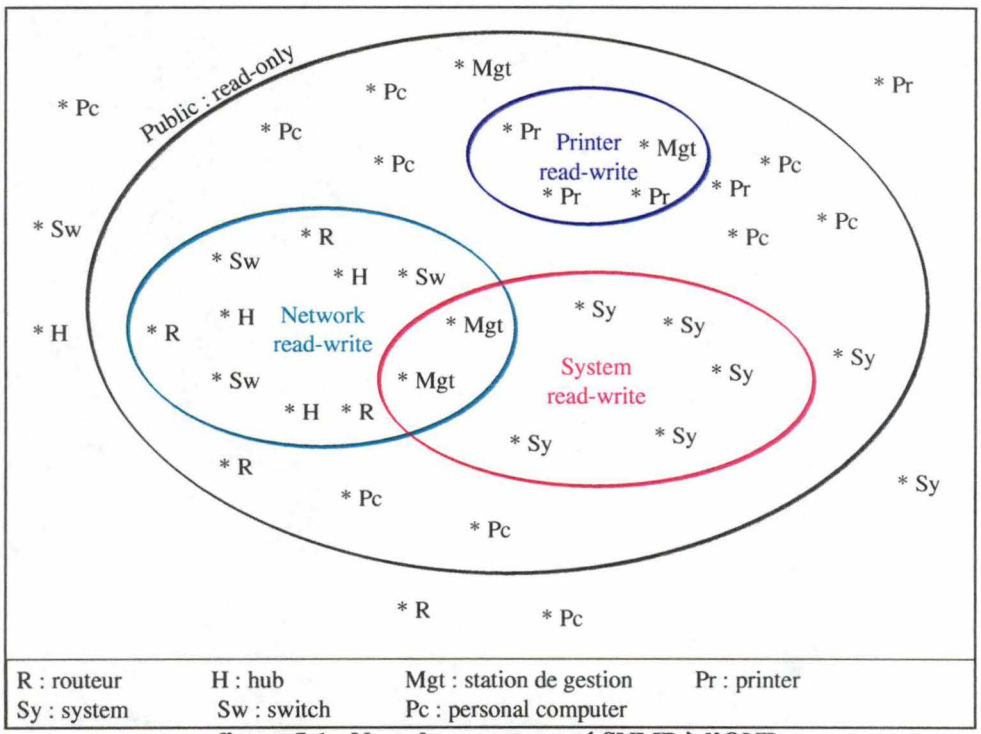

figure 5-1 : Nom de communauté SNMP à l'ONP

En considérant, ces différents points, on a pu établir une stratégie de gestion des systèmes et des appareils constituant le réseau. Dans un premier temps, il a fallut déterminer les communautés, les droits d'accès et la destination des *trap*. Ces considérations sont reprises dans le tableau ci dessous et dans la figure 5-1. On a pris la décision de récupérer tous les *trap* sur les deux stations de gestion, car un événement survenu à un endroit peut avoir des conséquences à d'autres niveaux. Il est vrai que cette solution risque d'inonder les stations de gestion avec des *trap* inutiles, mais il est possible d'établir des règles pour définir le niveau de gravité des trap reçus et des actions à prendre selon les *trap*.

Nom de communauté	Droits d'accès	Eléments ciblés	Destination des *trap*
Public	Read-Only	Tous les éléments à gérer actuellement sous notre responsabilité	
Network	Read-Create	Les appareils réseau et les deux stations de gestion	Station de gestion 1 Station de gestion 2
System	Read-Create	Les systèmes informatiques et les deux stations de gestion	Station de gestion 1 Station de gestion 2

5.2.2 Les éléments actifs du réseau

Pour mettre en place SNMP sur les différents composants du réseau qui nous intéressent, plusieurs possibilités ont été envisagées :

- **Command Line Interface (CLI):**
  Pour configurer SNMP sur les SSR 8600 et les SSR 2000, j'ai utilisé l'interface de commandes. Les commandes utilisées servaient à définir les communautés SNMP et la ou les destinations des *trap* SNMP.

```
snmp set community public priviledge read
snmp set community network priviledge Read-write
snmp set target v.w.48.134 community network status enable
snmp set target v.w.50.33 community network status enable
```

  Afin de pouvoir analyser au mieux le trafic passant au travers des SSR 8600, RMON a été configuré sur les ports gigabits Ethernet pouvant apporter le plus d'informations. Ce sont ceux marqués par un rond rouge sur la figure 4-8 du chapitre précédent. La mise en place de RMON s'est faite au moyen des commandes suivantes où les ports configurés dépendent de la configuration SSR 8600 :

```
rmon set ports gi.1.1-2,et.6.3-5
rmon set lite default-table yes
rmon set standard default-table yes
rmon set pro default-table yes
rmon set memory 4
rmon enable
```

- **Via un *browser* Internet**
  Pour configurer SNMP sur les *Switch Fast Ethernet* placés à tous les étages de la Tour du Midi, nous avons pu utiliser un browser Internet (figure 5-2). En effet si l'option permettant la configuration par le Web est activée et si on connaît le nom de communauté SNMP de l'appareil, alors il est possible de modifier la configuration des *switch*. Comme les *switch* sont configurés par défaut pour permettre ce type d'action, il n'y a eu aucun problème pour modifier les configurations.

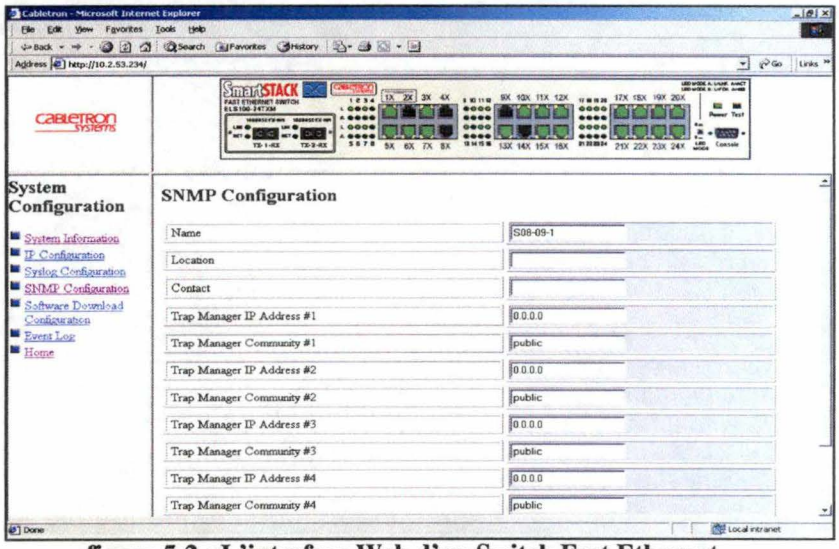

figure 5-2 : L'interface Web d'un Switch Fast Ethernet

- **Via une interface propriétaire**
  Pour modifier les configurations des Smart Switch Router (SSR) il n'est pas possible d'utiliser une interface Web. Par contre Cabletron fournit un utilitaire propriétaire pour configurer ou simplement accéder à la configuration actuelle des SSR. Cet utilitaire, Config Expert (figure 5-3), a donc permis de modifier la configuration SNMP des SSR à la seule condition de détenir le nom de communauté SNMP permettant l'opération *Set*.

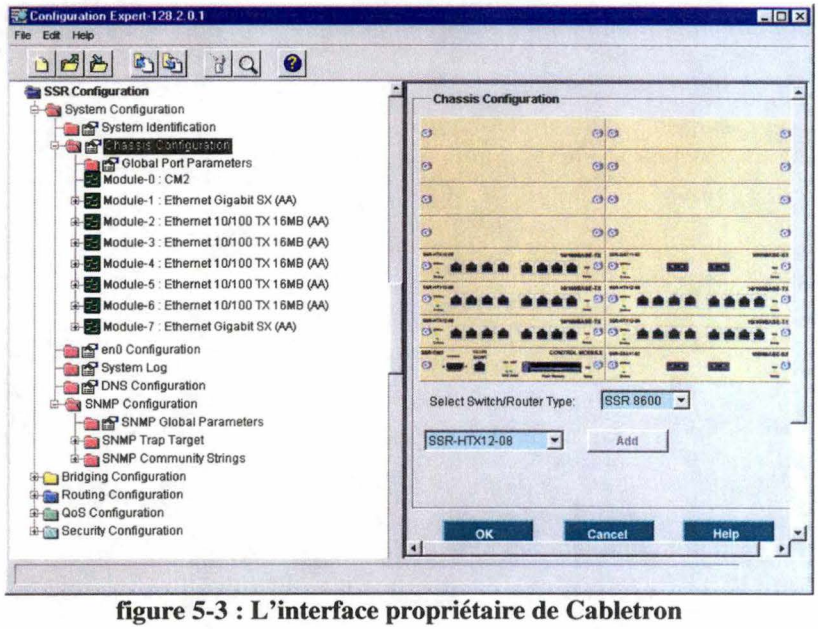

figure 5-3 : L'interface propriétaire de Cabletron

Pour configurer SNMP et RMON sur les systèmes Windows NT 4.00 il y a aussi une interface propriétaire. Celle-ci est décrite et montrée au point 5.2.3.

5.2.3 Les stations de gestion

Il a fallut aussi configurer les stations de gestion afin qu'elles puissent accéder aux informations contenues dans les MIB et aussi pour qu'elles reçoivent les trap SNMP provenant des systèmes visés. Les deux stations de gestion ont Windows NT4 SP6a comme système d'exploitation. SNMP a été configuré via l'interface Windows (figure 5-4) prévue à cet effet. [voir mur1998]

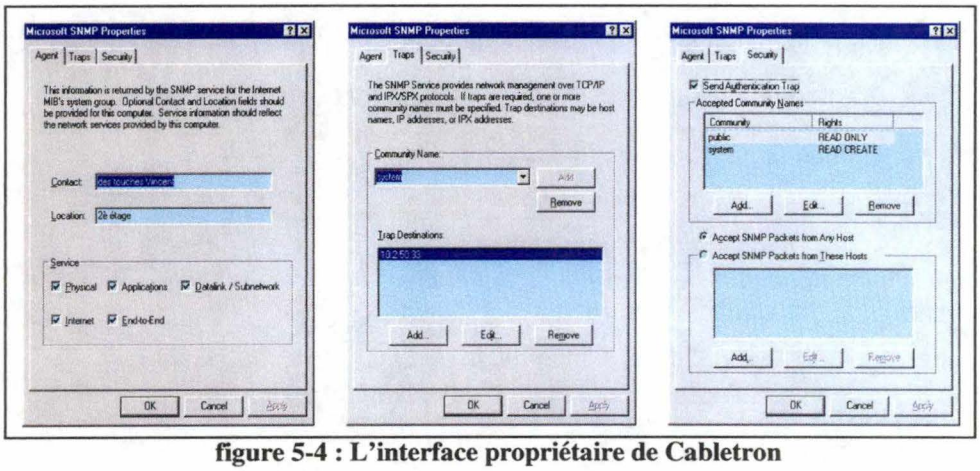

figure 5-4 : L'interface propriétaire de Cabletron

Cette configuration a été double :

- Modifier les paramètres SNMP pour faire partie des mêmes communautés que les systèmes visés ;
- configurer le module SNMP pour recevoir les trap SNMP provenant des systèmes visés.

5.2.4 Remarques

Les opérations *Get* et *GetNext* de SNMPv1 rendent difficile la compréhension des informations récupérées sur les agents. Afin de mieux saisir le sens des informations reçues il est bien souvent nécessaire de les mettre sous une autre forme pour mieux les comprendre. C'est le cas des informations qui forment un tableau, car l'ordre lexicographique de lecture des informations a comme résultat de fournir les informations colonne par colonne et non rangée après rangée. (ce qui est peut habituel).

D'un autre côté, en configurant les agents SNMP et en leur donnant une adresse IP de destination pour les *trap*, on a pu se rendre compte de différents problèmes survenant aussi bien sur les appareils actifs du réseau que sur les serveurs Windows NT. On a aussi reçu des *trap* en provenance d'applications. Le plus grand problème rencontré avec la centralisation de ces événements a été d'en faire un tri tant au niveau de la pertinence des informations qu'au niveau de la gravité de l'événement. Dans une même mesure, il a fallut décider d'actions à entreprendre face aux différentes situations.

Il est évident que l'on pourrait obtenir de meilleurs résultats si on avait utilisé les autres fonctionnalités liées à SNMP et surtout à RMON. La mise en place de ce dernier fût surtout bénéfique pour l'étude du trafic à partir de l'analyseur de protocole qui pouvait de manière automatique tirer un meilleur profit des informations relatives à RMON.

5.3 Etude du trafic à l'ONP

5.3.1 Via les applications

Il est possible d'avoir une idée du trafic généré par les applications si celles-ci enregistrent, par exemple dans des journaux, les données nécessaires. Ce système, s'il est performant, permet d'avoir par application une idée plus ou moins précise du trafic généré. Dans le cas de l'ONP on peut obtenir ce genre d'informations pour plusieurs types d'applications :

- **à partir des journaux des serveurs d'imagerie (Capucine)**

  En travaillant avec les données provenant des quatre serveurs images, on peut établir un graphique (figure 5-5) permettant de visualiser le niveau des activités liées à l'application Capucine en fonction du moment de la journée. Par niveau d'activité on entend ici le nombre de requêtes effectuées par les utilisateurs sur les différents serveurs d'images.

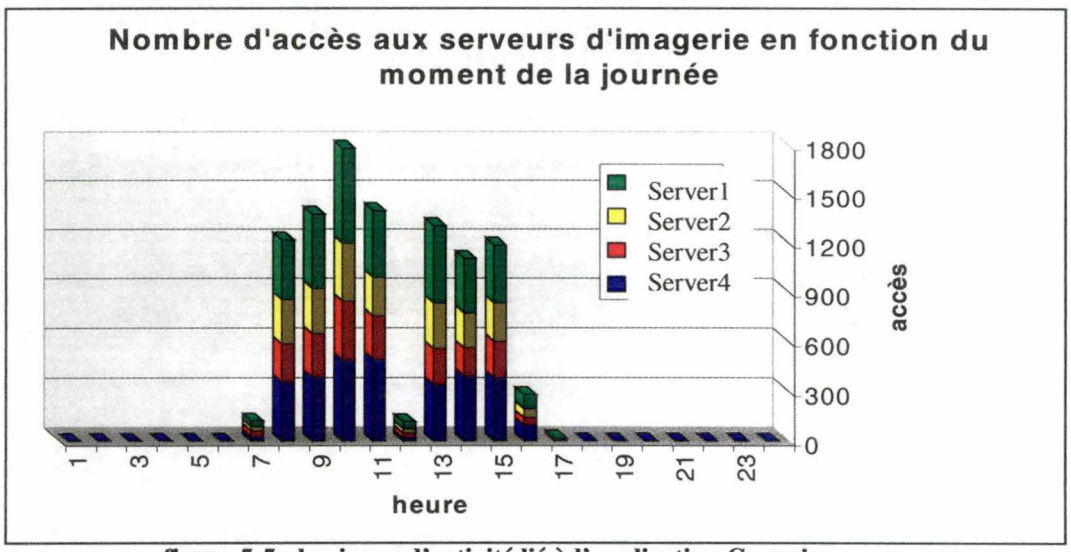

figure 5-5 : le niveau d'activité lié à l'application Capucine

Les utilisateurs de l'application Capucine effectuent leurs requêtes de manière plus ou moins identiques sur les quatre serveurs image, mais avec une préférence pour les serveurs 1 et 4. On pourrait essayer de rétablir un équilibre entre les différents serveurs. On remarquera aussi que les utilisateurs sont actifs pendant deux périodes distinctes : le matin et l'après-midi avec une plus grande activité le matin. Actuellement, nous n'avons aucune idée de l'impact sur le réseau, en terme de volume de données, d'une requête générée par l'application Capucine.

- **à partir des journaux du *Proxy Server***

Les données provenant des journaux du *Proxy Server* ont été analysées par un outil spécifique qui établit différents rapports sur le trafic passant par lui : les sites les plus visités, les utilisateurs les plus actifs, les fichiers les plus téléchargés, etc. Les deux graphiques ci-dessous ont été repris d'un de ces rapports.

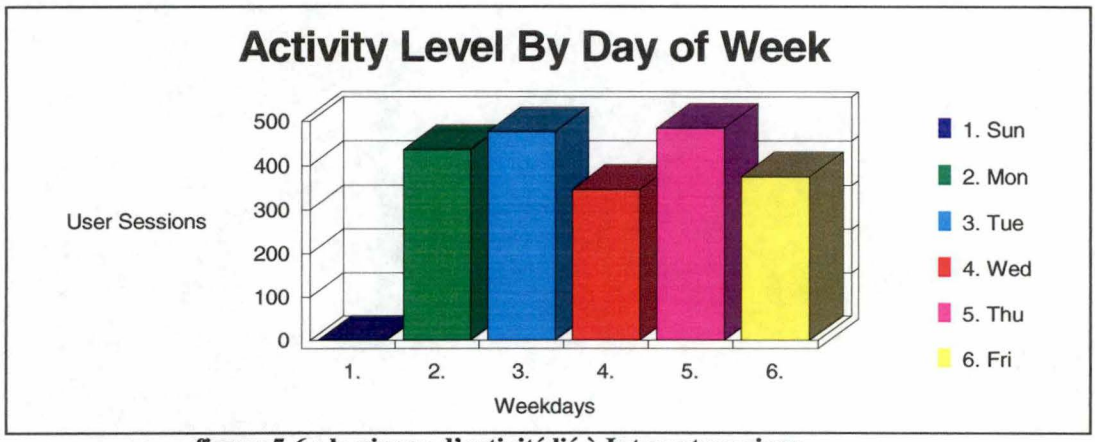

figure 5-6 : le niveau d'activité lié à Internet par jour

Ce premier graphique (figure 5-6) reprend le niveau d'activité lié à Internet par jour de la semaine. On remarquera que le nombre des sessions (une session est un ensemble d'activités Web (hits) comprenant les activités HTTP et FTP par utilisateur ; par défaut, une *user session* est terminée quand l'utilisateur est inactif pendant plus de 30 minutes) est quasi équivalent pour tous les jours de la semaine mais avec une diminution sensible le mercredi et le vendredi. Cela correspond fortement aux habitudes du personnel qui prend plus volontairement congé le mercredi et le vendredi (les personnes travaillant à 4/5 prennent, pour la plupart, leur jour de congé hebdomadaire un de ces deux jours-là).

En ce qui concerne le deuxième graphique (figure 5-7), on remarquera que le trafic Internet est aussi découpé en deux parties (matin et après-midi), comme pour le trafic lié à l'application Capucine mais avec une moins grande netteté.

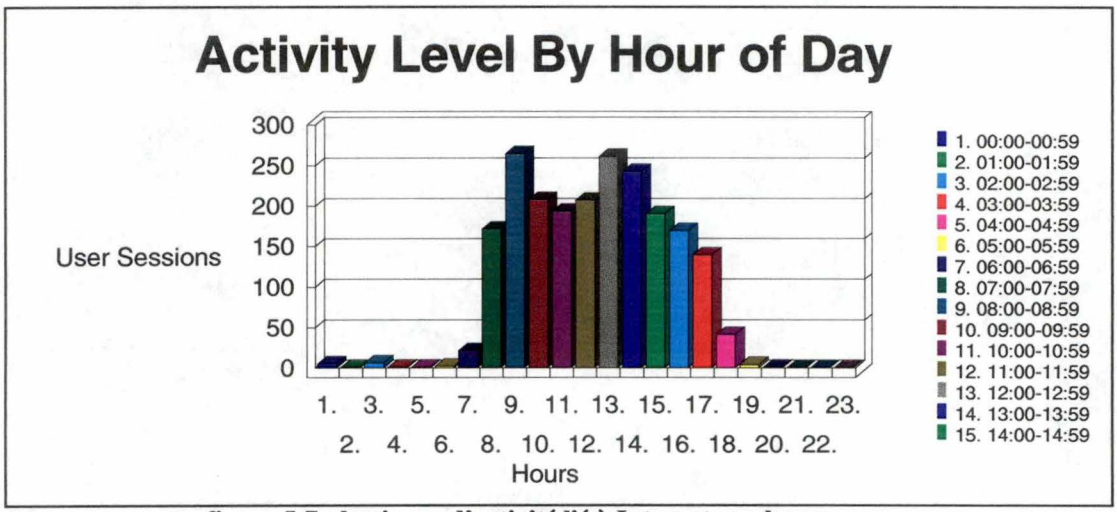

figure 5-7 : le niveau d'activité lié à Internet par heure

De manière générale, on remarquera que le niveau de trafic généré par les applications dépend fortement des us et coutumes des utilisateurs : c'est à dire que le trafic lié aux applications correspond aux horaires de travail des membres du personnel de l'ONP. Il faut aussi remarquer que les trafics générés par différentes applications sont en concurrence et que donc il est nécessaire de donner des priorités différentes à ces trafics. De plus, comme les niveaux de trafics correspondent aux plages horaire de l'ONP, on pourrait aussi envisager d'avoir des configurations réseau différentes en fonction du moment de la journée ou de la semaine.

- **à partir des journaux des backup**

Dans le cas des applications de backup des systèmes, il n'est pas nécessaire d'illustrer le niveau d'utilisation du réseau à l'aide d'un schéma. Il faut simplement partir du principe que les backup se font

de nuit, c'est à dire en dehors des plages horaires des membres du personnel de l'ONP. (Les backup ne sont pas localisés par VLAN mais bien centralisé).

5.3.2 Via un analyseur de protocoles

L'utilisation de l'analyseur de protocoles avait pour but de mettre en évidence différents flux de données sur le réseau de l'ONP. L'idée principale était de pouvoir classer le trafic en fonction des adresses IP source et destination, des ports source et destination, ainsi qu'en fonction des applications. Cette différenciation de flux devait se faire en plaçant l'analyseur de protocoles à plusieurs endroits du réseau afin d'obtenir des résultats plus précis ; par exemple en ciblant le trafic provenant d'un VLAN particulier et à destination d'un autre, ou en ciblant le trafic de et vers les bureaux régionaux, ...

Les résultats obtenus ne furent pas à la hauteur des espérances, et ce pour différentes raisons. Par exemple, cet analyseur de protocole ne tient compte que des protocoles qu'il connaît et donc il est difficile d'apprécier les données obtenues car elles ne reprennent pas la totalité du trafic (le trafic lié à l'application Capucine n'est pas pris en compte). L'arrivée tardive et partielle du matériel ne nous a pas permis de mettre en œuvre une analyse systématique des différents points faibles du réseau (figure 4-8).

D'autres solutions alternatives ont été envisagées : un analyseur de protocoles sous LINUX (solution refusée car LINUX n'est pas permis dans le réseau de production de l'ONP) ; un autre analyseur de protocoles sous Windows NT (mais lui non plus n'identifie que les protocoles connus), ...

Néanmoins, il a été possible d'analyser le trafic en connectant l'analyseur de protocoles sur le *hub* du VLAN6. Ce *hub* est connecté directement sur l'interface Fast Ethernet 6.1 sur le SSR2 (Figure 5-11) et représente la connexion entre les bureaux régionaux et le site central. Du fait du réseau switché en place à l'ONP, ce *hub* du VLAN 6 était le seul endroit où l'on pouvait effectuer une analyse du trafic avec le matériel actuellement en possession de l'ONP.

Les données obtenues ont quand même permis, compte tenu des éléments ci-dessus, de mettre en évidence certains flux, de détecter quelques "anomalies" et de tirer quelques conclusions. Les données obtenues seront traitées afin de mettre en évidence:

- Les protocoles utilisés;
- Les hôtes générant du trafic ;
- Les conversations entre hôtes.

5.3.2.1 Les protocoles utilisés

Une première utilisation de l'analyseur de protocoles a permis de détecter certaines anomalies du réseau. Par anomalie, on entend la détection de protocoles non autorisés à l'ONP comme les protocoles IPX/SPX, Netbeui, ... ou la détection d'adresses IP non attribuées à l'ONP. La correction de ces anomalies a été effectuée rapidement dans certains cas, mais dans d'autres cas il faut analyser plus en détails les résultats obtenus.

A propos de ces résultats, on a pu déterminer que :

- Le trafic IP représente quasi 99.9% du trafic analysé (donc reconnu) que ce soit en nombre de paquets ou d'octets.
- Le trafic TCP représente 99.34 % du trafic IP en terme de paquets et 98,18% en terme d'octets. Le trafic UDP quant à lui ne représente que 0.42% du nombre de paquets IP transmis et 1.38% du nombre d'octets transmis. Les protocoles ICMP et OSPF sont encore moins représentatifs.
- Le trafic TCP qui peut-être mis en évidence concerne les protocoles :
  - NETBIOS-SSN : 96.65% des paquets TCP transmis et 91.56% des octets transmis;
  - PCMAIL-SRV : 3.06% des paquets TCP transmis et 6.28% des octets transmis
  - HTTP : 0.18% des paquets TCP transmis et 1.20% des octets transmis
  - ISO-TSAP : 0.07% du trafic TCP transmis et 0.53% des octets transmis
  - Autres protocoles : il existe toute une série de protocoles détectés utilisant TCP mais dont le nombre de paquets et d'octets transmis sont insignifiants par rapport au quatre protocoles décrits ci-dessus.

Remarque : la proportion du trafic lié au protocole NETBIOS-SSN par rapport à la totalité du trafic est énorme. Cela provient du fait que l'analyseur de protocole ne tient compte que des protocoles qu'il reconnaît. Donc il n'est en aucun cas tenu compte, par exemple, du trafic de Capucine.

- Le trafic UDP qui peut-être mis en évidence concerne les protocoles :
  - SNMP : 70.79% des paquets UDP transmis et 49.18% des octets transmis
  - NETBIOS-DGM : 19.51% des paquets UDP transmis et 43.11% des octets transmis
  - NETBIOS-NS : 9.53% des paquets UDP transmis et 7.39% des octets transmis
  - SNMPTRAP : 0.17% des paquets UDP transmis et 0.32% des octets transmis
  - Autres protocoles : il existe toute une série de protocoles détectés utilisant UDP mais dont le nombre de paquets et d'octets transmis sont insignifiants par rapport au quatre protocoles décrits ci-dessus.

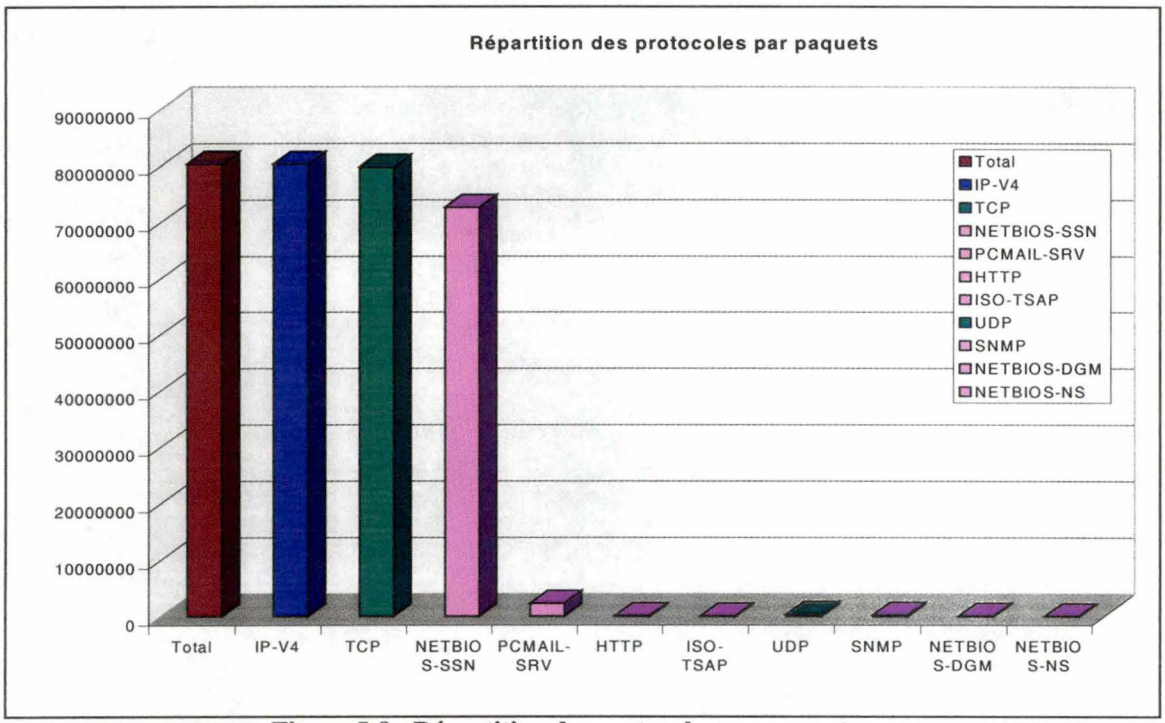

Figure 5-8 : Répartition des protocoles par paquets

La figure 5-8 permet de visualiser l'importance du trafic NETBIOS-SSN en fonction du nombre de paquets transmis. On aurait pu aussi établir un graphique sur base du nombre d'octets transmis, mais le résultat obtenu par ce graphique aurait été quasi identique à celui basé sur les paquets et on en aurait tirer les mêmes conclusions.

Le protocole NETBIOS-SSN est utilisé par les applications et les services propres à Windows NT et à son infrastructure. Par exemple, les communications entre les contrôleurs de domaine utilise ce protocole pour échanger des informations.

De manière générale on peut déduire des constations précédentes (basées sur les seuls protocoles reconnus) que le trafic IP est majoritaire, que le trafic IP le plus employé est celui concernant TCP et que celui-ci est essentiellement constitué du trafic NETBIOS-SSN (figure 5-8) Cette remarque concerne les protocoles les plus utilisés, mais il faut aussi pouvoir expliquer le pourquoi de certains protocoles même s'ils sont peu employés. Ces résultats commentés ne sont pas suffisant, il est nécessaire de pouvoir déterminer quels sont les hôtes qui génèrent du trafic.

5.3.2.2 Les hôtes générant du trafic

La deuxième phase de cette analyse sera basée sur le nombre de paquets envoyés par les hôtes. De cette manière on pourra mettre en évidence les hôtes qui génèrent les plus de trafic et ce en fonction des protocoles les plus utilisés. On peut aussi se servir des données obtenues pour détecter les hôtes générant du trafic basé sur des protocoles peu utilisés (IPX/SPX, Netbeui, ...).

Répartition du trafic IP par hôtes.

En analysant les données obtenues et comme on peut le voir sur la figure 5-9 l'hôte générant le plus de trafic IP (43%) est celui qui porte l'adresse IP v.w.55.135 : il s'agit du serveur ISIS qui est le *Primary Domain Controller* (PDC) du domaine Windows NT du production. Il est aussi *DHCP Server* pour les PC du site central et *WINS Server* pour toutes les stations de l'ONP (site central et bureaux régionaux).

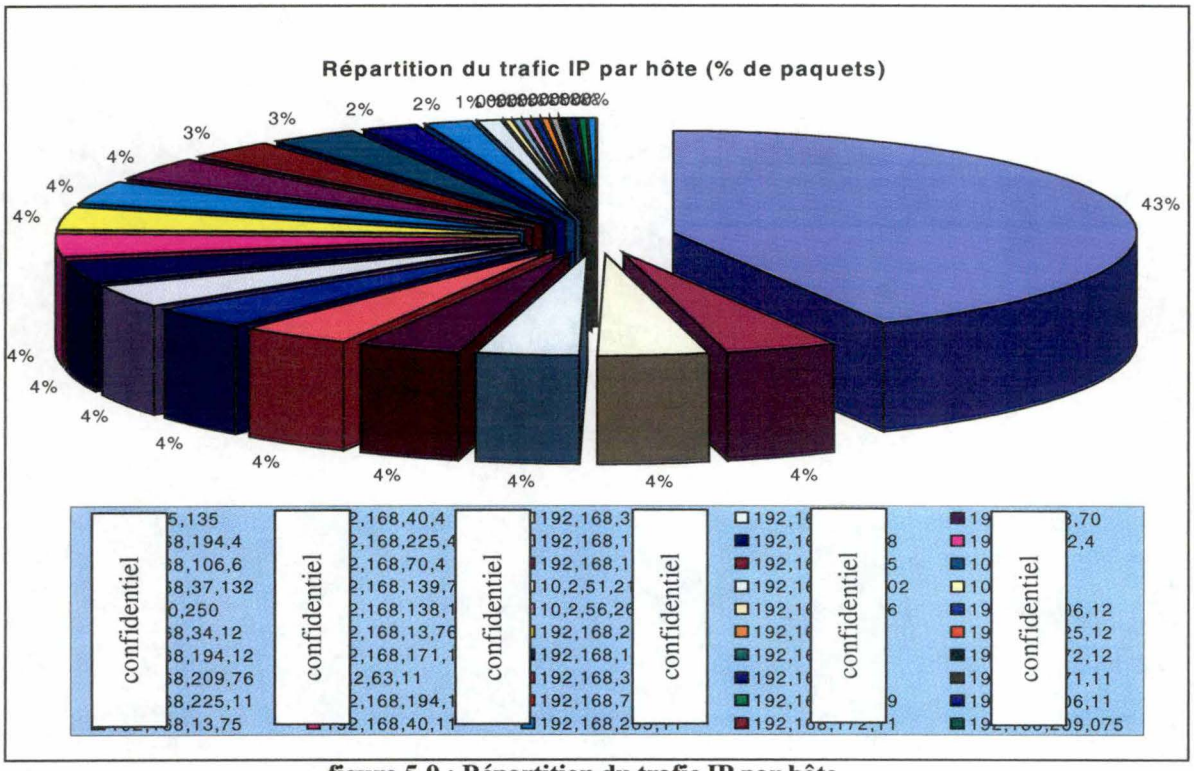

figure 5-9 : Répartition du trafic IP par hôte

Répartition du trafic TCP par hôtes.

En descendant dans la hiérarchie des protocoles on peut aussi mettre en évidence la répartition du trafic TCP généré par les différents hôtes. Sur base des résultats obtenus on remarque que le trafic TCP est généré à environ 43% par l'hôte ayant comme adresse IP v.w.55.135 qui correspond au serveur ISIS, le PDC du domaine Windows NT de production. De manière générale le trafic IP est constitué majoritairement de trafic TCP (figure5-10)

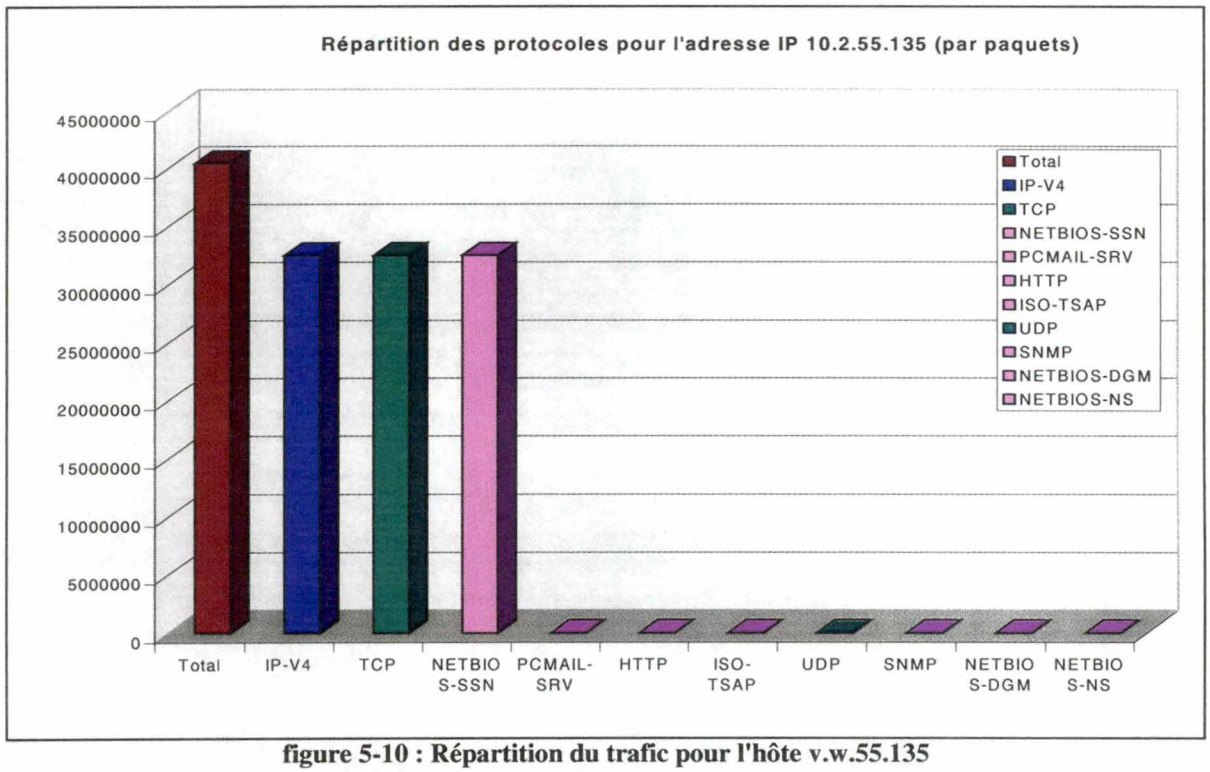

figure 5-10 : Répartition du trafic pour l'hôte v.w.55.135

Répartition du trafic NETBIOS-SSN par hôtes.
Les mêmes constatations viennent après analyse du trafic lié au protocole NETBIOS-SSN : la plus grande partie de ce trafic est généré par l'hôte ayant l'adresse IP v.w.55.135 : le serveur ISIS (le PDC du domaine Windows NT de production).(figure 5-10)

Répartition du trafic HTTP par hôtes.
On peut obtenir d'autres informations sur l'utilisation du réseau de l'ONP en analysant la répartition du trafic HTTP entre les différents hôtes ayant une connexion de ou vers le VLAN 6. Il ne faut pas oublier cependant que le trafic HTTP ne représente que 0.18% du nombre de paquets TCP transmis et 1.20% du nombre d'octets transmis.

Répartition du trafic ISO-TSAP par hôtes.
Le protocole ISO-TSAP est très faiblement utilisé par les différents hôtes, mais les données obtenues permettent de mettre en évidence que le plus "gros" générateur de trafic basé sur ce protocol est l'hôte dont l'adresse IP est r.s.1.4 : le mainframe Siemens contenant les informations concernant le paiement des pensions.

Répartition du trafic UDP par hôtes.
En analysant le trafic UDP plus en détails on remarque que le plus gros générateur de trafic UDP est une station de gestion (adresse IP : v.w.50.33) qui est utilisée pour la gestion des serveurs Bull via une produit propriétaire (ESMPRO) et aussi pour la gestion des sauvegardes dans les bureaux régionaux via une interface client du logiciel de sauvegarde.

Répartition du trafic SNMP par hôtes.
Bien que le protocole SNMP soit très peu utilisé (la mise en place SNMP sur les éléments actifs du réseau et sur les systèmes informatiques devrait changer cette situation), on peut remarquer que :

- 46.4% du trafic SNMP est généré par la station de gestion des serveurs Bull et des sauvegardes;
- en moyenne chacun des serveurs Windows NT des bureaux régionaux génère environ 3.6% du trafic SNMP
- que le reste, soit plus ou moins 7% est généré par d'autres hôtes.

L'étude de la répartition du trafic transitant par le VLAN 6 a permis de mettre en évidence, sur base du nombre de paquets générés par les hôtes, les hôtes les plus générateurs de trafic. Il a donc aussi été possible d'établir un lien entre les protocoles les plus utilisés et les hôtes générateurs de trafic concernant ces mêmes protocoles. On peut néanmoins essayer de mettre en évidence les conversations entre hôtes les plus importantes.

5.3.2.3 Les conversations entre hôtes

On va essayer de mettre en évidence les conversations les plus importantes entre les différents hôtes et en fonction des différents protocoles rencontrés. L'analyse étant de plus en plus détaillée, il n'est pas possible de la développer dans son intégralité. On ne va s'attarder que sur les protocoles les plus utilisés, sur les hôtes les plus "productifs" et sur les conversations les plus marquantes.

Conversations au niveau du protocole IP.
Au niveau du protocole IP, on peut facilement mettre en évidence quatre grands types de conversations :

- Entre ISIS (PDC du domaine Windows NT de production) et les serveurs Windows NT des bureaux régionaux qui jouent aussi le rôle de *Backup Domaine Controller* (BDC);
- Entre les serveurs Windows NT des bureaux régionaux et la station de gestion chargée de gérer les serveurs des bureaux régionaux (application propriétaire de Bull S.A.) et de gérer les sauvegardes;
- Entre le serveur de messagerie (Ms Exchange) et les serveurs Windows NT des bureaux régionaux;
- Entre le mainframe Escala2 et les serveurs Datapoint des bureaux régionaux.

Conversations au niveau du protocole TCP.
Avec les données récoltées, on peut aussi mettre en évidence les conversations les plus importantes au niveau du protocoles TCP. Comme vu précédemment, il n'est pas étonnant de remarquer que les conversations les plus importantes au niveau du protocole TCP sont les mêmes que celles décrites au niveau du protocole IP.

Conversations au niveau des protocoles applications.

- Au niveau du protocole NETBIOS-SSN, on peut effectuer le même genre de remarque que pour les protocoles TCP et IP. Il est donc assez facile de relever les grandes conversations qui existent entre les bureaux régionaux et le site central. Il s'agit des conversations entre :
  - ISIS (le PDC du domaine Windows NT de production) et les serveurs Windows NT des bureaux régionaux qui jouent aussi le rôle de *Backup Domaine Controller* (BDC);
  - Entre le serveur de messagerie (*Ms Exchange*) et les serveurs Windows NT des bureaux régionaux;
  - A ce niveau on peut aussi remarquer que d'autres applications utilisent le protocole NETBIOS-SSN pour communiquer, comme par exemple, le système de gestion du parc informatique *System Management Server* de Microsoft. Cette application utilise donc ce protocole pour établir des connexions entre le site central et les sites secondaires que représentent les bureaux régionaux.
- On remarque aussi que la station de gestion utilise le protocole PCMAIL-SRV pour communiquer avec les agents du logiciel de gestion des serveurs Bull qui sont situés dans les bureaux régionaux;
- Il semble que les communications entre le mainframe Escala2 et les serveurs Datapoint des bureaux régionaux se fassent via le protocole LOTUS-NOTES;
- L'analyseur de protocole a permis de mettre en évidence des conversations minimes mais pertinentes entre des hôtes bien précis :
  - Le protocole FTP semble être exclusivement utilisé entre le serveur Datapoint central et les serveurs Datapoint des bureaux régionaux;
  - Le protocole ISO-TSAP semble être exclusivement utilisé entre le mainframe Siemens et les PC ayant un émulateur de terminal;
  - ...
- ...

Conversations au niveau du protocole UDP.

Toutes proportions gardées, les conversations existantes au niveau du protocole UDP peuvent être divisées en quatre conversations de base :

- Entre les serveurs Windows NT des bureaux régionaux et la station de gestion de ces serveurs et des sauvegardes qui y sont faites;
- Entre les serveurs Windows NT des bureaux régionaux et l'hôte ayant comme adresse IP x.y.q.78. il s'agit du serveur JUKEBOX qui gère une collection de CD-ROM (documentation, software, ...)
- Entre les serveurs Windows NT des bureaux régionaux et le *Primary Domain Controller* ISIS;
- Entre les serveurs Windows NT des bureaux régionaux et l'hôte ayant comme adresse IP v.w.48.100.

Une analyse beaucoup plus fine, mais demandant plus de connaissances ou du moins plus spécialisée pouvait aussi être effectuée via le module de capture de trafic présent au sein du logiciel d'analyse de protocoles. Une fois la capture effectuée, il faut pouvoir comprendre les échanges de données. Comme on l'a vu précédemment la plupart du trafic concerne le protocole NETBIOS-SSN dont les principaux générateurs sont les contrôleurs de domaine de Windows NT4. Il existe une littérature spécialisée analysant plus en détails de tels échange de données [wil2000]. Cette documentation permet de tirer d'autres conclusions beaucoup plus précises mais aussi de mettre en évidence certains échanges de données par rapport à d'autres.

5.3.3 Remarques

Si l'on veut pouvoir gérer au mieux le trafic sur le réseau de l'ONP il faut aussi pouvoir dresser une liste des protocoles qui peuvent être employés et dans quelle mesure. Il est donc nécessaire de déterminer une ligne de base qui permettrait de détecter les modifications d'utilisation des ressources du réseau : tant au niveau de l'utilisation des ressources (quantité consommée, moment de la journée,...) qu'au niveau des protocoles utilisés. Dans une même mesure il faut aussi identifier quels sont les applications qui partagent les mêmes ports TCP ou les mêmes services réseau. On peut prendre à titre d'exemple le protocole NETBIOS-SSN qui semble être utilisé par plusieurs services ou applications propres à Windows NT.

Comme annoncé plus haut, les résultats obtenus sont assez éloignés des résultats espérés. Il est vrai que l'on voulait pouvoir mettre en évidence le trafic généré par l'application Capucine par rapport au trafic généré par d'autres applications. Néanmoins la méthode utilisée ici pourra servir pour ré-exécuter cette analyse lorsque l'on pourra configurer l'analyseur de protocoles en notre possession et que l'on aura le reste du matériel lié à cet analyseur (des modules clients, des fiches réseaux dédoublées, etc.).

Ce type d'analyse, même si elle n'est pas complète, n'a jamais été réalisé à l'office National des Pensions. Bien souvent le trafic était imaginé ou idéalisé par des réflexions du genre : "Normalement les données devraient

passer par là et elles ne doivent pas représenter une grande partie du trafic". Autrement dit, aucune quantification du trafic n'a été réalisée, même pas pour s'assurer que le trafic capucine provenant et allant vers les bureaux régionaux était bien dirigé sur les connexions ADSL. Tout cela pour dire que les résultats obtenus par cette analyse ont permis de lever un peu le voile sur le trafic réseau à l'ONP.

Nota Bene : Au départ il était question d'inclure dans une annexe les résultats obtenus par l'analyseur de protocole mais pour des raisons de sécurité et de confidentialité ils ne seront pas inclus dans ce document.

5.4 Exemples de politiques

Ci-dessous sont décrits des exemples de politiques pouvant être appliquées à l'Office National des Pensions. Ces politiques tiennent compte de la situation actuelle du réseau de l'ONP et des possibilités de gestion qui y sont envisagées. Il est possible que d'autres politiques soient réellement mises en place et il faut tenir compte du fait que la mise en œuvre de ces politiques ne fait pas partie du mémoire.

5.4.1 Rendre le trafic "imagerie" venant des bureaux régionaux prioritaire

- **Enoncé de la situation**

Pour rappel, une "politique" est déjà mise en place dans le bureaux régionaux : une différenciation de trafic est effectuée sur base des ports utilisés par l'application Capucine avec pour but de diriger le trafic "Capucine" vers la ligne ADSL ayant un débit plus élevé que la ligne PSTN. Cette différentiation n'est valable que pendant le transit du trafic via le réseau BiLAN de Belgacom. L'idée ici est de donner (ou de redonner) une priorité plus élevée au trafic "Capucine" provenant des bureaux à destination des serveurs de l'imagerie se trouvant dans le VLAN5 et vice versa.

- **Solution envisagée**

Configurer les SSR 8600 (le cœur du réseau) pour donner une plus haute priorité au trafic "Capucine" provenant des bureaux régionaux (VLAN6) et à destination des serveurs de l'imagerie (VLAN5). Afin de limiter le trafic vers les bureaux régionaux, il existe déjà une *Control Access List* (positionnée sur l'interface et.6.1 (figure 5-11)) dont les seuls membres ont accès aux bureaux régionaux.

- **Mise en place du concept de politique**

1) **associer un rôle aux interfaces**

Il faut associer un rôle aux interfaces des quatre SSR8600 impliqués dans ce trafic (figure 5-11). Une nomenclature a été mise en place pour désigner les interfaces : pour **`et.5.6`** où **`et`** désigne une interface de type Fast Ethernet, **`5`** désigne le module concerné dans le SSR, et le **`6`** désigne le numéro du port (d'interface) sur ce module. Certains noms d'interfaces commencent par **`gi`**, cela désigne des ports Gigabit Ethernet.

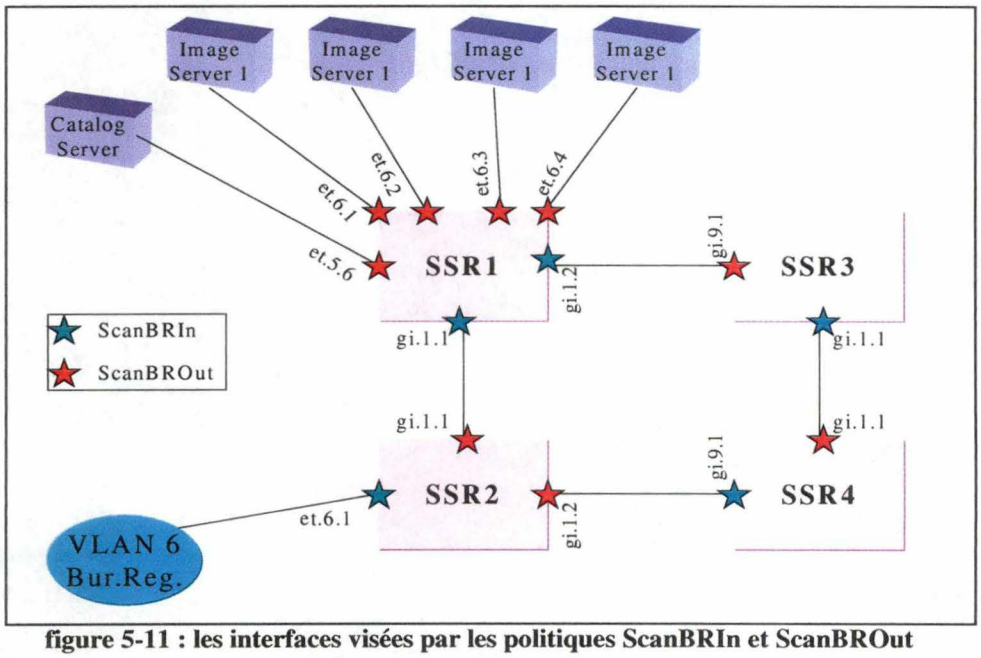

figure 5-11 : les interfaces visées par les politiques ScanBRIn et ScanBROut

Pour associer un rôle aux interfaces, on a utilisé la syntaxe suivante :

`<Nom de l'interface> : role = <nom du rôle> (Commentaire)`

SSR-B1-1
Interface et.5.6 : role = ScanBROut (Catalog Server)
Interface et.6.1 : role = ScanBROut (Image Server 1)
Interface et.6.2 : role = ScanBROut (Image Server 2)
Interface et.6.3 : role = ScanBROut (Image Server 3)
Interface et.6.4 : role = ScanBROut (Image Server 4)
Interface gi.1.1 : role = ScanBRIn (vers SSR2)
Interface gi.1.2 : role = ScanBRIn (vers SSR3)
...

SSR-B1-2
Interface gi.1.1 : role = ScanBROut (vers SSR1)
Interface gi.1.2 : role = ScanBROut (vers SSR4)
Interface et.6.1 : role = ScanBRIn (vers VLAN6 / bureaux régionaux)

SSR-19-3
Interface gi.1.1 : role = ScanBRIn (connexion inter SSR8600)
Interface gi.9.1 : role = ScanBROut (connexion inter SSR8600)

SSR-19-4
Interface gi.1.1 : role = ScanBROut (connexion inter SSR8600)
Interface gi.9.1 : role = ScanBRIn (connexion inter SSR8600)

2) **définir les politiques :**

Policy 1 : les paquets provenant des bureaux régionaux (VLAN6) et à destination des serveurs imagerie du VLAN5 seront traités en priorité par rapport aux autres paquets provenant des bureaux régionaux. Cela correspond au rôle ScanBRIn.

Policy 2 : les paquets provenant des serveurs de l'imagerie (VLAN5) et à destination des applications Capucine clients des bureaux régionaux seront traités en priorité par rapport aux autres paquets à destination des bureaux régionaux. Cela correspond au rôle ScanBROut.

3) **Pseudo algorithme :**

Policy 1 :

```
CapucineDestList = {x.y.z.1, x.y.z.2,
      x.y.z.3, x.y.z.4, x.y.z.5}

IF incommingPacketAdrdest = CapucineDestList
AND incommingPacketPortDest = 27759
THEN put incoming packet into a high priority queue
```

Policy 2 :

```
CapucineSrcBRList = {liste des ranges d'adresses Ip des
      stations des bureaux régionaux}

IF incommingPacketAdrdest = CapucineSrcBRList
AND IncommingPacketPortdest = 27750 OR 27751 OR ...
THEN put incoming packet into a high priority queue
```

4) **Extrait de la configuration à mettre en place :**

Voici un extrait de la configuration qui devrait être mise en place sur un des SSR 8600 (le SSR-B1-1). Cet extrait se base sur les fonctionnalités des SSR que l'on pourrait actuellement installer à partir de la *Command Line Interface* (CLI) avec la commande :

qos set ip *<name> <priority> <scraddr/mask>* | **any** *<dstaddr/mask>* | **any** *<srcport>* | **any** *<dstport>* | **any** *<tos>* | **any** *<portlist>* | *<interface-list>* | **any** *<protocol>* | **any** *<tos-mask>* | **any** *<tos-precedence-rewrite>* | **any** *<tos-rewrite>* | **any**.

Policy 1 :

```
qos set ip scanbrin1 high any x.y.z.1/255.255.255.0 any any any gi.1.1-2
qos set ip scanbrin2 high any x.y.z.2/255.255.255.0 any any any gi.1.1-2
qos set ip scanbrin3 high any x.y.z.3/255.255.255.0 any any any gi.1.1-2
```

```
qos set ip scanbrin4 high any x.y.z.4/255.255.255.0 any any any gi.1.1-2
qos set ip scanbrin5 high any x.y.z.5/255.255.255.0 any any any gi.1.1-2
...
Policy 2 :
qos set ip scanbrout1 high any x.y.13.0/255.255.255.192 any 27541 any et.5.6,
et.6.1-4
qos set ip scanbrout2 high any x.y. 13.0/255.255.255.192 any 27542 any et.5.6,
et.6.1-4
qos set ip scanbrout3 high any x.y. 13.0/255.255.255.192 any 27542 any et.5.6,
et.6.1-4
qos set ip scanbrout4 high any x.y. 13.0/255.255.255.192 any 27545 any et.5.6,
et.6.1-4
qos set ip scanbrout5 high any x.y. 13.0/255.255.255.192 any 27546 any et.5.6,
et.6.1-4
qos set ip scanbrout6 high any x.y. 13.0/255.255.255.192 any 27547 any et.5.6,
et.6.1-4
qos set ip scanbrout7 high any x.y. 13.0/255.255.255.192 any 27548 any et.5.6,
et.6.1-4
qos set ip scanbrout8 high any x.y. 13.0/255.255.255.192 any 27549 any et.5.6,
et.6.1-4
qos set ip scanbrout9 high any x.y. 13.0/255.255.255.192 any 27550 any et.5.6,
et.6.1-4
qos set ip scanbrout10 high any x.y. 13.0/255.255.255.192 any 27552 any et.5.6,
et.6.1-4
qos set ip scanbrout11 high any x.y. 13.0/255.255.255.192 any 27553 any et.5.6,
et.6.1-4
qos set ip scanbrout12 high any x.y. 13.0/255.255.255.192 any 27554 any et.5.6,
et.6.1-4
qos set ip scanbrout13 high any x.y. 13.0/255.255.255.192 any 27555 any et.5.6,
et.6.1-4
qos set ip scanbrout14 high any x.y. 13.0/255.255.255.192 any 27556 any et.5.6,
et.6.1-4
qos set ip scanbrout15 high any x.y. 13.0/255.255.255.192 any 27557 any et.5.6,
et.6.1-4
qos set ip scanbrout16 high any x.y. 13.0/255.255.255.192 any 27558 any et.5.6,
et.6.1-4
qos set ip scanbrout17 high any x.y. 13.0/255.255.255.192 any 27559 any et.5.6,
et.6.1-4
qos set ip scanbrout18 high any x.y. 13.0/255.255.255.192 any 27560 any et.5.6,
et.6.1-4
```

Les lignes de commandes ci-dessus représentent la mise en place de la politique `Policy 2` répondant au rôle `ScanBROut` pour un des treize bureaux régionaux. Il faudrait pour bien faire reprendre ces 18 lignes pour chacun des douze autres bureaux régionaux.

Cet exemple permet de mieux percevoir les atouts de l'utilisation du concept de politique. En effet, s'il y avait une PIB associée aux SSR 8600 de Cabletron, on pourrait définir les politiques sur base de la syntaxe IF <condition> THEN <action> et soit laisser le SSR 8600 mettre en place la configuration correspondante aux politiques (s'ils sont assez puissant pour jour le rôle de PDP et de PEP (modèle *Two-Tiers*)), soit laisser un PDP qui connaît les possibilités des SSR 8600 de le configurer via un PEP (modèle *Three-Tiers*).

- **Remarques**

Afin de favoriser un flux par rapport à un autre, on peut aussi garantir une bande passante minimum allouée à certain flux. Cette allocation se fait sur base d'un *Access Control List* (ACL) et permet de spécifier aussi le traitement à réserver aux paquets excédentaires : soit ils sont écartés , soit ils sont placés dans une queue prioritaire (Control, High, Medium, Low). Les critères de sélection des ACL sont basés sur différents champs des entêtes IP, TCP ou UDP : les adresses source et destination, les ports source et destination, le *Type of Service*, ...

Dans le cas qui nous intéresse on pourrait soit :

- Définir une bande passante minimum pour les flux "Capucine" et traiter les paquets excédentaires en les plaçant dans une queue haute priorité ;
- Définir une bande passante minimum pour les flux HTTP et traiter les paquets excédentaires en les écartant ou en les plaçant dans une queue basse priorité.

Pour rendre la configuration dynamique (un des avantages de la gestion basée sur une politique), on pourrait allouer plus ou moins de bande passante aux flux "Capucine" en fonction de la quantité de données transmises (par exemple, en se basant sur les historiques de RMON), et ce au détriment du flux HTTP.

La mise en place de ce type de QoS pour les mêmes politiques sur les SSR 8600 correspondrait aux lignes de commandes ci-dessous (via le *Command Line Interface*) en considérant que nous avons deux *Access Control List* : une pour les paquets provenant des bureaux régionaux à destination des serveurs de l'imagerie (scanBRInAcl) et une autre pour les paquets provenant des serveurs de l'imagerie et à destination des applications clients (scanBROutAcl).

```
rate-limit <name> input acl <acl list> rate <rate-limit> exceed-
action      drop-packets|set-priority-low|set-priority-medium|set-
priority-high
rate-limit <name> apply interface <interface>|all
```

```
rate-limit scanbrinrl input acl scanbrinacl rate 25000000 exceed-action set-
priority-high
rate-limit scanbroutrl input acl scanbroutacl rate 25000000 exceed-action set-
priority-high

rate-limit scanbrinrl apply interface gi.1.1
rate-limit scanbroutrl apply interface gi.1.2
```

Afin de s'assurer une bande passante maximale, on peut aussi, par exemple, limiter le trafic HTTP en considérant que l'on a une ACL (`httpBRAcl`) pour les paquets HTTP de et vers les bureaux régionaux.

```
rate-limit httprl input acl httpbracl rate 10000000 exceed-action dropped-
packets

rate-limit httprl apply interface all
```

La valeur *<rate-limit>* dépend évidemment du débit des interfaces : si on avait un système de gestion du réseau basé sur une politique, on ne devrait pas se soucier directement de cette valeur. Il suffirait de définir une limite sur base d'un pourcentage de la bande passante totale et le PDP, au courant des possibilités des PEP, configurerait de manière transparente pour nous les limites de débit.

5.4.2 Donner des priorités différentes aux flux liés aux applications

- **Enoncé de la situation**
  Actuellement, si on ne considère que le trafic intra site central, il n'y a aucune configuration mise en place permettant de classer les différents flux dans un but de gestion du réseau. Tous les flux, qu'ils soient importants du point de vue de la quantité de données transmises ou du point de vue de leur rôle dans les tâches des utilisateurs, sont traités de la même manière.

- **Solutions envisagées**
  Il faudrait, dans un premier temps, mettre en place un système de classification des flux sur base de l'importance des applications en terme de quantité de données transmises et de nécessité pour le travail quotidien. Dans second temps, il faudrait aussi mettre en place un système de prioritisation de ces flux.

  A titre d'exemple, on considérera quatre applications ou types d'applications :
  - Capucine ;
  - Thymon ;
  - Terminal Emulation ;
  - Accès à Internet.

  Si on devait classer ces applications en fonctions de la quantité de trafic et de leur importance à l'ONP, on obtiendrait, de manière générale, le tableau suivant (où ++ équivaut à une grande importance ou à une grande quantité de données et - - signifie peu d'importance ou peu de données transmises) :

Type d'applications	Quantité de données	Importance pour le travail	Priorités
Capucine	+ +	+ +	High
Thymon	+	+ +	High
Terminal Emulation	- -	+ +	Medium
Accès à Internet	-	- -	Low

- **Mise en place du concept de politique**

  1) **associer un rôle aux interfaces :**

  Les routeurs, qui sont ici concernés, sont aussi bien les SSR 8600 (cœur du réseau) que les SSR 2000. A titre d'exemple on ne reprend ici l'association de rôles que pour deux des SSR 8600 et un des SSR2000 (figure 5-12):

  SSR-B1-1 (8600)
  Interface et.2.2 : role = Http4to23 (FireWall)
  Interface et.5.2 : role = Thymon4to23 (Escala 1)
  Interface et.5.3 : role = Thymon4to23 (Escala 2)
  Interface gi.1.1 : role = Thymon23to5, Http23to4, Term7to23 (connexion vers SSR2)
  Interface gi.1.2 : role = Thymon23to5, Http23to4, Term23to7 (connexion vers SSR3)

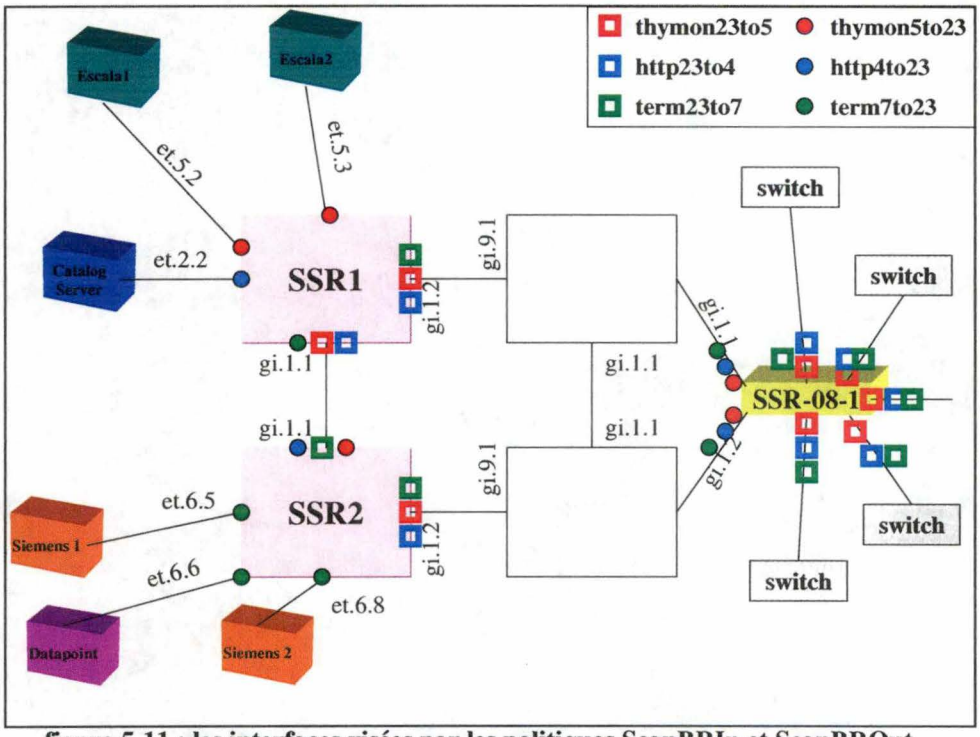

figure 5-11 : les interfaces visées par les politiques ScanBRIn et ScanBROut

SSR-B1-2 (8600)
Interface et.6.3 : role = Term7to23 (serveurs Siemens)
Interface et.6.4 : role = Term7to23 (serveurs Siemens)
Interface et.6.6 : role = Term7to23 (serveurs Datapoint)
Interface gi.1.1 : role = Thymon5to23, Http4to23, Term23to7 (connexion vers SSR1)
Interface gi.1.2 : role = Thymon23to5, Http23to4, Term23to7 (connexion vers SSR4)

SSR-08-1 (2000)
Interface gi.3.1 : role = Thymon5to23, Http4to23, Term7to23 (vers le SSR3)
Interface gi.3.2 : role = Thymon5to23, Http4to23, Term7to23 (vers le SSR4)
Interface et.1.* : role = Thymon23to5, Http23to4, Term23to7 (PC)
Interface et.2.* : role = Thymon23to5, Http23to4, Term23to7 (PC)
Interface et.4.1-8 : role = Thymon23to5, Http23to4, Term23to7 (PC)

Il est également possible de regrouper les rôles de façon à les rendre plus généraliste. Il est donc envisageable de définir deux rôles distincts :

- Le rôle `DataFrom23` qui reprendrait les rôles attribués aux interfaces devant traiter le trafic provenant des VLAN2 et 3;
- Le rôle `DataTo23` qui reprendrait les rôles attribués aux interfaces devant traiter le trafic à destination des VLAN2 et 3.

Il faut remarquer alors que, pour certaines interfaces, une ou plusieurs politiques (ou une partie) attribuées à un de ces deux rôles généralistes seront superflues.

2) **définir les politiques :**

`Policy 1` : les paquets provenant des VLAN2 et 3 (utilisateurs du site central) générés par l'application Thymon et à destination des serveurs Escala1 et 2 du VLAN5 seront traités en haute priorité par rapport aux autres paquets provenant des VLAN 2 et 3. Cela correspond au rôle `Thymon23to5`.

`Policy 2` : les paquets provenant des VLAN2 et 3 (utilisateurs du site central) générés par les émulateurs de terminal et à destination des serveurs Siemens et Datapoint du VLAN7 seront traités en priorité moyenne par rapport aux autres paquets provenant des VLAN 2 et 3. Cela correspond au rôle `Term23to7`.

`Policy 3` : les paquets provenant des VLAN2 et 3 (utilisateurs du site central) générés par les *browser* et à destination du FireWall (VLAN4) seront traités en basse priorité par rapport aux autres paquets provenant des VLAN 2 et 3. Cela correspond au rôle `Http23to4`.

`Policy 4` : les paquets provenant des serveurs Escala1 et 2 du VLAN5 et à destination des clients Thymon des VLAN2 et 3 (utilisateurs du site central) seront traités en haute priorité par rapport aux autres paquets provenant des VLAN 2 et 3. Cela correspond au rôle `Thymon5to23`.

`Policy 5` : les paquets provenant des serveurs Siemens et Datapoint du VLAN7 et à destination des émulateurs de terminal des VLAN2 et 3 (utilisateurs du site central) seront traités en priorité moyenne par rapport aux autres paquets provenant des VLAN 2 et 3. Cela correspond au rôle `Term7to23`.

`Policy 6` : les paquets provenant FireWall (VLAN4) et à destination des *browser* des VLAN2 et 3 (utilisateurs du site central) seront traités en basse priorité par rapport aux autres paquets provenant des VLAN 2 et 3. Cela correspond au rôle `Http4to23`.

Dans le cas où on voudrait regrouper les politiques, il suffirait de rassembler les politiques 1, 2 et 3 pour obtenir une politique traitant le trafic provenant des VLAN 2 et 3 (rôle `DataFrom23`), et de réunir les politiques 4, 5 et 6 pour obtenir une politique s'occupant du trafic à destination des VLAN 2 et 3 (rôle `DataTo23`)

3) **Pseudo algorithme :**

`Policy 1 :`

```
ThymonDestList = {x.y.z.201, x.y.z.202}
                 /* adresses des serveurs Escala */

IF incommingPacketAdrdest = ThymonDestList
AND IncommingPacketPortSrc = ThymonPortSrc
THEN put incoming packet into a high priority queue
```

`Policy 2 :`

```
TermDestList = {r.s.1.4, r.s.1.5,r.s.1.30}
      /* adresses des serveurs Siemens et Datapoint */

IF incommingPacketAdrdest = TermDestList
AND IncommingPacketPortSrc = TermPortSrc
THEN put incoming packet into a medium priority queue
```

Policy 3 :

```
HttpDestList = {x.y.q.77}
      /* adresses du proxy serveur */

IF incommingPacketAdrdest = HttpDestList
AND IncommingPacketPortSrc = 80
THEN put incoming packet into a low priority queue
```

Policy 4 :

```
ThymonSrcList = {v.w.48.* à v.w.63.*}
      /* adresses des PC des VLAN 2 et 2*/

IF incommingPacketAdrdest = ThymonSrcList
AND IncommingPacketPortDest = ThymonPortSrc
THEN put incoming packet into a high priority queue
```

Policy 5 :

```
TermSrcList = {v.w.48.* à v.w.63.*}
      /* adresses des PC des VLAN 2 et 2*/

IF incommingPacketAdrdest = TermSrcList
AND IncommingPacketPortSrc = TermPortSrc
THEN put incoming packet into a medium priority queue
```

Policy 3 :

```
HttpSrcList = {v.w.48.* à v.w.63.*}
      /* adresses des PC des VLAN 2 et 2*/

IF incommingPacketAdrdest = HttpSrcList
AND IncommingPacketPortDest = 80
THEN put incoming packet into a low priority queue
```

De la même manière qu'auparavant, si on choisit de regrouper les politiques, il faut le faire en rassemblant les politiques 1, 2 et 3 car elles traitent du trafic provenant des VLAN 2 et 3, et en rassemblant les politiques 4, 5 et 6 car elles s'occupent du trafic à destination des VLAN 2 et 3

4) Extrait de la configuration à mettre en place :

Vu le nombre de politiques de cet exemple, le nombre d'adresse source et destination, le nombre de ports source et destination, il ne sera pas possible de reprendre toutes les lignes de commandes qui devraient être mise en place sur les différents SSR pour arriver à obtenir une configuration se rapprochant des politiques énoncées.

La configuration des politiques énoncées sur le SSR-B1-1 (ainsi que sur les autres SSR) se fera aussi à l'aide de la commande suivante :

qos set ip *<name>* *<priority>* *<scraddr/mask>*|**any** *<dstaddr/mask>*|**any** *<srcport>*|**any** *<dstport>*|**any** *<tos>*|**any** *<portlist>*|*<interface-list>*|**any** *<protocol>*|**any** *<tos-mask>*|**any** *<tos-precedence-rewrite>*|**any** *<tos-rewrite>*|**any**.

Policy 1 :

```
qos set ip Thymon23to5a high any x.y.z.201/255.255.255.0 any any any gi.1.1-2
qos set ip Thymon23to5b high any x.y.z.202/255.255.255.0 any any any gi.1.1-2
```

Policy 2 :

```
qos set ip Term23to7 medium any r.s.1.4/255.255.248.0 any any any gi.1.2
qos set ip Term23to7 medium any r.s.1.5/255.255.248.0 any any any gi.1.2
qos set ip Term23to7 medium any r.s.1.30/255.255.248.0 any any any gi.1.2
```

Policy 3 :

```
qos set ip Http23to4 low any x.y.q.77/255.255.255.192 any any any gi.1.1-2
```

Policy 4 :

```
qos set ip Thymon5to23 high any v.w.48.0/255.255.248.0 any any any et.5.2-3
```

Policy 5 :

```
qos set ip Term7to23 medium r.s.1.4/32 v.w.48.0/255.255.248.0 any any any gi.1.1
```

```
qos set ip Term7to23 medium r.s.1.5/16 v.w.48.0/255.255.248.0 any any any gi.1.1
qos set ip Term7to23 medium r.s.1.4/16 v.w.48.0/255.255.248.0 any any any gi.1.1

Policy 6 :
qos set ip Http4to23 low x.y.q.77/24 v.w.48.0/255.255.248.0 any any et.2.2
...
```

- **Remarque**

La façon de mettre en œuvre des politiques au sein même des différents composants du réseau dépendant des capacités de ceux-ci, il aurait aussi été possible d'utiliser les commandes permettant de limiter la bande passante en fonction d'ACL qui auraient été établies selon des critères tels que les adresses source et destination, les ports source et destination, ...

5.4.3 Réseau de nuit et réseau de jour

- **Enoncé de la situation**

Lors de l'analyse du réseau sur base des applications, il était assez facile de mettre en évidence que le réseau en général est utilisé d'une certaine manière le jour et d'une autre la nuit. L'idée ici est de mettre en place une configuration de jour (avec les utilisateurs normaux) et une configuration de nuit (sans ces mêmes utilisateurs). Par utilisateurs normaux, on entend les utilisateurs présents pendant les heures d'ouverture de l'Office National des Pensions et qui utilisent les ressources réseaux et informatiques pour leur travail quotidien.

Cette double configuration aurait surtout pour but d'ouvrir des voies rapides et sans encombre pour les applications de fond (*backoffice*).

- **Solution envisagée**

L'élément déclencheur serait donc simplement le temps : la configuration de jour serait mise en place à un moment précis et remplacée par la configuration de nuit à un autre.
Cette configuration de nuit mettrait aussi en place un classement du trafic en flux distincts sur base des adresses IP source et destination, des ports source et destination et sur base des applications. A ces différents flux, s'ils sont concurrentiels, on pourrait aussi associer différentes priorités.

- **Mise en place du concept de politique**

1) **associer les rôles aux interfaces**

Les interfaces visées ici sont les interfaces des quatre SSR 8600 impliqués dans cette double configuration. On ne tiendra compte ici que des interfaces utilisées pour les applications de type "backoffice", car le trafic de nuit est principalement constitué par ces applications (figure 5-13).

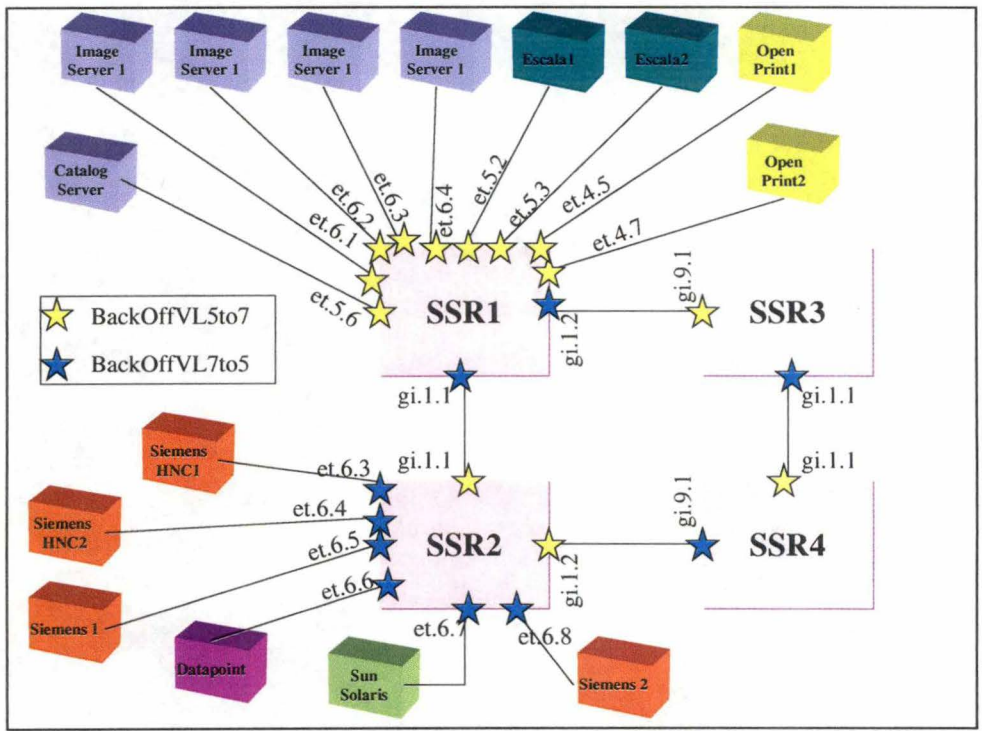

figure 5-13 : les interfaces visées par les politiques BackOffVL5to7 et BackOffVL7to5

SSR-B1-1

```
Interface gi.1.1 : role = BackOffVL7to5 (vers SSR2)
Interface gi.1.2 : role = BackOffVL7to5 (vers SSR3)
Interface et.4.5 : role = BackOffVL5to7 (Image OpenPrint 1)
Interface et.4.7 : role = BackOffVL5to7 (Image OpenPrint 2)
Interface et.5.2 : role = BackOffVL5to7 (Image Escala 1)
Interface et.5.3 : role = BackOffVL5to7 (Image Escala 2)
Interface et.5.6 : role = BackOffVL5to7 (Catalog Server)
Interface et.6.1 : role = BackOffVL5to7 (Image Server 1)
Interface et.6.2 : role = BackOffVL5to7 (Image Server 2)
Interface et.6.3 : role = BackOffVL5to7 (Image Server 3)
Interface et.6.4 : role = BackOffVL5to7 (Image Server 4)
```

SSR-B1-2

```
Interface gi.1.1 : role = BackOffVL5to7 (vers SSR1)
Interface gi.1.2 : role = BackOffVL5to7 (vers SSR4)
Interface et.6.3 : role = BackOffVL7to5 (Siemens HDC1)
Interface et.6.4 : role = BackOffVL7to5 (Siemens HDC2)
Interface et.6.5 : role = BackOffVL7to5 (Siemens)
Interface et.6.6 : role = BackOffVL7to5 (Datapoint via un Hub)
Interface et.6.7 : role = BackOffVL7to5 (Sun Solaris)
Interface et.6.8 : role = BackOffVL7to5 (Siemens)
```

SSR-19-3

```
Interface gi.1.1 : role = BackOffVL7to5 (vers SSR 4)
Interface gi.9.1 : role = BackOffVL5to7 (vers SSR 1)
```

SSR-19-4

```
Interface gi.1.1 : role = BackOffVL5to7 (vers SSR 3)
Interface gi.9.1 : role = BackOffVL7to5 (vers SSR 2)
```

2) Définir les politiques

`Policy 1` : le trafic en provenance des systèmes informatiques du VLAN5 à destination des systèmes informatiques du VLAN7 sera traité en priorité par les quatre routeurs SSR 8600 ;

`Policy 2` : le trafic en provenance des systèmes informatiques du VLAN7 à destination des systèmes informatiques du VLAN5 sera traité en priorité par les quatre routeurs SSR 8600 ;

3) Pseudo algorithme

`Policy 1 :`

```
VLAN5AdrSrc = {x.y.z.1, x.y.z.2, x.y.z.3, x.y.z.4,
               x.y.z.5, x.y.z.201, x.y.z.202, …}
VLAN7AdrDest : {r.s.1.4, r.s.1.5, r.s.1.30, …}

IF incomingPacketAdrSrc = VLAN5AdrSrc
AND incomingPacketAdrDest = VLAN7AdrDest
THEN put incoming packet into high priority queue
```

`Policy 2 :`

```
VLAN5AdrDest = {x.y.z.1, x.y.z.2, x.y.z.3, x.y.z.4,
                x.y.z.5, x.y.z.201, x.y.z.202, …}
VLAN7AdrSrc : {r.s.1.4, r.s.1.5, r.s.1.30, …}

IF incomingPacketAdrSrc = VLAN7AdrSrc
AND incomingPacketAdrDest = VLAN5AdrDest
THEN put incoming packet into high priority queue
```

4) Extrait de la configuration à mettre en place

La configuration à mettre en place sera faite en partie par la commande :

```
qos set ip <name> <priority> <scraddr/mask>|any
<dstaddr/mask>|any <srcport>|any <dstport>|any <tos>|any
<portlist>|<interface-list>|any <protocol>|any <tos-mask>|any
<tos-precedence-rewrite>|any <tos-rewrite>|any.
```

Celle-ci sera utilisée pour configurer les différentes interfaces des SSR 8600 auxquelles un des rôles a été attribué. De la même manière que pour le premier exemple, il est possible de favoriser les flux entre les VLAN 5 et 7 au moyen de la réservation de bande passante. Cette réservation se faisant aussi sur base d'ACL qui seraient constituées sur base des éléments définies dans les politiques.

5.4.4 Remarques.

On pourrait bien évidemment mettre en place un système de configuration dynamique du réseau qui reprendrait les trois idées précédentes. La mise en place d'un tel système ne serait pas simple et ne tiendrait compte que des performances et que de l'utilisation en temps réel du réseau.

La mise en place d'une configuration basée sur des politiques doit aussi pouvoir tenir compte des autres formes de gestion des réseaux de manière tout aussi pro active et dynamique. Si un problème est détecté (panne, taux erreurs élevés, ...), il est aussi nécessaire de réagir en modifiant, par exemple, les tables de routage des autres éléments actifs du réseau. Dans une autre mesure, il faut aussi tenir comte de la gestion des comptes et de la sécurité lors de la mise en place de configurations réseau.

La mise en place de politiques est ici facilitée par le fait que les différents éléments actifs du réseau concerné par les politiques étudiées proviennent du même constructeur et que les différents routeurs sont configurables de la même manière. Les commandes spécifiques à mettre en place sont donc identiques d'un routeur Cabletron à un autre routeur Cabletron quel que soit le modèle (du moins à l'ONP). Il aurait été plus complexe d'envisager la mise en place de politiques sur des éléments actifs du réseau provenant de différents constructeurs ou implémentant des QoS différentes. La mise en place de politiques dépend fortement du degré d'abstraction que l'on peut obtenir des différents éléments actifs du réseau.

Afin de simplifier la configuration à mettre en place au sein des SSR, on aurait pu utiliser des masques d'adresse pour les sous réseaux au lieu de cibler les machines une à une, mais cela n'aurait pas permis de cibler finement le trafic.

5.5 Conclusion

La mise en place de SNMP sur les différents systèmes et appareils était assez simple, mais vu le nombre de modifications de configuration à faire, cette manipulation fût assez rébarbative et a nécessité assez bien de temps. C'est surtout la centralisation de la réception des *trap* SNMP qui a permis de souligner la nécessité de mettre en place un système de gestion. La mise en place de RMON fût tout aussi laborieuse que celle de RMON, mais afin d'obtenir de meilleurs résultats, il aurait fallu personnaliser la configuration des sondes RMON par système ou appareil à gérer. Cette personnalisation aurait pu se faire sur base d'informations provenant de l'analyse du réseau, des protocoles les plus utilisés, etc.

Afin d'optimiser au mieux l'utilisation des ressources du réseau, il est d'abord nécessaire d'obtenir des informations à propos du trafic transitant en son sein. Ces informations peuvent être obtenues via différentes sources (journaux d'applications, analyseur de protocoles, etc.). La plus grande tâche à effectuer après la collecte des ces informations, a été d'en tirer des conclusions pertinentes. De plus, les renseignements obtenus ont permis de corriger certaines "anomalies" comme la présence d'hôtes générant du trafic IPX ou ayant des adresses IP non autorisées.

Les exemples de politiques imaginés pour illustrer le concept de gestion de réseaux via des politiques, montrent qu'il serait possible de mettre en place un tel type de gestion à l'Office National des Pensions. Ceci étant facilité, il est vrai, par l'homogénéité des capacités des appareils devant être gérés. Cette partie pratique de l'étude a mis aussi en évidence la complexité de la mise en œuvre de PBNM : il est nécessaire de bien connaître les capacités des appareils actifs du réseau, d'avoir une idée précise de l'utilisation des ressources du réseau, etc. On doit aussi pouvoir maîtriser le processus de traduction des politiques en ligne de commandes compréhensibles par le software de configuration.

La mise en place de PBNM ne serait en aucun concurrentielle à la mise en place de SNMP car PBNM pourrait utiliser les informations contenues dans les MIB (de SNMP, de RMON ou propriétaires) comme paramètres dans l'énoncé des politiques.

Conclusion

La gestion des réseaux n'a cessé d'être un problème proportionnel à leur taille et à leurs complexités. En effet depuis les premiers hôtes connectés de l'ARPANET jusqu'à l'Internet d'aujourd'hui en passant par lés réseaux d'entreprise, la gestion des réseaux a fortement évolué. Les solutions proposées, de ICMP à PBNM, de plus en plus utiles et efficaces, sont aussi basées sur des concepts de plus en plus complexes.

Mais il faut d'abord discerner les différentes formes de gestion qui se cachent derrière le nom générique de gestion des réseaux. Il existe communément cinq formes de gestion : la gestion des problèmes (situations anormales qui nécessitent une action ponctuelle), la gestion des comptes (pour facturation de l'utilisation des ressources réseau au sein de l'entreprise), la gestion de la configuration (démarrage et arrêt du réseau, mise à jour et maintenance des relations entre les composants du réseau, ...), la gestion des performances (surveillance et contrôle) et la gestion de la sécurité (protections et droits d'accès, autorisations et contrôles des accès, ...)

Le protocole SNMP est devenu rapidement, par sa popularité due à sa simplicité, le protocole de fait dans la gestion des réseaux. Ce protocole a été décliné en trois versions successives. SNMPv1, la première version de ce protocole, permet l'échange de données de gestion entre un gestionnaire et des agents situés sur les éléments du réseau à gérer. Les informations de gestion, décrites dans une MIB à partir de la structure SMI et d'ASN-1, sont échangées au moyen de trois opérations (*Get*, *GetNext* et *Set*) sur base d'une initiative du gestionnaire. La seule initiative permise à l'agent est l'envoi d'une alerte, au moyen de l'opération *Trap*, vers le gestionnaire dés qu'un événement prédéfini survient. L'accès en lecture ou en écriture aux informations de gestion est accordé sur base d'un nom de communauté SNMP.

Cette première version de SNMP a pourtant quelques lacunes : elle ne convient pas pour la gestion de grands réseaux, n'est pas conçue pour récupérer un grand nombre de données, les avis d'alertes ne sont pas confirmés, l'authentification est insuffisante, les gestionnaires ne savent pas communiquer entre eux, etc. La version 2 de SNMP va tenter de remédier à ces différentes lacunes. En effet, l'ajout de deux nouvelles opérations (GetBulk et Inform) vont permettre de combler les lacunes liées à la gestion de grands réseaux et à la communication entre gestionnaires. Néanmoins cette deuxième version, après plusieurs tentatives, n'a pas permis de résoudre les problèmes liés à l'authentification et à l'accès aux données. C'est donc sur base du nom de communauté que SNMPv2 régule les accès aux données. Il faut aussi noter que la coexistence entre SNMPv1 et SNMPv2 est assurée, soit par un agent de proximité, soit par un gestionnaire bilingue. La dénomination de cette version de SNMP correspondant à la situation actuelle est SNMPv2c.

Ce sont plusieurs RFC qui définissent une nouvelle architecture. Cette architecture doit satisfaire aux implémentations existantes de SNMPv1 et de SNMPv2c, doit fournir un cadre de travail pour la troisième version de SNMP et les révisions futures apportées par les améliorations des éléments composant SNMP. Cette architecture définit une entité SNMP comme un élément composé d'applications et d'un moteur SNMP. Cette entité SNMP, en fonction des applications qui la compose, joue, soit le rôle d'un gestionnaire, soit le rôle d'un agent SNMP traditionnel. Le moteur SNMPv3 est composé de sous-systèmes ayant un rôle bien défini : le *dispatcher* est un gestionnaire de trafic, le *message Processing Subsystem* traite les messages entrants et sortants afin d'y inclure ou d'en extraire le PDU, le *security Subsystem* contient les fonctions d'authentification et de chiffrement et l'*Access Control Subsystem* contient les services de contrôle d'accès.

Les applications sont au nombre de cinq et ont aussi des fonctions bien précises : le *Command Generator* génère les requêtes, le *command Responder* y répond, le *notification Originator* surveille le système pour détecter les événements ou les conditions particulières pour envoyer des PDU *trap* ou *Inform*, le *notification Receiver* écoute les messages de notification et génère une réponse dans le cas d'un PDU *Inform* et le *proxy Forwarder* transmet les messages SNMP.

Le point fort de SNMPv3 se situe dans la définition de deux aspects relatifs à la sécurité. Le modèle USM fournit les services d'authentification et ainsi que la garantie de l'intimité des informations (par chiffrement) et se situe au niveau du message. Le modèle VACM détermine quel type d'accès est accordé à l'initiateur d'une requête pour atteindre un objet particulier et y exécuter une action précise (cela se situe au niveau du PDU).

Bien que les versions successives de SNMP apportent des améliorations à ce protocole de gestion, il semble que la version la plus répandue soit SNMPv1. Les constructeurs ne semblent pas du tout enclin à implémenter SNMPv2 et SNMPv3 : il se tournent actuellement vers la gestion des réseaux basée sur des politiques.

L'extension la plus significative apportée à SNMP vient des spécifications concernant la surveillance d'un réseau distant. Ces spécifications sont reprises dans plusieurs RFC qui définissent une MIB de surveillance à distance. Il s'agit de RMON (Remote Network Monitoring) qui se décline en deux versions. La première version de RMON permet de stocker, dans la MIB associée, des informations concernant le trafic transitant au travers de l'appareil contenant une sonde RMON. Ces informations sont stockées localement dans différentes tables et permettent d'établir différentes statistiques sur l'utilisation des ressources du réseau. Cette version de RMON décode le trafic au niveau de la couche liaison de données (2).

La deuxième version de RMON (RMON2) fonctionne sur les mêmes principes que RMON, mais décode les paquets des couches réseau (3), transport (4) et application (7). RMON2 étend donc les possibilités de la MIB RMON d'origine pour y inclure les protocoles supérieurs à la couche MAC. L'ajout des protocoles de la couche réseau, comme IP, permet à la sonde de surveiller le trafic entre routeurs connectés au réseau local. La sonde peut alors surveiller les sources et les destinations du trafic hors réseau local arrivant ou sortant via un routeur. L'ajout de protocoles des couches supérieures, comme le couche application, permet à la sonde de fournir des statistiques détaillées sur le trafic au niveau des applications.

Du point du vue pratique, la mise en place de SNMP, sur les différents systèmes et appareils à gérer au sein de l'ONP, fût assez simple mais a nécessité beaucoup de temps étant donné le nombre assez élevé de configurations à modifier. De plus, il faut aussi remarquer que modifier les configurations une par une est une opération assez rébarbative du fait de la répétition des mêmes commandes ou des actions à entreprendre sur chaque élément du réseau. La mise en place de RMON s'est faite de la même manière que pour SNMP et avec les mêmes désavantages. Une différence entre la mise en place de SNMP et de RMON est que pour SNMP, il ne faut configurer qu'un agent par éléments à gérer, et que pour RMON, il faut configurer la sonde pour chacune des interfaces à propos ou à partir desquelles on veut obtenir des informations.

Les concepts qui modélisent la gestion des réseaux via des politiques sont des concepts récents ou en cours de standardisation. Néanmoins, la définition de politiques se fait à différents niveaux. Le niveau réseau (le plus abstrait) énonce des politiques de gestion de haut niveau à propos de la topologie, de la connectivité, des objectifs de performance, et de l'état dynamique du réseau. Le niveau suivant subdivise cette politique générale en politiques des nœuds (ou injonctions atomiques) en fonction des objectifs et des spécificités des appareils du réseau. Au niveau inférieur, le plus abstrait, on retrouve une traduction des injonctions en instructions compréhensibles par les appareils du réseau.

Il existe différentes architectures permettant la gestion des réseaux via des politiques. Le modèle *three-tiers* est composé d'un PEP capable d'actions comme le filtrage et le marquage des paquets, d'un PDP responsable de la détermination des actions à entreprendre après interprétation des règles de politique et d'un *repository* où sont stockées les politiques. Le modèle *two-tiers* est identique au modèle *three-tiers* mais le PDP et le PEP sont combinés en une seule entité. Le modèle multidomaine est utilisé par les ISP pour garantir les SLA qu'ils ont contractés entre eux. L'IETF a, lui aussi, fourni une architecture permettant la gestion du réseau via des politiques. Cette dernière architecture se base sur les protocoles existants comme SNMP (de nouvelles MIB et de nouvelles relations entre le gestionnaire et l'agent ont été créées), alors que les autres architectures proposent de se baser sur de nouveaux protocoles comme COPS et LDAP.

L'IETF a défini un modèle de stockage des données, le *Core Policy Schema* , qui est un schéma d'annuaire. A l'aide de ce schéma, les applications de gestion peuvent rechercher les conditions et les actions associées aux politiques. Les opérandes des expressions de politiques doivent pouvoir être non-ambigue, c'est pourquoi, il est nécessaire de modéliser différents concepts comme les utilisateurs, les appareils physiques, les services, etc. Pour la DMTF, ces modèles sont connus sous le nom de *Common Information Model*. Cette modélisation par la DMTF est connue comme l'approche *bottom-up* de cette problématique. En ce qui concerne l'approche *top-down*, l'IETF propose l'utilisation de la *Policy Information Base* (PIB). Cette PIB permet de spécifier les politiques en terme de fonctionnalités de l'interface.

Pour mettre en place une gestion du réseau, il existe deux modèles. Le plus connu et le plus ancien est celui utilisé par SNMP : c'est le gestionnaire qui débute les actions de configuration (*push*). Ce modèle nécessite une forte implication des administrateurs dans les opérations de reconfiguration et de vérification. Le second modèle, plus récent, se base sur les protocoles COPS et LDAP. Les informations de configuration sont centralisées dans un annuaire et ce sont les appareils eux-mêmes qui vont rechercher leur configuration. C'est ce principe qui est utilisé par le PDP et le PEP dans les architectures *three-tiers* et *two-tiers*. Il est aussi possible de mettre en place un système mixte.

Les annuaires sont surtout utilisés pour stocker des informations hiérarchisées qui sont plus souvent lues que modifiées. Cela convient donc parfaitement pour des listes téléphoniques, mais pas pour des informations qui sont souvent modifiées. Comme les annuaires ne sont pas pourvus de mécanismes permettant de

renforcer l'intégrité des données qui y sont stockées, les applications accédant aux données doivent donc contenir ce genre de mécanismes. Il n'y a pas non plus de mécanismes permettant d'avertir les applications des changements survenus dans l'annuaire. C'est pour toutes ces raisons que les annuaires et LDAP sont un mauvais choix de stockage de données malgré l'engouement des constructeurs pour ce genre d'infrastructure.

Les exemples de politiques, que l'on retrouve dans la partie du mémoire consacrée à l'étude pratique, ne représentent qu'une approche ou une réflexion sur les possibilités de mettre en place une gestion du réseau de l'ONP via des politiques en terme de qualité de service.

Mais avant de mettre en place un gestion du réseau, il faut d'abord maîtriser différents aspects importants concernant l'environnement que l'on veut pouvoir gérer. En effet, il est nécessaire de déterminer, avec précision, ce que l'on va gérer et ce que l'on ne va pas gérer. Il faut aussi savoir comment on va le gérer et pourquoi. Les connaissances à détenir sont d'abord celles qui font référence au réseau lui-même, c'est à dire sa topologie, les caractéristiques et les capacités des appareils le composant, les protocoles et l'adressage autorisés, etc. Ensuite il est utile, sur base d'informations diverses provenant de sources comme les journaux d'applications et l'analyseur de protocoles, d'obtenir une vision plus précise de l'utilisation des ressources du réseau. Il faut aussi tenir compte des développements futurs afin d'avoir une vue à plus long terme de la gestion du réseau.

La collecte de toutes ces informations est assez difficile car cela nécessite du temps et donc des ressources humaines. Pour beaucoup de personnes, la gestion du réseau, comme le réseau lui-même, est une chose obscure et donc bien souvent mal comprise ou interprétée.

Si on tient compte du nombre de paramètres pouvant se trouver dans une configuration et du nombre de configurations à mettre en place et à maintenir, il est assez aisé de se rendre compte que ce n'est pas avec les quatre opérations de base de SNMPv1 que l'on va pouvoir gérer efficacement le réseau dans sa globalité et les appareils (individuellement) qui le composent. Néanmoins, certains aspects de SNMP, comme la génération de *trap* et la MIB RMON, peuvent servir à établir une situation en temps réel du réseau.

Par rapport à SNMP, où les configurations sont statiques, PBNM permet d'avoir des configurations dynamiques et proactives. En effet, on peut déterminer des politiques qui modifient la configuration du réseau sur base d'informations sur l'état actuel du réseau et sur l'état probable de celui-ci. Cette probabilité étant obtenue à partir de situations vécues ou théoriques.

SNMP et PBNM ne sont pas à considérer comme des frères ennemis. Les conditions des politiques peuvent faire référence à des éléments contenus dans des MIB comme RMON (accessibles via SNMP), et les actions associés aux conditions peuvent aussi modifier des paramètres contenus dans une MIB (via SNMP).

L'utilisation de PBNM sur le réseau de l'ONP est, par contre, envisageable. Comme le matériel qui compose le réseau est plus ou moins homogène en terme de fonctionnalités , il est assez facile d'obtenir un niveau d'abstraction suffisant des capacités communes des différents appareils. PBNM pourrait résoudre le problème du peu de ressources humaines allouées à la gestion du réseau. La centralisation des configurations, la définition de politiques générales vont dans le sens d'une gestion plus aisée.

SNMP restera-t-il le protocole de fait ? PBNM sera-t-il le concept de gestion du future ? Ce sont des questions qu'il est légitime de se poser. Bien que SNMP soit fortement implémenté, différents aspects ne plaident pas en sa faveur : les constructeurs n'ont pas implémenté les dernières versions de SNMP, il existe au sein de l'IETF un groupe de travail chargé de l'avenir de SNMP, une adaptation de SNMP (par exemple, les MIB) doit être faite pour tenir compte des capacités actuelles des réseaux, ... De son côté PBNM, bien qu'étant un concept récent et en plein développement, semble être la piste à suivre. Il faudra tout de même être attentif car il y a encore beaucoup de groupes de travail au sein de l'IETF qui réfléchissent encore à la gestion des réseaux via des politiques (il y a beaucoup de documents de type *Internet-Darft* à ce sujet), et aussi car les constructeurs ont tendance à proposer une solution propre à leur matériel.

Glossaire

Agent
Un software qui fournit des informations sur un appareil spécifique ou un processus. Un agent SNMP fournit des informations de gestion en utilisant le protocole SNMP.

Alarme
Un signal utilisé pour indiquer l'occurrence d'un problème associé à un appareil ou à un processus.

ARPANET
Advanced Research Projects Agency Network. Un prototype de réseau à commutation de paquets, sur une grande échelle, sponsorisé par le département de la défense américain.

Authentification
Traitement qui permet de déterminer si l'information reçue a comme origine une source reconnue et autorisée.

Command Line Interface (CLI)
Un moyen d'accéder, pour un système de gestion, à la configuration d'éléments du réseau. Ce moyen implique une interface textuelle de type question/réponse via un transport de type terminal. Il n'existe pas de standardisation des CLI qui sont invariablement spécifiques aux constructeurs.

DIAMETER
Le protocole DIAMETER est un protocole (proposition) qui peut être utilisé pour les politiques, l'authentification, l'autorisation et la comptabilité, et le contrôle des ressources. Ce protocole coexiste avec RADIUS.

Differentiated Services (DiffServ)
Un moyen de définir les spécifications de transmission des paquets dans une entreprise ou inter entreprise dans le but de réaliser des niveaux de services ou d'autres spécifications de trafic. Développé par l'IETF, DiffServ est réalisé sur base des champs de l'entête IP et sur base de contrats entre les différents nœuds du réseau coopérants au traitement de ces paquets.

Entité réseau
Un nœud réseau ou un hôte qui supporte la gestion à distance, via un réseau et utilisant un agent de gestion.

Hôte
Dans un réseau TCP/IP, tous les nœuds ne font pas transiter les paquets vers un autre réseau (les routeurs, gateways, bridge, ...). Un hôte supporte typiquement toutes les couches du protocole TCP/IP, jusqu'à la couche application. Les routeurs ne supportent le protocole TCP/IP que jusqu'à la couche réseau.

Information de gestion
Données utilisées pour déterminer l'identité, l'état, la situation et l'historique d'une entité réseau.

Integrated Services (IntServ)
L'IETF *Integrated Services Working Group* développe une série de standards dans le but de déterminer comment les services d'application définissent leurs exigences de QoS, comment cette information est rendue disponible au routeur de saut en saut et les façons de tester et de valider que les contrats QoS sont maintenus. Avec l'approche IntServ, chaque élément du réseau est nécessaire pour identifier le jeu coordonné des capacités de contrôle des QoS (fourni en termes de fonctions qu'ils remplissent, d'informations qu'ils ont besoin et d'informations à exporter). Les routeurs ayant des capacités IntServ doivent classer les paquets sur base d'un certain nombre de champs et maintenir cette information pour chaque flux individuellement.

Mécanisme
Une opération ou un algorithme spécifique qui est implémenté dans un noeud dans le but de réaliser un ou plusieurs *Per-Hop-Behaviour.*

Nœud
Point de connexion ou terminal d'un ou de plusieurs liens de communication réseau.

Paquet
Un bloc de données qui comporte les informations nécessaires pour qu'il soit délivré sur un réseau à commutation.

Per-Hop-Behaviour (PHB)
Le traitement de transit donné à une classe spécifique de trafic, basé sur un critère défini dans le champs DiffServ. Les routeurs et les switch utilisent PHB pour déterminer les priorités de services de différents flux de trafic.

Polling
L'action d'interroger un nœud ou un appareil à intervalle régulier et périodique dans le but de collecter des données de gestion. Le *polling* est une fonction de base exécutée par les systèmes de gestion et une opération de base dans la plupart des modèles de gestion des réseaux.

Problème
Différence entre ce qui est désiré et ce qui est perçu.

Protocol Data Unit (PDU)
Un terme générique pour un message utilisé par un protocole de communication ou une couche spécifique du modèle OSI. Un PDU contient les informations utiles pour identifier et authentifier les données qu'il a encapsulé.

Proxy
Un processus ou un appareil qui agit en lieu et place d'un autre processus ou appareil. En réseau, un proxy permet au appareil non-réseau d'accéder (ou d'être accéder par) le réseau.

RADIUS
Protocole servant à transporter des informations d'authentification, d'autorisation et de configuration entre un *Network Access Server* (NAS) et un serveur d'authentification. Les questions et les réponses qui sont transportées par le protocole sont exprimées en terme d'attributs RADIUS comme le *User-Name*, le type de service, etc. Ces attributs fournissent les informations nécessaires au serveur RADIUS pour authentifier les utilisateurs et pour établir les services réseaux autorisés pour eux.

Request for Comments (RFC)
Les spécifications officielles des standards Internet. Tous les RFC sont crées, examinés et mis à jour par l'IAB et l'IETF. Tous les RFC sont disponibles gratuitement sur Internet.

Shaping
Retardement des paquets des flux ne correspondant pas aux règles de priorités des flux dans un routeur.

Station des gestion
Un ordinateur ou une station de travail utilisée pour exécuter les programmes de gestion.

Abréviations

AF - Assured Forwarding
APDU - Application Protocol Data Unit
ASN.1 - Abstract Syntax Notation number One
BDC - Backup Domain Controller
BER - Basic Enconding Rules.
BGP - Border Gateway Protocol
CIM - Common Information Model
CLI - Command Line Interface
CMIP - Common Management Information protocol
CMOT - CMIP over TCP/IP
COPS - Common Open Policy Service
CPS - Core Policy Schema
DHCP - Dynamic Host Configuration Protocol
DMTF - Distributed Management Task Force
DNS - Domain Name System
DoD - Departement of Defense
EF - Expedited Forwarding
EGP - External Gateway Protocol
FTP - File Transfert Protocol
HEMS - High-Level Entity Management System
HMP - Host Monitoring Protocol
IAB - Internet Architecture Board
ICMP - Internet Control Message Protocol
IETF - Internet Engineering Task Force
IESG - Internet Engineering Steering Group
IP - Internet Protocol
IRFT - Internet Research Task Force
ITU-T - International Telecommunication Union - Telecommunication standardization sector
ISO - International Organization for Standardization
ISP - Internet Service Provider
LAN - Local Area Network
LDAP - Lightweight Directory Access Protocol
MAC - Media Access Control
MIB - Management Information Base
OSI - Open Systems Interconnection
PDP - Policy Decision Point
PEP - Policy Enforcemnt Point
PER - Packet Enconding Rules
PDC - Primary Domain Controller
PDU - Protocol Data Unit
PIB - Policy Information Base
PING - Packet Internet Groper
PNM - Policy-based Network Management
PPDU - Presentation Protocol Data Unit
PRC - Provisioning Classe
QoS - Quality of Service
RAP - Ressource Allocation Protocol
RFC - Request For Comment
RMON - Remote Network Monitoring
RSVP - Resource Reservation Protocol
SGMP - Simple Gateway Management Protocol
SLA - Service Level Agreement
SMI - Structure of Management Information
SNMP - Simple Network Mangement Protocol
SPDU - Session Protocol Data Unit
TCA - Traffic Control Agreement
TCP - Transport Control Protocol
TFTP - Trivial File Transfer Protocol
TMN - Telecommunications Management Network

TPDU - Transport Protocol Data Unit
USM - User-based Security Model
VACM - View-based Access Control Model
VLAN - Virtual Local Area Network
VPN - Virtual Private Network
WAN - Wide Area Network
WINS - Windows Internet Name Service

Références

Les ouvrages et les articles

[bor] S. Boros, "Policy-Based Network Management With SNMP", Computer Science Departement, University of Twente (Netherlands).

[ID-cndwsnmp2001] Internet-Draft, "Configuring Networks and Devices With SNMP", draft-ietf-snmpconf-bcp-04.txt, mars 2001, (*Work in progress*)
[ID-dspmib2001] Internet-Draft, "The DiffServ Policy MIB", draft-ietf-snmpconf-diffpolicy-04.txt, mars 2001, (*Work in progress*)
[ID-dsqospib2001] Internet-Draft, "Differentiated Services Quality of Service Policy Information Base", draft-ietf-diffserc-pib-04.txt, juillet 2001, (*Work in progress*)
[ID-fpib2001] Internet-Draft, "Framework Policy Information Base", draft-ietf-rap-frameworkpib-04.txt, mars 2001, (*Work in progress*)
[ID-iptepib2001] Internet-Draft, "An IP Traffic Engineering Policy Information Base", draft-jacquenet-ip-te-00.txt, juin 2001, (*Work in progress*)
[ID-pbmmib2001] Internet-Draft, "Policy based Management MIB", draft-ietf-snmpconf-pm-05.txt, mars 2001, (*Work in progress*)
[ID-pt2001] Internet-Draft, "Policy Terminology", draft-ietf-policy-terminology-04.txt, novembre 2001, (*Work in progress*)
[ID-rpms2000] Internet-Draft, "Requirements for a Policy Management System", draft-ietf-policy-req-02.txt, novembre 2000, (*Work in progress*)

[jud2001] Michael Jude, "Policy-Based Management : Beyond The Hype", Business Communication Review, Mars 2001, 52-56.

[joh1999] Vicki Johnson, "Quality of Services – Glossary of Terms", Stardust Technologies, (www.qosforum.com) 1999

[lew2000] Lundy Lewis, "Policy-based Configuration Management : A Perspective from a Network Management vendor", *The Simple Times*, Septembre 2000, Volume 8 n°1

[mur1998] James D. Murray, *Windows Nt SNMP*, O'Reilly ™, Sebastopol, janvier 1998

[pra1995] Aiko Prass, "Network Management Architectures", Center for Telematics and Information Technology, University of Twente, Thesis N° 95-02, 1995

[puj1997] Guy Pujolle, *Les Réseaux (2ème édition revue et augmentée)*, Editions Eyrolles, Paris, 1997

[raj1999] Raju Rajan, Dinesh Verma, Sanjay Kamat, Eyal Felstaine, Shai Herzog, "A Policy Framework for Integrated and Differentiated Services in the Internet", *IEEE Network*, Septembre/Octobre 1999, 36-41

[RFC0792] RFC 0792, "Internet Control Message Protocol", septembre 1981
[RFC1157] RFC 1157, "Simple Network Management Protocol (SNMP)", mai 1990
[RFC1156] RFC 1156, "Management Information Base for network management of TCP/IP-based internets", mai 1990
[RFC1213] RFC 1213, "Management Information Base for Network Management of TCP/IP-based internets:MIB-II", mars 1991
[RFC1354] RFC 1354, "IP Forwarding Table MIB", juillet 1992
[RFC1398] RFC 1398, "Definitions of Managed Objects for the Ethernet-Like Interface Types", janvier 1993
[RFC1493] RFC 1493, "Definitions of Managed Objects for Bridges", juillet 1993
[RFC1513] RFC 1493, "Definitions of Managed Objects for Bridges", juillet 1993
[RFC1757] RFC 1757, "Remote Network Monitoring Management Information Base", février 1995
[RFC2011] RFC 2011, "SNMPv2 Management Information Base for the Internet Protocol using SMIv2", novembre 1996
[RFC2021] RFC 2021, "Remote Network Monitoring Management Information Base Version 2", janvier 1997
[RFC2074] RFC 2074, "Remote Network Monitoring MIB Protocol Identifiers", janvier 1997
[RFC2571] RFC 2571, "An Architecture for Describing SNMP Management Frameworks", avril 1999
[RFC2572] RFC 2572, "Message Processing and Dispatching for the Simple Network", avril 1999
[RFC2573] RFC 2573, "SNMP Applications", avril 1999

[RFC2574] RFC 2574, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", avril 1999
[RFC2575] RFC 2575, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)", avril 1999
[RFC2576] RFC 2576, "Coexistence between Version 1, Version 2, and version 3 of the Internet Standard Network Management Framework", mars 2000
[RFC2748] RFC 2748, "The COPS (Common Open Policy Service) Protocol", janvier 2000

[ssrgsg1] Cabletron, "SSR 8000/8600 Getting Started Guide : Features Overview" (ch 1, p.19-50)
[ssrurm1] Cabletron, "SmartSwitch Router User Reference Manuel : SSR Product Overview (ch 1; p.29-48)
[ssrurm17] Cabletron, "SmartSwitch Router User Reference Manuel : Access Control List Configuration Guide (ch 17; p.255-269)
[ssrurm19] Cabletron, "SmartSwitch Router User Reference Manuel : QoS Configuration Guide (ch 19; p.283-294)
[ssrurm21] Cabletron, "SmartSwitch Router User Reference Manuel : RMON Configuration Guide (ch 21; p.299-314)
[ssrclirm48] Cabletron, "SSR Command Line Interface Reference Manuel : RMON Commands (ch 48; p.675-774)
[ssrclirm53] Cabletron, "SSR Command Line Interface Reference Manuel : SNMP Commands (ch 53; p.799-808)

[sta1999] William Stallings, *SNMP, SNMPv2, SNMPv3 and RMON1 and 2 (thhird edition)*, Addison Wesley, Reading (Massachusetts),1999

[ste1999] Mark L. Stevens and WalterJ. Weiss, "Policy-Based Management for IP Networks", Bell Labs technical Journal, Octobre-Décembre 1999, 75-94.

[tan1997] Andrew Tannenbaum, *Réseaux (3ème édition)*, InterEditions, Paris, 1997

[wil2000] Ed Wilson, *Network Monitoring and Analysis (a protocol approach to troubleshooting)*, Prentice Hall, Upper Saddle River (New Jersey), 2000

[WP-iqp1999] White paper,"Introduction to Qos Policies", Stardust Technologies, (www.qosforum.com) 1999
[WP-pna] White paper,"Policy-based Network Architecture", Allot Communications, (www.allot.com)
[WP-eanm] White paper,"l'Ethernet atteint une nouvelle maturité", Network&Telecom, été 2001, 34-38

Les liens internet

Bell Laboratories : http://www.bell-labs.com

Cabletron : http://www.cabletron.com

Centre National de le Recherche Scientifique (CNRS) : http://www.urec.cnrs.fr

Cisco : http://www.cisco.com

Distributed Management Task Force (DMTF) : http://www.dmtf.org

ITU : http://www.itu.ch

ISO : http://www.iso.ch

Internet Engineering Task Force (IETF) : http://www.ietf.org
Internet Research Task Force (IRTF) : http://www.irtf.org

Office National des Pensions : http://www.onprvp.fgov.be

QoSForum : http://www.qosforum.com

Simple Times : http://www.simple-times.org
Simple Web : http://www.simpleweb.org

University of Twente (Netherlands) : http://www.utwente.nl

Annexes

Plusieurs des annexes ci-dessous ne sont pas nécessaires pour les personnes qui ont déjà de bonnes connaissances sur les réseaux et sur la suite de protocole TCP/IP. Néanmoins, ce mémoire ayant une double destination (FUNDP et ONP), il s'est avérer nécessaire de les garder dans ce document.

Annexe A : Le modèle de référence OSI

Le modèle de référence OSI, fondé sur une proposition élaborée par l'ISO, traite de la connexion entre systèmes ouverts à la communication avec d'autres systèmes.

Le modèle OSI est composé de sept couches (figure A.1) et décrit simplement ce que chaque couche doit faire. Cependant l'ISO a également produit des normes pour toutes les couches.

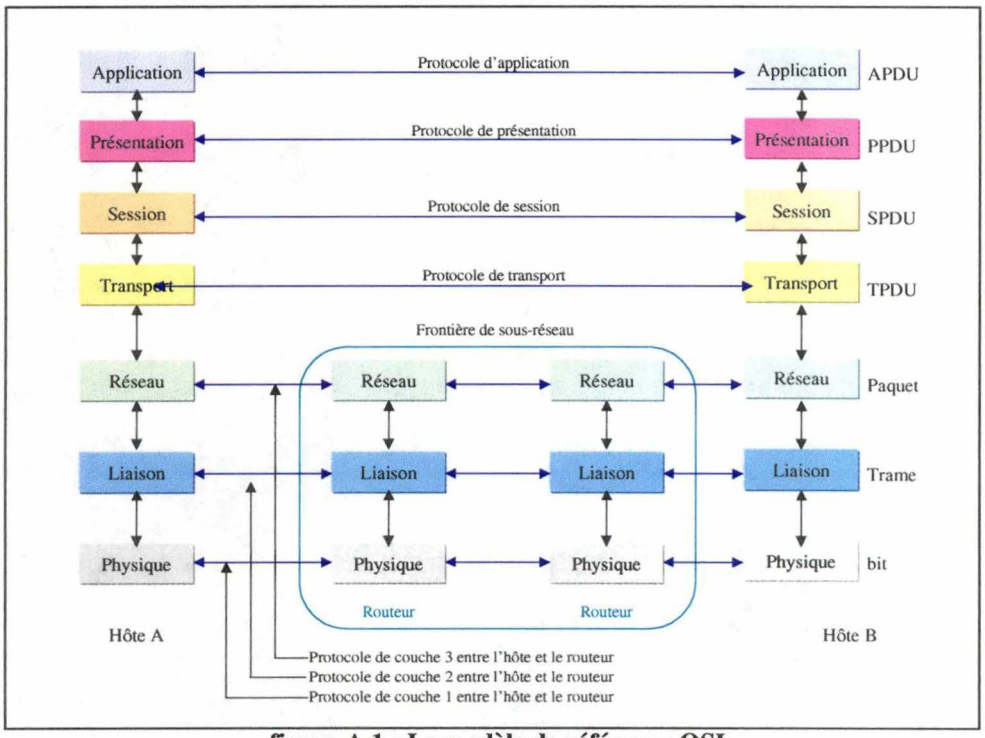

figure A.1 : Le modèle de référence OSI

La **couche physique** s'occupe de la transmission des bits de façon brute sur un canal de communication. Le nombre de volts nécessaires pour représenter un 1 ou un 0, la durée de transmission d'un bit, le(s) sens de la transmission, l'initialisation et la clôture de la connexion, le nombre de broches des connecteurs réseau, ... sont des exemples des éléments pris en compte à ce niveau afin de s'assurer que la réception d'un bit à 1 correspond à l'émission d'un bit à 1

Prendre un moyen de transmission brut et le transformer en une liaison qui paraît exempte d'erreurs de transmission (pour la couche réseau) est le rôle essentiel de la **couche liaison de données**. Ce rôle est accompli en fractionnant les données de l'émetteur en trames de données, en les transmettant en séquence et en gérant les trames d'acquittement renvoyées par le récepteur. La couche liaison de données doit donc aussi être capable de reconnaître les limites des trames. Les problèmes posés par les trames endommagées, perdues ou dupliquées seront résolus par la couche liaison de données. Une autre fonction lui est encore attribuée : empêcher un émetteur rapide de saturer un récepteur lent (mécanisme de régulation du trafic).

La façon dont les paquets sont acheminés de la source au destinataire constitue l'élément clé de la **couche réseau** qui permet de gérer le sous réseau. Les routes peuvent être fondées sur des tables statiques (déterminées ou non au début de chaque conversation) ou dynamiques (recalculées pour chaque paquet en fonction, par exemple, de la charge du réseau). La couche réseau est aussi responsable du contrôle de la congestion. On peut aussi ajouter une fonction de comptabilité à cette couche, ce qui permet la facturation de services par les opérateurs. Le passage des paquets d'un sous réseau à un autre peut engendrer différents problèmes car, par exemple, l'adressage peut être différent, les paquets trop volumineux, les protocoles incompatibles entre eux, ...

La fonction de base de la **couche transport** est d'accepter des données de la couche session, de les découper, si nécessaire, en plus petites unités, de les passer à la couche réseau et de s'assurer que tous les morceaux arrivent correctement de l'autre côté. De plus, tout cela doit être fait de façon efficace et en préservant les couches supérieures des inévitables évolutions technologiques du matériel. La couche transport détermine, entre autres, le type de services à fournir à la couche session et aux utilisateurs du réseau (connexion point à point ou non, sans erreurs, dans l'ordre d'émission ou non, ...). Le type de services est déterminé à l'établissement de la connexion. La couche transport gère aussi l'établissement et la clôture des connexions sur le réseau. Il est donc nécessaire d'avoir un mécanisme d'adressage permettant de cibler le destinataire. Un autre mécanisme important permet de contrôler le flux afin qu'un émetteur rapide ne sature un récepteur plus lent.

La **couche session** permet à des utilisateurs travaillant sur différentes machines d'établir des sessions entre eux, un peu comme la couche transport mais en fournissant des services supplémentaires. La gestion du dialogue, un de ces services, symbolisée par la gestion d'un jeton dans certains protocoles, permet d'établir les règles de dialogue. Un autre service de la couche session est la synchronisation qui à l'aide de points de reprise, permet, par exemple, de reprendre un transfert de données interrompu par une panne.

A la différence des autres couches qui sont concernées par la transmission fiable des bits d'un point à un autre, la **couche présentation** s'intéresse à la syntaxe et à la sémantique de l'information transmise. Elle permet, entre autres, de convertir la représentation des données internes vers la norme du réseau.

La **couche application** comporte de nombreux protocoles fréquemment utilisés. Le transfert de fichiers est un exemple permettant d'illustrer le fonctionnement de cette couche : les différents systèmes de fichiers ont leurs propres conventions (dénomination, représentation des lignes de textes, etc. ...). Donc, le transfert de fichiers entre différents systèmes requiert la gestion de ces incompatibilités qui est effectuée au niveau de la couche application.

Afin que la transmission de données au travers du modèle OSI se fasse correctement, il faut que le processus émetteur fournissent certaines données vers le processus récepteur. De ce fait, chaque couche transmet les données reçues de la couche supérieure à la couche inférieure en y ajoutant une en-tête qui lui est propre (et qui peut être nulle). Cette transmission de données, visualisée dans la figure A.2, se répète jusqu'à la couche physique. Le processus inverse se passe sur la machine réceptrice, qui enlève les en-têtes lorsque l'information remonte d'une couche à l'autre.

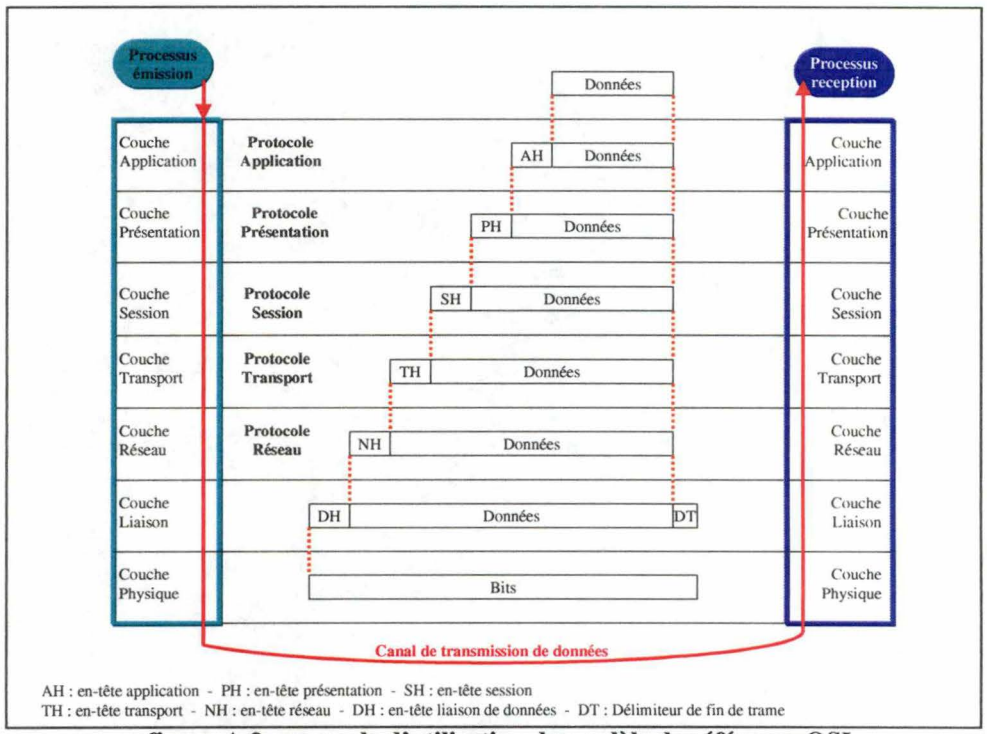

figure A.2 : exemple d'utilisation du modèle de référence OSI

La clé de ce concept réside dans le fait que chaque couche est programmée comme si elle était réellement horizontale, alors qu'en fait la transmission de données est verticale. (figure A.1)

Annexe B : Le protocole TCP/IP

B.1 le modèle TCP/IP

Le réseau ARPANET était un réseau de recherche patronné par le ministère américain de la défense (DoD), reliant plusieurs centaines d'universités et de sites administratifs utilisant des lignes téléphoniques louées. Depuis le démarrage de ce réseau, on cherchait surtout à relier des réseaux très divers de la façon la plus transparente possible. Cette architecture finit par être connue sous l'appellation de **Modèle de référence TCP/IP** du nom de ses deux principaux protocoles.

Le DoD voulant que les connexions restent intactes du moment que l'ordinateur source et l'ordinateur destination fonctionnaient, on a porté la plus grande attention à la disponibilité du réseau. De plus, on voulait disposer d'une architecture très souple permettant d'utiliser des applications aussi différentes que le transfert de fichiers et la transmission de la parole en temps réel.

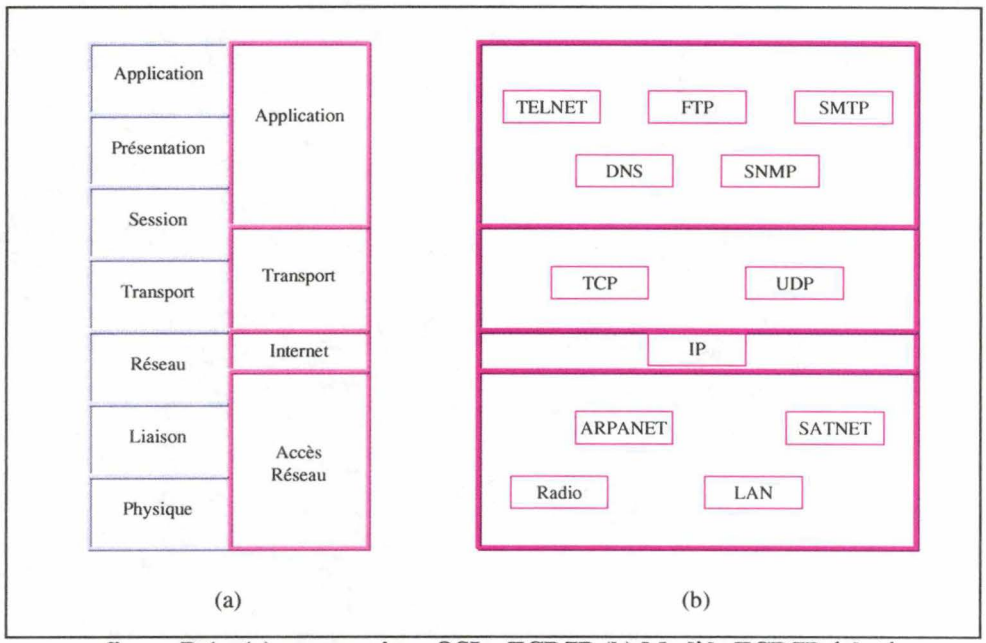

figure B.1 : (a) comparaison OSI – TCP/IP (b) Modèle TCP/IP éclaté

Le modèle de référence TCP/IP est composé de quatre couches (Application, Transport, Internet et Accès Réseau) qui reprennent les fonctionnalités des sept couches du modèles OSI (figure B.1 (a)).

La **couche accès réseau** est utilisée pour échanger des données entre un système final et le réseau auquel il est relié. L'émetteur doit fournir au réseau l'adresse de la destination , ainsi le réseau pourra acheminer les données jusqu'à destination. Le software utilisé à ce niveau dépend beaucoup du type de réseau utilisé (différents standards ont été développés pour les réseaux à commutation de paquets ou de circuits, les réseaux locaux (e. a. Ethernet), etc.

Les spécifications du modèle TCP/IP ont conduit à choisir un réseau à commutation de paquets fondé sur une couche d'interconnexion de réseaux sans connexion. Le rôle de la **couche Internet** est de permettre l'injection de paquets dans n'importe quel réseau et de les acheminer indépendamment les uns des autres jusqu'à destination. Les couches supérieures se chargeront de réordonner les paquets arrivant en désordre. La couche Internet définit un format officiel de paquets et un protocole qu'on appelle IP (*Internet Protocol*). Le routage des paquets et le fait d'éviter les congestions sont des point cruciaux de cette couche comme ils le sont pour la couche réseau du modèle de référence OSI.

Permettre à des entités paires sur des ordinateurs source et destination de soutenir une conversation est le rôle de la **couche transport** (identique au modèle OSI). Deux protocoles de bout en bout ont été définis. Le premier, TCP, est un protocole fiable orienté connexion qui fragmente le flux entrant en message qu'il passe à la couche Internet et qui sont réassemblés à la destination. TCP s'occupe également du contrôle de flux afin d'éviter qu'un émetteur rapide ne submerge un récepteur plus lent. Le deuxième protocole, UDP, est quant à lui non fiable, sans connexion et destiné aux applications qui ne veulent pas du séquencement et du contrôle de flux de TCP (parce

qu'elles ont le leur) et aussi pour les applications de type client serveur (question réponse) ou pour la transmission du son ou de l'image.

Le modèle TCP/IP n'a pas de couche de présentation et de session : la plupart des applications n'utilisent pas ces couches. On retrouve donc au dessus de la couche transport la **couche application** qui contient tous les protocoles de haut niveau. Au nombre de ceux-ci, on compte des protocoles comme TELNET, FTP, SMTP,... qui étaient les premiers à être développés (figure B.1(b)). A ceux-ci sont venu s'ajouter DNS, NNTP, HTTP, SNMP, ...

B.2 Les standards TCP/IP

Beaucoup de protocoles faisant partie de la suite TCP/IP ont été standardisés ou sont en voie de l'être. Une organisation est responsable du développement et de la publication de ces standards : il s'agit de l'*Internet Architecture Board* (IAB) qui les publie dans une série de documents appelés R*equests For Comments* (RFC).

Internet étant composé d'un grand nombre de réseaux interconnectés utilisant la suite TCP/IP, l'IAB est donc le comité chargé de coordonner la conception, la mise en œuvre et la gestion de l'Internet. L'IAB est composée de deux forces : l'*Internet Engineering Task Force* (IETF) et l'*Internet Research Task Force* (IRTF) dont le travail est pris en charge par des groupes de travail qui sont composés de volontaires intéressés.

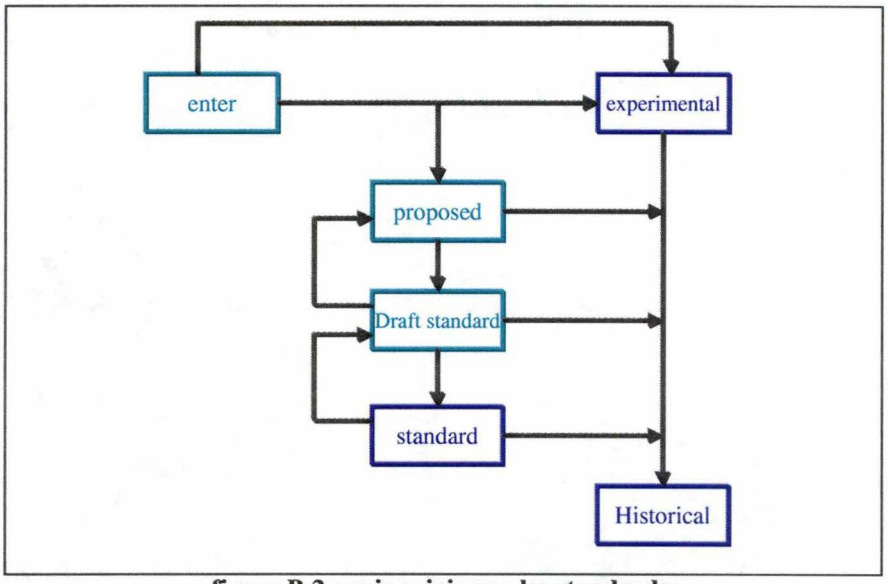

figure B.2 : voie suivie par les standards

L'IETF est responsable de la publication des RFC qui sont les notes de travail de la communauté de recherche et de développement de l'Internet. C'est l'IAB qui en définitive donne son approbation pour qu'un RFC devienne un standard, mais seulement si les spécifications de cet RFC remplissent les critères suivants :

- Il doit être stable et bien compris ;
- Il doit être techniquement compétent ;
- Il doit avoir plusieurs implémentations indépendantes et interopérables avec des tests opérationnels ;
- Il doit être intéressant pour une grande partie de la communauté Internet ;
- Il doit être reconnu utilisable dans une partie ou dans toutes les parties de l'Internet.

La différence entre ces critères et ceux utilisés pour les standards internationaux est la grande considération faite sur les expériences opérationnelles.

Avant de devenir un standard, une nouvelle spécification suit une série d'étapes. Tout d'abord une nouvelle spécification qui n'est pas prête à être standardisée sera publiée comme un RFC expérimental. Après plusieurs travaux sur cette spécification, et si elle correspond plus ou moins aux critères de standardisation susmentionnés, elle peut alors devenir une proposition de standard. Une proposition de standard devient une version *draft* (brouillon) si au moins deux implémentations indépendantes et interopérables pour lesquels des tests adéquats ont été réalisés. Ce n'est que si des tests concluants sont réalisés que l'on pourra élever cette spécification au niveau de standard qui aura alors un numéro de standardisation et son numéro RFC. Quand un standard, finalement devient obsolète, il est lui alors assigné l'état d'historique.(figure B.2)

Annexe C : Abstract Syntax Notation number One (ASN.1)

C.1 Concept généraux

La *notation de syntaxe abstraite numéro 1*(en anglais, *Abstract Syntax Notation One*) est une norme qui définit un formalisme de description de types de données abstraits.
ASN.1 est une notation formelle qui permet de spécifier les données transmises par les protocoles de télécommunications indépendamment des langages informatiques et de la représentation physique de ces données, pour toutes sortes d'applications communicantes et de données aussi complexes (ou aussi simples) soient-elles.

La notation offre un certain nombre de types prédéfinis tels que :

- Les entiers (`INTEGER`),
- Les booléens (`BOOLEAN`),
- Les chaînes de caractères (`IA5String`, `UniversalString`...),
- Les chaînes de bits (`BIT STRING`),
- etc,

et permet de définir des types composés tels que :

- Des structures (`SEQUENCE`),
- Des listes (`SEQUENCE OF`),
- Un choix de types (`CHOICE`),
- etc.

Pour chaque type, des contraintes de sous-typage permettent de restreindre l'ensemble des valeurs qui constitue son domaine.

La notation ASN.1 couvre uniquement les aspects de structuration de l'information (il n'existe pas d'opérateurs pour manipuler les valeurs ou faire des calculs sur celles-ci). Ce n'est donc pas un langage de programmation.

L'une des raisons du succès d'ASN.1 est que cette notation est associée à plusieurs ensembles de règles de codages normalisés comme les BER (*Basic Encoding Rules*), ou plus récemment les PER (*Packed Encoding Rules*) pour les utilisations où la bande passante est un objectif prioritaire. Ces règles de codage permettent de sérialiser les données sous forme de chaînes d'octets à transmettre.

> « Parce qu'il comble une lacune importante, ASN.1 est destiné, pour le meilleur et pour le pire, à devenir le langage de programmation réseau des années 90, tout comme le langage de programmation C est largement reconnu comme ayant été le langage de programmation système des années 80. Le choix d'ASN.1 est indubitablement un bon choix. »
>
> (M.T. Rose, « The Simple Book », 1990)

C.2 Domaines d'utilisation

Normalisée à l'origine pour spécifier les données des protocoles de l'interconnexion de systèmes ouverts (OSI), ASN.1 a su imposer ses qualités intrinsèques dans de nombreux autres domaines dont certains sont aujourd'hui très en vogue :

- La messagerie électronique X.400 ;
- L'annuaire X.500 ;
- Les réseaux intelligents ;
- Le commerce et le paiement électroniques : protocoles SET (*Secured Electronic Transaction*), authentification X.509... ;
- Les communications multimédias : norme MHEG, visioconférence (série de normes ITU-T T.120), communications multimédias en temps réel sur Internet (norme H.323)... ;
- C'est aussi la notation de typage de données d'autres langages formels comme GDMO (*Guidelines for the Definition of Managed Objects*) dans le domaine de la gestion des réseaux, LDS (*Langage de Description et de Spécification*) pour spécifier le comportement de systèmes télécommunicants et TTCN (*Tree and Tabular Combined Notation*) pour l'écriture de suites de tests de protocoles.

C.3 État de la normalisation

ASN.1 (qui ne portait pas encore ce nom à l'époque) a été normalisé pour la première fois en 1984 par le CCITT (*Comité Consultatif du Télégraphe et du Téléphone*, devenu aujourd'hui *ITU-T, Union Internationale des*

Télécommunications - Secteur de normalisation des Télécommunications) sous le nom de « Recommandation X.409 ». Peu après, l'ISO (*Organisation internationale de normalisation*) choisit de l'utiliser dans ses activités et scinde le document en deux : la syntaxe abstraite (ASN.1) et la syntaxe de transfert (BER). En 1985, le CCITT décide de travailler en collaboration avec l'ISO sur ces deux documents.

En 1987, l'ISO publie ces documents sous les numéros 8824 et 8825 (seuls trois nouveaux types de chaînes de caractères sont ajoutés). En 1988, il fusionne avec la CEI (*Commission Électrotechnique Internationale*) et fonde un comité technique nommé *ISO/IEC* JTC 1 *(Joint Technical Committee)*, désormais responsable de la norme ASN.1.

Dans son *Blue Book* de 1989, le CCITT publie, sous les numéros X.208 et X.209, une nouvelle version d'ASN.1 munie d'extensions résultant du travail en commun avec le JTC 1.

Pour l'édition de la version suivante, dite ASN.1:1994 (disponible depuis début 1995), la norme ISO 8824 a été scindée en 4 parties :

- ISO 8824-1 | ITU-T X.680 : *Spécification de la notation de base*,
- ISO 8824-2 | ITU-T X.681 : *Spécification des objets informationnels*,
- ISO 8824-3 | ITU-T X.682 : *Spécification des contraintes,*
- ISO 8824-4 | ITU-T X.683 : *Paramétrisation de spécifications ASN.1.*
-

La première partie améliore la norme ASN.1:1990 : la notation de macro et le type `ANY DEFINED BY` disparaissent, ils sont remplacés par la notion de classe d'objets informationnels ; l'étiquetage (*tagging*) automatique sur tout un module, la Paramétrisation des types et les caractères multi-octets font leur apparition.

En ce qui concerne les règles de codage, la norme ISO 8825 a été scindée en deux parties :

- ISO 8825-1 | ITU-T X.690 : *Règles de codage de base (BER), règles de codage canoniques (CER) et règles de codage distinctives (DER)*,
- ISO 8825-2 | ITU-T X.691 : *Règles de codage compact (PER).*

Enfin, une nouvelle édition de la norme, dite ASN.1:1997, peut être achetée auprès des comités nationaux de normalisation depuis avril 1999. Hormis quelques petits ajouts au sein de la notation, les modifications sont majoritairement éditoriales et ont consisté à intégrer les amendements de la version de 1994.

C'est cette édition ASN.1:1997 qu'il est désormais fortement conseillé d'utiliser. L'édition 1990 ne devrait plus être disponible dans un avenir proche.

C.4 Grammaire BNF

```
ModuleDefinition ::= modulereference DEFINITIONS "::=" BEGIN ModuleBody END
ModuleBody ::= AssignmentList | empty
AssignmentList ::= Assignment | AssignmentList Assignment
Assignment ::= Typeassignment | Valueassignment
Typeassignment ::= typereference "::=" Type
Valueassignment ::= valuereference Type "::=" Value
Type ::= BuiltinType | DefinedType
BuiltinType ::= BooleanType | IntegerType | BitStringType | OctetStringType |
NullType
                SequenceType | SequenceOfType | SetType | SetOfType | Choicetype |
                SelectionType | TaggedType | AnyType | ObjectIdentifierType |
                CharacterStringType | UsefulType | EnumeratedType | RealType
Value ::= BuiltinValue | DefinedValue
BuiltinValue ::= BooleanValue | IntegerValue | BitStringValue | OctetStringValue |
NullValue |
                SequenceValue | SequenceOfValue | Setvalue | SetofValue |
ChoiceValue |
                SelectionValue | TaggedValue | AnyValue | ObjectidentifierValue |
                CharacterStringValue | EnumeratedValue | RealValue
DefinedValue ::= Externalvaluereference | valuereference
BooleanType ::= BOOLEAN
BooleanValue ::= TRUE | FALSE
IntegerType ::= INTEGER | INTEGER (NamedNumberList)
NamedNumberList ::= NamedNumber | NamedNumberList, NamedNumber
NamedNumber ::= identifier(SignedNumber) | identifier(DefinedValue)
SignedNumber ::= number | -number
IntegerValue ::= SignedNumber | identifier
BitStringType ::= BIT STRING | BIT STRING (NamedBitList)
NamedBitList ::= NamedBit | NamedBitList, NamedBit
NamedBit ::= identifier(number) | identifier(DefinedValue)
BitStringValue ::= bstring | hstring | (IdentifierList) | {}
IdentifierList ::= Identifier | IdentifierList, identifier
OctetStringType ::= OCTETSTRING
```

```
OctetStringValue ::= bstring | hstring
NullType ::= NULL
NullValue ::= NULL
SequenceType ::= SEQUENCE {ElementTypeList} | SEQUENCE {}
ElementTypeList ::= ElementType | ElementTypeList, ElementType
ElementType ::= NamedType | NamedType OPTIONAL | NamedType DEFAULT Value |
                COMPONENTS OF Type
NamedType ::= identifier Type | Type | SelectionType
SequenceValue ::= (ElementValueList)
ElementValueList ::= NamedValue | ElementValueList, NamedValue
SequenceOfType ::= SEQUENCE OF Type | SEQUENCE
SequenceOfValue ::= {ValueList} | {}
SetType ::= SET{ElementTypeList} | SET {}
SetValue ::= {ElementValueList} |{}
SetOfType ::= SET OF Type | SET
SetOfValue ::= {ValueList} |{}
ChoiceType ::= CHOICE {AlternativeTypeList}
AlternativeTypeList ::= NamedType | AlternativeTypeList, NamedType
ChoiceValue ::= NamedValue
SelectionType ::= identifier < Type
SelectionValue ::= NamedValue
TaggedType ::= Tag Type | Tag IMPLICIT Type
Tag ::= {Class ClassNumber}
ClassNumber ::= number | DefinedValue
Class ::= UNIVERSAL | APPLICATION | PRIVATE | empty
TaggedValue ::= Value
AnyType ::= ANY | ANY DEFINED BY identifier
AnyValue ::= Type Value
ObjectIdentifierType ::= OBJECT IDENTIFIER
ObjectIdentifierValue ::= {ObjIdComponentList} | {DefinedValue ObjIdComponent
List}
ObjIdComponentList ::= ObjIdComponent | ObjIdComponent ObjIdComponentList
ObjIdComponent ::= NameForm | NumberForm | NameAndNumberForm
NameForm ::= identifier
NumberForm ::= number | DefinedValue
NameAndNumberForm ::= identifier(NumberForm)
CharacterStringType ::= typereference
CharacterStringValue ::= cstring
UsefulType ::= typereference
EnumeratedType ::= ENUMERATED (enumeration)
Enumeration ::= NamedNumber | NamedNumber, Enumeration
EnumerationValue ::= identifier
RealType ::= REAL
RealValue ::= NumericRealValue | SpecialRealValue
NumericRealValue ::= {Mantissa, Base, Exponent} | 0
Mantissa ::= SignedNumber
Base ::= 2 | 10
Exponent ::= SignedNumber
SpecialRealValue ::= PLUS-INFINITY | MINUS-INFINITY
```

C.5 Un exemple

Il est assez aisé de retrouver un ou plusieurs exemples de MIB, soit dans la documentation des différents systèmes à gérer (si les MIB sont propriétaires), soit dans les RFC définissant des MIB standards (il existe plusieurs sites Internet où l'on retrouve ce genre d'informations : http://www.ietf.org en est un exemple).

Annexe D : Inventaires réseau de l'ONP

Les éléments actifs du réseau

Cabletron SmartSwitch Router 8600 (SSR 8600)

- SNMP : les SSR 8600 supportent SNMPv1 et plusieurs MIB standards

- RMON : les SSR 8600 supportent RMON v1 et RMON v2 : il existe trois niveaux de configuration pour les groupes RMON :
    - le niveau *lite* pour les groupes `Etherstats`, `Event`, `Alarm` et `History` ;
    - le niveau *standard* pour les groupes `Host`, `Host Top N`, `Matrix`, `Filter` et `Packet Capture` ;
    - le niveau *professional* pour les groupes `Protocol Directory`, `Protocol Distribution`, `Application Layer host`, `Network Layer Host`, `Application Layer Matrix` (and `Top N`), `Network Layer Matrix` (and `Top N`), `Address Map` et `User History`.

- QoS : les SSR 8600 fournissent trois particularités différentes pour satisfaire les spécifications de qualités de service :

    - *Traffic prioritization* permet aux administrateurs de réseaux d'identifier et d'isoler des trafics réseau critiques dans des queues de différentes priorités par rapport aux trafics réseau non-critiques. Une fois qu'un paquet a été identifié, il peut être assigné à une des quatre priorités dans le but d'assurer la transmission. Les priorités peuvent être allouées sur de combinaisons de trafics des couches 2,3 et 4.

      La politique de gestion des queues des SSR est basée strictement sur la priorité (*Control, High, Medium* et *Low*). Pour changer cette politique en *Weighted Fair Queueing* (WFQ), il est nécessaire d'entrer une ligne de commande spécifique.

    - *Type of Service(TOS) rewrite* fournit aux administrateurs de réseaux l'accès à l'octet ToS des paquets IP. L'octet ToS est conçu pour fournir un *Feed-back* aux applications des couches supérieures. L'administrateur peut ainsi marquer les paquets en utilisant la particularité *ToS rewrite* de telle façon qu'une application (un protocole de routage) puisse manipuler les paquets sur base d'un mécanisme prédéfini.

    - *Traffic rate limiting* fournit aux administrateurs de réseaux un outil de gestion de la bande passante. L'administrateur peut créer une limite supérieure pour un profile de trafic, qui est basé sur les informations de la couche 3 et 4. Le trafic excédentaire à cette limite supérieure peut soit être abandonné soit "re-prioritisé" dans une autre queue.

Cabletron SmartSwitch Router 2000 (SSR 2000)

- SNMP : les SSR 2000 supportent SNMPv1 et plusieurs MIB standards

- RMON : les SSR 2000 supportent RMON v1 et RMON v2 : il existe trois niveaux de configuration pour les groupes RMON :
    - le niveau *lite* pour les groupes `Etherstats`, `Event`, `Alarm` et `History` ;
    - le niveau *standard* pour les groupes `Host`, `Host Top N`, `Matrix`, `Filter` et `Packet Capture` ;
    - le niveau *professional* pour les groupes `Protocol Directory`, `Protocol Distribution`, `Application Layer host`, `Network Layer Host`, `Application Layer Matrix` (and `Top N`), `Network Layer Matrix` (and `Top N`), `Address Map` et `User History`.

- QoS : les SSR 8600 fournissent trois particularités différentes pour satisfaire les spécifications de qualités de service :
    - *Traffic prioritization* permet aux administrateurs de réseaux d'identifier et d'isoler des trafics réseau critiques dans des queues de différentes priorités par rapport aux trafics réseau non-critiques. Une fois qu'un paquet a été identifié, il peut être assigné à une

des quatre priorités dans le but d'assurer la transmission. Les priorités peuvent être allouée sur de combinaisons de trafics des couches 2,3 et 4.
La politique de gestion des queues des SSR est basée strictement sur la priorité (*Control, High, Medium* et *Low*). Pour changer cette politique en *Weighted Fair Queueing* (WFQ), il est nécessaire d'entrer une ligne de commande spécifique.

- *Type of Service(TOS) rewrite* fournit aux administrateurs de réseaux l'accès à l'octet ToS des paquets IP. L'octet ToS est conçu pour fournir un *feedback* aux applications des couches supérieures. L'administrateur peut ainsi marquer les paquets en utilisant la particularité *ToS rewrite* de telle façon qu'une application (un protocole de routage) puisse manipuler les paquets sur base d'un mécanisme prédéfini.

- *Traffic rate limiting* fournit aux administrateurs de réseaux un outil de gestion de la bande passante. L'administrateur peut créer une limite supérieure pour un profile de trafic, qui est basé sur les informations de la couche 3 et 4. Le trafic excédentaire à cette limite supérieure peut soit être abandonné soit "re-prioritisé" dans une autre queue.

Cabeltron SmartSTACK 100 (ELS100-24TXM) Ethernet Switch

RFC No	Titre	Groupes supportés
1213	MIB – II	System, Inerfaces, Address Translation, IP, ICMP, TCP, UDP, Transmission et SNMP
1398	Ethernet MIB	Ethernet-specific statistics subgroup of the MIB II Transmission group
1493	Bridge MIB	Spanning Tree, Forwarding Table, information and configuration
1757	RMON MIB	Statistics, History, Alarm and Event
Cabletron rpivate enterprise MIB	CtELS-NG-mib.txt (1.02.00)	Provides management information for the next generation of the ELS100 product line, part of the SmartStack product line

RFC No	Titre
1157	IETF SNMP Coldstart → generic 0 LinkDown → 2 LinkUp → 3 Authentification Failure → 4 Enterprise specific 6
1493	IEFT Bridge, ENTERPRISE dot1dBridge – 1.3.6.1.2.1.17 NewRoot → specific 1 TopologyChange → 2
1757	IEFT RMON, ENTERPRISE rmon – 1.3.6.1.2.1.16 RisingAlarm → 1 fallingAlarm → 2

Les systèmes d'exploitations

Microsoft Windows NT 4.0 server

- SNMP : la version supportée par les versions de Windows NT 4.0 server installées à l'ONPP est la version SNMPv1

- RMON : le système d'exploitation Windows NT 4.0 Server peut supporter les RMON v1 et RMON v2

- QoS : pas dinformations

Annexe E : Informations obtenues à partir des journaux du Proxy Server

Cette annexe composée d'informations confidentielles n'est disponible que sur demande auprès de l'auteur du document.

Annexe F : Résutats obtenus par l'analyseur de protocoles

Cette annexe composée d'informations confidentielles n'est disponible que sur demande auprès de l'auteur du document.

Protocol Distribution

Top Conversations by Packets

Top Hosts by Packets Sent