Technologie des Réseaux d'Entreprise

Gestion de Réseaux

K. Marcus, D. Loisel

Denis Arnaud, Sébastien Fabre, Gregory Kuhlmey

Institut Eurécom, Décembre 1997

Table des matières

Table des figures

1 Présentation de l'étude

1.1 Introduction

Le cadre de notre étude consiste en la mise en place d'une infrastructure permettant la gestion du réseau d'une organisation répartie sur plusieurs sites. Dans notre cas, il s'agit de l'Institut Eurécom, connecté par Internet à ses Ecoles mères, l'ENST à Paris et l'EPFL à Lausanne. Et la gestion du réseau doit permettre la surveillance de l'état opérationnel de certaines ressources critiques réparties sur l'ensemble des trois sites.

1.2 Existant — Architecture du réseau

Le réseau est constitué de trois sites reliés par l'Internet. Chacun de ces sites fonctionne avec:

Cinq serveurs de fichiers SUN

Cinq serveurs de calcul HP

Un routeur CISCO

Trois switchs FORE

Un LAN ATM

De plus, le site d'Eurécom possède un serveur WEB.

Nous avons réparti les ressources critiques (les serveurs de fichiers et de calcul) sur les trois switchs pour éviter de se trouver sans aucune ressource en cas de défaillance d'un de ces derniers. D'autre part, la charge des différents liens est ainsi mieux répartie.

Le routeur peut être associé à un Firewall pour filtrer les accès au réseau interne depuis l'extérieur. A Eurécom le serveur Web est directement connecté sur le routeur pour que les requêtes destinées à ce serveur restent dans la partie du réseau ouverte au public. Nous avons installé une sonde RMON à la sortie du routeur pour effectuer des mesures et des filtrages sur le trafic sortant et uniquement sur celui-ci.

Les ressources sont gérées par SNMPv1 qui fonctionne avec de l'IP sur ATM. Le réseau est administré depuis une machine connectée sur le switch central pour limiter les flux d'informations liés à l'administration. Dans le

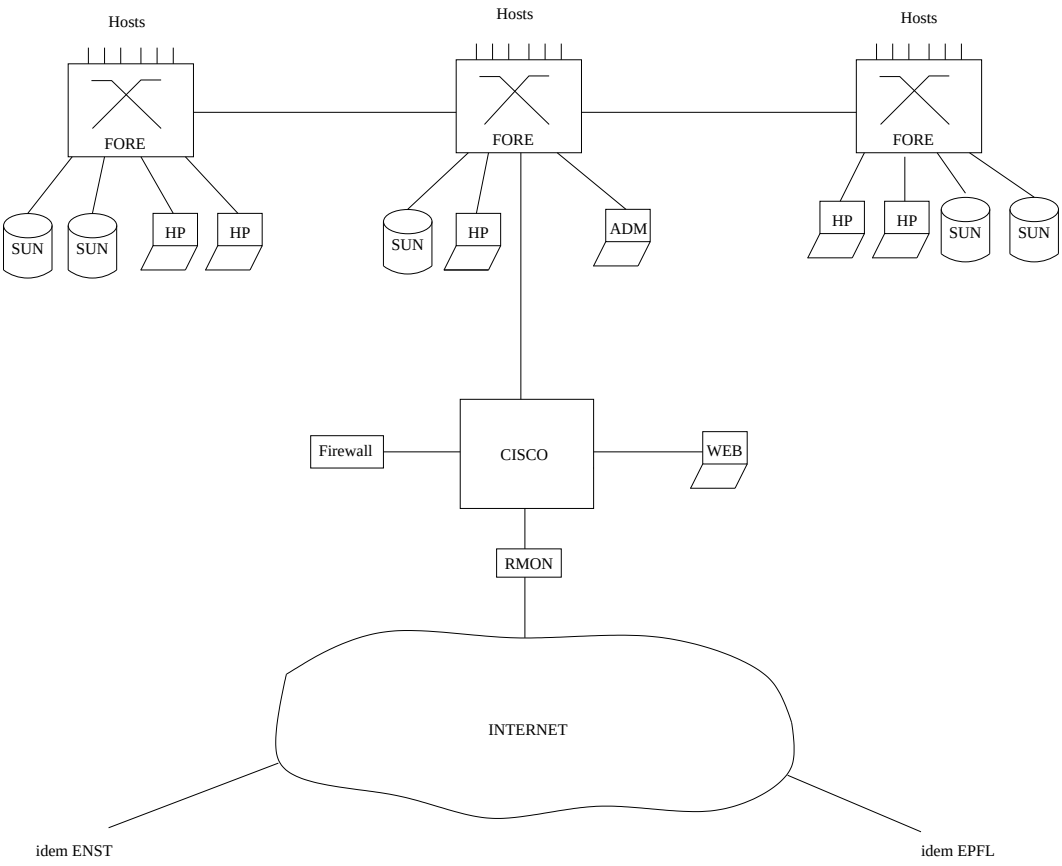

FIG. 1 – *L'architecture du réseau*

cas où l'ensemble des ressources est géré depuis un seul site, le site d'administration est équipé d'une console d'administration supplémentaire.

2 Plates-formes

2.1 Comparaison - Présentation des produits

Nous avons eu à comparer deux plates-formes, à savoir *TME 10 NetView* d'IBM/Tivoli et *Spectrum* de Cabletron. Plates-formes sur lesquelles nous devions étudier la pertinence d'ajouter le produit Optivity de Bay Networks.

2.1.1 Cabletron Spectrum

Intégration - Développement

Spectrum est une plate-forme construite sur le modèle client/serveur. La plate-forme s'organise autour du *SpectroSERVER* en modules clients séparés. Le SpectroSERVER est lui-même organisé en une *Virtual Network Machine (VNM)*, une base de données, des piles de communication et une API de communication pour les clients.

Les clients peuvent aussi bien être des modules de Spectrum, comme par exemple *SpectroGRAPH* qui est la GUI de la plate-forme, que des applications qui tournent sur une autre plate-forme (voir figure 2). L'adaptation entre les modules externes et le cœur de la plate-forme, le SpectroSERVER, est faite par des *Spectrum Portable Management Applications (SPMA)*. Celles-ci permettent de compléter la gamme des applications déjà fournies par Spectrum.

Le SpectroSERVER supporte les piles SNMP, ICMP, EPI (un protocole propriétaire de Cabletron) par l'intermédiaire du *Device Communications Manager (DCM)*, mais la pile CMIP n'est encore qu'à l'état de prototype.

La base de données du SpectroSERVER est orientée objet et très riche, puisqu'elle permet d'appréhender le réseau en considérant les liens et interactions qui existent entre les éléments gérés grâce à l'*Inductive Modeling Technology (IMT)*. Le défaut de cette base de données est d'être propriétaire. Ce n'est que par une API supplémentaire que les informations peuvent être exporter vers d'autres systèmes de gestion de données.

Cabletron fournit deux types d'outils pour le développement:

- Une boîte à outils de premier niveau qui permet d'étendre Spectrum à de nouveaux éléments de réseau et de configurer le noyau de Spectrum.
- Une boîte à outils de deuxième niveau qui contient des APIs et qui sert à développer en C++ des extensions avancées aux applications de management ou aux représentations des données.

Le large recours à des moyens de communication ou de représentation de données propriétaires ne facilite pas l'intégration de cette plateforme. En fait, elle se limite à la possible interopérabilité avec des applications externes par le moyen de SPMA et la fourniture d'APIs. Bien que certains outils de développement soient fournis, les possibilités offertes demeurent restreintes.

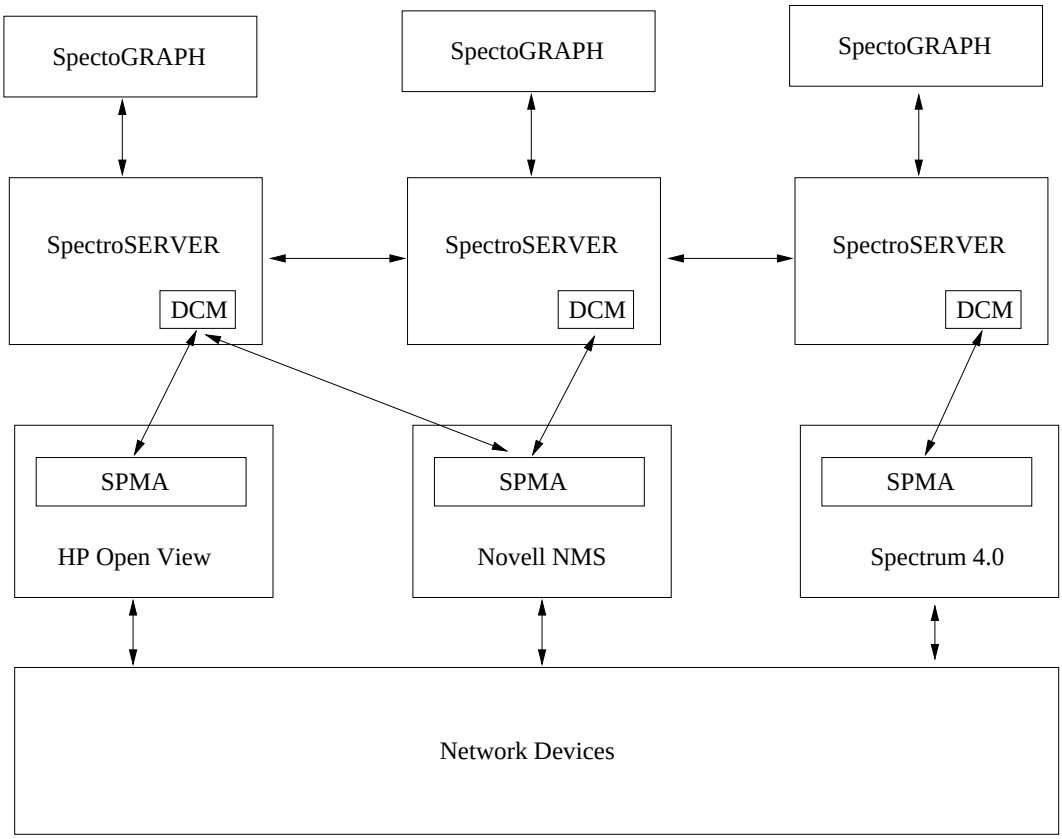

FIG. 2 – *Communications entre serveurs et avec les modules clients*

Communication entre managers

Le mécanisme de communication entre managers est propriétaire.

Gestion distribuée

Spectrum a été conçue pour la gestion distribuée de réseau. En effet la gestion du réseau peut véritablement être répartie en affectant à différents serveurs des tâches ou bien des sous-réseaux spécifiques. Les serveurs dédiés s'occupent alors du polling sur certains éléments seulement et récupèrent les alarmes les conçernant. Après analyse, seules les informations pertinentes sont retransmises aux autres SpectroSERVERs. On peut accéder aux SpectroSERVERs un par un ou de façon regroupée depuis une console client unique.

La modélisation interne du réseau tient compte non seulement des éléments qui le constituent, mais aussi des liens qu'il peut y avoir entre eux. La plate-forme a ainsi une approche plus "intelligente" des informations. En plus, ces informations sont traitées par une technologie originale (IMT) qui permet de réduire le nombre d'alarmes en cas d'avalanches en isolant la source du problème.

2.1.2 TME 10 NetView

Intégration - Développement

Cette plate-forme est née de la fusion entre IBM et Tivoli, et a repris les points forts des produits de ces deux firmes, respectivement NetView (aussi appelé SystemView) et TME 3.0. Alors que NetView se voyait dévolu à la gestion de réseaux, TME 3.0 était spécialisé pour l'administration système. La fusion entre les deux sociétés a permis d'intégrer les deux types d'administration (système et réseaux) sur un seul produit, nommé du coup *TME 10 NetView*.

En fait, cette plate-forme reprend la base du produit de Tivoli (*TME 10 Framework*), base sur laquelle vient s'intégrer le produit d'IBM et d'autres applications Tivoli (distribution de logiciels, gestion des utilisateurs, gestion distribuée des configurations, etc.). Certains éléments de gestion de réseaux de NetView sont ou vont être directement intégrés dans TME 10 Framework, permettant notamment à d'autres applications d'inter-opérer avec NetView. Un des intérêts majeurs de

TME 10 Framework est justement que Tivoli fournit une API (*Application Programming Interface*), la *TME 10 Framework API*, permettant d'intégrer différentes applications de différentes sociétés (Tivoli délivre un label "*Tivoli Ready*" aux sociétés qui développent des applications s'intégrant à TME 10 Framework, et les développeurs bénéficient d'une boîte à outils d'intégration (*Integration Toolkit*) ainsi que de l'environnement d'un club de développeurs créé par Tivoli: la *Tivoli's 10/Plus Association*).

Ainsi, non seulement il est possible de développer soi-même des applications s'intégrant étroitement avec TME, mais l'on bénéficie aussi des applications des autres plates-formes et des autres développeurs.

FIG. 3 – *TME 10 Framework: l'intégration simplifiée*

Communication entre managers

TME 10 Framework donne la base de communication entre managers, et permet par exemple de configurer les traps SNMP ou les règles de corrélation d'évenements, sur tous les composants NetView à la fois, et tout cela à partir d'un seul site.

Alors que les managers (les *Mid-Level Managers*) communiquent avec les agents en utilisant le protocole SNMP, les managers utilisent entre eux un protocole basé sur CORBA (comme le montre la figure 4), ce qui permet de sécuriser et de rendre robustes les communications.

D'autre part, les données relatives notamment aux événements, à la topologie et à la configuration SNMP, sont stockées dans une base de données qui s'interface avec les principaux produits du marché (Oracle, Sybase, Informix et MS SQL).

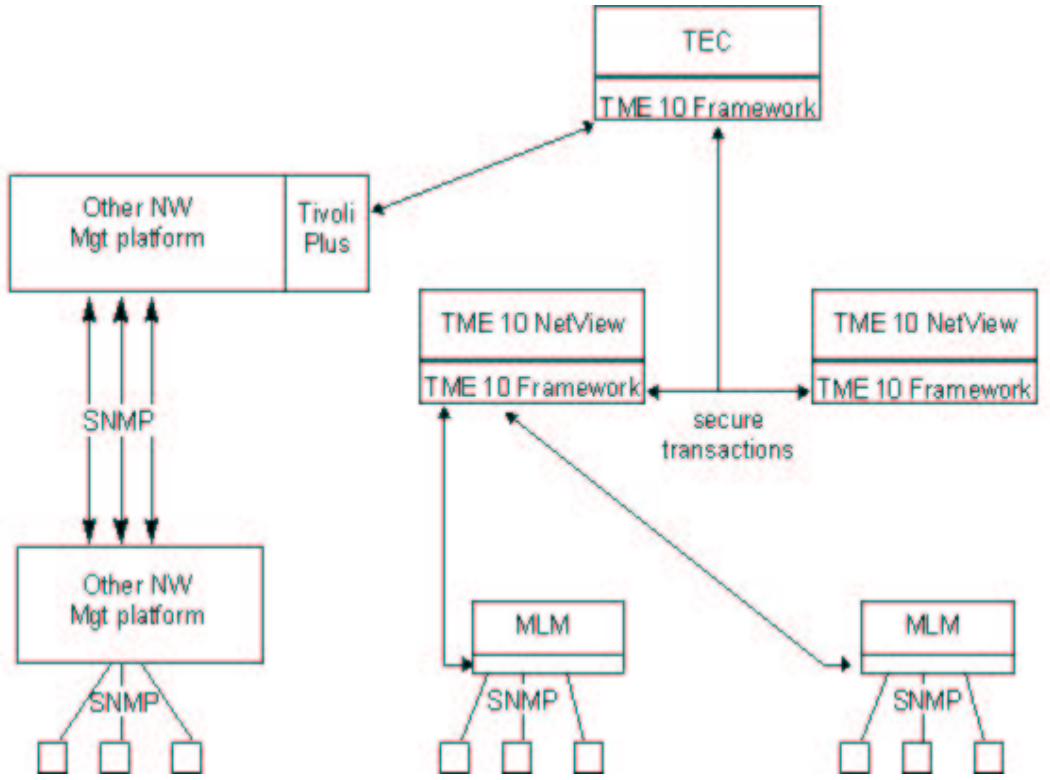

FIG. 4 – *Communications inter-applications basées sur des transactions CORBA*

Gestion distribuée

L'architecture répartie de l'implantation des différents éléments NetView permet de positionner les managers, les MLMs (*Mid-Level Managers*), au plus près des ressources réseaux. Comme les MLMs demandent peu de ressources système (HP-UX et Sun/Solaris font partie des OS supportés par les MLMs), et qu'ils communiquent avec la station principale NetView sur le mode client/serveur, cette architecture peut facilement supporter la croissance du réseau.

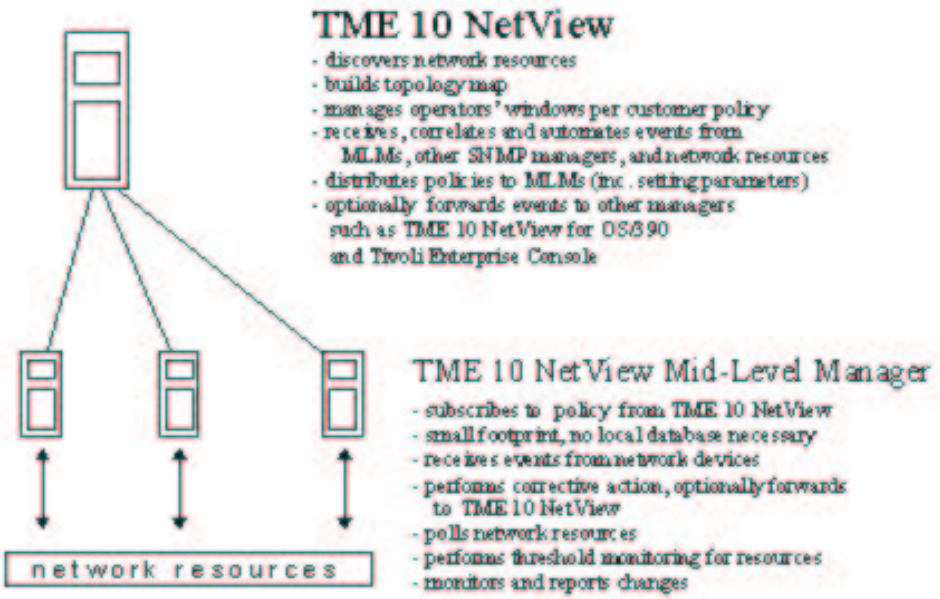

FIG. 5 – *Architecture distribuée*

D'autre part, dans notre étude, nous devons envisager deux types d'installation:

- chaque site gère localement ses ressources,
- l'ensemble des ressources est géré par un seul site avec délégation sur les sites locaux.

Or, TME 10 NetView permet justement d'envisager sereinement ces deux types d'installation à partir d'une seule architecture.

En effet, dans le cas d'une gestion locale, des MLMs gérés localement par un système NetView suffit. Et si l'on souhaite alors passer à une configurartion de gestion centralisée, il suffit d'ajouter sur le site central la *TME 10 Enterprise Console (TEC)*, qui permet de fédérer et de corréler les événements provenant de nombreuses sources, y compris de managers SNMP autres que NetView (voir la figure 6).

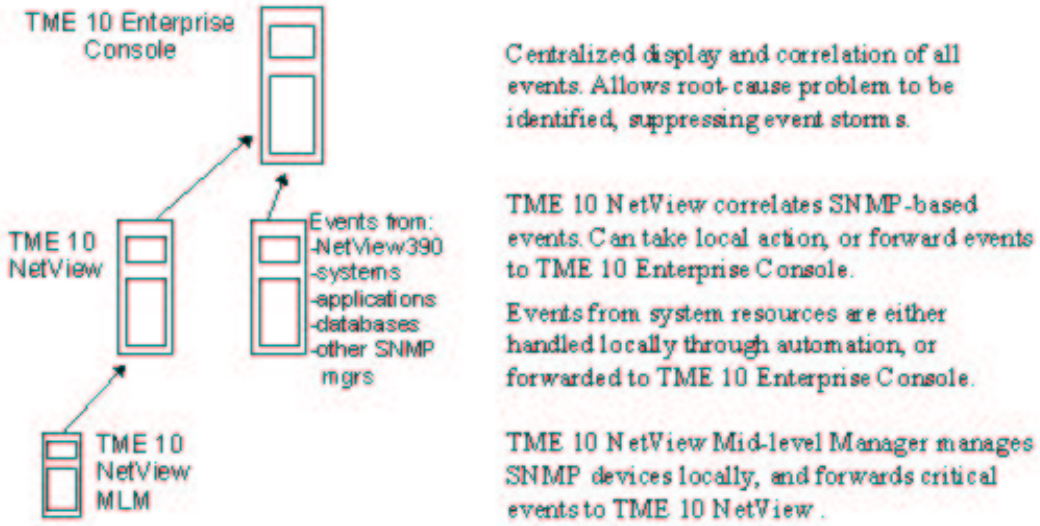

FIG. 6 – *Gestion centralisée - Intégration des événements*

Ainsi, les événements peuvent être filtrés et analysés à chacun des stades (MLM, NetView, TEC), avec la possibilité de les faire suivre au stade supérieur ou d'entreprendre des actions correctives. Aussi, seuls les événements pertinents sont retenus à chacun des niveaux, et leur corrélation permet de trouver et de traiter plus facilement leur source.

De même, les sondages (*polling*) sont effectués par les MLMs, au plus près des ressources, ce qui évite d'encombrer inutilement le réseau.

Les plus

En plus des caractéristiques énoncées répondant au cahier des charges, TME 10 présente quelques avantages intéressants:

Robustesse: Si un MLM ou un NetView tombe en panne, un autre peut récupérer de manière automatique ses responsabilités (no-

tamment de sondages et de relais/analyse d'événements).

Intégration système/réseau: Cela résulte de la fusion entre Tivoli et IBM, et donne donc un moyen de gérer d'un seul endroit à la fois les systèmes (*hosts*) et les ressources réseau. Un clic de souris sur la carte topologique permet par exemple d'accèder au contenu du disque et des fichiers de configuration d'un système.

Gestion par action unique: Pouvoir regrouper les objets sous formes de collections (par exemple, tous les switchs Fore, ou encore les systèmes en panne) permet de distribuer de manière transparente les configurations et autres logiciels à tous les membres du groupe automatiquement, comme si l'on ne s'adressait qu'à une seule entité.

2.1.3 Optivity

Optivity est un produit à part. On pourrait le qualifier de logiciel d'aide à l'optimisation. Cette plate-forme d'administration vient se placer au dessus d'une autre. Les plate-formes compatibles avec Optivity sont HPOpenView, Novell NMS, SunNet Manager, et Netview.

Cette plate-forme offre de très nombreuses facilités en ce qui concerne la gestion des switchs, routeurs et sondes RMON; l'évaluation du trafic et de la configuration du réseau; l'évolution et l'optimisation du réseau grâce à un outil de simulation permettant le test à l'avance de ses solutions réseau; et l'administration distribuée.

Optivity propose trois niveaux d'administration, un niveau local, un niveau Campus (réseau étendu sur un site) et un niveau Entreprise (réseau partagé sur plusieurs sites).

Nous aurions pu opter pour l'achat de ce produit pour faciliter la gestion de nos switchs ATM et de notre sonde RMON. Mais l'achat d'une autre plate-forme, sa mise en place, et sa maintenance nous ont apparus comme un investissement superflu considérant la taille actuelle de notre réseau, qui nous semble pouvoir se passer d'Optivity pour son optimisation.

2.2 Choix retenu

Bien que les deux plates-formes, *Spectrum* et *TME 10 NetView*, permettent de remplir le cahier des charges imposé dans des conditions sensiblement équivalentes, nous avons retenu pour notre étude le choix de la

plate-forme d'IBM/Tivoli.

Ce choix se justifie d'abord par la possibilité réduite, avec Spectrum, de développement et de personnalisation des applications. Ces possibilités existent, mais un manque d'API de développement se fait ressentir, alors que pour TME 10, la base d'intégration de Tivoli (TME 10 Framework) fournit tous les outils, APIs et supports (notamment avec la Tivoli's 10/Plus Association) pour le développement d'applications pouvant inter-opérer.

Le label "*Tivoli Ready*", délivré par Tivoli en cas de conformance à l'API TME 10 Framework, permet de garantir l'inter-opérabilité entre les développements que l'on va effectuer et les applications de gestion de réseau déjà existantes. Dans le cas de Spectrum, l'intégration se fait par "le haut", en ajoutant une surcouche aux applications existantes: Spectrum peut récupérer des objets sur d'autres plates-formes, mais les échanges dans l'autre sens sont plus limités.

Ensuite, la communication entre managers est robuste, sécurisée et suit des standards (Corba) dans le cas de TME 10, alors qu'elle reste propriétaire pour Spectrum.

De même, l'organisation des données et le recueil des objets s'interfacent facilement avec les produits de bases de données les plus répandus dans le cas de TME 10, alors que Spectrum utilise des systèmes propriétaires.

Enfin, lors de notre recherche de documentation, nous avons pu nous rendre compte que la disponibilité des informations était assez inégale: autant il est facile de trouver des livres blancs, rouges et autres documentations techniques sur NetView, autant obtenir une description précise de Spectrum apparaît plus délicat. Si nous devions installer une plate-forme, nous nous sentirions ainsi beaucoup mieux épaulés avec NetView qu'avec Spectrum (l'on trouve par exemple sur Internet des documentations complètes et précises sur l'installation et l'utilisation de NetView pour tel ou tel OS, sur les changements dans les dernières versions, *etc.*).

En ce qui concerne Otpivity, il est clair que cette plate-forme ne peut s'intégrer qu'à NetView, et non pas à Spectrum. Nous n'avons pas estimé devoir retenir Optivity, car la seule plate-forme d'IBM/Tivoli permet d'effectuer allégrement tout ce qu'exige le cahier des charges. Optivity apporterait un plus, mais de l'ordre de l'agrément et du confort, plus que de celui de la fonctionnalité. Ainsi, le plus d'Optivity vient surtout de la facilité de configuration par

des aides graphiques évoluées, et d'effectuer des simulations/analyses sur le réseau. Cette plate-forme trouve toute sa justification dans le cas de réseaux hétérogènes construits par petits bouts indépendants sans planification véritablement centralisée. Mais, comme dans notre cas, le réseau est relativement homogène, cohérent et jeune, nous ne voyons pas d'atout indispensable que pourrait apporter Optivity.

2.3 Présentation

Nous avons donc retenu la mise en œuvre de la plate-forme TME 10 NetView d'IBM/Tivoli uniquement. Il reste à expliciter l'architecture correspondante à notre cas, que nous explicitons sur la figure 7.

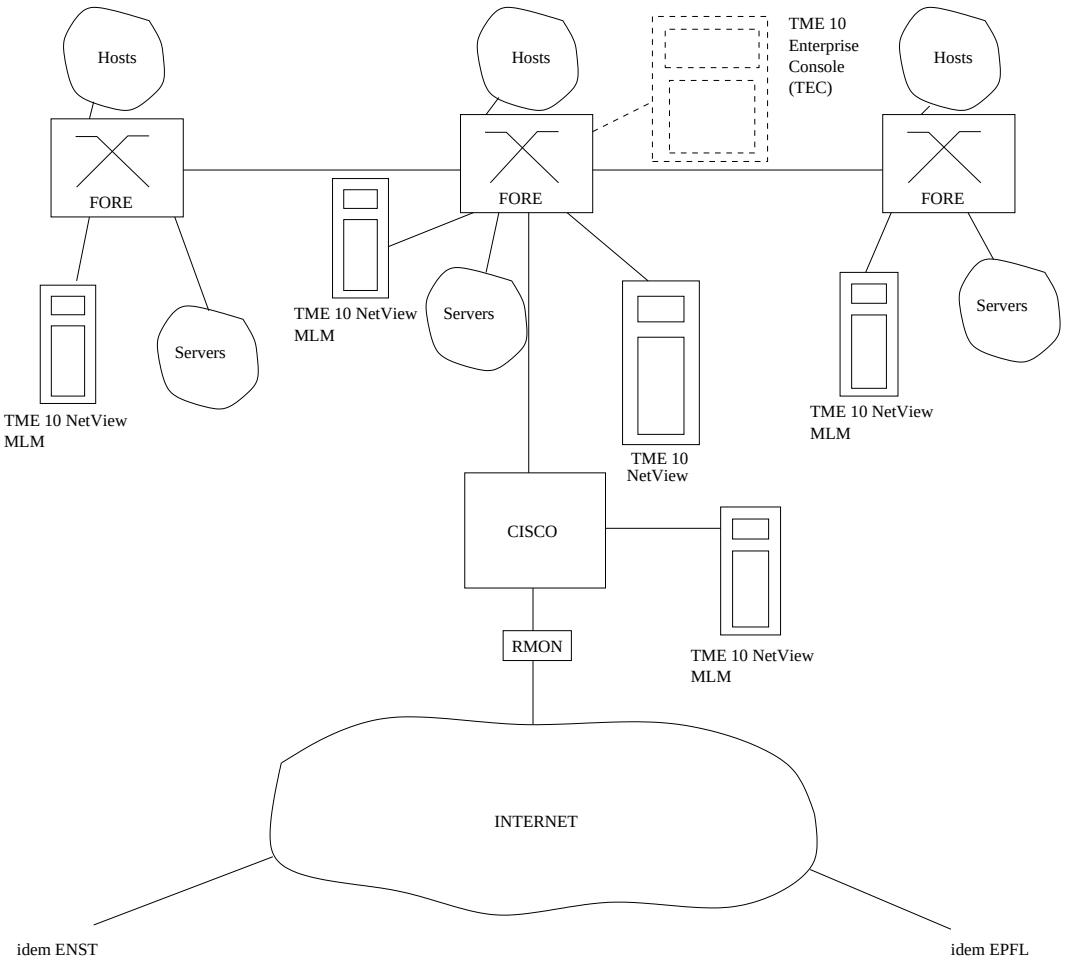

FIG. 7 – *Architecture retenue*

Les MLMs (*Mid-Level Managers*) sont des programmes qui nécessitent relativement peu de ressources systèmes (et pas de bases de données locales). Ils peuvent donc être placés directement sur une des machines déjà en place (host ou serveur). Dans notre architecture, nous avons placé un seul MLM par switch, et un sur le routeur. Si ces MLMs parviennent à saturation, *i.e.*, si la charge de travail (analyse des événements et actions correctives, sondages des ressources, détection de franchissement de seuils, rapports de modifications perçues, etc.) devient trop lourde pour un seul de ces MLMs par switch, l'on peut sans problème en ajouter d'autres sur les autres hosts ou serveurs. Par exemple, l'on pourra placer un MLM pour gérer les serveurs, et un pour les hosts. Ainsi, le MLM du routeur gère plus spécifiquement les tâches affairant au réseau extérieur.

Cette gestion par un MLM sur chacun des switchs permet d'éviter une génération trop importante de trafic sur le réseau. Toutefois, si ce trafic s'avérait encore trop important (dans le cas d'ajouts de hosts et de serveurs, par exemple), l'on pourrait alors placer un système TME 10 NetView par switch.

Pour l'instant donc, un seul TME 10 NetView permet de gérer localement tous les MLMs, et dans le cas d'une gestion centralisée, l'on placera de plus une station avec TME 10 Enterprise Console. Cette structure hiérarchique de collection et de traitement de l'information permet d'étendre facilement le réseau en cas de croissance future, et surtout d'intégrer facilement l'architecture multi-sites de notre cas d'étude.

3 L'administration du réseau

Pour l'administration de notre réseau nous avons besoin d'effectuer plusieurs mesures sur les switchs, le routeur et les serveurs, d'être prévenus au plus tôt des pannes survenant sur le réseau et de connaître le trafic généré sur le WAN (la part de trafic http et le trafic total).

3.1 Taux de disponibilité des switchs et du routeur

Sur chaque site, nous disposons de trois switchs ATM Fore et d'un routeur Cisco. Ces matériels implémentent une MIB propriétaire en plus de la MIB II.

Pour effectuer la mesure du taux de disponibilité sur les switchs Fore (avec la MIB Fore), nous disposons d'une variable $UpTime$ (voir figure 8) indiquant le temps écoulé depuis la dernière ré-initialisation du switch.

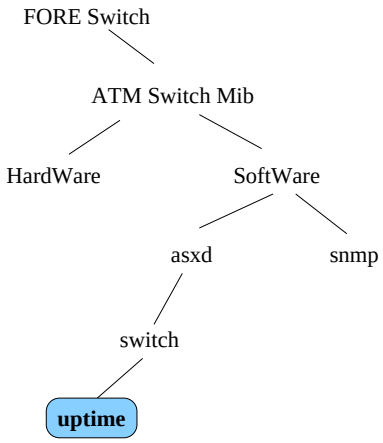

FIG. 8 – *Variable UpTime dans la MIB Fore*

Grâce à un sondage (*polling*) fréquent (toutes les 5 minutes à l'installation, puis moins souvent si le réseau s'avère fiable) sur cette variable, nous pouvons mesurer la diponibilité des switchs avec une granularité égale à celle du polling.

Du côté du routeur, la MIB Cisco ne fournit aucune variable pour la mesure de la disponibilité. Par contre, Avec la MIB II, nous retrouvons une variable $SysUpTime$ (voir figure 9), équivalente a l'$UpTime$ de la MIB Fore que nous utilisons de la même façon.

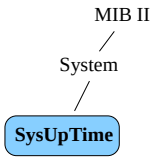

FIG. 9 – *Variable SysUpTime dans la MIB II*

3.2 Connaître l'état opérationnel des liens et des équipements

Il s'agit ici de surveiller les liens physiques et les matériels.

3.2.1 Les switchs

Sur les switchs, disposant de la MIB Fore, on utilise la variable *Carrier* (voir figure 10) qui indique pour chaque port l'état du lien auquel il est connecté. On peut ainsi connaître l'état des trois liens principaux du réseau de chaque site (deux liens inter-switch, et un lien switch central – routeur). Pour surveiller l'état des switchs eux-mêmes on interroge la variable *OperStatus* (voir figure 10) qui nous donne l'état opérationnel de chaque port.

De plus, la MIB Fore comporte des alarmes à trois états: *noalarm*, *minoralarm* et *majoralarm*. Ces alarmes sont présentes dans tous les groupes sous la forme d'une variable appelée *alarmstate*. En interrogeant ces variables on peut connaître l'état opérationnel de chaque groupe du switch. Par exemple, dans le groupe *powerGroup*, si $alarmstate = noalarm$ tout va bien, si $alarmstate = minoralarm$, certaines alimentations sont en pannes et si $alarmstate = majoralarm$, toutes les alimentations sont hors service. Ces alarmes permettent une gestion "fine" des problèmes ; en fonction de la gravité des alarmes on prévient automatiquement certaines personnes.

3.2.2 Le routeur

Sur le routeur, on utilise la variable $ifOperStatus$ (voir figure 11), qui indique elle aussi l'état opérationnel de chaque interface.

3.2.3 Les serveurs

Les serveurs de calcul et de fichiers, comme décrit dans la section 1, sont portés par des *hosts*. Nous avions donc la possibilité de prendre ou de laisser la MIB *Host Resources*. Cette dernière est très riche et nous permettrait d'effectuer facilement toutes nos mesures, et en particulier celles sur la CPU et la

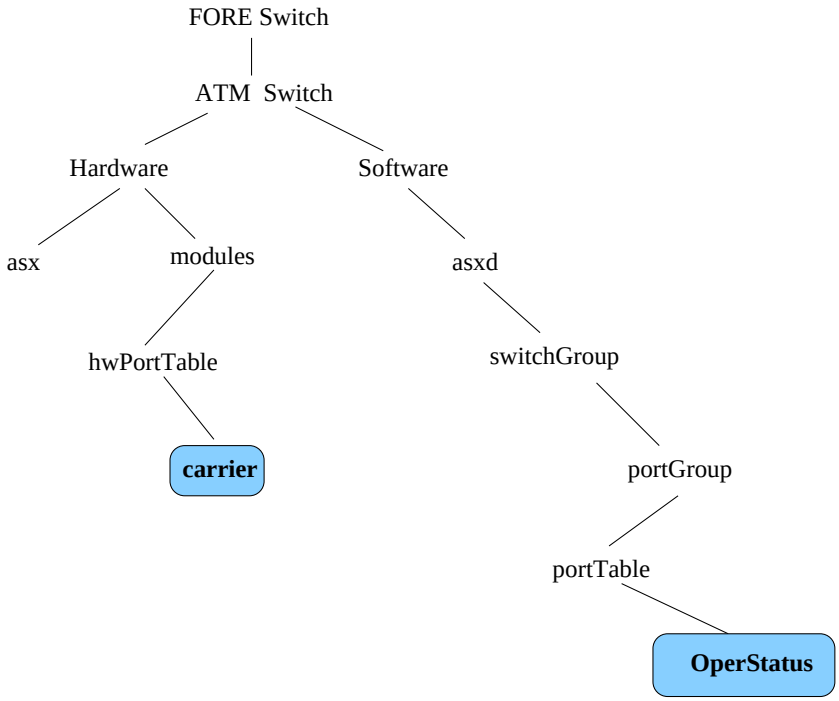

FIG. 10 – *Variables Carrier et OperStatus de la MIB Fore*

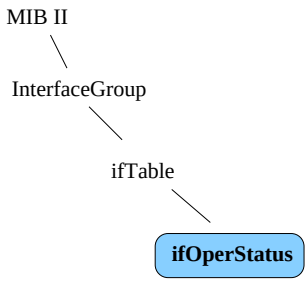

FIG. 11 – *Variable ifOperStatus de la MIB II*

mémoire. Toutefois, avant d'opter pour une MIB, il faut vérifier qu'elle soit réellement implémentée par des constructeurs. C'est le cas d'Empire Technologies qui développe cette MIB. Nous avons donc choisi d'utiliser la MIB Host Resources (RFC 1514).

Il est à noter, cependant, que dans la plupart des cas, les constructeurs (Sun, HP, IBM, DEC, etc.) implémentent une MIB propriétaire sur leurs *hosts*, qui reprend les principaux avantages de la MIB Host Resources. Dans le cadre de cette étude, nous avons tout de même pris la MIB Host Resources afin de prendre un exemple "pédagogique". Si nous avions à implémenter ce système, nous prendrions bien sûr les MIBs propriétaires fournies en standard sur les *hosts*, plutôt que d'aller acheter l'implémentation d'une MIB supplémenatire.

Cette MIB comprend la variable $hrSystemUptime$ (voir figure 12) équivalente aux variables $upTime$ de la MIB Fore et $SysUpTime$ de la MIB II. En interrogeant régulièrement cette variable on surveille l'état des hosts.

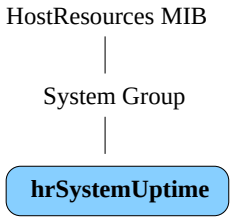

FIG. 12 – *Variable $hrSystemUptime$ de la MIB Host Resources*

Connaître l'état opérationnel des liens et des équipements est important car une panne peut avoir de graves conséquences. Toutefois, il ne faut pas surcharger le réseau de requêtes. Ainsi, nous pensons qu'un polling assez fréquent (tous les quarts d'heure) correspond à un bon compromis sécurité – trafic. Nous renforcerons tout de même cette surveillance par un système de traps SNMP (voir la section 3.3) car la panne d'organes vitaux comme les switchs ou le routeur doit être connue de l'administrateur au plus tôt.

3.3 Prévenir le ou les administrateurs au plus tôt

Pour prévenir l'administrateur on utilise les traps SNMP. Il existe 6 traps ($coldStart$, $warmStart$, $linkDown$, $linkUp$, $authentificationFailure$, et $egpNeighborlossTrap$) prédifinies dans SNMP et utilisables par tous les agents (s'il les supporte). En outre, les MIB propriétaires peuvent fournir

d'autres traps, qui ne peuvent être traitées que par les plates-formes d'administration compatibles avec le matériel (*i.e.*, pouvant traiter les informations contenues dans les traps propriétaires).

- Le routeur Cisco implémente tous ces traps sauf le $warmStartTrap$. On va donc valider l'emission des traps classiques SNMP sur l'agent du routeur, et on sera prévenu des pannes sur les liens ou les interfaces du routeur par les $linkDowntrap$.

- Les switchs Fore sont gérés par des agents utilisant la MIB Fore. Cette MIB comprend 6 traps ($asxSwLinkDown$, $asxswLinkUp$, $asxHostLinkDown$, $asxHostLinkUp$, $asxNetModuleDown$, $asxNetModuleUp$). De même, on valide l'émission des traps sur les agents Fore pour être prévenu de la moindre panne dès qu'elle survient.

- Pour les serveurs, la MIB Host Resources ne prévoyant aucun trap, on utilisera les traps génériques SNMP.

3.4 Surveiller le taux de remplissage des partitions des disques

Il s'agit ici d'accéder à la taille de chaque partition et à la taille de sa zône occupée. Grâce à la MIB Host Resources ces informations sont directement disponibles. La variable $hrStorageSize$ (voir figure 13) donne la taille de n'importe quel type de zône mémoire (mémoire vive ou partition disque). La variable $hrStorageUsed$ (voir figure 13) en donne elle la taille de la zône utilisée.

On a donc comme taux de remplissage : $\frac{hrStorageUsed}{hrStorageSize}$.

On effectue sur ces variables un polling de faible fréquence (3 fois par jours), car en utilisation normale, ces données évoulent assez lentement. De plus, les quotas offrent une relative protection contre les débordements. Toutefois, si le taux de remplissage d'un disque atteignait 90%, il faudrait alors augmenter significativement la fréquence du polling sur ce disque.

3.5 Surveiller l'utilisation CPU et mémoire

3.5.1 La mémoire

Pour l'utilisation de la mémoire, on a vu dans le paragraphe précédent que la mémoire était traitée de la même façon que les partitions disques.

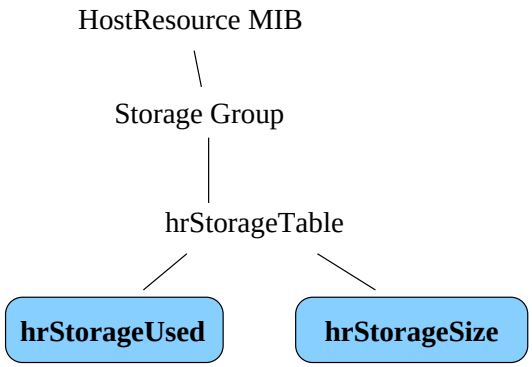

FIG. 13 – *Variables hrStorageUsed et hrStorageSize de la MIB Host Resources*

On effectuera par contre un polling plus fréquent car les variations de la zône mémoire utilisée sont plus rapides que celles de la zône disque utilisée. Un polling de période 15 minutes devrait nous permettre une surveillance moyenne. Pour la mémoire, il faudrait avoir une vision presque instantannée, or le trafic généré par un tel polling serait trop important. Ici, on se limitera à la prise "d'une photo" à un instant t et au relevé des problèmes survenus sur une longue période. Si le nombre de problèmes relevés sur une assez longue période est élevé, on peut en déduire un sous-dimensionnement du matériel ou un mauvaise répartition de la charge.

3.5.2 Le processeur

Pour la charge du processeur, nous sommes soumis au même problème, car l'utilisation CPU est une donnée évoluant très vite. On adopte donc la même politique de polling. Par contre c'est une autre variable que l'on va interroger : la variable *hrProcessorLoad* (voir la figure 14). Elle nous donne en pourcent le taux d'utilisation du processeur.

3.6 Mesurer le trafic vers le WAN et le trafic Web

Comme nous l'avons vu dans le premier chapitre nous avons décidé d'insérer une sonde RMON à la sortie du routeur Cisco vers le WAN. Cette sonde nous servira à mesurer le trafic WAN généré par chaque site (ce qui pourrait-être fait directement par le routeur), mais surtout à mesurer le trafic WEB en particulier (protocole http dans IP) (ce que ne peut pas faire le routeur). Cette dernière mesure justifie à elle seule l'achat d'une sonde RMON.

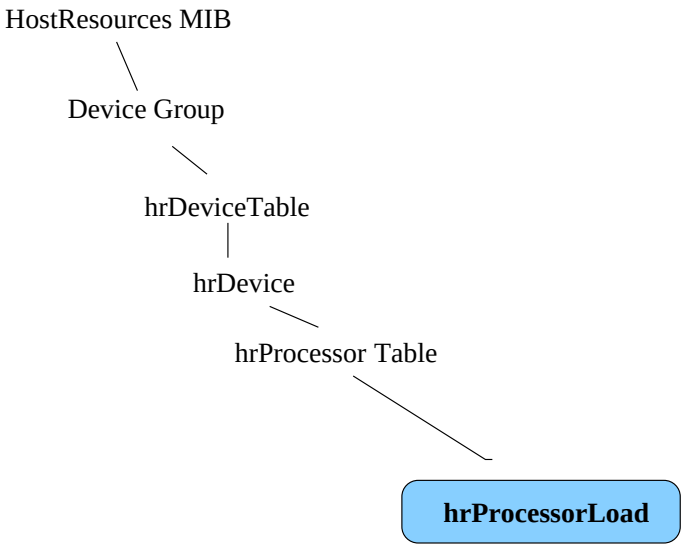

FIG. 14 – *Variable hrProcessorLoad de la MIB Host Resources*

3.7 Politique d'administration: les noms de communauté

Dans le deuxième chapitre de ce rapport nous avons expliqué que notre réseau pouvait être géré de deux façon, soit avec une gestion séparée de chaque site, soit avec une gestion centralisée de sites répartis quasiment autonomes. Nous avons donc choisi un système de noms de communauté pouvant être utilisé dans les deux cas.

Nous avons défini trois communautés:

La communauté PUBLIC: avec des droits read-only seulement et une vision de la plupart des variables (excepté les variables critiques comme la configuration de la sonde RMON par exemple).

La communauté ADMINDELEGUE: ayant une vision sur les mêmes variables que la communauté PUBLIC mais avec des droits Read-Write sur certaines d'entre elles pour permettre à un "administrateur délégué" d'effectuer des modifications simples (par exemple la configuration des imprimantes).

La communauté ADMINISTRATEUR: ayant tous les droits sur toutes les variables.

La communauté ADMINDELEGUE peut voir sa vision et ses droits aug-

mentés dans le cas de la gestion répartie du réseau. Cette communauté correspondra alors aux administrateurs locaux de chaques sites.

3.8 Vision globale des équipements et leur(s) MIB(s) sur le réseau

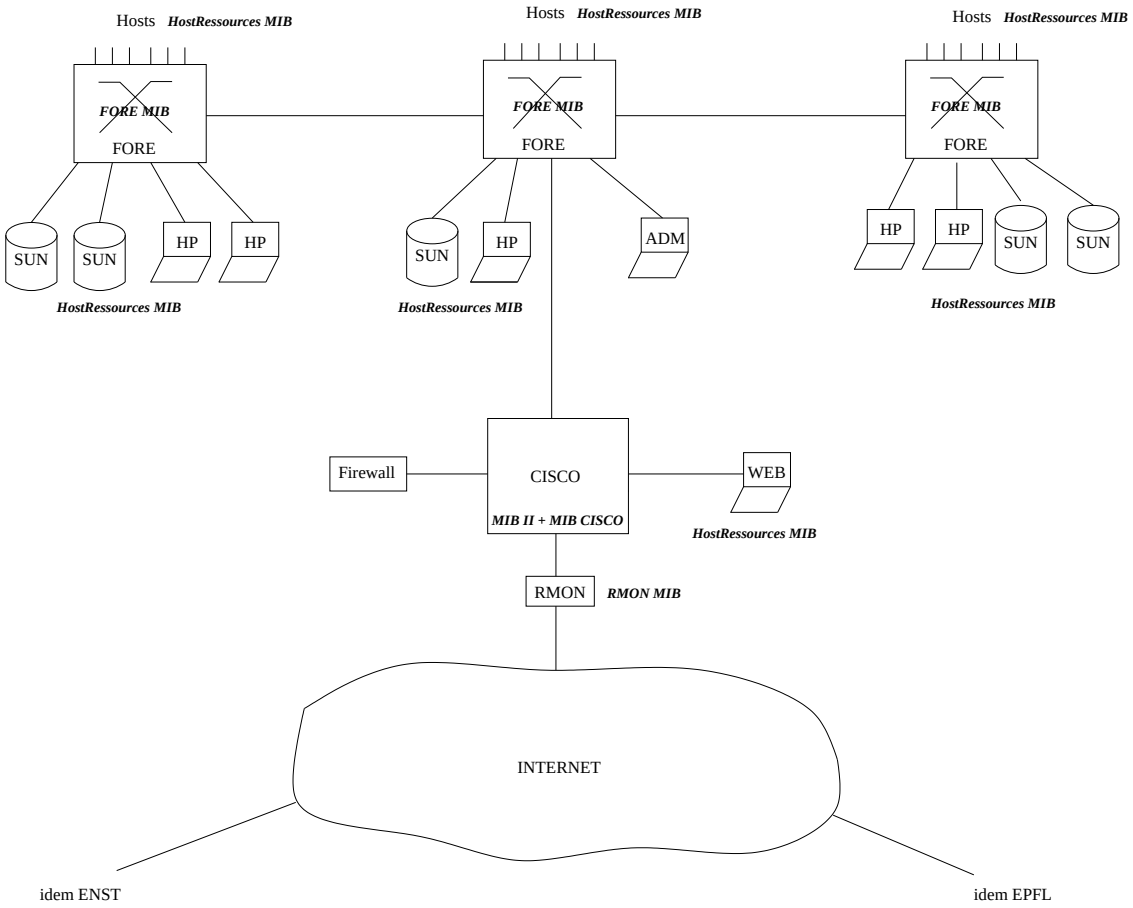

FIG. 15 – *Les équipements et leur(s) MIB(s)*

4 Conclusion

Les installations se voient de plus en plus réparties entre différents sites, et leur interconnexion pose un véritable défi à la gestion de réseaux. IBM, en fusionnant avec Tivoli, a compris les enjeux de ce nouveau paradigme, afin de fournir, dans le cas de notre étude notamment, une solution cohérente, homogène, robuste, sécurisée et évolutive. Nous savons que le travail n'en est pas pour autant terminé, et que la phase d'installation du produit prendra probablement un certain temps, dévolu notamment à la mise au point du matériel et à la formation du personnel. Mais nous restons convaincus que vous trouverez entière satisfaction à l'utilisation quotidienne de cette installation. Et n'est-ce pas là le principal?