MASTER EN SCIENCES INFORMATIQUES

Développement d'un Agent de Mesures de Performances pour la Gestion des Systèmes Distribués

Nosbusch, Marc

Award date:
1995

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix, NAMUR

Institut d'Informatique

Développement d'un Agent de Mesures de Performances pour la Gestion des Systèmes Distribués.

Marc NOSBUSCH

Promoteur : Jean RAMAEKERS

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

Année Académique 1994-1995

Résumé.

Il n'y a pas de doute que les systèmes distribués joueront un rôle stratégique dans les entreprises de demain. Mais pour pouvoir tirer pleinement profit de ces systèmes fortement hétérogènes, une gestion appropriée avec des outils puissants s'impose. Cependant il est impensable de concevoir la gestion des systèmes distribués sans considérer la gestion des performances.

Ce mémoire va montrer comment un agent de mesures de performances peut être intégré dans une plate-forme de gestion existante. Afin de situer le contexte, les systèmes distribués et leur gestion vont d'abord être introduits. Ensuite, nous allons étudier en détail une partie de la gestion, à savoir la gestion des performances. L'étude de cas qui va suivre s'articule autour de deux grandes parties. La première étant la présentation de la plate-forme de gestion ISM de Bull, la seconde, quant à elle va exposer la gestion des performances sur cette même plate-forme. Dans cette dernière partie, le développement et le fonctionnement d'un agent de mesures de performances sera décrit.

Abstract.

There is no doubt that distributed systems will play a strategic role in tomorrow's enterprises. In order to take advantage of these highly heterogenous systems, an appropriate management with powerful tools is necessary. Nevertheless it is unimaginable to conceive the management of distributed systems without considering performance management.

This thesis will show how to integrate a performance measuring agent into an existing management platform. To situate the context, distributed systems and their management will be introduced first. Afterwards, we will focus on a specific part of the management, namely the performance management. The case study that follows is divided into two parts: on the one hand the presentation of Bull's management platform ISM and on the other hand the performance management on it. In this latter part, the development and operation of a performance measuring agent is described.

Remerciements.

Avec ces quelques lignes j'aimerais remercier toutes les personnes qui d'une façon ou d'une autre ont contribué à la réalisation de ce mémoire.

Tout d'abord Monsieur Jean Ramaekers, mon promoteur, grâce à qui j'ai pu effectuer mon stage chez Bull à Paris. Ses conseils judicieux et son appui m'ont aidé beaucoup au cours de l'élaboration du mémoire.

Je tiens à remercier sincèrement Hugues Deghorain qui a fait de son mieux durant et même après le stage pour me soutenir et pour m'offrir les meilleures conditions de travail.

Merci aussi à tous ceux chez Bull qui m'ont aidé et encouragé: Philippe, Alain, Etienne, Jean, Jean-Pierre, Marc, Marie-Dominique, Thierry, Raphaël et tous les autres.

J'aimerais remercier également Joël Hubin pour ses remarques pertinentes et ses conseils. De même, un grand merci pour les encouragements ainsi que pour les critiques nombreuses et constructives de Stefan Vincent lors de la rédaction du mémoire.

Merci enfin à Sandra, Luc et à tous mes amis de la Fondation Biermans-Lapôtre pour leur soutien amical. J'aimerais également exprimer ma profonde gratitude à mes parents pour leurs affection et encouragements tout au long de mes études.

Table des matières.

Table des figures.

Chapitre 1

Introduction

1. Introduction.

Le sujet de ce mémoire est le développement d'un agent de mesures de performances pour la gestion des systèmes distribués. Notre but est de montrer comment nous pouvons étendre une solution de gestion existante en intégrant un agent. Naturellement nous exposerons aussi le fonctionnement de l'agent.

Pour pouvoir situer l'agent, il fallait d'abord introduire les concepts clés de système distribué et de sa gestion. Ensuite, comme l'agent a été conçu pour la solution de gestion *Integrated System Management* (*ISM*) de Bull, nous allons présenter cette solution.

L'agent peut servir à effectuer des mesures de performances. Par conséquent nous avons consacré un chapitre à la gestion des performances dans le cadre des systèmes distribués.

Dans le chapitre 2 nous allons introduire les systèmes distribués et leur gestion. Nous ferons la découpe de l'activité de gestion selon trois dimensions. Ensuite nous aborderons la gestion un peu plus concrètement, en partant des informations de gestion. Pour conclure nous exposerons la gestion dans le monde du *Internet Activities Board* (IAB).

Le chapitre suivant sur la gestion des performances comprend 7 sections. D'abord nous essayerons de définir la notion de performance. Dans la section suivante nous détaillerons l'activité de gestion des performances en dégageant ses différentes phases. Par après nous examinerons les outils et les méthodes de mesures qui sont utilisés pendant l'activité de gestion des performances. Pour mieux l'illustrer, nous donnerons des exemples d'outils aux différents niveaux d'un système distribué. Finalement nous évoquerons quelques problèmes relatifs aux mesures de performances.

La première partie de l'étude de cas qui va suivre, traite la plate-forme de gestion *Integrated Systems Management* (ISM) de Bull. Après une présentation générale, l'architecture du manager/ISM et des agents va être détaillée. La description du manager comprend l'étude des services, des agents intégrateurs et du CMIS Dispatcher.

La deuxième partie de l'étude de cas analyse les performances sous ISM. D'abord nous décrirons le développement et le fonctionnement d'un agent SNMP de mesures de performances pour ISM. Ensuite nous examinerons le service de performances *PMS* existant sous ISM.

Chapitre 2

La gestion des systèmes distribués

2. La Gestion des Systèmes Distribués.

2.1. Introduction.

Le thème principal de ce chapitre est la ***gestion des systèmes distribués***. Mais, avant de nous lancer dans le vif du sujet, nous devons d'abord expliciter la notion de système distribué. Dans la suite, nous donnerons un aperçu de la gestion des systèmes distribués. L'activité de gestion peut être vue selon trois dimensions, que nous allons exposer dans la section 2.4. Dans la section suivante nous aborderons par le biais d'un modèle de gestion, le fonctionnement concret de la gestion. Ce modèle va être decrit en partant des informations de gestion. Pour illustrer cet exposé nous donnerons un exemple de gestion.

2.2. La notion de système distribué.

Dans un premier temps, nous essayerons d'éclaircir le concept de système distribué par la présentation de ses caractéristiques principales. Pour l'illustrer, nous donnerons un exemple concret. De cet exemple nous pourrons déduire les différentes composantes intervenant dans un système distribué. Les éléments d'un sous-ensemble de ces composantes peuvent être découpés en différentes couches. Ensuite, nous allons donc décrire ces différentes couches. Pour finir nous discuterons les avantages et les désavantages d'un système distribué.

2.2.1. Un essai de définition.

De nombreuses définitions d'un ***système distribué*** ont été données, mais aucune de celles-ci n'a reçu l'accord de toute la communauté scientifique. Nous allons essayer de donner une définition assez générale d'un système distribué. Elle est inspirée de celle qu'*Amjad Umar* donne au début de son livre[1] et des caractérisations que *Coulouris*[2] et *Hegering*[3] évoquent.

[1] [Umar,1993] page 5

[2] [Coulouris,1988]

J'aimais bien ces définitions complémentaires, puisqu'une approche pas trop restrictive, basée sur les caractéristiques principales me paraissait intéressante.

Un système informatique est ***centralisé***, si ses *composantes*[4] se trouvent sur un même site. Par contre un système est ***décentralisé***, si ses composantes se trouvent sur des sites différents. Pour qu'on puisse parler de système décentralisé, il faut en plus que la coordination entre les composantes soit limitée voir inexistante. Un système décentralisé peut être qualifié de ***système distribué*** si ses *éléments*[5] sont autonomes et leurs opérations sont coordonnées par un mécanisme global.

Pour mieux comprendre cet essai de définition, analysons dès à présent plus en détail les caractéristiques d'un système distribué. Ces caractéristiques sont: la transparence, l'hétérogénéité, la séparation spatiale et l'autonomie.

Une première caractéristique est donc la ***transparence***. Nous qualifions un système comme étant transparent s'il peut être vu comme un ensemble, plutôt qu'un assemblage de parties indépendantes. Par conséquent un utilisateur ne devrait pas se rendre compte qu'il accède à une ressource distante. Tout devrait se passer comme s'il se trouvait devant une grande machine virtuelle. Les systèmes non transparents par contre n'offrent pas cette possibilité. En effet l'utilisateur doit spécifier explicitement les ressources auxquelles il veut accéder. Les deux stades extrêmes sont les systèmes totalement transparents et les systèmes non transparents. Entre ces deux types se situe tout un ensemble de systèmes distribués partiellement transparents. Un système non-transparent peut encore être qualifié de réseau d'ordinateurs. Dans les systèmes partiellement transparents uniquement certaines fonctions le sont, comme par exemple le système de fichiers. Pour plus de détails sur les différents aspects de la transparence, le lecteur intéressé peut se référer au livre de *Coulouris*[6] et au mémoire de *Stefan Vincent*[7].

L'***hétérogénéité*** est la deuxième caractéristique d'un système distribué. En effet nous trouvons souvent une multitude de fournisseurs, de protocoles, d'architectures, de périphériques, d'interfaces homme-machine, de systèmes d'exploitation différents. Bien que l'hétérogénéité soit une caractéristique fréquente, un système distribué peut tout aussi bien être parfaitement homogène.

---

[3] [Hegering,1994]

[4] La définition d'une "composante" se trouve à la sous-section: *2.2.3 Les composantes d'un système distribué.*

[5] La définition d'un "élément" se trouve à la sous-section: *2.2.4 Les différentes couches.*

[6] [Coulouris,1988]

[7] [Vincent,1994]

La caractéristique suivante d'un système distribué est la ***séparation spatiale***. Nous supposons que les différents éléments, qui interviennent dans le système distribué, ne possèdent pas de mémoire commune et qu'elles ne communiquent que par des connexions réseaux. Ces systèmes sont appelés systèmes distribués à processeurs faiblement couplés ("*loosely coupled*"). De cette manière nous excluons les systèmes multiprocesseurs, qui sont encore qualifiés de fortement couplés ("*tightly coupled*")[8]. Vu cette propriété nous devons toujours prendre en considération des délais non-négligeables lors des échanges de messages.

L'autonomie de ses éléments est la dernière caractéristique d'un système distribué, que nous allons évoquer. En effet les composantes n'ont pas de connaissances exactes de l'état des autres parties du système distribué. Ceci est dû principalement aux délais de transmission des messages. Pendant ce délai un nombre élevé d'instructions a été exécuté dans les ordinateurs connectés et l'état du système a changé. Mais nous supposons qu'il existe des mécanismes de coordination entre les différents éléments.

Dans la suite quand nous parlerons d'un système distribué, il s'agira d'un système distribué faiblement couplé, partiellement transparent qui opère dans le domaine de l'informatique de gestion.

[8] [Coulouris,1988] donne plus de précisions sur les systèmes faiblement et fortement couplés.

2.2.2. Un exemple de système distribué.

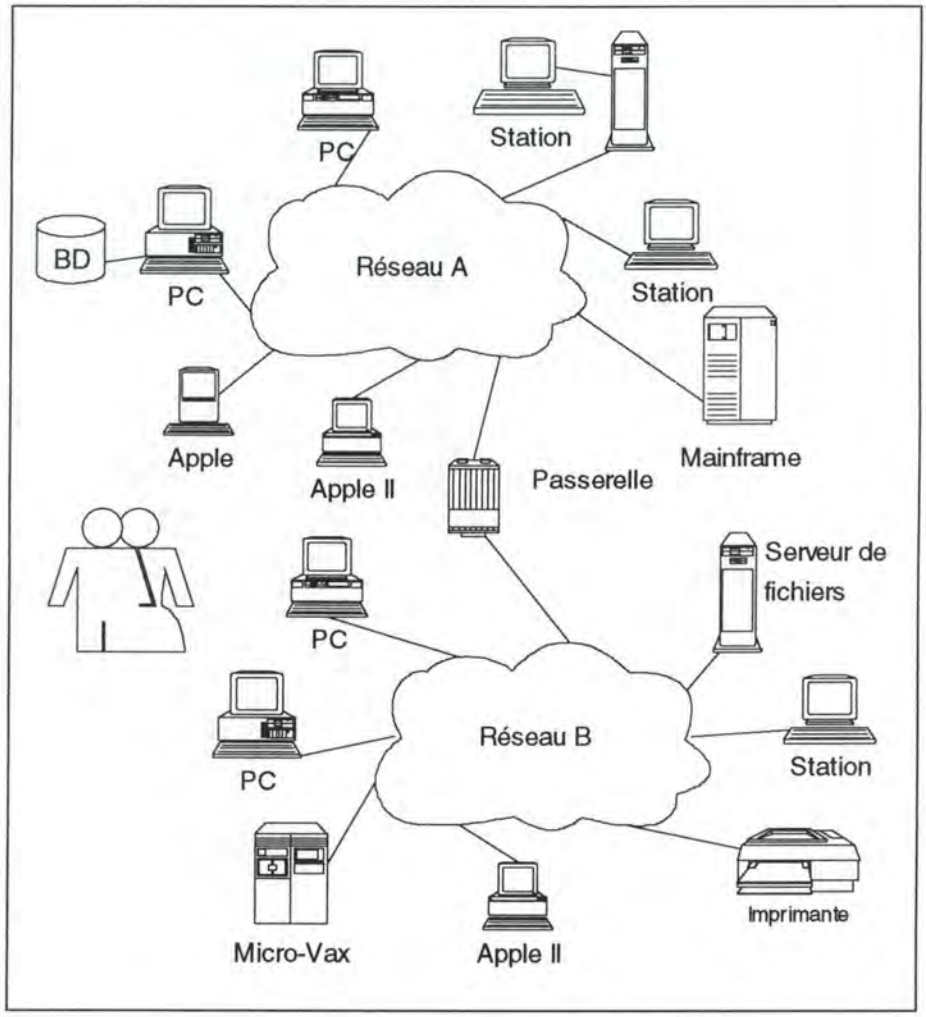

Figure 1 : Un exemple de système distribué.

Physiquement un système distribué se concrétise par un assemblage des différentes composantes qui sont énumérées dans la sous-section suivante. Au lieu de donner une définition stricte d'un système distribué, nous avons choisi une approche moins restrictive de définition par les caractéristiques ou symptômes. Sur l'exemple représenté sur la Figure 1 nous dégagerons les caractéristiques précitées.

La première caractéristique qui est bien reflétée par l'exemple, est ***l'hétérogénéité*** de notre système distribué. En effet nous voyons une myriade d'éléments différents. La ***séparation spatiale*** des éléments est la deuxième caractéristique que nous remarquons. Elle entraîne que tous les éléments communiquent entre eux par des réseaux. La caractéristique suivante est la ***transparence***. Ainsi par exemple l'accès au serveur de fichiers se fait de façon transparente, sans que l'utilisateur se rende compte qu'il accède à une ressource distante. ***L'autonomie*** des éléments est la dernière caractéristique. Les différents éléments doivent fonctionner de

manière indépendante. Les composantes que nous pouvons dégager de cet exemple seront présentées maintenant.

2.2.3. Les composantes d'un système distribué.

Un système distribué comprend concrètement:

- Des ***systèmes de communication***. Ce sont tous les dispositifs qui servent à mettre en place des réseaux. Les réseaux comprennent des réseaux locaux (exemples: Ethernet, Token Ring, Token Bus), des réseaux métropolitains et des réseaux à grande distance.
- Des ***systèmes intermédiaires***. Par systèmes intermédiaires nous entendons des dispositifs qui permettent d'interconnecter des réseaux. Des exemples de tels systèmes sont des passerelles, des routeurs.
- Des ***systèmes terminaux***. Les ordinateurs, les stations de travail et l'ensemble des périphériques constituent les systèmes terminaux.
- Des ***données***. Les données englobent les applications, le système d'exploitation et les données au sens propre.
- Des ***hommes***. N'oublions pas les hommes. Finalement ce sont eux qui utilisent et gèrent le système distribué. Notons que le terme technique de "composantes" n'est pas le mieux adapté pour qualifier des êtres humains. Dans la suite de notre exposé nous ne considérerons donc pas les hommes parmi les "composantes".

2.2.4. Les différentes couches.

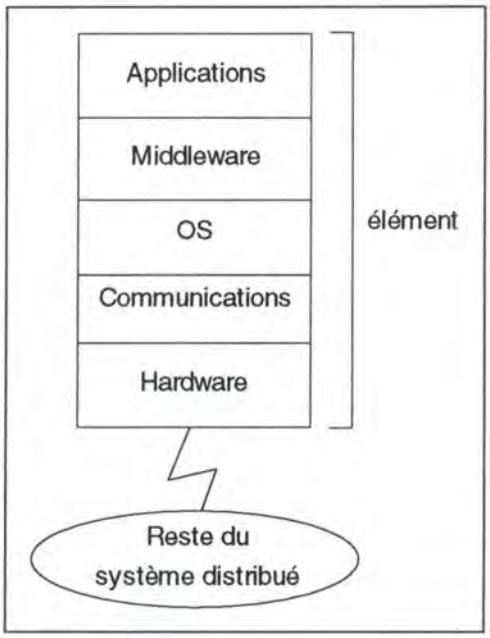

Figure 2 : Les couches d'un élément de système distribué.

Un ***élément*** d'un système distribué est restreint à un des types de composantes suivants: système intermédiaire ou système terminal. Il faut en outre que l'élément soit autonome. Comme nous pouvons le constater sur la Figure 2, un élément peut être découpé en cinq ***couches*** différentes. Toutes ces couches ne sont pas nécessairement présentes dans chaque élément. Mais nous trouvons au sein de tout élément au moins les couches hardware et communication. Nous sommes ici en présence du principe de la boîte noire. En effet chaque couche offre un certain nombre de services qu'elle met uniquement à la disposition des couches supérieures. Celles-ci peuvent invoquer les services, sans devoir connaître la façon selon laquelle ils sont implémentés.

Passons maintenant en revue les diverses couches. Au niveau ***hardware*** nous trouvons par exemple des ordinateurs, des périphériques, des modems, des routeurs, etc. La couche de ***communication*** permet d'interconnecter les composantes à l'aide de protocoles de communication. Des exemples de tels protocoles sont les stacks OSI et TCP/IP. Au niveau de la couche ***OS*** (*"Operating System"*) se trouvent les fonctionnalités du système d'exploitation comme par exemple la gestion des fichiers, de la mémoire, des processus. La couche au-dessus, le ***middleware***, s'occupe de tâches aussi variées comme la gestion des bases de données et de la gestion de la distribution. Les ***applications***, avec une interface homme-machine se trouvent au niveau supérieur.

2.2.5. Les avantages.

Maintenant que nous avons planté le décors en caractérisant les systèmes distribués, citons quelques-uns de leurs avantages. Les trois premiers sont inspirés par ceux de *John Donovan*[9]:

- ***Facteurs stratégiques***. Un système distribué se greffe très bien sur la structure des organisations. Les centres de décision d'une entreprise sont souvent répartis à différents endroits, dans des filiales ou des antennes décentralisées. De même, les entreprises avec leurs clients et fournisseurs se trouvent rarement sur un seul site. Un système distribué qui permet de "souder" ces différents acteurs peut donner un avantage compétitif aux organisations qui le détiennent.
- ***Flexibilité***. Un des atouts majeurs des systèmes distribués est leur flexibilité. On peut ajouter ou retirer facilement des composantes sans devoir arrêter tout le système. Par conséquent une organisation peut se confectionner son propre système distribué, qui croît de façon incrémentale selon ses besoins.
- ***Coûts d'acquisition du matériel***. Un autre point important sont les coûts d'acquisition du matériel. Par exemple le coût par MIPS[10] d'un mainframe est beaucoup plus élevé que celui d'une station de travail. *Donovan* affirme que 20 MIPS sur mainframe coûtent environ 4 millions de dollars, tandis que 20 MIPS sur une station de travail coûtent moins que 10000 dollars. Notons que cette différence de prix non négligeable est en train de diminuer fortement pour le moment. En plus l'investissement dans un système distribué peut se faire graduellement, car il permet une croissance continue. Un mainframe par contre, doit être remplacé complètement après un certain nombre d'extensions.
- ***Liberté***. Les systèmes distribués permettent une grande diversité de composantes. Des composantes de types et de fournisseurs différents peuvent cohabiter. De cette manière les entreprises ne sont pas liées à des fournisseurs particuliers. Elles peuvent choisir les meilleures composantes sur le marché en ce qui concerne les performances et le prix.

Les autres avantages que *Donovan* cite ne me semblaient pas appropriés puisqu'ils ne sont pas spécifiques aux systèmes distribués.

---

[9] John Donavan cité dans [Umar,1993] page 7

[10] MIPS : *Million Instructions Per second* (indicateur des performances d'un processeur)

2.2.6. Les désavantages.

Après avoir vu les avantages, nous montrerons aussi quelques inconvénients des systèmes distribués.

- ***Coûts de développement***. Les coûts de développement de larges applications distribuées sont plus élevés que dans le cas des systèmes centralisés. Actuellement des méthodologies de développement d'applications distribuées ne sont pas encore arrivées dans un stade de maturité et des langages et des outils ne commencent qu'à émerger. Dès que des outils et des méthodes puissantes seront au point, les coûts de développement vont diminuer fortement.
- ***Manque de standards***. Le développement et le renforcement de standards pour les applications, les ordinateurs et les réseaux est un processus lent et difficile. Un manque de standards et des standards imprécis entraînent que la compatibilité et l'interopérabilité au sein du système distribué ne sont pas toujours garanties.
- ***Contrôle de la sécurité et de l'intégrité***. L'informatique distribuée intensifie les problèmes relatifs au contrôle de l'intégrité et à la sécurité des données et des applications. Ceci est un des grands problèmes qu'il faudra résoudre.
- ***Gestion alourdie***. Finalement la gestion d'un système distribué devient plus lourde. D'où la nécessité d'une gestion puissante qui tient compte de tous les aspects de la distribution. Dans la section suivante nous allons justement nous focaliser sur la gestion.

2.3. La gestion des systèmes distribués.

2.3.1. La notion de gestion.

La définition de la gestion des systèmes distribués qui va suivre est inspirée de celle que *Hegering*[11] donne au début de son livre. Elle reflète bien les différents aspects de la gestion, mais elle dépasse le cadre de notre analyse sur certains points. Je me suis donc permis de ne reprendre que les points qui me paraissaient essentiels.

[11] extrait de [Hegering,1994] page 64

Le terme de **gestion des systèmes distribués** *est défini comme étant l'ensemble des* **tâches** *et* **outils** *utilisés pour planifier, configurer, contrôler, monitorer des systèmes distribués et pour enlever des erreurs dans ceux-ci. L'activité de gestion des systèmes distribués englobe toutes les tâches et les outils qui permettent une utilisation efficace, optimale et sûre des composantes distribués.*

Les différentes ***tâches*** de l'activité de gestion des systèmes distribués s'inscrivent dans les trois dimensions de la gestion qui seront présentées à la section 2.4.

Dans la sous-section 2.3.3 nous allons proposer une classification des ***outils*** selon leur degré d'intégration. Dans la suite nous allons désigner l'ensemble des outils qui servent à la gestion d'un système distribué particulier, par le terme: ***solution de gestion***.

N'oublions pas qu'il faut organiser et planifier le travail du personnel impliqué dans la gestion. Les différents ***intervenants humains*** seront présentés à la sous-section 2.3.5.

La forme que prend la gestion dans une organisation particulière est conditionnée par de nombreux facteurs. Nous ne détaillerons que les suivants:

- ***la complexité du système distribué sous-jacent*** influence directement la gestion. Le nombre de composantes, leur répartition géographique et le "degré" d'hétérogénéité ont un impact direct sur la forme de la gestion.
- ***la qualification du personnel*** détermine quelle solution de gestion va être retenue. Souvent le personnel qui s'occupe de la gestion a encore d'autres missions.
- ***l'importance que l'organisation attache à son système distribué***. Ce facteur est important et ne serait-ce que pour les moyens budgétaires qui sont libérés pour la gestion.

Après avoir défini la gestion et énuméré quelques facteurs qui la conditionnent nous nous concentrerons maintenant sur les solutions de gestion. Nous allons commencer par les motivations qui sont à l'origine de l'utilisation d'une solution de gestion.

2.3.2. Les motivations.

Dans les paragraphes qui suivent nous allons exposer pourquoi il est important d'utiliser une solution de gestion efficace.

La gestion des systèmes distribués est ***plus difficile*** que dans le cas des systèmes centralisés. Mais une bonne solution de gestion devrait permettre un meilleur rendement et un amortissement rapide des composantes distribuées. Dans le cadre des grands systèmes centralisés elle est bien maîtrisée. Ceci est dû à deux constatations: premièrement toute "l'intelligence" du système est concentrée sur une machine. En plus les équipements proviennent le plus souvent d'un seul constructeur. Dans la plupart des cas un produit de gestion propriétaire suffit aux besoins. Deuxièmement la longue expérience d'utilisation avec de tels systèmes et leur stabilité fait que leur gestion est relativement bien maîtrisée.

L'***hétérogénéité*** des systèmes distribués entraîne de nouveaux problèmes. En effet pour la plupart des composantes il existe des outils de gestion variés avec tout un arsenal d'interfaces homme-machine différentes. Un gestionnaire, même avec un savoir éclectique dans le domaine de la gestion, risque de ne plus avoir une vue d'ensemble cohérente du système distribué. D'où la nécessité d'utiliser une bonne solution de gestion.

La ***répartition géographique*** des composantes nécessite aussi une bonne solution de gestion. En effet comment configurer, contrôler et monitorer des composantes distribués sans disposer d'outils adéquats.

Dans la suite nous distinguerons les différents degrés d'intégration des outils qui composent une solution de gestion. Ensuite nous examinerons les objectifs d'une solution de gestion.

2.3.3. Les outils.

Nous venons de voir qu'une solution de gestion se matérialise par un ensemble d'outils. Dans ce paragraphe nous allons décrire les grandes catégories d'outils de gestion, qui servent à la gestion des systèmes distribués. Trois catégories sont crées pour délimiter les outils de gestion selon leur degré d'intégration[12]. Les catégories reprennent les outils isolés, coordonnés et intégrés.

Une première approche consiste à avoir des outils ***isolés***. Ces outils individuels sont souvent hétérogènes, propriétaires et ne disposent pas d'une interface homme-machine uniforme. L'inconvénient est que le responsable ne dispose pas d'une vue globale du système distribué mais

[12] extrait de [Hegering,1994] page 92

uniquement de vues partielles sur quelques parties (p.ex.: le système de fichiers) ou une vue partagée selon les constructeurs. Cette approche se justifie pour des petits systèmes qui proviennent d'un seul fournisseur.

Pour améliorer le rendement, des outils isolés peuvent être ***coordonnés*** et regroupés. Ainsi les résultats d'un outil peuvent être utilisés par un autre. La coordination, obtenue de cette manière, définit les relations entre les différents outils. Ces deux façons de procéder ont l'inconvénient que dans la plupart des cas le gestionnaire doit connaître la multitude des outils hétérogènes et leurs différentes interfaces. De ce fait le temps d'apprentissage est augmenté. Pour remédier à ce problème tous les outils peuvent être ***intégrés*** dans un seul, à savoir une plate-forme de gestion avec une seule interface homme-machine cohérente. Ceci entraîne que le gestionnaire n'a plus que la logique d'utilisation d'un seul outil à apprendre. Dans le chapitre 4 nous allons décrire la solution de gestion intégrée *ISM* proposée par Bull.

2.3.4. Les objectifs d'une solution de gestion.

Une solution de gestion efficace devrait garantir un système distribué performant et fiable. Ce dernier est nécessaire pour permettre aux entreprises de rester compétitives, de s'adapter plus rapidement aux fluctuations du marché. Pour avoir une avance sur la concurrence, un système distribué performant est crucial. Pour répondre à ces exigences une solution de gestion doit être:

- ***conviviale*** : La gestion doit pouvoir recourir à des outils qui offrent une interface homme-machine cohérente et conviviale. Ceci pour réduire le temps d'apprentissage, d'augmenter la facilité d'utilisation et la rapidité d'intervention.
- ***fiable*** : La solution de gestion devrait être plus fiable que le système distribué. En principe même si le système distribué tombe en panne, la solution de gestion (ou une partie de celle-ci) devrait rester opérationnelle pour pouvoir réparer la panne.
- ***appropriée*** : La solution de gestion doit être appropriée au système distribué. Il faut que ses coûts d'acquisitions soient en rapport avec ceux du système distribué. Les fonctionnalités qu'elle offre doivent aussi être appropriées. Comparée au système distribué, la solution de gestion ne doit pas être sur-dimensionnée.

- ***sûre*** : La solution de gestion doit garantir la sécurité du matériel et des informations. C'est-à-dire elle doit offrir une protection contre les accidents, les erreurs et le sabotage. Pour pouvoir faire cette gestion de la sécurité[13], la sécurité de la solution de gestion doit d'abord être assurée.

Les outils et les solutions de gestion étaient au centre de notre propos dans les trois dernières sous-sections. Dorénavant nous présenterons les intervenants humains qui manipulent ces outils.

2.3.5. Les intervenants humains.

Jusqu'à présent nous avons toujours utilisé le terme générique de "gestionnaires" pour qualifier le personnel impliqué dans la gestion. Maintenant nous analyserons plus en détail les différentes groupes de "gestionnaires". Un large éventail de tâches intervient dans la gestion d'un système distribué. Celles-ci sont accomplies par des intervenants qui se voient attribué différents rôles. Un intervenant peut aussi remplir plusieurs rôles. Examinons dès à présent ces rôles:

- ***Le responsable stratégique*** s'occupe de la gestion à moyen et à long terme. Il acquiert des outils de gestion qui correspondent aux besoins des utilisateurs du système distribué. En outre il doit garantir la cohérence et la compatibilité de la configuration du système. Pour accomplir sa tâche il utilise des rapports de gestion et sporadiquement les outils de gestion.
- ***L'administrateur*** a pour mission l'installation et la configuration des outils de gestion.
- ***L'opérateur*** s'occupe de la gestion quotidienne du système distribué. Il doit être disponible pour répondre aux requêtes des utilisateurs. D'autres tâches comprennent le contrôle des composantes, des tests, l'évaluation des alarmes et du niveau de performance du système. Il est impossible d'énumérer la multitude des tâches qui interviennent au niveau opérationnel. Les outils de gestion sont utilisés intensément à ce niveau pour surveiller et contrôler le système distribué.
- ***Le développeur d'applications*** de gestion étend ou adapte les solutions de gestion existantes. Il utilise des interfaces de programmation, des boîtes à outils et des compilateurs.

Nous venons d'introduire la gestion des systèmes distribués. Dans notre exposé nous nous sommes attardés sur les outils de gestion et les intervenants humains. Etudiant dès lors les trois dimensions de la gestion.

---

[13] Nous reparlerons de la gestion de la sécurité dans la section *2.4.1 La dimension fonctionnelle.*

2.4. Les trois dimensions de la gestion.

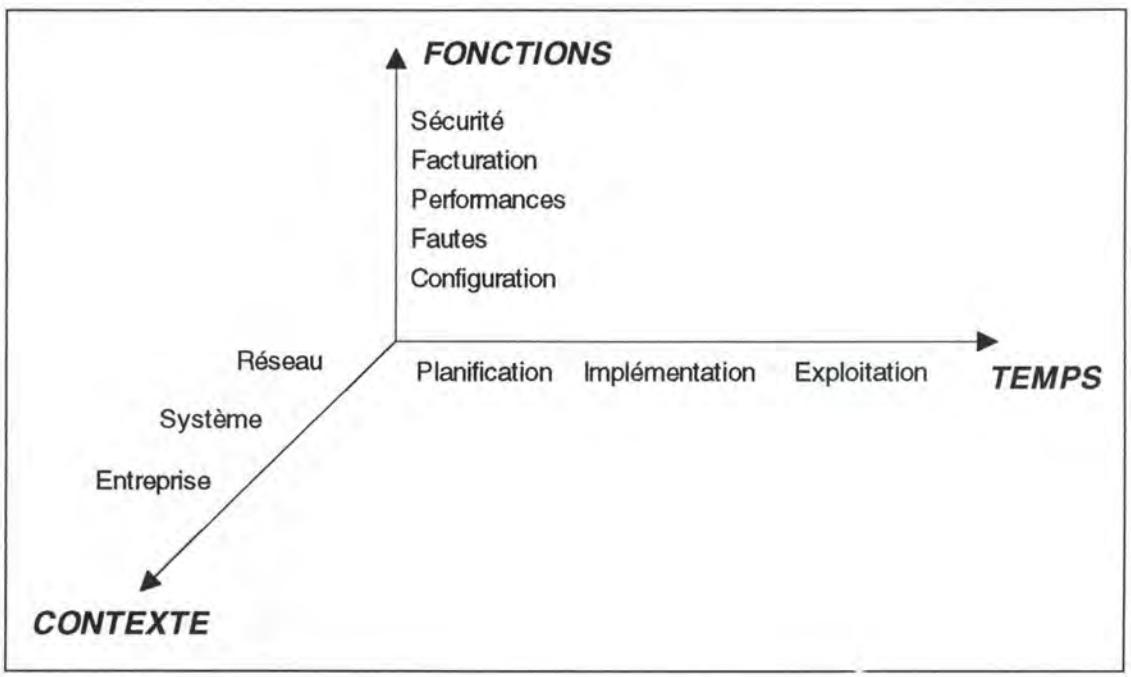

Figure 3 : Les trois dimensions de la gestion.

Dans cette section nous allons essayer de caractériser l'activité de gestion selon trois critères. Nous ne donnerons pas une nouvelle représentation de l'activité de gestion mais nous faisons seulement une classification de ses tâches. Cette classification de la gestion[14] se fera selon les dimensions représentées par la Figure 3.

2.4.1. La dimension fonctionnelle.

L'ISO a découpé la gestion selon les cinq fonctions[15] suivantes: configuration, fautes, performances, facturation et sécurité. Cette taxonomie est la plus intéressante et en plus elle a été reprise par presque toutes les organisations concernées par la gestion.

- ***La gestion de la configuration.***

Les composantes d'un système distribué doivent coopérer de façon efficace. La gestion de la configuration doit permettre d'établir des liens entre les composantes diverses. Pour accomplir cette tâche nous avons

[14] inspirée de [Hegering,1994] pages 84 et suivantes

[15] source: "*ISO Management Framework*" (norme ISO 7498-4)

besoin de la description du système distribué. Trouver une représentation adéquate de la configuration est devenue un problème crucial de la gestion de la configuration. La Figure 4 ci-dessous donne un aperçu des informations nécessaires rien que pour la partie réseau. Nous constatons que les informations géographiques, topologiques et organisationnelles sont représentées de façon hiérarchique.

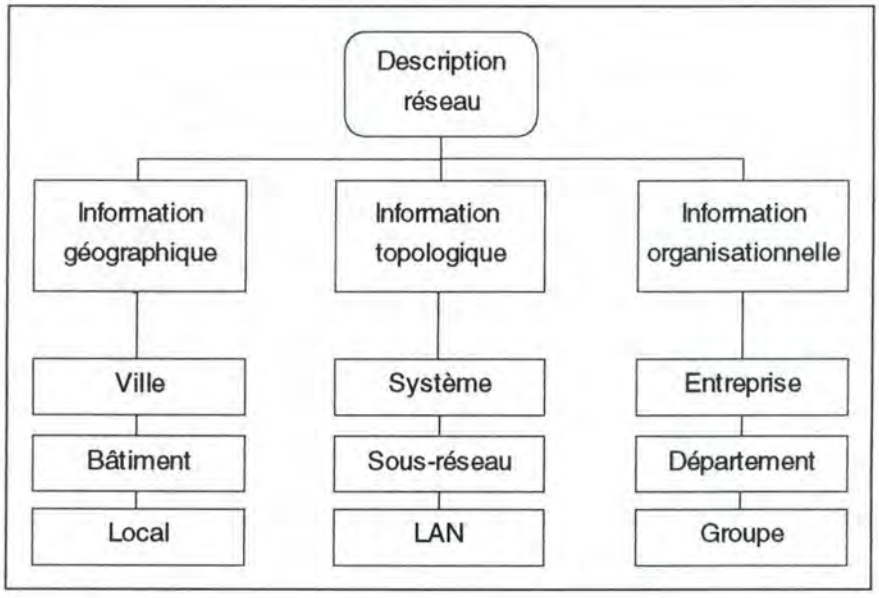

Figure 4 : Une description schématique d'un réseau.

La gestion de la configuration comprend les sous-tâches suivantes: la mise-à-jour automatique de la configuration qui permet de diminuer et de faciliter le travail des opérateurs, la configuration à distance, la reconfiguration dynamique des ressources (en cas de problèmes, de surcharges) , l'initiation de travaux par lots et le suivi de leur exécution.

- ***La gestion des fautes.***

L'objectif de la gestion des fautes est de maintenir le système distribué à une disponibilité maximale. La Figure 5 montre les différentes sous-tâches de la gestion des fautes. Celles-ci contribuent à la réalisation de cet objectif: la réception et le traitement d'alarmes venant de composantes, le monitoring de l'état du système, le diagnostic des erreurs ainsi que la détection d'une propagation d'erreurs, la fourniture d'un service de support aux utilisateurs, l'introduction d'un système de documentation d'erreurs (*"Trouble Ticket System"*), la gestion de la récupération des erreurs.

Un *Trouble Ticket System*[16] est un outil qui sert à documenter des erreurs survenues. Il doit pouvoir enregistrer les erreurs et les analyser le cas échéant. Chaque *Trouble Ticket System* comprend une base de données qui

[16] Le lecteur intéressé par les *Trouble Ticket Systems* pourra consulter [Hegering,1994] pages 303 et suivantes.

contient des Trouble Tickets ou encore des documentations d'erreurs, générés par des personnes ou des applications de gestion. Dans une configuration minimale nous avons au moins un module d'entrée qui permet de remplir la base de données et un module de sortie qui permet de lire les enregistrements.

Le problème principal, dans la gestion des fautes est le diagnostic. L'utilisation de techniques d'intelligence artificielle, comme les systèmes experts, n'est pas évidente. Le lecteur qui est intéressé par l'utilisation de systèmes experts dans la gestion des fautes pourra consulter l'article [Tarouco,1989].

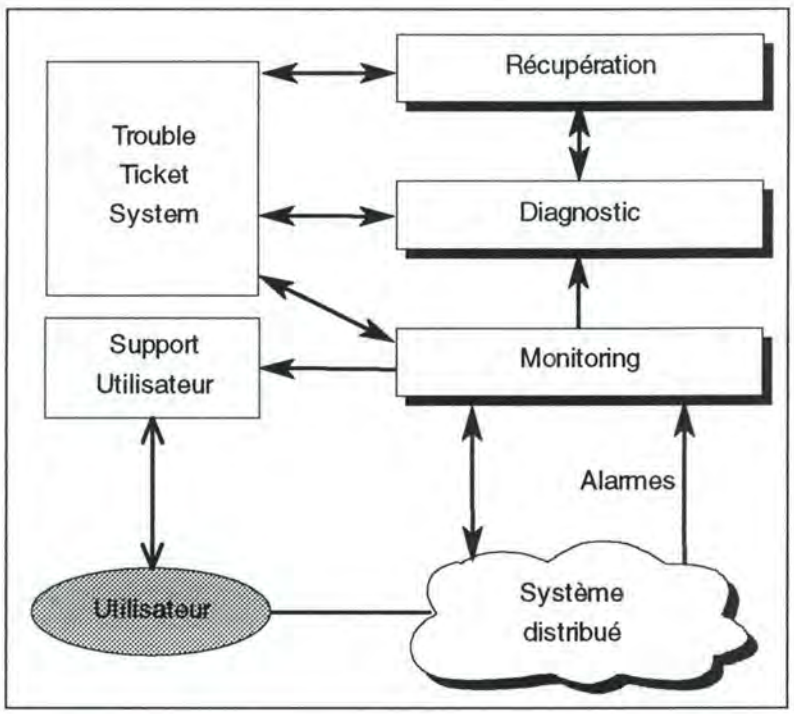

Figure 5 : Un système de gestion de fautes.

- ***La gestion de la facturation.***

Les frais d'exploitation d'un système distribué comprennent, entre autres, l'utilisation d'un serveur d'impression, d'un réseau local, de temps processeur, de papier, d'encre pour imprimante, etc.. Pour garantir une répartition équitable des coûts entre les différents services, départements et organisations utilisatrices, une première tâche consiste à élaborer des stratégies de facturation, qui reposent sur une politique de facturation propre à l'entreprise. Dans un premier temps, des informations, concernant l'utilisation des ressources, doivent être collectées et enregistrées dans des comptes utilisateurs. Ce n'est qu'ensuite que des coûts y seront imputés. D'autres tâches consistent à assigner des quotas d'utilisation, le monitoring de ceux-ci et la maintenance de statistiques d'utilisation. La collecte des informations nécessaires à la facturation peut être couplée à celle des mesures de performances pour plus d'efficacité et pour éviter le gaspillage

de bande passante. Finalement la facturation permet aussi de donner une idée sur le coût global de l'exploitation du système distribué. De cette manière la rentabilité d'un système distribué peut être évaluée plus exactement.

- ***La gestion de la sécurité.***

La sécurité est un des facteurs les plus importants de la gestion. En effet elle n'est plus uniquement nécessaire dans les domaines militaires et financiers, mais partout où transitent des informations "sensibles" par des réseaux et à travers des systèmes informatiques. Souvent des solutions à des problèmes de sécurité existent déjà. Tout l'art consiste à mettre en musique tous ces éléments pour fournir la meilleure protection possible du système distribué dans le cadre de la politique de sécurité de l'organisation. La gestion de la sécurité comprend entre autres les sous-tâches suivantes: le monitoring du système distribué pour détecter des attaques contre la sécurité, le chiffrement des informations, l'exécution de procédures d'authentification et l'implémentation de mesures de sécurité.

- ***La gestion des performances.***

Nous arrivons maintenant au point central de notre propos: la gestion des performances. Comparée à son objectif, la gestion des performances peut être considérée comme étant la suite de la gestion des fautes (voir Figure 6).

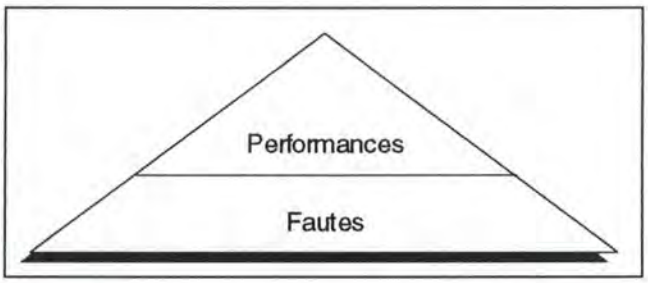

Figure 6 : La gestion des fautes et des performances.

En effet elle ne se borne pas uniquement à veiller à ce que le système distribué tourne et qu'il reste disponible, elle offre un service supplémentaire à valeur ajoutée. Ce service consiste à s'assurer que le système distribué tourne de façon optimale et efficace. Comment définir des performances optimales et efficaces d'un système distribué? Il y a beaucoup de facteurs qui interviennent, à savoir: les temps de réponse, les taux d'utilisation des composantes, le nombre de demandes de ressources rejetées, la capacité maximale des composantes, les temps de service, la longueur des files d'attente, etc.. La fonction de gestion des performances doit aussi tenir compte de la spécificité des utilisateurs humains. Bien que des mauvaises langues appellent l'homme le "périphérique" le plus lent

(pourtant elles ont raison), il faut néanmoins que les temps de réponse[17] à ses requêtes soient minimaux.

Dans la gestion des performances sont inclus les paramètres de définition de la qualité de service, le monitoring du système distribué pour déterminer des goulots d'étranglement, les mesures, le traitement des données ainsi captées, la génération de rapports, la gestion de l'évolution des capacités du système distribué. Dans le chapitre 3 nous reviendrons plus en détail sur les différents aspects de la gestion des performances.

2.4.2. La dimension temporelle.

Le cycle de vie de l'activité de gestion est représenté par la dimension temporelle. Il comprend les phases de planification, d'implémentation et d'exploitation.

La ***phase de planification*** peut se présenter sous deux cas de figure. Un premier, très rare, consiste à concevoir une nouvelle solution de gestion. Cette conception pourra se faire en parallèle avec la mise en place d'un nouveau système distribué. Dans le deuxième cas de figure, une solution doit être adaptée à un système distribué existant, avec déjà des solutions de gestion partielles existantes. Par conséquent ces solutions doivent être prises en compte et intégrées dans la nouvelle. Ceci entraîne que la phase de planification doit commencer par une analyse de l'existant. Pour cette analyse il faut répondre aux questions suivantes: Qui utilise quels outils pour effectuer quelles tâches? Quelles sont les informations qui circulent? Quels sont les points faibles de la solution de gestion existante? L'étape suivante consiste à dégager les besoins futurs de la gestion. Finalement après une évaluation le responsable stratégique[18] retient des outils de gestion et définit les tâches.

Lors de la ***phase d'implémentation*** une solution de gestion est mise en oeuvre. Elle comprendra l'implémentation des parties matérielles et logicielles des outils de gestion. Comme nous venons de l'exposer à la sous-section 2.3.3 les outils peuvent être distingués selon trois degrés d'intégration.

La solution de gestion est opérationnelle durant la ***phase d'exploitation***. Pendant les opérations quotidiennes nous constatons si la solution retenue présente encore des faiblesses. Une réitération des phases de planification et d'implémentation peut être envisagée dans ces cas-là.

---

[17] Ben Shneiderman analyse les temps de réponse et la productivité des utilisateurs dans [Shneiderman,1984].

[18] voir à la sous-section *2.3.5 Les intervenants humains*.

2.4.3. Le cadre ou contexte de la gestion.

Les différents contextes de la gestion sont ciblés sur des objets tels que les réseaux, les systèmes, les applications ou l'entreprise toute entière. Cette dimension permet une vue plus générale de la gestion. Nous constatons qu'à chaque niveau correspondent des stratégies et des outils différents.

- Un système distribué est impensable sans une structure de communication, un réseau sous-jacent. Par conséquent une ***gestion réseau*** s'impose. Nous pouvons la décomposer en une gestion des applications réseau et en une gestion des composantes physiques réseau. Des exemples d'applications réseau sont: le courrier électronique et le transfert de fichiers.
- La migration d'une informatique centralisée vers une informatique distribuée a entraîné aussi la création d'une nouvelle discipline de ***gestion système*** (d'un système distribué).

  Celle-ci s'occupe entre autres de la gestion:

  de la transparence des éléments interconnectés,
  de la sécurité et de l'intégrité des données,
  des entrées/sorties,
  du système de fichiers,
  de l'archivage,
  des utilisateurs,
  de l'installation et de la configuration du hardware et du software,
  des fautes,
  etc.

  Bref de tout ce qui ne relève pas du domaine de la gestion réseau.
- Finalement nous avons la ***gestion de l'entreprise***, qui elle peut être décomposée en la gestion stratégique, la gestion du personnel, la gestion des stocks, la gestion des clients, etc.. La gestion de l'entreprise concerne directement ou indirectement le développement du système distribué et par conséquent aussi sa gestion.

Maintenant que nous nous sommes familiarisés avec les dimensions fonctionnelles, temporelles et le contexte de la gestion nous arrivons aux informations de gestion. Celles-ci constituent le coeur même du modèle de la gestion que nous allons présenter.

2.5. Les informations de gestion.

Dans le modèle que nous exposerons, les composantes sont gérées par des agents. Il existe au moins une machine "gérante" qui abrite un manager et des applications de gestion. Le manager et les agents s'échangent des informations de gestion. Nous allons appréhender le fonctionnement général[19] de ce modèle manager-agent en partant des informations de gestion.

En premier lieu nous détaillerons les différents types d'informations de gestion et leur représentation. Ensuite nous examinerons la base d'informations de gestion (*MIB* : "*Management Information Base*"). Pour finir nous décrirons l'accès aux informations de gestion dans le modèle manager-agent et nous présenterons deux exemples de gestion..

2.5.1. Les types et leur représentation.

Des informations de gestion sont nécessaires dans l'activité de gestion. Nous distinguons trois types d'informations de gestion. D'abord nous avons les informations ***statiques*** qui fournissent une description d'une ***composante réelle***[20]. Dans le cas d'une imprimante il pourrait s'agir de sa localisation dans l'entreprise. Ensuite nous disposons d'informations ***dynamiques*** qui évoluent au cours du temps. Citons par exemple le nombre de trames envoyées sur un réseau Ethernet. Des agrégations de ces dernières constituent des informations ***statistiques***. Une application de gestion doit pouvoir recourir à ces données en lecture et en écriture. L'accès d'une application de gestion aux informations sera traité à la sous-section 2.5.3.

Comme nous l'avons expliqué au début de ce mémoire, nous travaillons sur un système distribué dont une des caractéristiques est l'hétérogénéité. Il faut donc que les informations de gestion ne soient pas spécifiques aux différentes composantes. L'approche, retenue généralement, consiste à fournir un modèle conceptuel de la composante gérée, qui offre une interface uniforme du côté de l'application de gestion[21] (voir Figure 7). Cette interface est indépendante de la composante gérée. De l'autre côté les informations doivent être extraites de la composante réelle et les modifications de l'information de gestion, initiées par l'application de

---

[19] inspiré de [Seitz,1994].

[20] Par "composante réelle" nous entendons les composantes matérielles (imprimantes, ordinateurs, etc.) et logiques (processus, tables de routage, circuits virtuels, etc.). Nous avons choisi le terme de "composante réelle" pour marquer une opposition avec une "composante modélisée".

[21] Une application de gestion remplit une ou plusieurs des cinq fonctions de gestion qui ont été présentées à la section *2.4.1 La dimension fonctionnelle*.

gestion, devront se répercuter aussi sur la composante. Une telle représentation modélisée d'une composante est appelée ***objet géré*** (“***Managed Object***”). L'objet géré doit reflèter le ***comportement*** de la composante réelle. L'information de gestion sur un objet géré est contenue dans un ou plusieurs ***attributs***. Chaque attribut peut être déclaré comme étant accessible ou non de l'extérieur. Dans le premier cas l'accès peut être en lecture ou en écriture. Par exemple dans la figure ci-dessous, des attributs comme “*l'état opérationnel*”, “*la résolution*”, “*le nombre de pages imprimées*” pourraient caractériser l'imprimante.

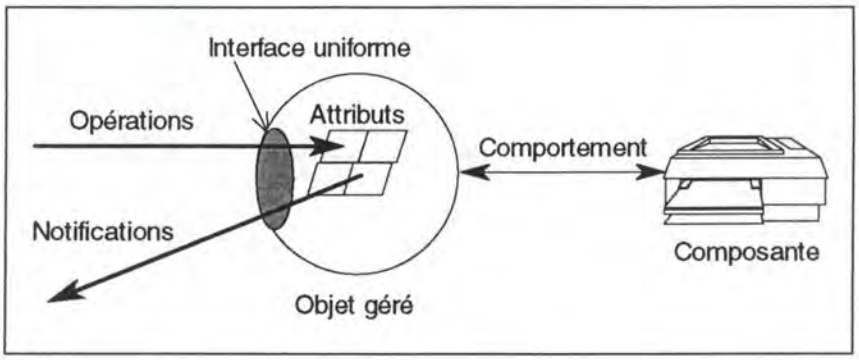

Figure 7 : Une représentation schématique d'un objet géré.

Des ***opérations***, offertes à l'interface de l'objet géré, permettent un accès à ses informations. Ces opérations permettent d'écrire ou de lire des attributs, de créer ou de supprimer des objets gérés entiers. Les opérations comprenent aussi des actions accessoires dépendant de l'objet géré.

Ainsi on pourrait par exemple éteindre une imprimante, tout en imprimant encore les travaux en suspens. Tous les nouveaux travaux soumis à l'imprimante seront refusés.

Jusqu'à présent nous avons toujours admis que l'initiative venait du côté des applications de gestion. Dans le cas d'événements exceptionnels, l'objet géré peut prendre l'initiative et envoyer des ***notifications*** pour en aviser l'application de gestion. Ainsi une imprimante peut envoyer une alarme quand elle ne dispose plus de papier. Ou bien, si le nombre de trames erronées sur un segment de réseau dépasse un seuil prédéfini, une alarme peut être envoyée.

Quelles sont les relations entre les applications de gestion, l'objet géré et la composante réelle? Comme nous venons de le voir, l'accès à l'objet géré est défini du côté de l'application de gestion. Les liens entre l'objet géré et la composante peuvent au contraire être implémentés librement. Il faut cependant que le couplage entre les deux se fasse dans les deux sens. Des modifications de l'état de la composante doivent se répercuter sur l'objet géré et vice versa.

2.5.2. La MIB.

L'ensemble des objets gérés répartis dans le système distribué est regroupé dans une "***Management Information Base***" (*MIB*). La MIB est une base d'informations de gestion organisée en forme d'arbre. Notons que la MIB est indépendante de toute implémentation physique. En effet les objets gérés peuvent être stockés en mémoire, sur fichiers ou dans une base de données. L'adressage des objets gérés dans une MIB nécessite des noms univoques, qui selon l'architecture, suivent un certain schéma d'adressage.

2.5.3. Les accès aux informations de gestion.

Une application de gestion, qui essaie d'extraire des informations sur un objet géré, ne peut pas le faire de façon directe. En effet dans la plupart des cas, la composante ciblée ne se trouve pas sur le même élément. Il faut donc qu' une communication soit établie à travers le réseau entre l'application de gestion et la composante distante. Pour ce faire, des ***protocoles de gestion*** permettent de lire ou de modifier des attributs d'objets gérés, d'exécuter des actions spécifiques à un objet géré, d'envoyer des notifications. Les entités qui communiquent par un protocole de gestion, peuvent être distingués suivant deux rôles à savoir le ***manager*** et ***l'agent*** (voir la Figure 8 ci-dessous).

Un manager reçoit les requêtes d'une ou de plusieurs applications de gestion. Il envoie les requêtes à l'agent en utilisant un protocole de gestion. En outre le manager fait passer les réponses et les notifications qu'il reçoit à l'application de gestion.

Un agent s'occupe de la gestion des objets gérés qui forment la partie locale d'une MIB. En outre il reçoit des requêtes du manager et envoie des notifications.

Le gestionnaire peut utiliser une application de gestion à travers l'interface homme-machine pour gérer une composante. L'application accède au manager pour communiquer avec n'importe quel agent du système. Ainsi la gestion est possible à partir d'un point central.

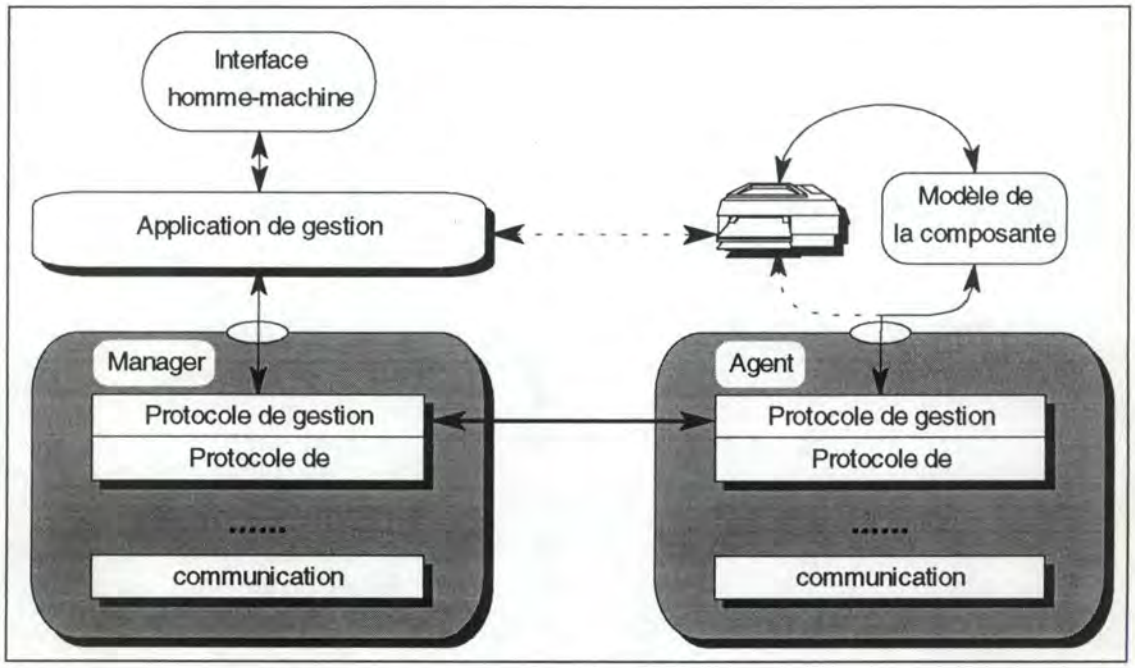

Figure 8 : Le schéma général du modèle Manager-Agent.

Des implémentations concrètes du modèle manager-agent ne possèdent pas forcément toutes les propriétés et fonctionnalités décrites ci-dessous. Nous pourrons le constater dans l'illustration qui va suivre à la section 2.6.

2.5.4. Deux exemples de gestion.

L'idée de cette sous-section est d'introduire la gestion dans les mondes IAB[22] et OSI[23]. La gestion dans le monde IAB sera présentée plus en détail dans la section suivante. Ainsi le lecteur pourra mieux comprendre la suite du mémoire (les chapitres 4 et 5).

a) La gestion dans le monde IAB.

Dans la section suivante nous allons discuter brièvement la première version du *Simple Network Management Protocol*[24]. La philosophie de ce protocole est, comme son nom l'indique , la simplicité. *L'axiome fondamental*[25] de SNMP reflète bien cette idée:

---

[22] IAB : Internet Activities Board

[23] OSI : Open Systems Interconnection

[24] SNMP : Simple Network Management Protocol

[25] énoncé dans [Rose,1991] page 71 et [Rose,1994] page 62.

"The impact of adding network management to managed nodes must be minimal, reflecting the lowest common denominator."

L'*Internet Activities Board* (IAB) qui fût fondé en 1983, est une organisation qui s'occupe de la standardisation de TCP/IP et des domaines apparentés. En tant que tel l'IAB a standardisé aussi SNMP. Les standards issus de cet organisme sont publiés dans les *Request For Comments* (RFCs).

Pour plus d'informations sur SNMP le livre [Rose,1991] est une bonne référence. Les ouvrages suivants donnent à côté de la description de SNMP, également un aperçu de la nouvelle version du protocole (SNMPv2): [Hegering,1994], [Rose,1994], [Seitz,1994] et [Stallings,1993].

b) La gestion dans le monde OSI.

"OSI Management: the facilities to control, coordinate and monitor the resources which allow communications to take place in the OSI environment"[26]

L'OSI Management Framework de l'ISO offre des définitions étendues et puissantes qui permettent la gestion réseau ainsi que la gestion de systèmes distribués. Les entités de gestion utilisées sont très complexes et difficiles à implémenter. Ceci est un des grands désavantages du Framework, qui explique pourquoi les fournisseurs hésitent d'équiper leurs composantes avec des fonctionnalités de gestion OSI. Cependant la gestion ISO/OSI est utilisée de plus en plus par les opérateurs des télécommunications. Nous n'allons pas détailler les services CMIS (*"Common Management Information Services"*), ni le protocole correspondant CMIP (*"Common Management Information Protocol"*). Les ouvrages suivants sont une bonne aide pour le lecteur intéressé par la gestion ISO/OSI: [Hegering,1994], [Seitz,1994] et [Stallings,1993].

Le lecteur intéressé par une étude comparative concise des protocoles SNMP et CMIP, trouvera dans l'article [Gering,1993] une bonne référence.

[26] source: *Addendum 4 (Management Framework)* de *l'ISO/OSI reference model document* (ISO 7498-4)

2.5.5. La gestion dans le monde IAB.

a) La représentation des informations de gestion.

Les objets gérés SNMP qui peuplent la MIB ne sont en fait que des variables, qui peuvent être lues et écrites. Le concept d'*attribut* que nous avons introduit à la section précédente n'existe pas en SNMP.

b) La MIB.

Quand une nouvelle MIB est définie, les objets doivent obtenir des identifiants, appelés *Object IDentifier* (OID). La structuration de ces variables se fait à l'aide d'un arbre global d'enregistrement (Figure 9):

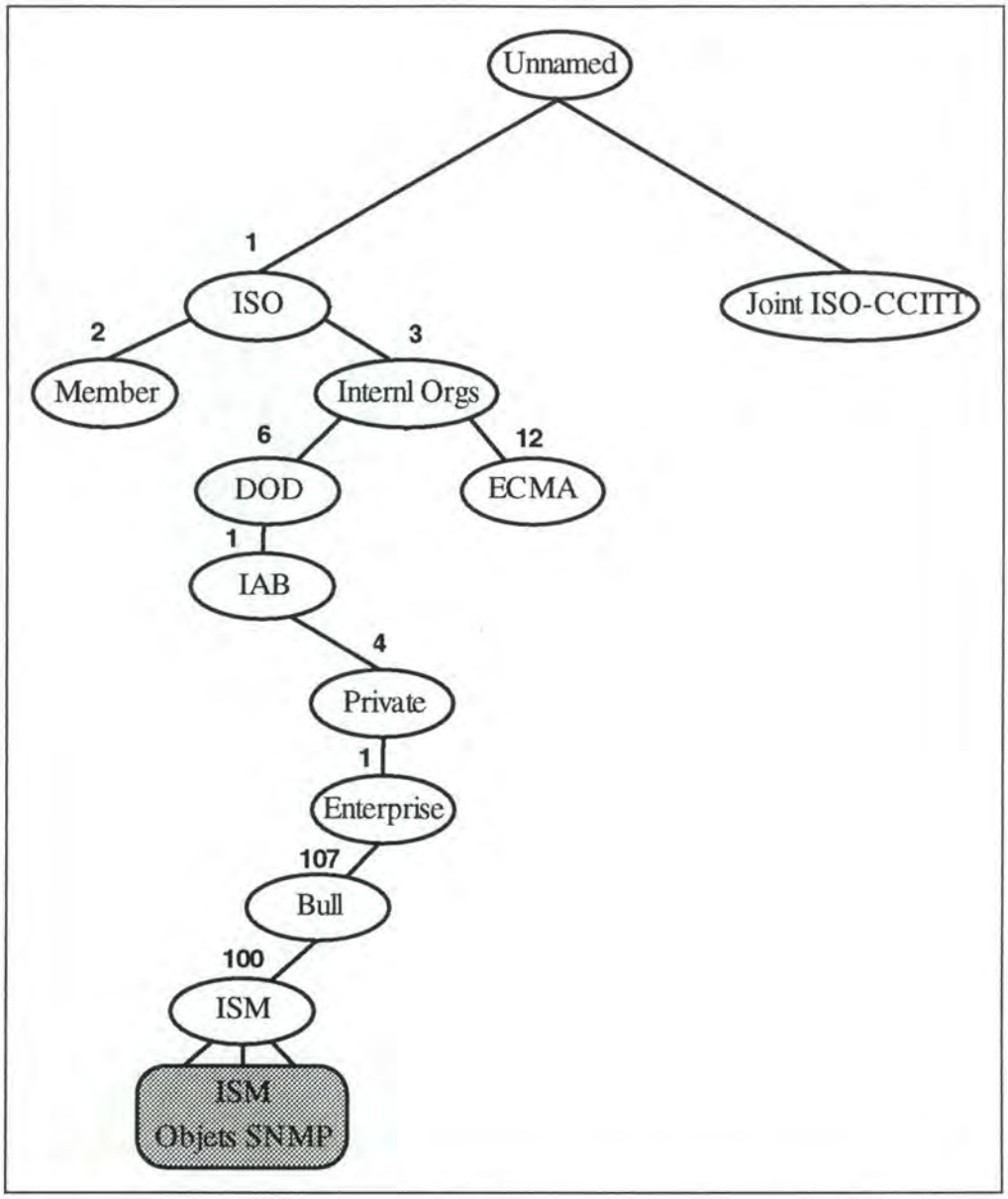

Figure 9 : L'arbre global d'enregistrement d'objets.

Les OIDs de SNMP font partie de l'arbre global d'enregistrement d'objets (*"Object Registration Tree"*), dont une partie est visible sur la figure 9. Chaque noeud est sous la tutelle d'une autorité d'enregistrement (*"Registration Authority"*) reconnue, qui a le droit d'allouer les noeuds sous son noeud à d'autres individus ou organisations, créant ainsi des nouvelles autorités d'enregistrement subordonnées. Un des noeuds d'ISM[27] dans l'arbre s'appelle:

- 1.3.6.1.4.1.107.100 (iso international dod iab private entreprises bull ism), il sert à l'enregistrement d'objets SNMP.

[27] La plate-forme de gestion ISM de Bull va être présentée dans le chapitre 4.

Chaque MIB obtient un noeud dans l'arbre. Le propriétaire de la MIB devient alors autorité d'enregistrement responsable de l'allocation des OIDs dans cette MIB.

c) Les accès aux informations de gestion.

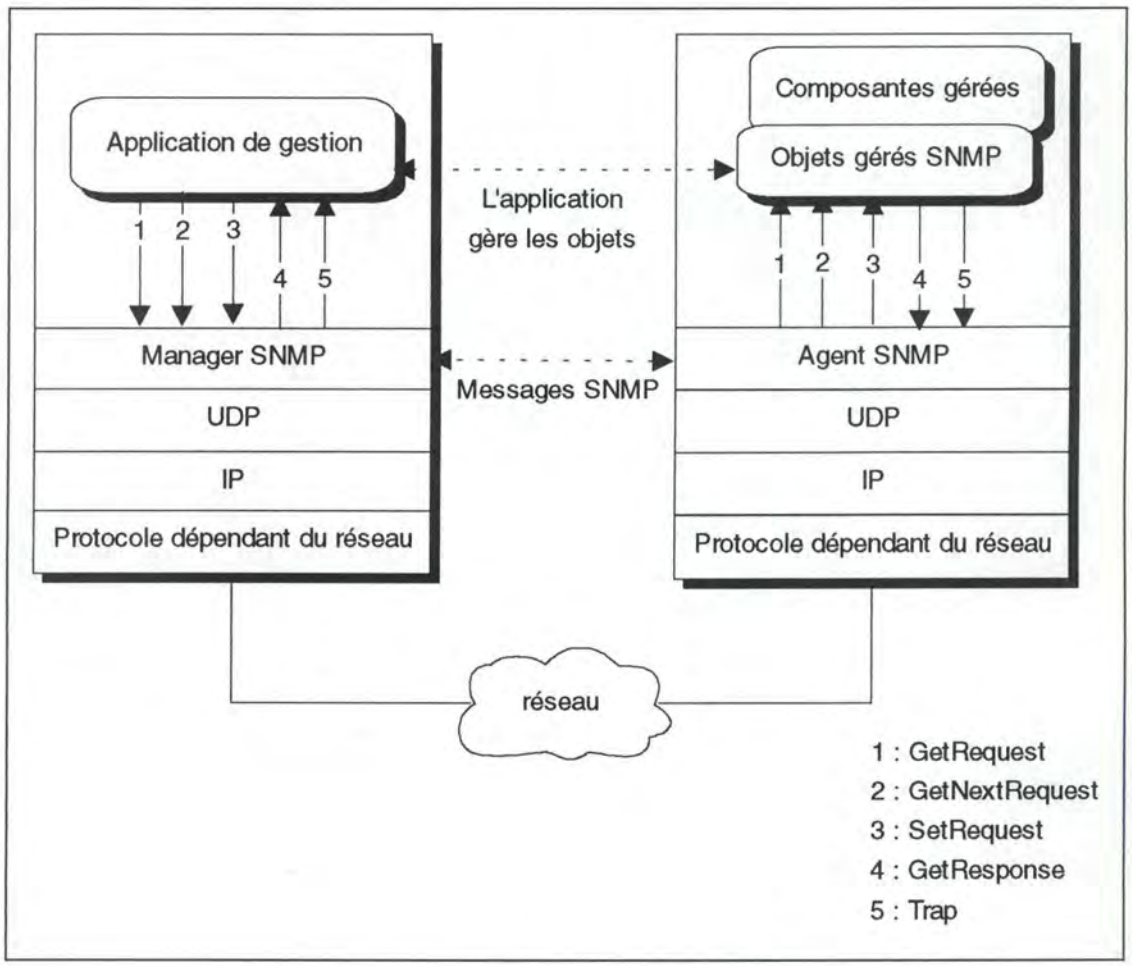

Figure 10 : Le rôle de SNMP[28].

Comme nous pouvons le constater sur la Figure 10, SNMP ressemble fort au modèle général manager-agent que nous avons présenté à la section précédente. En effet nous avons une (ou plusieurs) station(s) de gestion qui contiennent les applications de gestions et les managers SNMP. Le(s) managers communique(nt) avec les agents à l'aide du protocole SNMP. Ce dernier utilise comme protocole de communication UDP/IP[29]. Les ***opérations*** suivantes sont implémentées dans SNMP:

---

[28] source: [Ben-Artzi,1990]

[29] UDP ("User Datagram Protocol") et IP ("Internet Protocol") travaillent tous les deux dans le mode "connectionless". Le lecteur intéressé par ces protocoles pourra se référer par exemple à [Comer,1991].

1. Le manager veut lire la valeur d'une instance d'un objet (*GetRequest*).

2. Avec l'opération *GetNextRequest* le manager peut lire la valeur de l'instance suivante.

3. Le manager peut assigner une valeur à une instance avec l'opération *SetRequest.*

4. L'agent répond par une opération *GetResponse* à une requête de lecture du manager.

Nous constatons qu'il n'existe pas d'opération qui permette d'effectuer une action sur un objet. Le seul moyen d'implémenter une action est par le biais d'un *SetRequest*. Dans ce cas un objet SNMP sert comme interrupteur. A chaque changement de valeur de cet "interrupteur" par une opération *SetRequest,* l'agent vérifie sa valeur. Selon la valeur de l'interrupteur l'agent peut déclencher l'action correspondante, qui est codée en dur dans l'agent.

La survenance d'un événement exceptionnel est ***notifiée*** par une Trap.

Chapitre 3

La gestion des performances

3. La Gestion des Performances.

3.1. Introduction.

> *"We cannot hope to manage and control a system or an activity unless we can monitor its performance."*[30]

En effet, comment peut-on gérer ou contrôler un système sans surveiller de près le niveau de performances? Dans un système distribué, un manque de ressources par exemple, peut entraîner une chute de performances qui entrave le bon fonctionnement du système entier. Avec une gestion adéquate des performances des différentes composantes, de telles situations critiques pourraient être évitées. Nous allons étudier maintenant plus en détail la gestion des performances dans le cadre des systèmes distribués.

Pour commencer nous essaierons d'élucider la notion de ***performance*** et de présenter les indicateurs de performances les plus utilisés. La section suivante va donner une vue sur l'activité de ***gestion de performances*** en dégageant ses phases. Par après nous caractériserons les ***outils*** et les ***méthodes*** de mesures qui sont utilisés lors des différentes phases. Pour illustrer ces outils nous donnerons quelques exemples concrets. Nous allons essayer d'appliquer ces ***exemples d'outils*** aux différentes couches[31] d'un élément de système distribué. Finalement nous évoquerons quelques ***problèmes*** relatifs aux mesures de performances.

---

[30] source: [Stallings,1993] page 28

[31] Les différentes couches sont expliquées à la section "*2.2.4 Les différentes couches*".

3.2. La notion de performance.

3.2.1. Une définition difficile.

L'importance du concept de ***performance*** est souligné par l'extrait suivant:

"Performance is one of the three fundamental categories of attributes that are indispensable for the viability of any technical system, the other two being functionality (with its equally important aspects of correctnes and reliability) and economicity."[32]

La *performance* est donc un des trois attributs qui garantissent la "viabilité" d'un système technique tel qu'un système distribué. Mais comment peut-on caractériser la *performance* d'un système informatique? Il n'est pas évident de trouver "**la**" définition de la *performance* d'un système informatique. Dans [Encyclopedia,1978] nous trouvons une affirmation qui nous paraît intéressante:

"There are as yet no techniques for characterizing the overall performance of a computing system in the abstract. The trouble is that we do not have any adequate and accepted method of quantifying the 'information work' that a given job entails."[33]

Ce qui était vrai en 1978, reste ou redevient actuel de nos jours avec l'avènement de l'informatique distribuée, comme le prouve l'affirmation suivante:

"... reasonable models for the study of centralized systems and their workloads were barely becoming available when the world was already moving toward distributed systems, for which even a definition of performance has not been agreed upon yet, ..."[34]

---

[32] source: [Ferrari,1986]

[33] source: [*Encyclopedia,1978*] page 1063

[34] source: [Ferrari,1986]

En effet comment étudier la performance d'un système distribué, si la communauté scientifique n'est même pas d'accord sur une définition commune? Comment comparer par exemple les performances de deux systèmes distribués? Certes nous pouvons utiliser des indicateurs de performances comme le temps de réponse, la disponibilité, etc. Mais ceux-ci nous fournissent qu'une vue partielle du niveau de performance global. Pour mieux expliquer la quantification du niveau de performance, nous allons donner quelques exemples d'indicateurs dans la sous-section suivante.

3.2.2. Les deux types d'évaluation des performances.

La performance d'un système informatique peut être évaluée ou quantifiée par des indicateurs qui se présentent sous deux optiques[35] différentes. Nous exposerons maintenant quelques exemples d'indicateurs.

a) une évaluation orientée vers le niveau de service offert aux utilisateurs:

1. La ***disponibilité*** indique le pourcentage de temps pendant lequel une composante est disponible pour un utilisateur. Elle peut être exprimée en fonction du temps moyen entre deux pannes ("*Mean Time Between Failures*" : *MTBF*) et du temps moyen nécessaire pour effectuer des réparations après une panne ("*Mean Time To Repair*" : *MTTR*) (Figure 11).

$$D = \frac{MTBF}{MTBF + MTTR}$$

Figure 11: Calcul de la disponibilité D.

La disponibilité totale d'un système distribué dépend de la disponibilité des diverses composantes ***et*** de leur disposition. Il ne suffit pas que certaines composantes soient redondantes pour augmenter leur disponibilité, il faut aussi qu'elles soient configurées de façon à tirer profit de la redondance (voir Figure 12).

[35] source: [Stallings,1993] pages 28 et suivantes

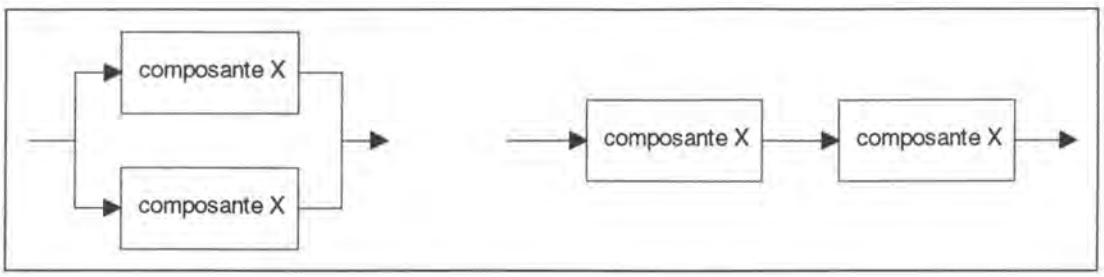

Figure 12 : Une disposition en parallèle versus une disposition en série.

Dans la disposition parallèle, si une des deux composantes tombe en panne, l'autre peut prendre automatiquement la relève. Par contre l'agencement en série n'apporte pas de disponibilité supérieure. Au contraire, pour que l'ensemble travaille bien, il faut que les deux composantes redondantes travaillent bien.

2. Le ***temps de réponse*** constitue l'intervalle de temps entre une requête initié par l'utilisateur et la réponse apparaissant sur son écran. Des études ont été menées sur l'interaction de l'homme avec les systèmes informatiques et les critères ergonomiques à respecter. Dans l'introduction d'une étude menée à cet égard, Ben Shneiderman affirme:

"Time is precious. When unexpected delays impede the progress of a task, many people become frustrated, annoyed and eventually angry."[36]

Le temps de réponse a donc un impact direct sur la motivation des utilisateurs et sur leur productivité.

3. L'***exactitude*** ("*Accuracy*") est exprimée par le pourcentage de temps pendant lequel aucune erreur apparaît. Normalement l'utilisateur ne devrait pas se rendre compte des erreurs qui surviennent aux différents niveaux du système distribué. Ces erreurs sont corrigées en général par des algorithmes de détection et de correction d'erreurs. Néanmoins il peut être intéressant d'éliminer des sources d'erreurs fréquentes, car celles-ci peuvent provenir de défauts matériels. Il faut que de tels défauts soient écartés pour qu'ils n'entravent pas les performances globales du système.

b) une évaluation orientée vers l'efficacité proprement dite du système:

1. Le ***'throughput'*** est le nombre de transactions d'un certain type, effectuées au niveau des applications, pendant une durée déterminée. Il est intéressant de connaître le *'throughput'* maximal possible ainsi que sa valeur actuelle pour éviter des situations critiques.

[36] source: [Shneiderman,1984]

2. Le ***taux d'utilisation*** est le pourcentage utilisé de la capacité théorique de la composante. C'est un indicateur intéressant pour prévoir d'éventuels goulots d'étranglement. Sachant que le temps de réponse croît exponentiellement avec l'augmentation du taux d'utilisation d'une composante, il est important de surveiller de près l'évolution de ce taux pour pouvoir intervenir rapidement. En observant le profil d'utilisation des différentes composantes, le gestionnaire peut déterminer celles qui sont sur-utilisées et celles sous-utilisées. Ces informations lui permettront de réorganiser le système et d'obtenir ainsi de meilleures performances.

Notons qu'à côté des quelques indicateurs que nous venons de présenter, il existe encore une myriade d'autres. Leur signification est loin d'être bien définie. En effet elle change en fonction du système analysé, des fournisseurs d'outils, etc.

Nous venons de voir un petit échantillon d'indicateurs qui nous paraissait intéressants pour illustrer la quantification des performances. Passons à l'activité de gestion des performances qui a besoin d'indicateurs significatifs.

3.3. Les phases de la gestion des performances.

3.3.1. L'enchaînement des phases.

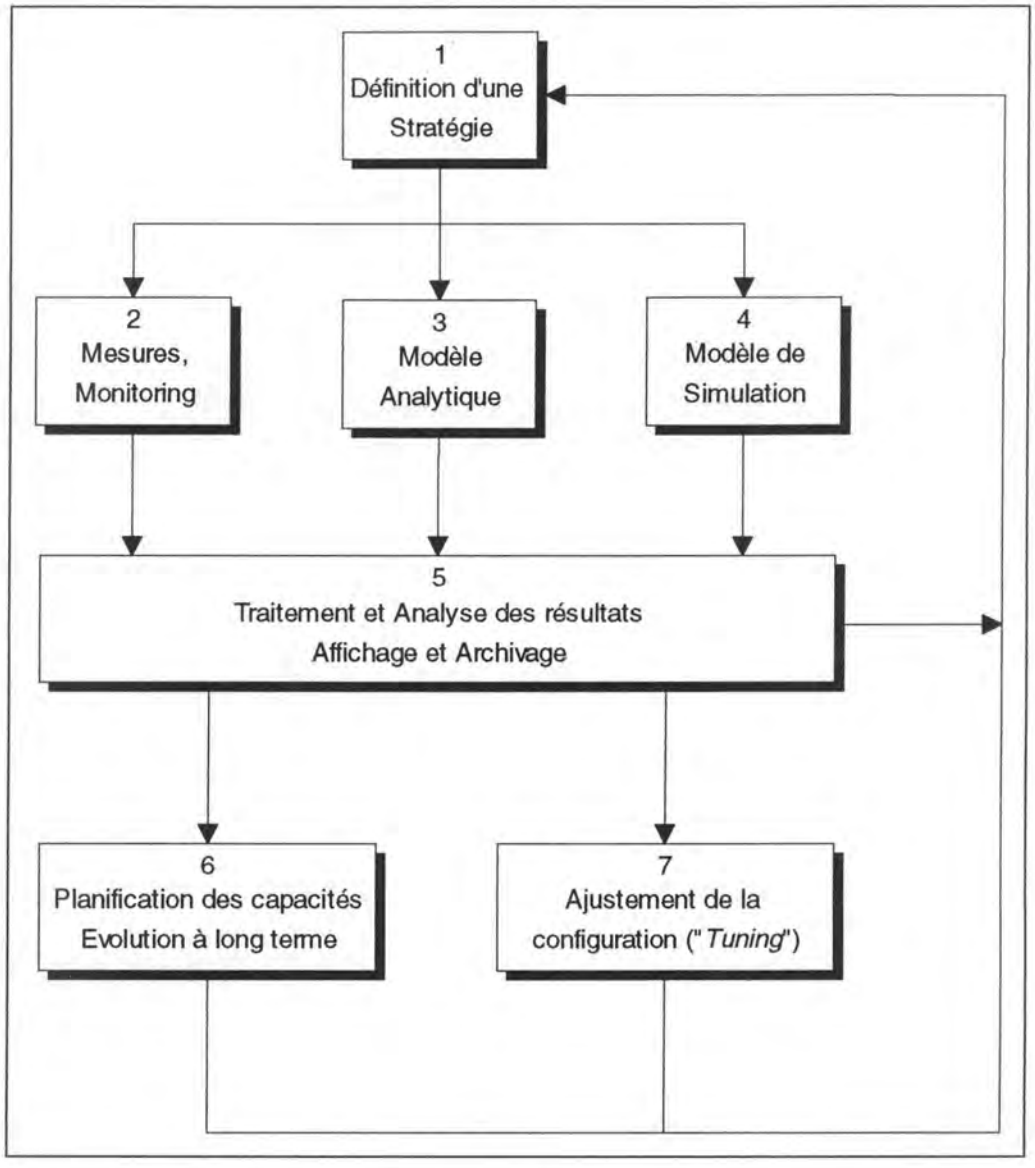

Figure 13 : Les phases de la gestion des performances.

Analysons dès à présent l'***activité*** même de gestion des performances. Le gestionnaire essaie, sur base de symptômes, de déterminer les causes d'une dégradation éventuelle des performances et puis de rétablir un niveau de performances acceptable.

A la suite de cette constatation nous pouvons observer la cristallisation de deux grands pôles de tâches. Dans un premier temps, la tâche du gestionnaire consiste à évaluer le niveau de performance du système. Dans un second temps, il peut ajuster les paramètres et changer le système

distribué pour améliorer les performances. Pour notre analyse nous découperons l'activité de gestion des performances en phases, que nous allons enchaîner ensuite (voir Figure 13). Nous décrirons dans les paragraphes qui suivent les différentes phases.

a) La définition d'une stratégie.

L'évaluation du niveau des performances d'un système peut se faire à l'aide d'outils de mesure ou à l'aide d'un modèle du système. Mais avant de commencer l'évaluation, il faut déterminer la ***stratégie*** selon laquelle le système distribué va être analysé (phase 1). Cette stratégie consiste à fixer la période pendant laquelle le système va être analysé, comment il va être analysé, l'objectif de l'évaluation des performances et les paramètres qui vont être évalués.

b) Les modèles analytiques et les simulations.

Pour observer l'évolution des performances, le gestionnaire dispose de ***modèles analytiques*** (phase 3). La solution à de tels modèles est trouvée à l'aide de règles transformationnelles qui sont basées sur les mathématiques formelles. Cette méthode a l'avantage que des relations fonctionnelles entre les variables peuvent être dégagées, grâce aux formules mathématiques. Des exemples de tels modèles sont les processus stochastiques et la théorie des files d'attente.

Si le système devient trop complexe, les ***simulations*** constituent une aide intéressante (phase 4). Les simulations permettent de traiter d'autres problèmes que les méthodes analytiques, puisqu'elles permettent des descriptions procédurales des interactions entre variables. Pour faciliter des simulations sur ordinateurs, il existe des outils et des langages spécialisés.

Cependant il faut remarquer que ces deux méthodes dépendent très fortement de la qualité du modèle et de l'interprétation correcte des résultats.

c) Les mesures.

Une autre façon de procéder consiste à effectuer des ***mesures*** directement sur la composante ciblée (phase 2). Les mesures se font en monitorant l'activité de celle-ci. Cette activité peut être naturelle ou être générée de manière artificielle. La mesure de ressources sous une charge artificielle[37] peut servir entre autres à la comparaison de deux entités identiques ou à des tests avant la mise en service. Nous allons revenir plus en détail dans la section suivante sur la typologie des outils de mesure.

[37] Ces types de mesures sont encore appelés "*benchmarks*".

d) Le traitement et l'analyse des résultats.

L'étape suivante sert à ***traiter*** et à ***analyser*** les résultats obtenus (phase 5). Ce traitement consiste à tirer des inférences statistiques, à éditer des rapports, à filtrer, à archiver, à agréger, à interpréter, à présenter graphiquement les données, etc. L'archivage des informations significatives est important pour suivre et pour prévoir l'évolution du système. Pour illustrer l'importance des données "historiques", citons *George Orwell*:

"Who controls the past controls the future."

e) La planification des capacités.

Le but de la ***planification des capacités*** est de disposer toujours d'assez de capacités pour subvenir aux besoins des utilisateurs d'un côté et de n'avoir pas de surcapacités trop coûteuses de l'autre (phase 6). Des données historiques peuvent fournir un aperçu de l'évolution des capacités et déceler d'éventuelles fluctuations périodiques.

f) L'ajustement de la configuration.

Par opposition à la planification des capacités, ***l'ajustement de la configuration*** (***"Tuning"***) opère à court terme (phase 7). L'ajustement de la configuration sert à améliorer les performances du système distribué. Cette tâche englobe des activités très variées comme par exemple: étendre les capacités matérielles des machines, relocaliser des composantes (ordinateurs, périphériques, etc.), relocaliser des processus et/ou des données, changer l'infrastructure de communication, etc.

3.3.2. Conclusion.

Nous constatons que l'activité de gestion des performances peut être considérée comme un cycle de mesures et/ou de contrôles successifs qui convergent après un certain temps vers un équilibre stable et performant.

Comme nous l'avons annoncé plus haut, passons aux outils de mesures utilisés à la phase 2.

3.4. Les outils et les méthodes de mesures.

3.4.1. Typologie.

Traditionnellement, les outils de mesures étaient essentiellement des dispositifs ***hardware*** indépendants. Ils servaient surtout à capter des informations de bas niveau comme l'observation du trafic sur les bus système. Avec la complexité croissante des systèmes informatiques des outils ***software pur*** et ***hybrides***, c'est-à-dire avec des parties hardware et software mélangées, apparurent. Des dispositifs hardware, contrairement aux outils hybrides et logiciels, ont l'avantage de générer pas ou que très peu d'overhead sur le système mesuré. Des exemples concernant des outils de monitoring hybride et hardware se trouvent dans les articles décrivant les systèmes TMP [Haban,1990] et ZM4 [Hofmann,1994]. Les méthodes de mesures qui vont être décrites dans les sous-sections suivantes sont implémentées par des outils correspondants aux trois types que nous venons de voir.

3.4.2. Le monitoring.

Une première méthode pour évaluer les performances, qu'on appelle encore ***monitoring***[38], consiste à observer simplement ce qui se passe dans le système mesuré. Les valeurs obtenues, par des interrogations périodiques, sont ou bien enregistrées ou bien elles peuvent déclencher des actions. Un paramètre important de cette méthode d'interrogation, qui est appelée encore ***polling***, est la fréquence de prise des échantillons. Elle doit être choisie à ce qu'elle permette de tracer d'un côté le changement entre deux états significatifs du système. Mais d'un autre côté elle ne doit pas trop surcharger le système non plus.

Prenons le cas[39] d'un manager SNMP qui fait du polling de composantes dans un intervalle T. Le nombre N de composantes qui peuvent être surveillées est donné par:

$$N \leq T / \Delta$$

Δ : représente le temps minimal nécessaire pour accomplir un poll et comprend les intervalles suivants:

---

[38] Le terme de "*monitoring*" peut être traduit par "*surveillance*" ou "*observation*"

[39] Ce cas est exposé dans [Ben-Artzi,1990].

a = initiation du poll par le manager (requête sortante)
b = délai réseau du manager à l'agent
c = réception et interprétation par l'agent
d = réponse de l'agent (réponse sortante)
e = délai réseau de l'agent au manager
f = réception et interprétation par le manager

$$\Delta = a + b + c + d + e + f$$

3.4.3. Le monitoring basé sur les événements.

Une variante de la méthode précédente, moins consommatrice en ressources, ne considère que les changements d'état significatifs que le gestionnaire veut tracer. Lors de chacun de ces changements, un événement[40] est généré de manière asynchrone. La station de gestion décide à la réception d'un tel événement si elle doit initier un polling sur l'objet en question pour surveiller son comportement de près ou si elle doit exécuter une action spécifique. Un exemple de monitoring basé sur les événements est le suivi des opérations de création et de destruction de processus décrit par [Haban,1990].

3.4.4. Le benchmarking.

Une autre stratégie consiste à mesurer le système sous une charge artificielle. De cette manière il est possible de comparer deux systèmes sous une même charge. La difficulté de cette technique, connue encore sous le nom de benchmarking, est de déterminer une charge significative pour le système qu'on veut analyser. Cette méthode a l'inconvénient d'être consommateur en temps et en ressources.

> *"In the performance community, benchmarking is always a dreaded but necessary activity."*[41]

Le benchmarking est donc un procédé nécessaire, mais redouté par ceux qui sont concernés par les mesures de performances. C'est un procédé redouté car d'un côté il permet de générer facilement des nombres, mais d'un autre côté ce n'est pas évident de les interpréter. D'autre part il est nécessaire parce que de cette manière toutes les dépendances réelles entre

[40] "*événement*" est une notion similaire de "*interruption*"

[41] source: [Molloy,1992] page 29.

paramètres peuvent être étudiées. Mais pour faire des bonnes comparaisons il faut que le processus de benchmarking soit planifié convenablement. Il est important de ce rappeler qu'une benchmark livre des mesures de performances valables pour **cette** même benchmark. Une application peut se comporter de façon similaire à la benchmark ou non.

3.5. Les performances et les systèmes distribués.

Dans cette section nous allons présenter pour chaque couche d'un système distribué (voir la Figure 2 à la sous-section: *2.2.4 Les différentes couches.*) un ou plusieurs exemples d'outils qui servent à l'évaluation des performances.

3.5.1. Au niveau Hardware.

Au niveau matériel, les paramètres physiques des composantes vont être analysés. Nous pouvons distinguer entre les outils qui travaillent dans le mode analogique et le mode digital. Les outils en mode analogique ne peuvent analyser que des conducteurs opérant en mode analogique tandis que des outils en mode digital peuvent être utilisés dans les deux cas. Dans la suite nous allons décrire quelques représentants de ces deux familles[42].

a) En mode analogique.

- Le générateur de fréquences. L'outil génère une fréquence de test et mesure ensuite l'énergie perdue par le signal. Ce test est appliqué surtout pour les paires torsadées.
- Le multimètre. Il permet de mesurer le voltage, l'intensité du courant et la résistance d'un conducteur électrique. Si les valeurs mesurées diffèrent trop de la moyenne alors le conducteur a probablement un défaut. Des câbles endommagés, des courts-circuits, etc. peuvent affecter les performances.
- L'oscilloscope est un autre outil pour analyser des signaux. A l'aide de la forme caractéristique de son motif visuel le type de l'interférence perturbatrice peut être déduite (par exemple, une induction électro-mécanique).

[42] Les exemples sont tirés de [Hegering,1994].

b) En mode digital.

- La breakout box. Elle permet d'observer l'état de la connexion au niveau physique entre deux composantes réseau. La breakout box passive se limite à l'observation des signaux. La breakout box active permet en plus de couper et de recomposer les fils de manière arbitraire.
- Le BERT (Bit Error Rate Tester). Cet outil envoie une séquence de bits vers une composante de communication qui travaille en mode écho. Si la séquence revient elle est comparée avec celle qui a été envoyée et ainsi le nombre de bits erronés peut être déterminé.

3.5.2. Au niveau Communications.

Les mesures de performances des communications se font essentiellement à l'aide d'analyseurs de protocole. Ceux qui peuvent être utilisés pour enregistrer des données de protocole et pour les évaluer en utilisant des logiciels d'analyse appropriés (et l'expérience d'un gestionnaire réseau). Un analyseur trace par exemple le nombre de trames émises, le nombre d'erreurs de transmission, etc.. Dans la suite nous allons exposer des outils qui se basent sur la *RMON*[43] MIB décrite dans l'article [Waldbusser,1992]. Les ouvrages suivants expliquent aussi le fonctionnement et l'utilisation de *RMON*: [Hegering,1994], [Leinwand,1992], [Rose,1991], [Rose,1994] et [Stallings,1993]. Nous ne détaillerons pas ici les avantages de *RMON* par rapport aux analyseurs de protocole traditionnels. Ceux-ci sont exposés magistralement dans l'article [Waldbusser,1992b].

a) Le cas de RMON.

La RMON MIB étend SNMP au-delà de la simple gestion de composantes en introduisant le trafic de données sur les réseaux. Ce nouveau standard amène la gestion SNMP littéralement sur le câble[44]. RMON spécifie comment capter et stocker des informations concernant le trafic de données réseaux par une sonde[45] distante. Ces informations sont alors passées à une ou plusieurs stations de gestion SNMP pour effectuer des analyses et des actions. Les fonctions de sonde peuvent être effectuées par un appareil spécifique ou bien par un élément (ordinateur, routeur, etc.) qui remplit aussi d'autres tâches. La possibilité qu'ont les stations de gestion SNMP de

[43] RMON : Remote MONitoring

[44] "Now, a newly created standard (...) brings SNMP management literally down to the wire." in [Waldbusser,1992]

[45] appelée parfois aussi *moniteur réseau* (*"Network Monitor"*).

coopérer avec des sondes de fournisseurs différents est un des grands avantages de RMON.

Avec RMON une gestion préventive peut être mise en place. Des problèmes tels que des erreurs excessives de paquets peuvent être détectées et réparées avant que le réseau ne tombe en panne. RMON permet de faire la gestion de réseaux utilisant les protocoles Ethernet, Token Ring, FDDI.

La MIB RMON ne contient pas moins que neuf groupes de données différents. Ces groupes comprennent: les statistiques sur le trafic, les données historiques, les alarmes, les statistiques sur les hôtes, les listes des "top N" hôtes, les données de la matrice, les filtres de paquets, les paquets captés et les événements.

Le groupe de statistiques sur le trafic comprend les données mesurées par les agents distants sur le segment réseau auquel ils sont attachés. Ces statistiques donnent un aperçu de l'état de santé d'un réseau supportant un protocole particulier. Pour un réseau local *Ethernet* elles contiennent par exemple le nombre de collisions, le nombre de mauvais codes de détection d'erreurs, etc.

Le groupe de données historiques comprend des informations extraites périodiquement du groupe de statistiques. Ces informations sont stockées pour une analyse ultérieure.

Des seuils peuvent être définis dans le groupe des alarmes pour surveiller le niveau de performance du réseau. Si un seuil est dépassé, une alarme est générée. Des alarmes peuvent être rattachées à toutes les données statistiques de l'agent RMON.

Les agents RMON "découvrent" des nouveaux hôtes en gardant une liste avec les adresses physiques source et destination extraites de bons paquets reçus du réseau. Les statistiques sur les hôtes peuvent être utilisées aussi pour l'inventaire.

Le groupe des "top N" hôtes fait la liste ou le hit-parade des hôtes en fonction de la valeur numérique des statistiques associées. Par exemple, une table dans ce groupe pourra contenir les dix hôtes qui ont envoyés le plus de données pour un jour particulier.

Dans le groupe matrice sont répertoriés les données sur des communications entre un ensemble d'adresses au niveau physique. Ce type d'information permet de voir par exemple, quels hôtes utilisent le plus un serveur ou quels hôtes communiquent avec quelle composante.

Le groupe contenant les filtres de paquets, permet de déterminer quels paquets peuvent être interceptées et générer des événements. Le groupe des paquets captés spécifie comment les données sont acheminées vers les stations de gestion. Finalement le groupe des événements est une table de tous les événements générés par l'agent RMON. Il contrôle la génération des événements et il permet de garder un log de tous les événements.

Les informations des différents groupes sont accessibles par toute station de gestion SNMP et peuvent être traitées ultérieurement par des tableurs, des systèmes experts, etc..

3.5.3. Au niveau OS.

Les paramètres mesurés au niveau du système d'exploitation sont les performances du système de fichiers, l'utilisation des tampons,...

a) Le cas de nhfsstone.

L'outil *nhfsstone* contient des fonctionnalités pour évaluer le niveau de performances du *NFS* (*"Network File System"*) de SUN, un système de gestion de fichiers distribué. *NFS* rend l'accès aux fichiers transparents, c'est-à-dire que les applications ne voient pas de différence entre un accès local et un accès distant. Le fonctionnement de *nhfsstone* ainsi qu'une comparaison entre ses différentes versions existantes sont exposés dans l'article [Molloy,1992]. *Nhfsstone* génère une charge artificielle qui est composée d'un mélange spécial d'opérations *NFS*, et mesure le temps de réponse pour le serveur en question. Pour faciliter l'interprétation, les résultats sont affichés graphiquement. La benchmark *nhfsstone* devrait permettre de comparer les implémentations de *NFS* de différents fournisseurs. Les comparaisons de performances par *NFS* peuvent se faire selon trois approches:

- Nous pouvons nous concentrer sur le serveur en imitant le comportement des clients.
- Nous pouvons nous concentrer sur les clients en fournissant un serveur très rapide.
- Nous pouvons nous concentrer sur le couple client/serveur pour avoir l'optique d'une application

La charge artificielle produite par nhfsstone se base sur un pourcentage de codes opératoires, une collection de fichiers et un ensemble de tailles de fichiers. La charge est quantifiée en opérations d'entrée/sortie par seconde, appelé encore *IOPS*[46], et elle est associée à un certain délai. Une opération d'entrée/sortie est caractérisée par un mélange de différents types d'opérations et des tailles de ces opérations. C'est la raison pour laquelle des valeurs *IOPS* de deux charges différentes ne sont pas comparables. Il est difficile d'interpréter la valeur du délai puisqu'il est fonction du temps que le client met pour produire la requête, du temps de transmission de la requête, du temps de serveur nécessaire pour satisfaire la requête, du temps

[46] IOPS : Input/Output OPerations per Second.

de transmission de la réponse et finalement du temps que le client met pour s'apercevoir que la réponse est arrivée. Le délai global dépend donc de la taille de l'opération et du délai chez les clients.

D'autres recherches ont été menées pour évaluer le niveau de performances de *NFS*. Citons par exemple les travaux[47] de Bodnarchuk, qui a développé un modèle fournissant une charge synthétique d'opérations *NFS*.

3.5.4. Au niveau Middleware.

Le middleware comprend des fonctionnalités aussi diverses que la gestion de la distribution et des systèmes de gestion de bases de données. Des exemples d'évaluation des performances sont l'équilibration de la charge et les temps de réponse de requêtes SQL. L'équilibration de la charge est réalisée par la relocalisation de processus. Des outils qui gèrent la relocalisation des processus sont confrontés à deux types de problèmes[48]. Le premier est de savoir ***s'il*** faut migrer le processus et ***où*** il faut le migrer. Le deuxième problème consiste à connaître le moment ***quand*** il faut le migrer.

3.5.5. Au niveau Application.

Comment évaluer les performances d'une application? Comment assurer le monitoring d'une application distribuée? Un exemple d'outil est donné dans le paragraphe qui suit.

a) Le cas des applications distribuées.

Les processus d'applications distribuées peuvent aussi, comme toutes les autres composantes d'un système distribué, être modélisés par des objets gérés dans une *MIB*. La Figure 14 montre une implémentation où chaque agent gère un processus de l'application distribuée. Les objets gérés décrivent des informations concernant l'application distribuée. Comme protocoles de communication entre manager et agent, SNMP ou CMIP peuvent être utilisés.

---

[47] source: [Bodnarchuk,1991]

[48] source: [Hac,1992]

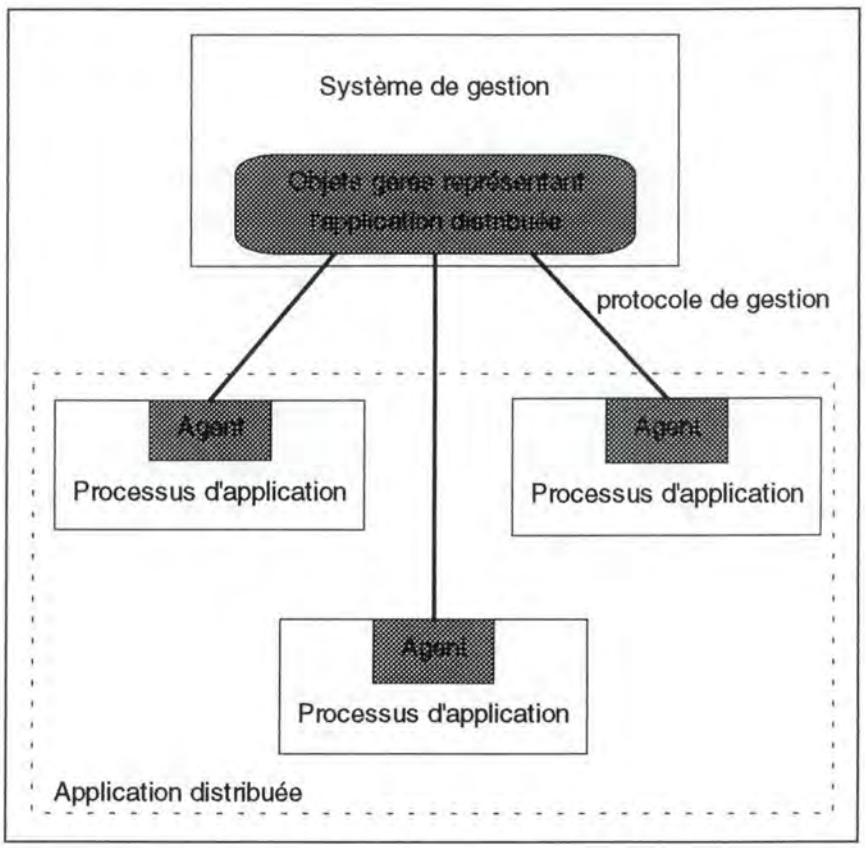

Figure 14 : La gestion d'une application distribuée[49].

3.6. Les problèmes des mesures de performances.

Avant de conclure nous allons évoquer quelques problèmes qui surgissent lors des mesures de performances.

Les mesures de performances souffrent d'un phénomène analogue au "principe d'incertitude" d'Heisenberg connu dans la physique quantique: "Pour prédire la situation future et la vitesse d'une particule, on doit mesurer sa situation actuelle et sa vitesse avec exactitude. Pour ce faire il faut l'éclairer. Quelques ondes de cette lumière incidente seraient éparpillées par la particule en question, indiquant ainsi sa situation. (...) on ne peut cependant pas utiliser une quantité arbitrairement petite de lumière et l'on doit faire appel au moins à un quantum. Celui-ci dérangera la particule et modifiera sa vitesse de façon imprévisible. Mais, plus on voudra mesurer la position précisément, (...) plus l'énergie du quantum

[49] source: [Hegering,1994] page 439

requis sera élevée. Aussi la vitesse de la particule sera fortement perturbée. En d'autres termes, plus vous essaierez de mesurer la position de la particule avec précision, moins vous disposerez d'une valeur précise pour sa vitesse et vice versa."[50]

Quel rapport y-a-t-il entre la physique quantique et les mesures de performances? Si on mesure les performances d'une composante (un ordinateur, un segment réseau, etc.) d'un système distribué, le résultat obtenu sera faussé par les interférences causées par le système mesureur. Cet effet sera d'autant plus fort qu'on veut obtenir des mesures plus précises. La surcharge générée est plus grand avec un mesureur software, qu'avec un mesureur hardware (voir [Haban,1990]). Un mesureur hardware ad-hoc indépendant n'a qu'un effet négligeable sur le système mesuré.

Au niveau du traitement des résultats des mesures de performances, il est difficile de les interpréter correctement. En effet les données captées localement à différents endroits ne sont pas facilement comparables, puisque les horloges des différents éléments ne sont pas forcément synchronisées. Un autre problème relatif à l'interprétation est la quantité de données collectées. Parmi celles-ci il faut garder uniquement les informations nécessaires et significatives. Comment choisir les données significatives?

Stallings énumère dans son livre[51] encore quelques problèmes concernant le choix et l'utilisation des indicateurs de performances:

- les indicateurs sont trop nombreux
- la signification de la plupart des indicateurs n'est pas bien comprise
- quelques indicateurs ne sont utilisées que par certains constructeurs
- souvent les indicateurs sont mesurés correctement mais mal interprétés
- dans de nombreux cas, le calcul des indicateurs consomme beaucoup trop de temps et le résultat final ne peut presque plus servir à des fins de contrôle

[50] source: [Hawking,1989] pages 80 et 81

[51] [Stallings,1993]

3.7. Conclusion.

"..the study of performance evaluation as an independent subject has sometimes caused researchers in the area to lose contact with reality."[52]

Pour ne pas perdre le contact avec la réalité nous traiterons le cas concret d'un agent de mesures de performances développé durant mon stage chez Bull. Mais d'abord nous allons présenter la plate-forme de gestion *ISM* de Bull, puis l'agent de mesures de performances. Nous finirons par un service de gestion de performances (*"Performance Monitoring Service"* : *PMS*) qui est fourni par *ISM*.

[52] source: [Ferrari,1986]

Chapitre 4

Etude de cas:

ISM de Bull

4. Etude de Cas: ISM de Bull.

4.1. Introduction.

Ce chapitre décrit le fonctionnement d'un produit de gestion concret. Nous allons commencer par une présentation générale d'*ISM*. Dans les sections suivantes, les composantes de base vont être exposées. D'abord nous détaillerons le manager *ISM* et les diverses parties de son architecture (les *Services*, les *Agents Integrators* et le *CMIS Dispatcher*). Ensuite nous décrirons les agents. Dans la conclusion nous aborderons les bénéfices d'une solution de gestion intégrée comme *ISM*.

Pour éviter des confusions, nous n'avons pas traduit en français les termes techniques spécifiques à *ISM* (comme par exemple des noms d'applications, de services ou d'autres composantes). (Que Jacques Toubon me pardonne.) Les objets qui sont ombrés sur les figures, représentent des occurrences multiples de ces objets.

4.2. La présentation d'ISM de Bull.

4.2.1. Introduction.

Integrated System Management (*ISM*) de Bull, qui était disponible à partir de la fin 1992[53], sert à gérer toutes les composantes d'un système distribué, c'est-à-dire les ressources système (système d'exploitation, utilisateurs, etc.) et les ressources réseaux qui les interconnectent. La description d'*ISM* version 3 qui va suivre, est basée essentiellement sur le document [GHLDD,1994]. Le lecteur intéressé par une vue d'ensemble des fonctionnalités *d'ISM* trouvera avec [ISM,1994] une bonne référence.

Après la brève description des différents types d'utilisateurs nous enchaînerons avec l'architecture de base et les caractéristiques principales d'*ISM*. Par après nous détaillerons les stratégies d'intégration au niveau des applications. essaierons

[53] source: [ISM,1995]

4.2.2. La classification des utilisateurs.

Les utilisateurs d'*ISM* sont répartis dans quatre catégories, à savoir: les utilisateurs, les administrateurs, les développeurs et les démonstrateurs. Un utilisateur particulier peut tomber dans une ou plusieurs de ces catégories.

a) Les utilisateurs d'ISM.

Cette catégorie comprend tout ceux qui sont chargés de gérer un système distribué à l'aide des outils d'*ISM*. Nous pouvons distinguer deux sous-groupes importants concernant:

la prise de décision tactique
Les tâches suivantes servent à la prise de décision tactique et nécessitent un haut niveau d'expertise:

- coordonner et acheter des composantes matérielles et logicielles en tenant compte de la qualité offerte par les fournisseurs
- assurer la disponibilité des composantes
- assurer un investissement efficace dans les composantes matérielles et logicielles
- garder un historique des problèmes survenus
- évaluer la qualité de service
- diagnostiquer et résoudre des problèmes qui sont de l'ordre opérationnel
- enregistrer les changements planifiés et réels dans la configuration
- assurer une configuration cohérente du système distribué

Pour remplir ces tâches l'utilisateur d'*ISM* devra disposer:

- d'outils de monitoring de haut niveau qui donnent par exemple une image globale du réseau
- de rapports de gestion, comme par exemple un rapport mensuel ou hebdomadaire sur la qualité de service, un résumé des événements exceptionnels
- de requêtes ad-hoc occasionnelles

les opérations quotidiennes
Ces tâches nécessitent moins de qualifications et d'expertise que les tâches d'analyse précitées, car elles sont répétitives et routinières. Citons en guise d'exemple:

- la collecte de données sur le niveau de service et l'utilisation des ressources

- l'évaluation des alarmes et la détection de problèmes
- l'exécution de tâches de récupération
- l'exécution de tests
- le contrôle des ressources
- la distribution des configurations, des applications et des logiciels système
- l'activation et la désactivation des composantes
- la maintenance

b) Les administrateurs d'ISM.

Ils sont responsables du bon fonctionnement d'*ISM* lui-même, pour que les utilisateurs d'*ISM* disposent de tous les outils. Les administrateurs d'*ISM* remplissent des tâches telles que:

- l'installation et la configuration des composantes matérielles et logicielles d'*ISM*
- l'enregistrement des utilisateurs d'*ISM*
- le lancement et la désactivation d'*ISM*
- la détection de problèmes des composantes d'*ISM*

L'ensemble de ces tâches peut être partagé entre le personnel qui s'occupe de tâches analogues dans le système distribué. Ainsi un administrateur est responsable de l'enregistrement des utilisateurs dans *ISM* et les autres systèmes. Pour des raisons de sécurité il s'avère être utile de faire la distinction entre les administrateurs d'*ISM* et les utilisateurs d'*ISM*.

c) Les développeurs d'ISM.

Les développeurs d'*ISM* développent et enrichissent des applications de gestion. Pour ne pas interférer avec le fonctionnement d'applications, ils travaillent dans un environnement de développement. Pour développer ils utilisent:

- des interfaces de programmation (API)
- des langages et des interpréteurs
- des outils de test et de debugging

L'environnement et les outils que j'ai utilisé, en tant que développeur, durant mon stage seront décrits plus tard.

d) Les démonstrateurs d'ISM.

Les démonstrateurs d'*ISM* sont ceux qui font des démonstrations des fonctionnalités offertes par *ISM* et qui assurent la formation. (le plus connu étant H. Deghorain)

4.2.3. L'architecture d'ISM.

Rappelons que la gestion d'un système distribué se base le plus souvent sur le modèle manager-agent. D'une part nous avons les agents qui tournent sur chaque machine du système distribué et qui sont responsables du contrôle de cette machine. D'autre part les managers commandent les agents. Un agent peut être contrôlé par un ou plusieurs managers. Un manager communique avec chaque agent en utilisant des protocoles de gestion tels que SNMP, CMIP[54]. La Figure 15 suivante montre schématiquement ce mécanisme dans le cas d'ISM.

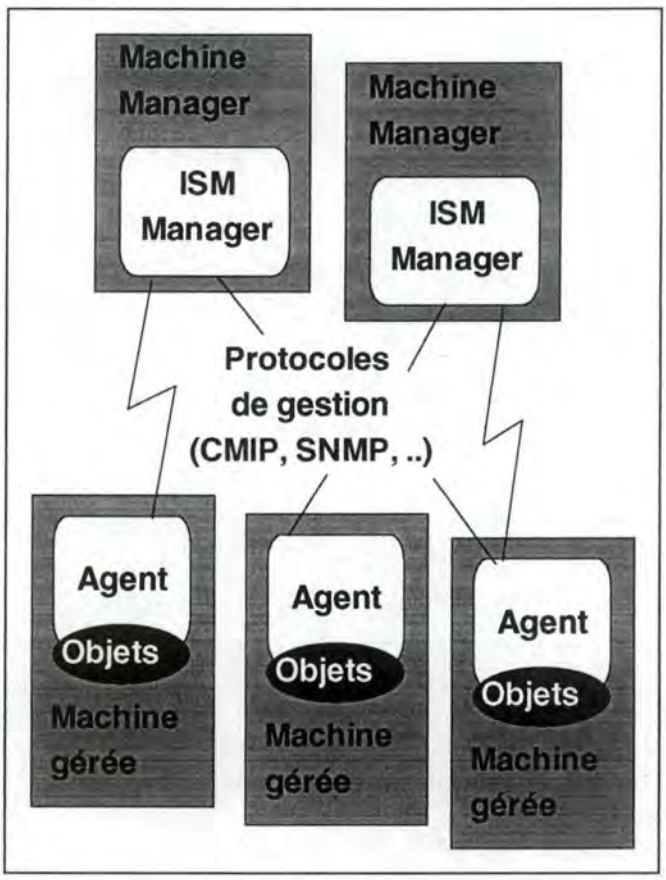

Figure 15 : L'architecture ISM.

[54] Le modèle manager-agent, ainsi que les protocoles SNMP et CMIP sont exposés dans la sous-section *2.5 Les informations de gestion*.

Le mécanisme manager-agent est basé sur le modèle objet. Toutes les composantes réelles sont représentées par des objets gérés qui sont regroupés dans une MIB. Les caractéristiques des objets (par exemple l'état d'une imprimante) sont appelées: ***attributs*** des objets.

Les objets sont regroupés dans des classes d'objets gérés ou encore ***MOC*** (*"**Managed Object Classes**"*). Chaque MOC représente un type de composante réelle. Les MOCs définissent les informations que la MIB va contenir pour chaque type de composante réelle, c'est-à-dire les attributs qu'elle va contenir.

Pour chaque MOC il y aura normalement plusieurs instances d'objets gérés, appelées ***MOI*** (*"**Managed Object Instances***"), qui représentent des instances des occurrences de composantes réelles. Prenons par exemple la MOC *imprimante* qui a les attributs *état imprimante* et *utilisateur courant*. On pourrait avoir les instances suivantes:

MOI	**état imprimante**	**utilisateur courant**
imprimante 1	on line	personne
imprimante 2	off line	personne
imprimante d'Hugo	on line	Hugo

Le manager/ISM contrôle et perçoit le système distribué à travers les objets gérés et leurs attributs. Les agents doivent gérer le lien qui existe entre la composante réelle et l'objet géré. Il peut y avoir une correspondance d'un objet géré avec plusieurs composantes réelles ou bien d'une composante réelle avec plusieurs objets MIB. Le manager/ISM envoie des requêtes et l'agent renvoie des réponses aux requêtes en utilisant un protocole de gestion. Il peut envoyer également des notifications d'événements exceptionnels au manager/ISM.

Chaque protocole de gestion comporte un certain nombre d'opérations de base. Ainsi *CMIP*[55] supporte les ***opérations*** suivantes:

1. Le manager peut lire des attributs (la fonction M-GET). L'agent répond à un M-GET avec l'information sur la composante réelle présentée sous la forme d'un objet géré.

2. Le manager peut modifier les attributs d'un objet MIB (la fonction M-SET). L'agent modifie alors la composante réelle en fonction des nouvelles valeurs d'attributs fournies par la fonction M-SET.

[55] CMIP : Common Management Information Protocol

3. Le manager peut exécuter une action sur un objet MIB (la fonction M-ACTION). L'agent exécute une action sur une composante réelle, en fonction de l'action sur l'objet géré demandé par le manager.

4. Le manager peut créer et supprimer un objet MIB (les fonctions M-CREATE et M-DELETE). L'agent crée ou supprime des composantes réelles[56].

5. L'agent peut signaler l'occurrence d'un événement sur un objet MIB par une ***notification*** (la fonction M-EVENT). L'agent envoie les informations de l'événement qui s'est produit sur un objet géré, pour refléter l'événement qui s'est produit sur une composante réelle.

4.2.4. Les caractéristiques principales d'ISM.

Cette section est axée sur les décisions de conception qui ont déterminé les caractéristiques principales de l'architecture d'*ISM*.

Un système distribué peut contenir beaucoup d'agents, utilisant une multitude de protocoles de gestion. Le système distribué peut être vu comme un ensemble hétérogène de sous-systèmes gérés, chacun utilisant son propre protocole de gestion. Ces sous-systèmes disposent d'outils indépendants, spécifiques pour un protocole. A la suite de cette constatation, *ISM*, en tant qu'outil de gestion, devrait remplir les exigences suivantes:

1. *ISM* devra fournir une gestion intégrée pour plusieurs sous-systèmes. Pour cela plusieurs protocoles de gestion devront être supportés. En plus *ISM* devrait rester ouvert pour l'introduction de nouveaux sous-systèmes avec de nouveaux protocoles de gestion dans le futur.

2. *ISM* devra fournir un ensemble cohérent d'applications qui géreront tous les sous-systèmes. C'est la raison pour laquelle les différences entre les protocoles de gestion devront être cachées à l'utilisateur final partout où c'est possible.

[56] Remarquons que les "composantes" réelles ne peuvent pas toutes être créées ou supprimées. Ces fonctions s'appliquent typiquement à des structures de données comme des tables de routage par exemple.

Pour satisfaire à ces exigences, les concepteurs d'*ISM* ont opté pour une architecture en couches:

1. La couche la plus basse du manager/ISM (l'agent intégrateur ou AI) contient du code qui est spécifique pour un protocole de gestion donné. La couche la plus haute du manager/ISM contient les applications qui seront le plus générique que possible. Les applications accèdent aux AIs en utilisant le service CMIS qui est transporté par un protocole commun interne au manager/ISM et indépendant du protocole de gestion du sous-système.

2. Les applications manager/ISM doivent voir tous les sous-systèmes gérés à travers une seule MIB composée. L'agent intégrateur est responsable de la traduction d'un protocole de gestion et d'une MIB spécifiques vers le protocole et la MIB du manager/ISM.

Une autre particularité de l'architecture d'*ISM* est qu'elle permet à plusieurs managers de coopérer à travers l'échange d'informations de gestion. *ISM* permet la communication de manager à manager, ce qui servira à construire des hiérarchies de managers. Dans une telle hiérarchie, le protocole standard CMIP est utilisé de telle manière à ce qu'un manager (appelé le supra-manager) voie les autres managers comme agents (voir la Figure 16). Ce mécanisme est d'autant plus intéressant qu'un système distribué est plus large. Il est utilisé d'ailleurs aussi pour la communication entre managers de sites miroir.

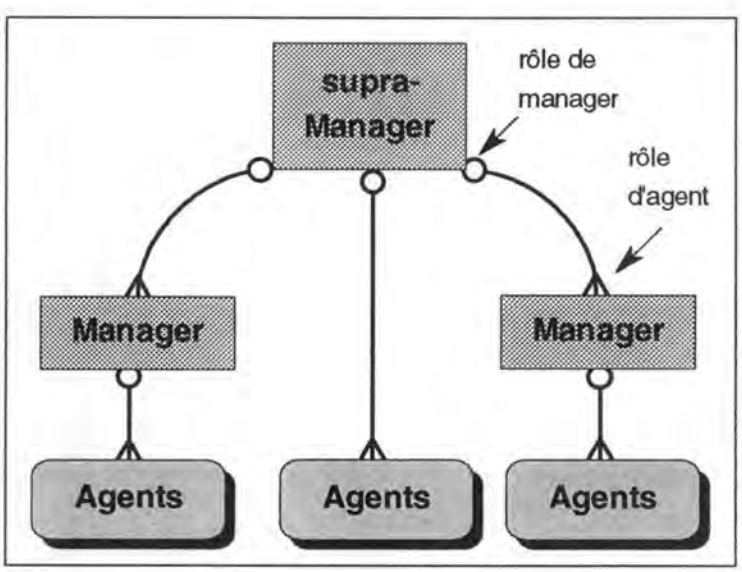

Figure 16 : Une hiérarchie de managers.

4.2.5. L'intégration au niveau des applications.

ISM supporte un grand nombre de fonctions qui agissent sur un large éventail de MOCs. Il est important, d'un point de vue ergonomique, que la diversité des fonctions n'entraîne pas une diversité des interfaces homme-machine. C'est la raison pour laquelle *ISM* fournit un ensemble d'outils qui peuvent servir à intégrer les applications sous une interface homme-machine commune.

1. *ISM* fournit une application (*ISM-Monitor*), qui est située à un haut niveau et qui est généralement la première application lancée quand un utilisateur se connecte sur *ISM*. Cette application va afficher une fenêtre contenant une image schématique du réseau. Une application associée (*ISM Application Board*) contient les icônes pour toutes les autres applications *ISM*. Ces deux applications permettent le lancement d'applications selon leur contexte. Le contexte est par exemple un objet sélectionné sur un graphique. Lors du lancement, des informations sur cet objet sont passées à l'application. Notons que toutes les applications d'*ISM* peuvent être accédées à partir de ce point unique.

   ISM-Monitor[57] affichera graphiquement les MOIs et leur statut. Ces informations peuvent être visualisées par la même application. *ISM-Monitor* peut montrer aussi les MOIs sous forme tabulaire. Les fonctions de base de lecture ou de modification des attributs peuvent être effectuées via la même application.

2. *ISM* fournit un service d'*Alarm Log* qui peut être utilisé par toute composante qui veut signaler une condition exceptionnelle. Une application de gestion des alarmes (*ISM-Alarm*) permet de visualiser et de gérer les alarmes. Toutes les occurrences de conditions exceptionnelles peuvent être visualisées par ce mécanisme.

3. *ISM* fournit aussi un langage de haut niveau spécifique pour la gestion de systèmes distribués (*SML*). Ce langage de haut niveau contient des primitives qui supportent l'interface homme-machine d'*ISM*. De cette manière les développeurs sont incités à créer des applications avec une interface homme-machine cohérente.

[57] voir au paragraphe 4.3.3.a)

4.3. L'ISM Manager.

Pour commencer nous donnerons un bref aperçu de l'architecture d'*ISM* et de ses composantes. Dans la suite la structure et le contenu de la MIB ainsi que les applications *ISM* vont être discutées.

4.3.1. L'architecture du Manager.

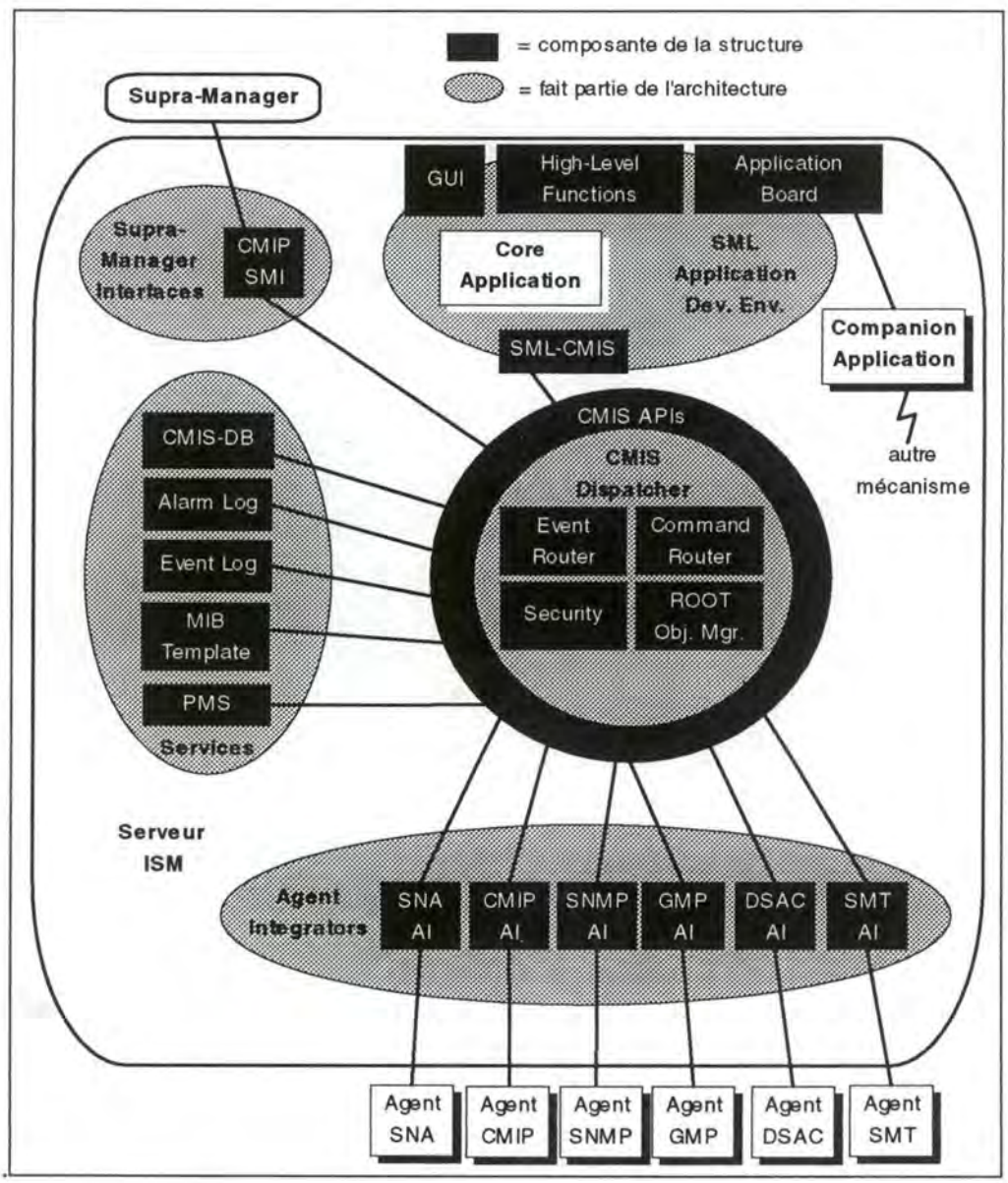

Figure 17 : L'architecture interne d'ISM.

Dans les paragraphes suivants nous allons expliquer brièvement les composantes principales de l'architecture du manager/*ISM*, telles qu'elles sont représentées sur la Figure 17.

a) Les applications ISM.

Les applications sont les composantes d'*ISM* qui ont une interface homme-machine. Plusieurs copies d'une même application peuvent être actives. Pour chaque fenêtre utilisateur il existe une copie active de l'application correspondante. Les utilisateurs peuvent se connecter via un terminal X ou une station de travail. Dans la plupart des cas les applications sont développées dans le langage *SML*. Ce langage interprété, basé sur COMMON LISP, utilise des librairies de fonctions compilées.

Les *Core Applications* constituent le noyau du manager/ISM et elles tournent sur les machines manager/ISM. Ces applications utilisent les protocoles standards *ISM* et la *Structure* commune du manager/ISM.

Les *Companion Applications* tournent aussi sur les machines manager/ISM, mais elles n'utilisent pas complètement les services de la plate-forme. Leur niveau d'intégration est plus ou moins élevé en fonction de leur ouverture fonctionnelle. Ce sont des applications (tableurs, systèmes *Trouble Ticket*[16], etc.) qui existent indépendamment d'*ISM*, cependant elles sont utilisées fréquemment par le personnel qui assure la gestion.

b) Les Services ISM.

Les services sont les composantes d'*ISM* qui n'ont pas d'interface homme-machine et qui fournissent des fonctions indépendantes d'un protocole de gestion. Les services existants sont: le MIB Template Service (voir à la sous-section *4.4.2*), l'Alarm Log Service (voir à la sous-section *4.4.3*), l'Event Log Service (voir à la sous-section *4.4.4*), la CMIS Database (voir à la sous-section *4.4.1*)et *PMS* (voir à la section *5.3*).

c) Les Agent Integrators ISM.

Les *Agent Integrators* (AIs) communiquent avec les agents d'un sous-système de gestion spécifique en utilisant le protocole de gestion de l'agent. Ils communiquent avec les autres composantes d'*ISM* à l'aide du service CMIS. Les AIs existants sont: *SNMP AI, SNA*[58] *AI, SMT*[59] *AI, DSAC/AEP*[60]

[58] SNA : Systems Network Architecture (IBM)

[59] SMT sert à gérer des réseaux FDDI (Fiber Distributed Data Interface).

[60] DSAC/AEP : Distributed Systems Administrative Control / Administrative Exchange Protocol

AI, CMIP AI et GMP[61] *AI*. Une boîte à outils permet de développer de nouveaux AIs qui se basent sur des protocoles standards ou propriétaires.

Les AIs assurent la traduction entre le protocole de l'agent et les primitives de service CMIS. Les AIs servent d'unique point d'entrée pour les agents et ils s'occupent de la communication interne avec les applications *ISM* via le *CMIS Dispatcher*. Les applications de leur côté n'ont pas besoin de connaître l'endroit où l'agent se trouve. L'AI fait la traduction d'une MOI vers une adresse dans un sous-système de gestion.

Finalement les AIs traduisent les représentations spécifiques des MOIs de l'agent vers la représentation interne des objets dans le manager/ISM (description *GDMO*[62]). (voir à la section *4.5*)

d) Les ISM Supra-Manager Interfaces.

Les *Supra-Manager Interfaces* (*SMI*) permettent à un manager/ISM de communiquer avec d'autres managers. Dans *ISM*3 le *CMIP SMI* est la seule interface supra-manager fournie.

e) L'ISM CMIS Dispatcher.

Toutes les composantes du manager/ISM communiquent à travers le *CMIS Dispatcher* en utilisant des protocoles et des services standards. Il comprend: les *CMIS Application Programmatic Interfaces* (API), un mécanisme de messages interne qui assure la communication entre les applications les services et les AIs, le *Command Router* qui aiguille les requêtes, le *Root Object Manager* et l'*Event Router* pour le routage des notifications. (voir à la section *4.6*)

4.3.2. La MIB ISM.

Cette section décrit la structure et le contenu de la *MIB ISM*.

Toutes les classes d'objets gérés (MOCs) dans la *MIB ISM* sont basées sur les spécifications standards des *Guidelines for the Definition of Managed Objects* (GDMO) de l'ISO. La traduction nécessaire de la syntaxe MIB du sous-système de gestion vers la syntaxe *ISM* est assurée par les AIs responsables.

L'*ISM MIB Template* (ou schéma) intègre toutes les MOCs nécessaires à la gestion d'un système distribué.

---

[61] GMP : GCOS Management Protocol (Bull)

[62] GDMO : OSI Guidelines for the Definition of Managed Objects

a) La structure MIB.

Comme nous venons de voir précédemment, la MIB *ISM* contient les classes d'objets qui sont spécifiques pour chaque sous-système de gestion. C'est la raison pour laquelle la MIB *ISM* va contenir des MOIs SNMP, DSAC/AEP, GMP, SNA, FDDI, CMIP, etc.

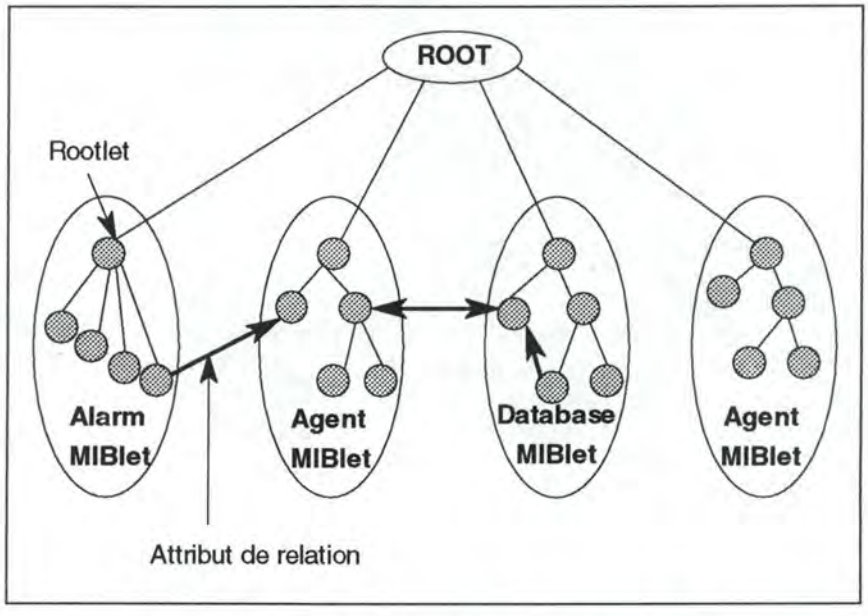

Figure 18 : La structure de la MIB ISM.

La Figure 18 ci-dessus montre que la MIB *ISM* peut être vue comme un ensemble de sous-MIBs, qui sont encore appelés *MIB-lets*. Chaque MIBlet est un sous-arbre de la MIB *ISM* et est rattachée directement à la racine (*root*). Les MOIs de la MIBlet qui sont directement rattachées à la racine, ont des propriétés spéciales et sont appelées *Rootlets*. Chaque MIBlet possède une seule Rootlet.

b) Les Object Managers.

Une composante d'*ISM*, qui est appelée *Object Manager* ou gestionnaire d'objets, gère chaque MIBlet. Les AIs sont des gestionnaires d'objets car ils rendent accessibles les objets relatifs aux agents. Les services *ISM* sont aussi des gestionnaires d'objets puisqu'ils implémentent des objets qui sont locaux à *l'ISM Manager*.

Les gestionnaires d'objets permettent un certain nombre d'opérations sur leur MIBlet (M-GET, M-SET, etc.). Les AIs agissent sur leur MIBlet Agent respective. Chaque MIBlet Agent contient les MOCs définies pour un sous-système de gestion spécifique. Les services suivants utilisent des MIBlets spécifiques: *CMIS Database, MIB Template Service, Alarm Log Service* et *Event Log Service*. Une MIBlet Service contient les MOCs nécessaires pour fournir le service.

c) Les opérations sur la MIB.

Malgré la division en MIBlets, la MIB *ISM* est vue comme une entité. Une application peut effectuer des opérations sur les MOIs partout dans la MIB *ISM* sans connaître la composante qui instancie la MIBlet. Il suffit que l'application donne le nom[63] de la MOI pour que le *CMIS Dispatcher* passe la requête au gestionnaire d'objet responsable pour la MIBlet. Le routage des notifications par contre se fait en fonction de filtres de notifications créés par les services et les applications. Dans ces filtres ils spécifient quelles notifications ils désirent recevoir. Les filtres contiennent par exemple le type de notification, la MOI qui est à la source de la notification, etc.

d) Le nommage des objets.

La MIB est organisée hiérarchiquement en forme d'arbre. Cet arbre est appelé *arbre de contenance* ("*Containment Tree*"). Les MOIs sont nommées en fonction de leur position dans l'arbre. Chaque MOI a un *nom relatif* ou *RDN* ("*Relatif Distinguished Name*"), qui est unique parmi les MOIs appartenant à une seule MOI supérieure. Un attribut de la MOC, appelé *attribut de nommage*, est choisi pour contenir le nom relatif par rapport à un supérieur donné. Chaque MOI a un nom absolu ou *DN* (*Distinguished Name*) qui est unique pour la MIB. Le DN d'un objet est obtenu récursivement à partir de la concaténation de son RDN avec le DN de son supérieur.

e) Les relations entre objets.

La MIB *ISM* permet d'utiliser d'autres relations que des relations hiérarchiques de nommage entre objets. Ces relations sont mises en oeuvre à l'aide d'*attributs de relation*. Les attributs de relation contiennent les DN d'objets, qui sont utilisés comme "pointeurs" d'un objet vers un autre. Pour illustrer ce concept, prenons un exemple:

L'*Equipment MOC* dans la *Database MIBlet* contient l'attribut de relation *Vendor Name*. Cet attribut contient le DN du fournisseur de l'équipement. De cette manière nous pouvons retrouver les informations sur le fournisseur dans la *Vendor MOC*, qui se trouve aussi dans la *Database MIBlet*.

Quel est l'intérêt des attributs de relation?

1. Grâce à ce mécanisme *ISM Monitor* peut traverser la MIB de manière transversale.

[63] Le mécanisme de nommage des MOIs sera traité dans le paragraphe d.

2. De même, *ISM Alarm* peut en partant d'un enregistrement d'une alarme, qui contient un attribut de relation, retrouver et visualiser la MOI sur laquelle l'alarme fut détectée.

3. L'utilisateur peut, en utilisant *ISM Monitor* ou d'autres applications spécifiques, modifier ces relations pour refléter des changements de topologie par exemple.

Notons cependant qu'il existent des restrictions à ce mécanisme. Par exemple dans la MIBlet SNMP il n'est pas possible d'avoir des pointeurs sur des objets d'autres MIBlets.

f) Les extensions MIB.

Des nouvelles MIBs peuvent être rajoutées à *ISM* pour diverses raisons:

- pour servir de support à une nouvelle composante qui contient un agent
- lors du développement d'un nouvel agent ou de méthodes d'agent
- lors de l'ajout d'une nouvelle MOC à *CMIS Database*
- etc.

Pour rajouter une MIB modifiée ou nouvelle, il faut la compiler et l'installer dans le *MIB Template Service*[64].

[64] voir au point *4.4.2 Le MIB Template Service*.

4.3.3. Les applications ISM

Ce chapitre décrit les applications qui font partie de la *Structure ISM*[65], c'est-à-dire celles qui sont utilisées par d'autres applications.

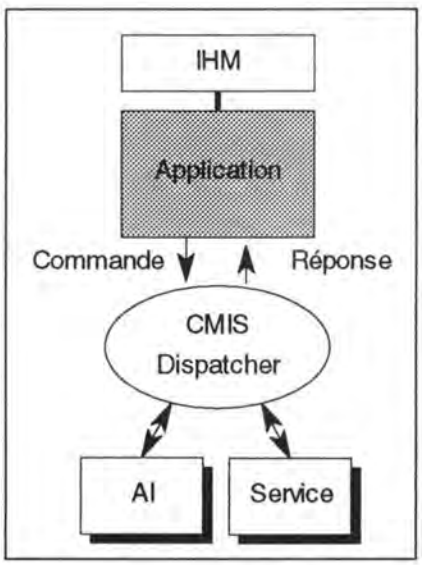

Figure 19 : Les applications ISM.

Une application utilise des fonctions *SML* pour l'interface homme-machine (IHM) graphique et l'API *CMIS Dispatcher* pour communiquer avec les autres éléments d'ISM (AIs et Services). Ce mécanisme est illustré sur la Figure 19.

a) L'ISM Monitor.

Cette application fournit la fenêtre *ISM* de plus haut niveau. Cette fenêtre contient une image du réseau montrant les systèmes, les connexions, etc., qui sont animés en fonction de leur état. Une fenêtre séparée montre *l'ISM Application Board* qui contient toutes les applications *ISM* disponibles.

ISM Monitor utilise un fichier graphique qui contient une image de tout ou d'une partie du système distribué géré. Les MOIs et leurs attributs sont représentés par des symboles. L'état d'une MOI sera indiqué avec des couleurs. L'image peut être animée en choisissant des attributs spécifiques et en les monitorant. De tels attributs sont par exemple l'état opérationnel, le pourcentage utilisé de la capacité, etc. Il est également possible d'animer les icônes avec d'autres mécanismes (par exemple avec des alarmes).

[65] ISM Framework

L'utilisateur peut sélectionner une MOI et faire afficher les détails sous forme tabulaire. Il peut passer à une autre MOI. Utilisé de cette manière, *ISM Monitor* sert à parcourir la MIB.

L'application *ISM Monitor* peut associer aussi un *panneau*[66] à une MOI, qui peut être visualisé sur demande. Ce panneau est représenté par une fenêtre qui contient une sélection d'attributs. Ces attributs peuvent provenir de diverses MOIs. Par exemple un tableau associé à un système d'ordinateur pourra montrer la charge CPU, le nombre d'alarmes survenues, le nombre de connexions TCP, etc.

En sélectionnant un symbole d'une MOI et en invoquant un menu, un utilisateur est guidé dans son choix d'actions possibles sur cette MOC. Ce choix comprend notamment: l'affichage du tableau de bord, une fonction de zoom sur la MOI pour voir une image plus détaillée, l'affichage des détails d'une MOI, la modification des attributs sélectionnés, la représentation des alarmes pour la MOI.

ISM Monitor peut être utilisé pour diverses tâches de gestion telles que:

- le monitoring d'état de haut niveau
- la présentation détaillée de bas niveau des objets et des attributs gérés (parcours MIB)
- la gestion de la configuration et le contrôle via les MIBlets Agents.
- la gestion de l'inventaire et de la topologie via *CMIS-DB*

b) L'ISM Application Board.

La fenêtre de cette application contient les applications *ISM* sous forme iconique. Les icônes de *l'Application Board* sont divisées en trois catégories:

- administrateur *ISM* : les applications pour administrer et configurer *ISM*
- utilisateur *ISM* : les applications pour la gestion système
- développeur *ISM* : les boîtes à outils

c) Les autres applications.

Dans ce paragraphe nous allons passer rapidement en revue trois autres applications moins importantes dans le cadre de notre étude.

- L'administrateur d'ISM peut utiliser *ISM Configurator* pour configurer *ISM*. La configuration est stockée dans *CMIS-DB*.

---

[66] Instrument Panel

- La composante *ISM Script* est une extension des shells scripts standards d'Unix. Les commandes, utilisées dans les scripts, permettent l'accès à l'interface homme-machine, au *CMIS Dispatcher*, au *CMIS MIB Template* et au gestionnaire d'événements.
- *ISM Data Export* sert à exporter les informations contenues dans les MIB *ISM* vers des fichiers ASCII.

4.4. Les services.

L'ISM Manager fournit un ensemble de services disponibles aux applications. Dans la suite nous allons aborder les services suivants: la *CMIS Database*, *le MIB Template Service*, *l'Alarm Log Service*, *l'Event Log Service* et le *Performance Monitoring Service*.

4.4.1. La CMIS Database (CMIS-DB).

Ce service maintient un repository de MOIs et en garantit la persistance. Les MOCs dans *CMIS-DB* sont locales à *l'ISM Manager* et n'ont pas obligatoirement de contrepartie "réelle". Elles sont utilisées pour représenter le système distribué du client, selon le type de la représentation choisie (par exemple: l'inventaire, la topologie réseau).

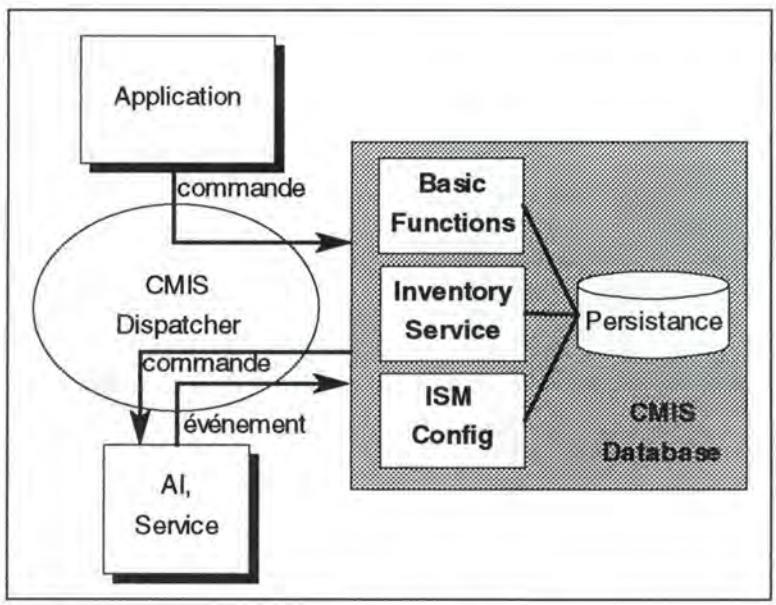

Figure 20 : La CMIS Database.

La Figure 20 ci-dessus montre la place de *CMIS-DB* dans *ISM*. Les paragraphes suivants présentent les trois types de services qu'offre *CMIS-DB*: les fonctions de base, *l'Inventory Service* et le *Configuration Service.*

a) Basic Functions.

Un ensemble de fonctions de base est offert à toutes les applications. En utilisant ces fonctions, un utilisateur est libre de manipuler les données qu'il désire dans *CMIS-DB*, à condition qu'il respecte les contraintes des définitions MOC *ISM*. Les fonctions suivantes sont disponibles:

- M-CREATE: Les applications peuvent créer de nouvelles instances d'un objet géré.
- M-DELETE: Les applications peuvent supprimer des instances d'un objet géré.
- M-GET: Les applications peuvent accéder aux attributs des instances d'objets gérés.
- M-SET: Les applications peuvent ajouter, effacer et modifier les attributs des instances d'objets gérés.

Ce service offre aussi les notifications ENROL et DEENROL qui sont utilisées par le *Root Object Manager* après un M-CREATE ou un M-DELETE sur une Rootlet (voir à la section 4.6.2).

D'autres fonctions permettent aussi d'enrichir le comportement de CMIS-DB et donc de la MIB:

- M-ACTION: Des méthodes SML peuvent faire partie de la MIB.
- MIB Import/Export: La MIB peut être importée de ou vers un fichier.

b) Inventory Service.

Ce service peut être utilisé pour le stockage d'informations sur l'inventaire et sur la topologie, en utilisant la *CMIS-DB Inventory* MIBlet.

c) Configuration Service.

Ce service est utilisé par les AIs et le SMI. Il sert à stocker les détails de la configuration de l'*ISM Manager*.

4.4.2. Le MIB Template Service.

Le contenu de chaque MOC est décrit par un schéma (*"Template"*), qui est utilisé par les applications génériques pour interpréter cette MOC. Ce schéma est fourni par un service du manager/ISM, le *MIB Template Service*, qui permet aux applications d'accéder au schéma contenant des informations sur les différents objets et attributs gérés par *ISM*.

Le *MIB Template Service* pourra contenir par exemple les informations suivantes:

- pour chaque ***classe***:
  un nom qui identifie la classe,
  une description textuelle,
  les opérations supportées par les objets de cette classe,
  le type de représentation (graphique, tabulaire),
  etc.
- pour chaque ***attribut***:
  un nom qui identifie l'attribut,
  une description textuelle,
  les opérations d'accès permises,
  la syntaxe (en ASN.1[67]),
  le type de représentation,
  etc.

Le *MIB Template Service* contient les définitions d'objets. Ce sont les définitions des objets MIB proprement parlé, englobant les attributs, la syntaxe, etc. Cette information, fournit par les développeurs d'agents et les gestionnaires d'objets, ne changera que si les MOCs changeront elles-mêmes (par exemple lors du changement du code d'un agent).

Comment étendre la MIB *ISM*? La Figure 21 ci-dessous illustre les différentes phases d'installation d'une nouvelle MIB. Une boîte à outils, appelée *MIB Integration Kit,* permet aux développeurs *ISM* et aux utilisateurs d'intégrer leurs propres MIBs dans le *MIB Template Service*. Cet outil crée un *MIB Package*. La collection de toutes les informations nécessaires à *ISM* pour prendre en charge un ensemble de MOCs s'appelle *MIB Package*. Il contient les éléments suivants: la définition *ISM GDMO* d'objets, les extensions graphiques, les icônes, la configuration SNMP AI (pour les objets dans la SNMP MIBlet).

[67] ASN.1 : Abstract Syntax Notation One (ASN.1 est le langage OSI pour décrire une syntaxe abstraite. Une syntaxe abstraite est une définition de types de données indépendante de toutes les structures et restrictions spécifiques à une machine particulière.)

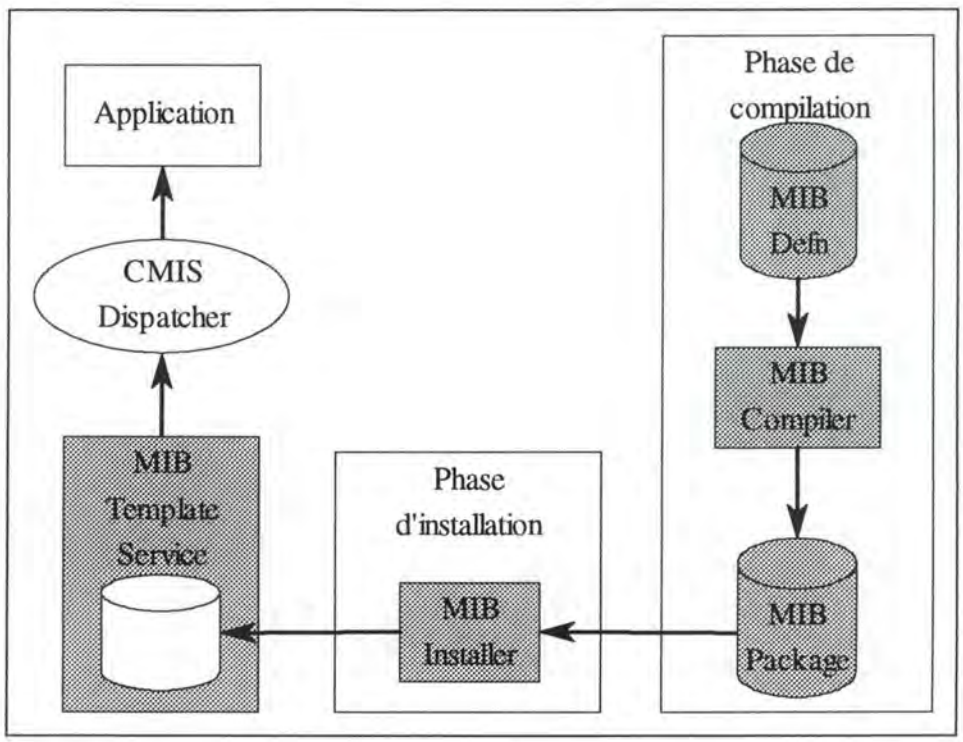

Figure 21 : Le MIB Template Service.

Le compilateur, appelé *MIB Compiler*, prend en charge les définitions d'objets ainsi que les extensions graphiques d'objets. Dans le cas de SNMP, une étape de compilation préliminaire permet de transformer une définition standard d'une MIB SNMP dans le format *ISM GDMO*. Ensuite cette définition de MIB en *ISM GDMO* est compilée pour obtenir le *MIB Package*. Une fois la MIB compilée, l'utilitaire *MIB Installer*, permet de l'installer dans le *MIB Template Service*.

Quand une nouvelle MIB est définie, les objets doivent obtenir des identifiants, appelés *Object Identifiers* (OIDs)[68]. Dans le cas de SNMP, la traduction des définitions SNMP MIB dans *ISM* GDMO nécessite la définition d'objets supplémentaires et donc l'assignation d'OIDs supplémentaires. Les OIDs de CMIS et de SNMP font partie de l'arbre global d'enregistrement d'objets (*"Object Registration Tree"*), dont une partie est visible sur la figure 9 (sous-section 2.5.5).

4.4.3. L'Alarm Log Service.

Ce service implémente la *MIB Alarm*, qui contient les MOCs pour gérer des alarmes. Ces MOCs comprennent, comme nous pouvons le constater sur la Figure 22, *l'Alarm Log*, *l'Alarm Record* et le *Stats Block*.

[68] voir à la sous-section *2.5.5 La gestion dans le monde IAB.*

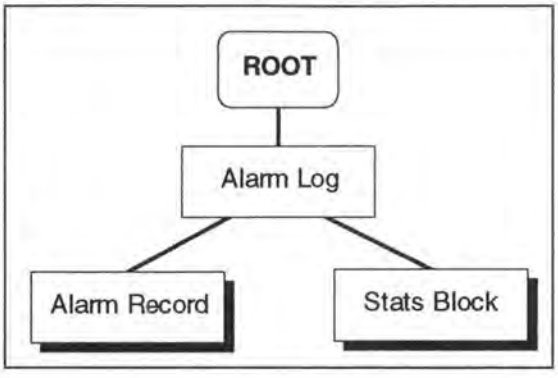

Figure 22 : L'Alarm MIBlet.

L'Alarm Log est un journal[69] d'alarmes qui est géré par *l'Alarm Log Service*. *L'Alarm Record* est un enregistrement qui représente une alarme. Le *Stats Block* contient des statistiques sur l'alarme dans le journal. Il contient une MOI pour chaque Rootlet dans la *MIB ISM*.

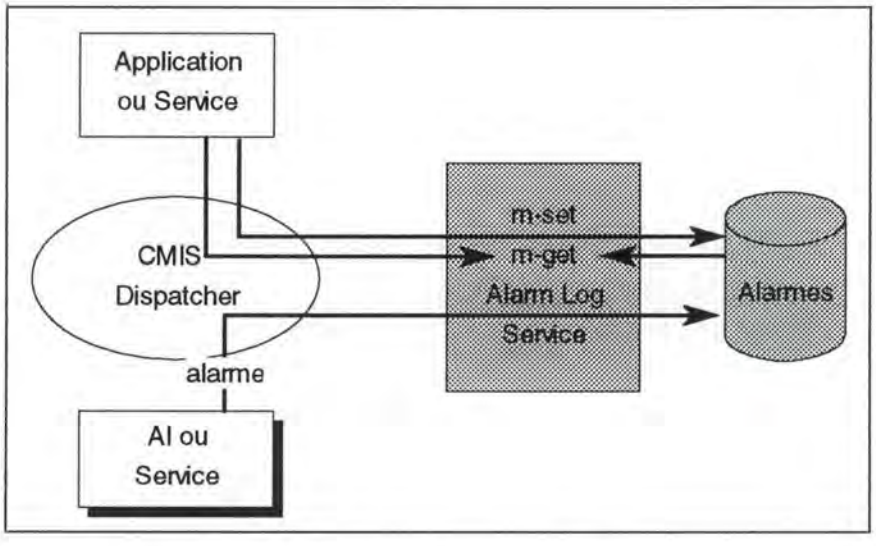

Figure 23 : L'Alarm Log Service.

Comme nous pouvons le voir sur la Figure 23, les alarmes sont reçues par les gestionnaires d'objets (service ou AI) et stockées dans le journal (Alarm Log) en tant qu'*Alarm Record* MOIs. Les applications peuvent accéder à ces *Alarm Records* par le biais des opérations M-SET et M-GET.

Des alarmes répétitives, c'est-à-dire plusieurs alarmes concernant le même problème sur la même MOI, sont comptées et stockées avec le temps de la première et de la dernière survenance. Pour éviter que le journal ne croisse indéfiniment, seule la dernière occurrence de l'alarme est gardée.

[69] Dans la suite nous utilisons "journal" et "(fichier) log" comme synonymes.

Quand une condition, qui a déclenchée une alarme, cesse d'exister une notification "clear" peut être générée par le gestionnaire d'objets en question. Cette notification sera traitée de la même manière qu'une alarme répétitive. Des alarmes qui ne sont pas "libérées" (cleared) par le gestionnaire d'objets responsable, continueront à exister dans le log. Une application peut utiliser l'attribut qui contient le temps de dernière survenance comme filtre, c'est-à-dire elle peut demander uniquement des alarmes qui se sont passées après un certain temps. *ISM Alarm* interroge périodiquement le journal d'alarmes avec ce filtre pour détecter de nouvelles alarmes. *L'Alarm Log Service* est lancé durant le début d'exécution de l'*ISM Manager* et il va rester opérationnel de manière continue.

4.4.4. L'Event Log Service.

Ce service maintient l'*Event MIBlet*, qui contient des MOCs utilisées pour gérer des événements. Comme nous pouvons le constater sur la Figure 24, les MOCs suivants peuplent *l'Event MIBlet*:

- *l'Event Log Manager* représente le service lui-même
- *l'Event Log*, représente un log d'événement géré par *l'Event Log Service*
- *l'Event Record*, représente une instance d'un événement

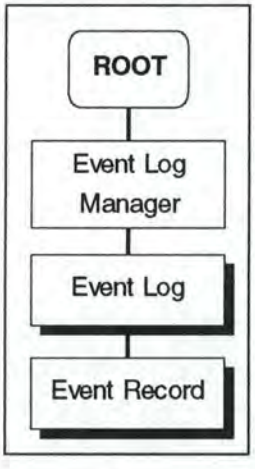

Figure 24 : L'Event MIBlet.

Les gestionnaires d'objets interceptent les notifications et les stockent comme enregistrements (*Event Record*) dans l'*Event Log*. La MOC *Event Log* contient un attribut, appelé *Discriminator Construct*, qui peut contenir un filtre pour sélectionner les événements qui seront loggés dans ce journal.

Les applications peuvent accéder à ces enregistrements avec les opérations M-GET, M-SET et M-DELETE. M-GET peut contenir un filtre qui permet de sélectionner un sous-ensemble d'événements.

Ce service est mis en route au début du lancement de *l'ISM Manager* et il reste opérationnel de façon continue.

4.4.5. Le Performance Monitoring Service.

Ce service va être exposé dans le chapitre 5 qui traite de la gestion des performances sous *ISM*.

4.5. Les Agents Integrators (AIs).

La couche la plus basse du manager *ISM* comprend les AIs, dont chacun est responsable pour un sous-système de gestion particulier. La Figure 25 montre les interactions de l'AI avec les autres composantes d'ISM. Au point suivant nous allons décrire l'architecture des AIs. Ensuite nous aborderons la génération d'alarmes. Pour finir nous expliquerons le fonctionnement d'un AI particulier: le SNMP AI.

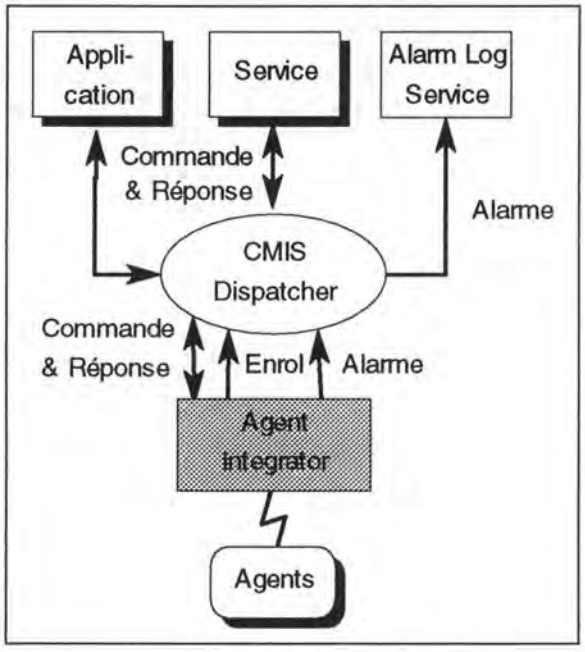

Figure 25 : Un ISM Agent Integrator.

4.5.1. L'architecture des AIs.

Un agent intégrateur va traduire les requêtes M-CREATE, M-DELETE, M-GET et M-SET dans des requêtes équivalentes du sous-système de gestion. Ensuite il va les passer à l'agent du sous-système pour exécution. Les réponses à ces requêtes vont être transformées dans les confirmations correspondantes. En plus des notifications seront générées en fonction de celles envoyées par les agents.

Dans certains cas, l'AI peut répondre immédiatement, si l'information requise est gardée localement en cache. Le choix, (1) de garder l'information localement en cache, (2) d'interroger le sous-système de gestion ou (3) d'utiliser une combinaison des deux techniques, dépend du protocole du sous-système de gestion et est par conséquent du ressort du concepteur de l'AI.

Comme nous allons le voir dans le chapitre 4.6.2, le *Root Object Manager* requiert que les gestionnaires d'objets envoient une notification ENROL pour chaque MOI sous leur contrôle qui est une *Rootlet,* c'est-à-dire une subordonnée directe de *Root.* La Rootlet existe alors jusqu'à ce qu'il arrive un M-DELETE pour cette *Rootlet* ou que l'AI soit arrêté. Dans de tels cas, une notification DEENROL est envoyée au *Root Object Manager.*

Remarquons que si la communication avec l'agent est rompue, la Rootlet existe toujours, mais change d'état (l'état est mis à "down"). Quand la communication reprend avec l'agent, l'état initial est rétabli (il est remis à "up"). Il se peut qu'entre temps la configuration de l'agent ait changée, car il a été arrêté et relancé avec un nouvel ensemble d'objets gérés. Ce genre de problèmes est réglé par l'application.

4.5.2. La génération d'alarmes.

Les alarmes qui proviennent de l'agent du sous-système de gestion, sont traduites dans le format *ISM* et puis passées vers le haut. Une deuxième source d'alarmes est l'AI lui-même quand il a détecté, par exemple, qu'un agent est tombé en panne.

Chaque AI doit être configuré. La configuration comprend notamment les noms et adresses des agents avec lesquels un AI donné va communiquer. Cette information est stockée dans l'*ISM Configuration Service*[70].

Pour l'instant les agents intégrateurs suivants existent sous *ISM*: SNMP AI, DSAC/AEP AI, CMIP AI, GMP AI, SNA AI, SMT AI, etc. Au point suivant nous allons traiter le SNMP AI.

---

[70] voir le paragraphe *4.4.1.c ISM Configuration Service.*

4.5.3. Le SNMP Agent Integrator.

Cet AI supporte les sous-systèmes de gestion qui utilisent le protocole SNMP. L'AI traduit les requêtes *ISM* en requêtes SNMP et les passe au système SNMP approprié. Les traps[71] SNMP sont converties dans des alarmes *ISM* et passées à *l'Alarm Log Service*. Des notifications ENROL sont générées dès que les Rootlets sont découvertes. La table dans l'annexe A résume quelles fonctions sont disponibles pour une application *ISM* qui utilise le protocole SNMP. L'AI traduit la MIB SNMP présentée par l'agent dans une MIB *ISM GDMO*.

4.6. Le CMIS Dispatcher.

Toutes les composantes de l'*ISM Manager* (Applications, Services et AIs) communiquent entre elles à travers le *CMIS Dispatcher* (CD). Il implémente les services suivants: le *Command Router*, le *Root Object Manager* et *l'Event Router*.

4.6.1. Le Command Router.

Les applications *ISM* devraient voir la MIB comme entité, sans devoir se soucier quel gestionnaire d'objets est responsable pour une MOI. Le *Command Router* connaît la localisation de chaque gestionnaire d'objets et il s'occupe du routage des commandes vers le gestionnaire d'objets approprié. Ce routeur prend en charge les requêtes suivantes: M-GET, M-SET, M-CREATE et M-ACTION.

Quand le *CMIS Dispatcher* reçoit une requête d'une application ou d'un service, il utilise un répertoire, appelé le *Rootlet Directory*, pour déterminer quel gestionnaire d'objets (Service ou AI) est responsable de l'objet spécifié dans la requête (voir Figure 26).

Pour chaque *Rootlet*, ce répertoire contient le nom et l'adresse du gestionnaire d'objets qui "possède" la *Rootlet* et tous ses subordonnés. Dès que le *Root Object Manager* obtient une notification ENROL, il met à jour le répertoire avec le nom de la nouvelle *Rootlet* et l'adresse de son gestionnaire d'objets. La *Rootlet* et son arbre de contenance seront donc accessibles aux applications, dès qu'une notification ENROL a été reçue par le *Root Object Manager*. De même des notifications DEENROL sont captées et traitées par le *Root Object Manager*. Dans ce cas la *Rootlet Directory* est mise à jour lors du prochain relancement du *CMIS Dispatcher*

[71] Les *traps* sont les notifications dans le protocole SNMP.

4.6.2. Le Root Object Manager.

Le *Root Object Manager* (*ROM*) est le gestionnaire d'objets pour l'objet *Root*. Le *ROM* maintient un répertoire de *Rootlets* (*Rootlet Directory*), comme nous l'avons anticipé au point précédent. Chaque gestionnaire d'objets doit faire connaître ses *Rootlets* au *ROM*, en envoyant des notifications ENROL au *ROM*. Toutes les *Rootlets* doivent être instanciées.

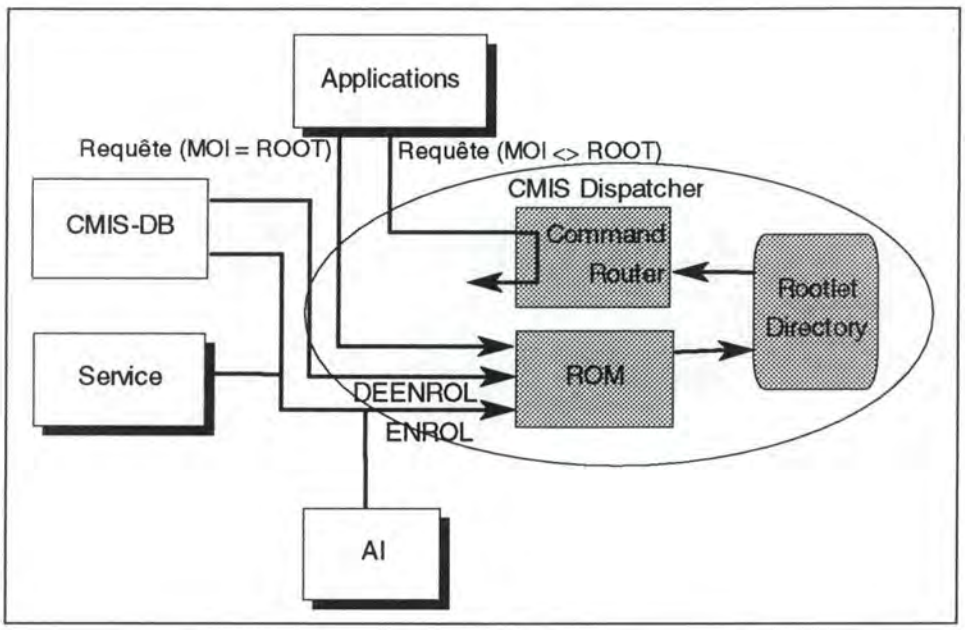

Figure 26 : Le Root Object Manager.

La Figure 26 montre le fonctionnement du *ROM* et du *Command Router* au sein du *CMIS Dispatcher*.

4.6.3. L'Event Router.

Le *CMIS Dispatcher* s'occupe aussi du routage d'événements pour les autres composantes *d'ISM Manager*. *L'Event Router* maintient des *Events Forwarding Discriminators* (*EFDs*) par gestionnaire d'objets, qui dirigent le flux des événements qui parviennent des gestionnaires d'objets. Une application ou un service peuvent accéder à ces *EFDs* pour déterminer quels événements ils vont recevoir. Le gestionnaire d'objets fait parvenir l'événement à *l'Event Router*, qui lui l'aiguille en fonction des *EFDs* en vigueur pour cet gestionnaire d'objets. Ce mécanisme est utilisé:

- par *l'Alarm Log Service* pour obtenir des notifications d'alarmes des AIs
- par le *ROM* pour recevoir des notifications ENROL et DEENROL

- par d'autres applications et services pour recevoir des événements selon leurs besoins

4.7. Les Agents.

Nous venons de décrire l'architecture du manager/ISM. Dans cette section les *Agents* vont être présentés en prenant le cas des agents SNMP. Nous avons choisi ces agents puisque dans le prochain chapitre nous allons décrire un agent SNMP de mesures de performances.

4.7.1. Les Agents SNMP.

Nous allons commencer par la description d'un outil de développement. Le paragraphe suivant montre le fonctionnement du *SNMP Dispatcher*, qui permet de faire cohabiter plusieurs agents SNMP sur une même machine. Par après nous allons exposer l'utilisation de traps SNMP.

a) Le SNMP Agent Toolkit (SAT).

Une boîte à outils est disponible pour développer des nouveaux agents SNMP. Elle est basée sur l'*Agent Toolkit* développé par le MIT. SAT est décrit dans le document [SAT,1993].

b) Le SNMP Dispatcher.

Le but du *SNMP Dispatcher* est de permettre à plusieurs agents SNMP indépendants de coexister sur une même plate-forme, c'est-à-dire sous une même adresse IP. Il peut être utilisé quand il est nécessaire d'intégrer un agent *ISM* et des agents existants non-*ISM* sur une même machine. Le résultat est un seul manager composite qui comprend un dispatcher et plusieurs sous-agents.

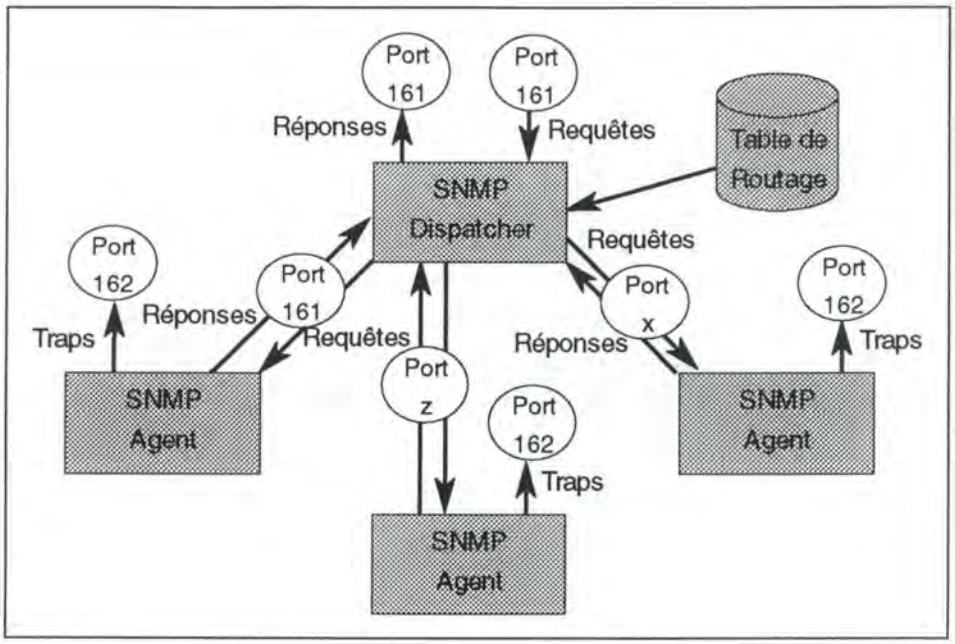

Figure 27 : L'architecture du SNMP Dispatcher.

L'architecture du dispatcher est montrée sur la Figure 27. Il reçoit les requêtes SNMP (GET, GET-NEXT, SET), les passe à l'agent SNMP approprié et traite sa réponse. Les traps sont envoyées de la façon habituelle (par le port 162), sans passer par le dispatcher.

Le dispatcher reçoit les requêtes SNMP sur un port bien défini, généralement le port 161. Les managers SNMP voient donc le *SNMP Dispatcher* comme un agent SNMP standard.

Un des agents SNMP subordonnés peut utiliser le port standard 161. Les autres agents doivent permettre que leurs numéros de port soient changés par une option de configuration. Les agents sont sur écoute sur des ports connus uniquement par le SNMP dispatcher. Le dispatcher utilise une table de routage pour faire passer les requêtes SNMP à l'agent approprié. La table de routage est construite, à partir d'un fichier de configuration, lors du lancement du dispatcher.

L'agent répond à la requête sur le même port où il a reçu la requête. Le dispatcher renvoie alors cette réponse au manager, en tenant compte éventuellement de plusieurs réponses d'agents différents.

c) L'utilisation de traps SNMP.

L'AI SNMP permet que de l'information pour *ISM Alarm* est véhiculée par une trap SNMP. L'AI génère une notification *ISM Alarm* quand il reçoit une trap SNMP. En particulier il attribue l'alarme à un objet dans la *MIB ISM*, c'est-à-dire il génère la MOC et le DN ("*Distinguished Name*") de l'objet *ISM* à partir de l'information contenue dans la trap et la met dans les champs *classe* et *instance d'objet* de la notification d'événement *ISM*.

4.7.2. Les autres agents.

D'autres agents peuvent être implémentés tels que: des agents DSAC/AEP, des agents CMIP, des agents GMP, des agents SNA, des agents SMT (FDDI), etc.

4.8. Conclusion.

Après la description d'ISM de Bull, nous discuterons quelles bénéfices cette plate-forme de gestion intégrée peut apporter aux gestionnaires d'un système distribué.

- *ISM* fournit une interface homme-machine cohérente et homogène. A l'aide du langage SML les développeurs peuvent maintenir la cohérence lors de la conception de nouvelles application de gestion. De cette manière le temps d'apprentissage de nouvelles applications et le temps de réaction face à des problèmes peuvent être réduits.
- Contrairement à une multitude d'outils isolés, *ISM* permet d'avoir une vue d'ensemble du système distribué. Ainsi les relations entre les différentes composantes et le système global deviennent visibles et facilitent ainsi le contrôle. Cette vue d'ensemble est rendue possible grâce à un repository unique: la MIB *ISM*. Une description unique a aussi l'avantage de pouvoir éliminer d'éventuelles redondances.
- Une autre caractéristique importante d'ISM est que la plate-forme est extensible à de nouveaux protocoles de gestion. De cette façon *ISM* reste ouvert pour des développements futurs. En outre les utilisateurs peuvent développer des agents et intégrer leur propre partie de la MIB dans ISM. Nous allons décrire le cas de l'intégration d'un agent de mesures de performances dans le chapitre suivant.

Chapitre 5

Etude de cas:

Les performances sous ISM

5. Etude de Cas: Les Performances sous ISM.

5.1. Introduction.

A la suite de la présentation de la plate-forme de gestion *ISM* de Bull, nous allons exposer dans ce chapitre deux exemples concrets de gestion de performances sous *ISM*. D'abord nous présenterons *Mesurix*, l'agent de mesures de performances que j'ai développé lors de mon stage. Par après nous décrirons le *Performance Monitoring Service* (PMS), qui s'occupe de la gestion des performances sous *ISM*. Nous finirons par une comparaison des deux implémentations.

5.2. Mesurix: Agent SNMP de mesures de performances.

5.2.1. Introduction.

Lors de mon stage dans le centre de recherche Bull des Clayes (département des Yvelines), j'ai pu travailler dans le domaine de la gestion des systèmes distribués. Concrètement j'ai découvert la plate-forme de gestion *ISM* et développé un agent SNMP de mesures de performances.

Cet agent SNMP[72], qui a été baptisé "*Mesurix*", permet de faire du polling sur un objet SNMP. Si la valeur de cet objet dépasse un seuil prédéfini, une notification[73] est envoyée. En outre l'agent offre un service de journal (log) sur fichier. Une session de mesures de performances, qui est caractérisée par les attributs suivants: l'objet à poller, le début/la fin du polling, etc., peut être entièrement paramétrée par l'utilisateur. Le comportement de l'agent est donc passif dans le sens où il se limite à interroger des objets, à envoyer des traps et à écrire dans un fichier journal.

[72] Le protocole SNMP a été présenté à la section *2.6. La gestion dans le monde IAB.*

[73] Une notification est appelée TRAP dans le domaine SNMP.

Mesurix ne fait donc aucun traitement statistique des données et ne déclenche aucune des actions suivants les valeurs de l'objet monitoré.

Pourquoi avoir développé un agent de mesures de performances, puisqu'il existe déjà un service (PMS) sous *ISM* qui effectue cette même tâche? L'agent SNMP Mesurix peut être intéressant dans le cas des:

- **petits systèmes distribués**. Pour ces systèmes une implémentation complète *d'ISM* est souvent trop consommateur en ressources. Il devrait exister une version plus légère *d'ISM* où certaines fonctions moins importantes seraient absentes ou remplacées. Dans le cadre d'une telle solution *Mesurix* pourrait couvrir partiellement les tâches de *PMS* (dans le domaine SNMP) tout en étant moins consommateur en ressources.
- **grands systèmes distribués**. Dans ces systèmes *ISM* est fort chargé par la gestion. Un agent comme *Mesurix* permet, comme nous allons le voir, de faire grâce au mécanisme des seuils une gestion par "délégation". En effet le manager/*ISM* n'a pas besoin de faire du polling de ressources. L'agent se charge du polling et quand un seuil critique est atteint il envoie une notification au manager/ISM. De cette manière le manager/ISM est déchargé puisque le polling est consommateur en ressources.

5.2.2. L'environnement de développement.

L'agent a été développé sur un *DPX/20*[74] par le biais d'un terminal X. Comme système d'exploitation nous avons utilisé une version UNIX[75] spécialement conçue pour des machines RISC.

Le langage de programmation C conjointement avec une boîte à outils nous a servi pour le développement de l'agent. Cette boîte à outils[76] facilite le développement d'agents SNMP. Le *Snmp Agent Toolkit* comprend des librairies de fonctions C qui s'occupent de l'implémentation des opérations SNMP. Il existe en outre des supports à la compilation (scripts, makefiles), au debugging et aux tests des fonctionnalités de l'agent. Les outils de test qui permettent de simuler les requêtes SNMP d'un manager, m'ont permis d'effectuer des essais sur l'agent.

Le code C de l'agent développé est propriété de la société Bull et ne pourra donc pas être publié dans le présent mémoire.

---

[74] Le *DPX/20* est une machine Bull équivalente à un *IBM Risc System 6000*.

[75] The *Base Operating System AIX* (*BOSX*) version 3.2

[76] La boîte à outils SAT (*Snmp Agent Toolkit*) est décrite dans le document [SAT,1993].

5.2.3. L'intégration dans ISM.

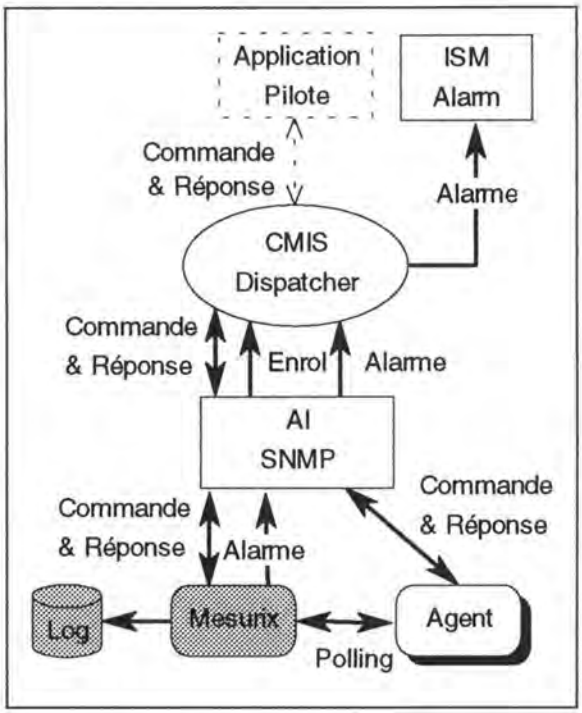

Figure 28 : L'agent Mesurix dans ISM.

Comme nous pouvons le constater sur la Figure 28, l'agent SNMP *Mesurix* communique à travers *l'AI SNMP* et le *CMIS Dispatcher* avec les autres parties d'*ISM*. Le polling des agents par *Mesurix* se fait sans passer par l'*AI SNMP*. Pour que *Mesurix* puisse communiquer directement avec les agents, il faut qu'il connaisse leur adresse. Ces adresses doivent être fixées lors de la configuration, avant le lancement de *Mesurix*. Ceci est contraire au modèle manager-agent où les agents ne communiquent jamais entre eux. Les traps générées par l'agent *Mesurix* peuvent être visualisées comme des alarmes dans *ISM Alarm*. Le log est écrit dans un fichier ASCII, où chaque ligne comprend des informations sur l'objet loggé à un instant donné.

Nous pouvons envisager une application pilote, écrite en SML, qui devrait permettre de contrôler l'agent. Ce contrôle comprendrait par exemple la définition de nouvelles sessions, l'affichage des sessions actives, etc.

5.2.4. La partie de la MIB gérée par Mesurix.

La *MIB Mesurix* est composée de trois catégories d'informations. Nous avons les informations qui concernent une session de mesures de performances en général, la définition des seuils et du fichier journal correspondants (voir Figure 29). Nous allons décrire, dans ce qui va suivre,

le comportement de l'agent sur base de ces informations contenues dans la MIB.

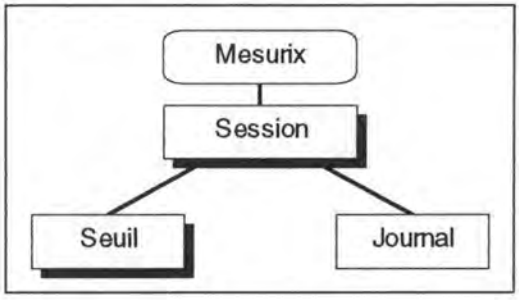

Figure 29 : La MIBlet Mesurix.

Une **session** est identifiée par un identifiant de session. Ce dernier sert aussi comme index dans la table des sessions. Les trois attributs suivants contiennent l'objet "cible", qui va être interrogé par l'agent, sous la syntaxe ISO (MOC et MOI) et sous la syntaxe SNMP (OID). Les deux premiers y figurent uniquement à titre "informatif", pour garnir une trap par exemple. Seuls les objets SNMP ayant une valeur positive sont permis. Un exemple d'un objet "cible" est la ***ifOutQLen***[77] qui représente la longueur de la file d'attente, exprimée en nombre de paquets, pour les paquets envoyés par l'interface réseau. Pour pouvoir utiliser des indicateurs de performances, tels que nous en avons décrits à la sous-section *3.2.2,* il faut qu'ils soient modélisés dans une MIB. Une session est caractérisée en outre par le temps de début et de fin. La fréquence des interrogations est donnée en secondes. Les trois états opérationnels suivants servent à caractériser une session:

- DISABLED (1) : la session est désactivée, des mises-à-jour sur les objets (ou attributs) de la session ne sont possibles que dans cet état.
- ENABLED (2) : la session peut être activée, le seul objet susceptible d'être changé est l'état opérationnel.
- BUSY (3) : la session est activée, c'est-à-dire le monitoring de l'objet cible est enclenché. Le seul objet qu'on peut modifier est l'état opérationnel.

Les transitions possibles entre les états opérationnels sont:

- (1) → (2) : L'utilisateur déverrouille une session de mesures de performances. La session est prête pour commencer les mesures.
- (2) → (1) : L'utilisateur verrouille une session. Maintenant des modifications peuvent être faites sur les paramètres de la session.

[77] Cet objet SNMP est issu de la MIB-II [RFC1213].

- (2) → (3) : Dès que l'utilisateur a rendu la session actionnable, l'agent l'active à partir du moment où le début de l'intervalle de temps, prévu pour les mesures, est atteint.
- (3) → (1) : Cette transition est réalisée, soit par l'utilisateur quand il veut désactiver une session, soit par l'agent quand les mesures sont finies. Cette dernière condition est remplie si le temps actuel est supérieur au temps de fin de mesures ou si l'objet à interroger n'existe pas ou si sa valeur n'est pas entière positive. Toutefois avant d'arrêter complètement la session, l'agent attend que toutes les réponses aux requêtes soient arrivées. Si après un temps, fixé arbitrairement, la réponse n'est toujours pas arrivée (parce qu'un paquet a été perdu par exemple), la session est quand même arrêtée.
- (3) → (2) Impossible.
- (1) → (3) Impossible.

L'agent Mesurix peut envoyer des traps suivant certaines conditions. Celles-ci sont matérialisées par la comparaison de la valeur obtenue avec un **seuil** prédéfini. Si par exemple, la valeur courante est supérieure ou égale à un seuil (appelé "*Rising Threshold*") et la valeur précédente était inférieure ou inexistante alors une trap est envoyée. Si par contre, la valeur courante est inférieure ou égale à un seuil (appelé "*Falling Threshold*") et la valeur précédente était supérieure ou inexistante alors une trap est envoyée. Ce principe est illustré sur la Figure 30.

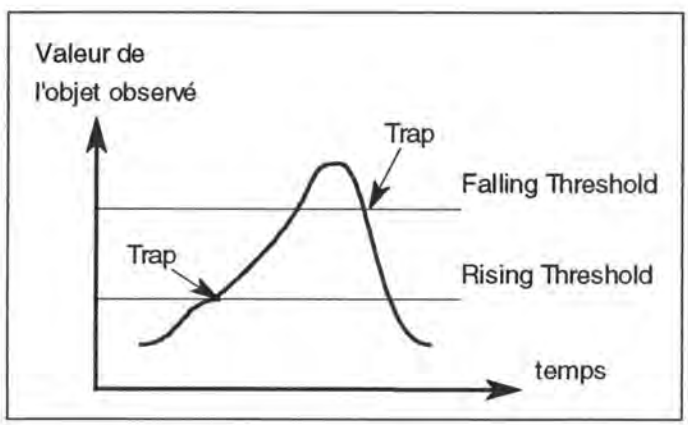

Figure 30 : Les seuils Rising et Falling.

Une troisième condition qui peut générer une trap, est le test d'égalité avec une valeur prédéfinie si toutefois la valeur précédente était différente de cette même valeur. Dans le cas inverse une trap de type "*clear*" est envoyée. La Figure 31 montre un exemple de seuil d'égalité.

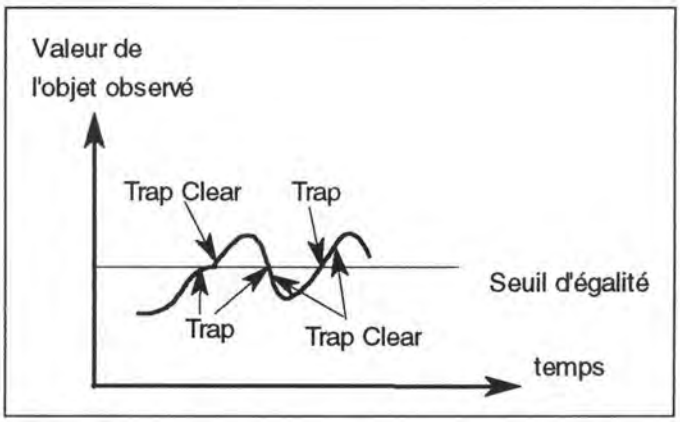

Figure 31 : Le seuil d'égalité.

A chacune des valeurs seuil ainsi définies peut être associée une "sévérité". La sévérité indique le type de la trap, tels que: indéterminé, alarme critique, alarme mineure, alarme majeure, avertissement (warning), libération d'une alarme (clear). Ces sévérités sont reconnues par *ISM Alarm* et *ISM Alarm Log Service*.

Pour éviter une charge inutile du réseau quand il y a des oscillations autour d'un seuil d'égalité, un temps de rémanence pour l'envoi de traps a été implémenté. En effet, une trap d'atteinte du seuil d'égalité n'est générée que si la dernière trap de ce type, pour une même valeur, a été envoyée dans un temps supérieur au temps de rémanence.

Une trap Mesurix contient les informations suivantes:

- un texte libre décrivant le problème spécifique à la trap
- un texte libre spécifique à une session
- l'objet qui est à l'origine de la trap (la MOC, la MOI et l'OID de l'objet SNMP)
- la valeur seuil dépassée
- la sévérité correspondante
- la valeur courante, celle qui a dépassé/atteint le seuil

Le **journal** (ou fichier log) peut être activé ou désactivé. Quand il est activé, il permet de garder une trace d'une session de mesures sur fichier. Une ligne dans ce fichier contient l'OID de l'objet monitoré, sa valeur actuelle, son type et le temps actuel. Ces données, se trouvant sous le format ASCII, peuvent être utilisées ultérieurement pour faire un traitement statistique, pour élaborer des rapports, pour analyser des situations critiques, etc.

5.2.5. L'utilisation des services offerts.

L'agent Mesurix peut être utilisé pour des usages multiples. Un exemple d'utilisation est la gestion quotidienne des imprimantes. Quand un certain nombre de pages a été imprimé, une trap avertit l'opérateur qu'il doit remplacer la cartouche d'encre.

Dans le domaine de la gestion des performances, nous pouvons citer la surveillance du taux d'utilisation d'une ressource, du nombre de demandes pour une ressource, du nombre de demandes d'une ressource rejetées, des indicateurs présentés à la sous-section 3.2.2, etc. Des journaux pris dans le courant d'une journée, sont par exemple une indication pour la répartition de la charge. Ils peuvent servir aussi pour fixer des valeurs seuils. La comparaison de journaux de sessions, sur un même objet à moyen ou à long terme permet de montrer l'évolution du niveau de performances.

5.3. Le Performance Monitoring Service (PMS).

5.3.1. Introduction.

Dans la présente section, nous allons décrire le *Performance Monitoring Service (PMS)* d'*ISM* et les composantes annexes. La présentation de ce service est basée sur les documents [PMS,1993] et [PMS,1994].

5.3.2. L'intégration dans ISM.

Comme nous le voyons sur la Figure 32, PMS est un service accessible à travers *le CMIS Dispatcher*. PMS est un gestionnaire d'objets responsable de sa propre MIBlet, contenant les objets des classes PMS (session, seuils, requêtes). Ces objets peuvent être créés, activés et consultés à travers des requêtes *CMIS* (M-GET, M-CREATE et M-SET). Le service est responsable de sessions de "polling" sur des objets *SNMP*, *DSAC/AEP* et *CMIS* via les AIs.

Il existe une application, appelée *Performance Monitoring Service-Control*, qui permet de contrôler ce service. L'interaction entre les différentes parties est décrite plus en détails dans la sous-section suivante.

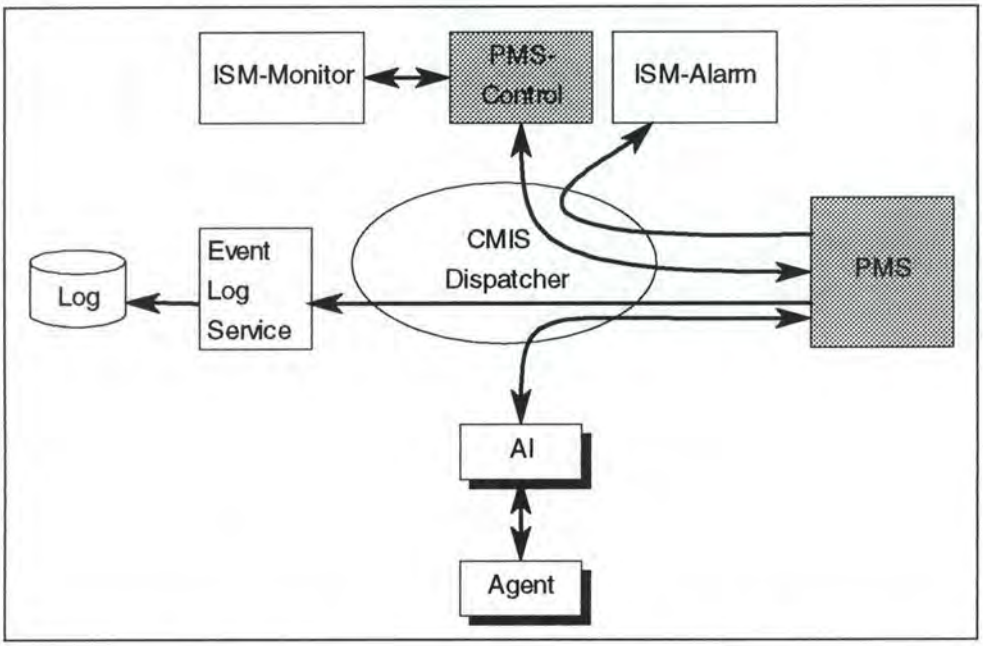

Figure 32 : L'intégration de PMS dans ISM.

5.3.3. Les fonctionnalités offertes.

PMS implémente les fonctionnalités suivantes:

- la génération automatique d'alarmes.

Les résultats des requêtes, envoyées périodiquement par PMS, sont comparés aux seuils définis sur les valeurs d'attributs. A chaque dépassement de seuil, une alarme est envoyée, accessible par *ISM Alarm*.

- la récolte de statistiques off-line.

Les résultats des requêtes peuvent être loggés dans des journaux (logs), gérés par *l'Event Log Service*. Après extraction, les données contenues dans ces journaux peuvent être traitées par des applications spécifiques comme par exemple des tableurs.

- l'aide à la récolte de données de performances on-line.

Les statistiques on-line sous *ISM*, sont visualisées par des panneaux[78] dans *ISM Monitor*. Il est possible de visualiser l'évolution des valeurs d'un attribut dans un panneau. L'utilisation de *PMS* permet aussi d'étendre cette fonctionnalité aux attributs "composites", c'est-à-dire calculés automatiquement à partir d'autres attributs.

[78] voir le paragraphe 4.4.3.a

5.3.4. La partie de la MIB gérée par PMS.

Les différentes possibilités offertes pour la définition des sessions peuvent être appréhendées à travers la MIBlet gérée par PMS. Nous représentons ici schématiquement sur la Figure 33, l'arbre de nommage de cette MIBlet:

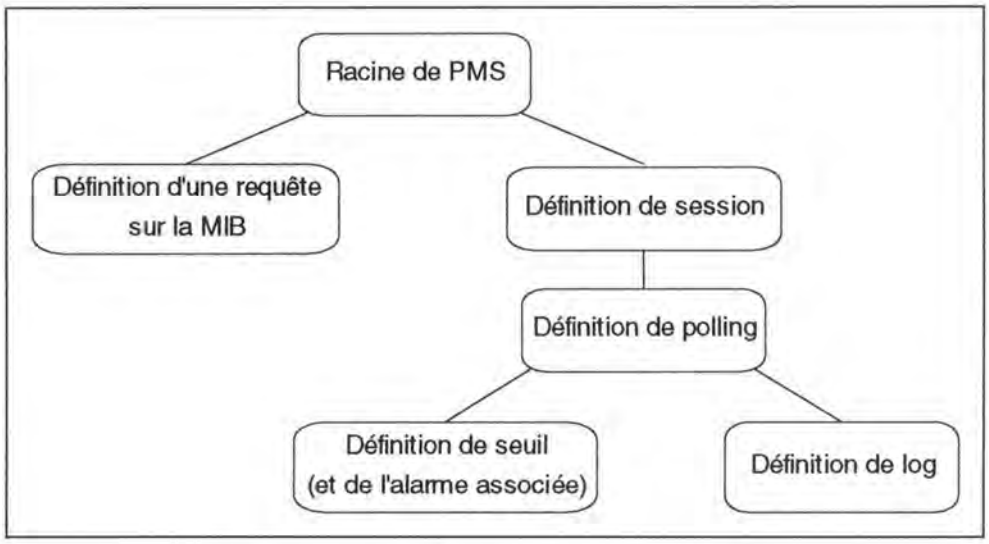

Figure 33 : La Miblet gérée par PMS.

La racine de la MIBlet est représentée par un objet *Performance (PMS)* unique. Une session, matérialisée par l'objet *Session*, peut contenir plusieurs définitions de polling. Ces définitions comprennent les caractéristiques des pollings de la session: état (actif/non actif), durée, heure de démarrage, etc.

Une définition de polling est formée par une association d'un objet *Requête CMIS,* avec un objet de départ pour cette requête. Cette définition comprend en outre la périodicité du polling. Pour un même polling, on peut définir une ou plusieurs définitions de seuils, portant chacune sur un attribut des objets retournés par la requête.

Une requête générique peut être construite à l'aide du *CMIS Query Builder.* I 'ensemble des parties d'une requête peut être stocké dans un objet géré par PMS. Une même requête générique peut être utilisée pour des pollings différents, appliquée chaque fois à des objets de départ différents.

5.3.5. L'application PMS-Control.

L'application *PMS-Control* permet de visualiser les caractéristiques des objets manipulés par le service PMS. Elle marche en coopération avec *ISM Monitor*, notamment pour la spécification des objets MIB sur lesquels

portent les requêtes. *PMS-Control* permet d'effectuer les opérations suivantes:

- visualiser la liste des sessions définies et afficher des résumés de sessions,
- gérer une session (créer, démarrer, arrêter et effacer),
- créer, modifier et dupliquer les définitions de polling d'une session,
- visualiser le traitement (loggé, notifié et/ou surveillé par rapport à des seuils) effectué sur chaque attribut, pour le polling sélectionné,
- saisir les caractéristiques des seuils (les valeurs de seuil, les propriétés de l'alarme associée, la liste d'attributs composites éventuels)

5.3.6. La définition d'une session.

Nous allons parcourir dans cette section les différentes étapes de définition d'une session. Dans une ***première*** étape, un objet *Requête générique* est construit. Cette construction se fait à l'aide du *CMIS Query Builder*, qui permet de définir graphiquement des requêtes, par navigation dans l'arbre de nommage. Pour que la requête soit utilisable par PMS, elle doit être sauvegardée dans *CMIS-DB*, dans un objet subordonné à l'objet PMS.

Ensuite nous devons créer un nouvel objet *Session*, dont les attributs vont être garnis avec les caractéristiques de la session que nous voulons définir. Après la création, les attributs ne sont pas encore définis et l'objet *Session* est dans l'état *inactif*.

La ***troisième*** étape consiste à définir les caractéristiques du polling. L'objet à définir, fait référence à une requête CMIS générique définie autre part. Les attributs de cet objet identifient cette requête générique et l'objet de base de cette requête, c'est-à-dire un objet "sommet" auquel tous les objets qu'on veut interroger seront subordonnés. Une même session peut comprendre plusieurs pollings avec des requêtes distinctes ou non.

Finalement nous pouvons éditer des seuils et/ou définir des logs sur les définitions de polling introduites précédemment. Quatre niveaux de "sévérité" peuvent être rattachés à un seuil.

Pour chaque niveau, nous pouvons définir une rémanence, c'est-à-dire le nombre de polls au bout desquels une alarme est envoyée. Nous pouvons utiliser aussi un mécanisme d'hystérésis, qui correspond à un pourcentage par rapport à la valeur du seuil à franchir, à partir duquel une nouvelle alarme est envoyée. Ce mécanisme sert à éviter le cas pathologique où une valeur d'attribut resterait autour de la valeur du seuil et générerait des alarmes inutiles.

5.3.7. Un exemple d'utilisation.

Nous allons montrer comment on peut utiliser PMS pour surveiller un ou plusieurs objets MIB. Avec l'aide de l'application *Query Builder* nous devons définir une requête générique.

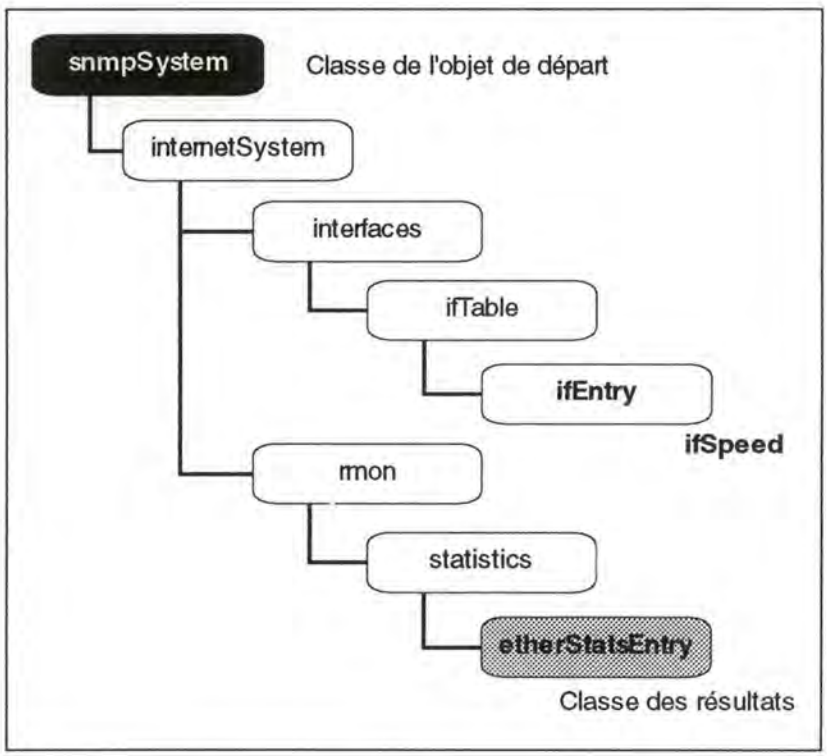

Figure 34 : La requête générique.

Nous prenons l'exemple de la *MIB RMON*[79] pour une sonde *Ethernet*, qui permet d'obtenir des informations sur l'utilisation de la ligne. Sur la classe "etherStatsEntry" nous allons définir la variable "permil_utilization" (Figure 34) comme étant:

*((DELTA (ifInOctets)*8000)/DELTA (vd_time) / v_speed*

qui est le nombre de bits par seconde, divisé par la vitesse de la ligne et transformée en "pour mille". La variable "v_speed" est égale à "ifSpeed" sur la classe "ifEntry" et "vd_time" à "sysUpTime/100" sur la classe "snmpSystem". La requête peut être sauvegardée sous PMS lorsqu'elle est prête.

Avec *PMS-Control* nous pouvons définir un polling dans une session. Sur les attributs retournés par la requête, en l'occurrence la variable "permil_utilization", nous pouvons définr des seuils. Ainsi nous pouvons

[79] Pour plus d'informations sur *RMON*, voir le paragraphe 3.4.2.a.

mettre par exemple un seuil critique haut à 300. Ceci signifie que lorsque l'utilisation de la ligne dépasse 30% une trap critique haut est envoyée. Maintenant il ne reste plus qu'à activer la session et d'animer l'objet sous *ISM-Monitor*.

5.4. Conclusion.

Dans cette section nous allons dans un premier temps comparer l'agent *Mesurix* et le service *PMS*. Ensuite nous allons proposer quelques extensions à l'agent *Mesurix*.

Une différence fondamentale entre les deux est leur nature. En effet *PMS* est un service *ISM*, tandis que *Mesurix* est un agent. De cette différence découle le fait que *PMS* peut supporter plusieurs protocoles de gestion (CMIS, SNMP, DSAC/AEP, etc.), tandis que *Mesurix* n'est opérationnel que dans le sous-système de gestion SNMP.

La performance de *PMS*[80] et de *Mesurix* dépend chez tous les deux du nombre d'objets pollés et de la fréquence des polls. Si la fréquence des pollings est haute, le nombre d'objets observés doit être petit. Les deux dépendent aussi étroitement de l'infrastructure de communication sous-jacente pour les pollings. Le polling effectué par *Mesurix* est plus performant puisqu'il ne doit pas passer par *l'AI SNMP* et le *CMIS Dispatcher* pour interroger les agents. Cependant il faut noter que cette façon de faire est moins propre puisque *Mesurix* doit connaître l'adresse des agents qu'il veut poller. Comme *PMS* est écrit en *SML*, il dépend aussi du niveau de performances de l'interpréteur *SML*.

Comme extensions nous pouvons citer une application SML qui piloterait l'agent. Elle devrait permettre aux utilisateurs de définir des sessions de mesures et d'en exploiter les résultats.

Une autre extension fort importante serait l'introduction d'objets composites, qui seront calculés à partir de plusieurs objets SNMP. Avec cette méthode il est possible de construire ses propres indicateurs pour les mesures de perfomances.

[80] voir le chapitre 1.4 dans le document [PMS,1993].

Chapitre 6

Conclusion

6. Conclusion.

Dans ce mémoire nous avons décrit un agent SNMP de mesures de performances pour la gestion des systèmes distribués. Nous avons montré qu'avec le développement d'un agent nous pouvons étendre la plate-forme de gestion. Une prochaine étape serait d'utiliser ce même agent et *ISM* pour étudier un cas réel de mesures de performances. De cette manière nous pourrions analyser d'abord le comportement de l'agent et l'améliorer le cas échéant. Similairement nous gagnerions des informations sur les mesures de performances. En effet nous pourrions examiner les répercussions sur les mesures induites par l'état du système distribué. Si le système est trop chargé par exemple, les mesures seront biaisées.

Ensuite il faudrait, après avoir gagné une certaine maîtrise de l'agent, apprendre à interpréter correctement les résultats des mesures. Nous pourrions utiliser ou développer des outils qui nous aiderons à dans l'analyse des résultats.

A la suite de ce propos, nous constatons qu'après le développement de l'agent, il reste encore beaucoup de travail à accomplir avant que ce dernier ne soit pleinement opérationnel.

Annexes

ANNEXES.

Annexe A.

Table résumant les fonctions disponibles aux applications ISM qui utilisent le protocole SNMP.

Comportement CMIS tel qu'il est vu par les applications	Traduction	Comportement SNMP tel qu'il est vu par l'agent
1. Fonctions:		
M-GET	complète	GET, GET-NEXT
M-SET	partielle	SET
M-CREATE	complète	SET (tables uniquement)
M-DELETE	complète	SET (tables uniquement)
M-ACTION	aucune[1]	(SET)
M-EVENT	partielle	TRAP
2. Paramètres:		
Etendue (Scoping)	partielle	GET-NEXT (tables)
Filtrage	partielle	fait par SNMP AI
Contrôle d'accès	aucune	
Synchronisation	partielle	
Confirmation	partielle	pas pour les TRAPs
3. MIB:		
Organisation hiérarchique	partielle	nombre limité de niveaux
Attributs multi-valués	aucune	attributs mono-valués
Grammaire ASN.1 complète	partielle	sous-ensemble d'ASN.1

[1] Certaines M-ACTION sont admises par les SNMP AI, mais elles ne peuvent pas être traduites directement dans SNMP.

Table des sigles

Table des sigles.

A

ACM	: Association of Computing Machinery
AEP	: Administrative Exchange Protocol (Bull)
AI	: Agent Integrator (Bull/ISM)
API	: Application Programmatic Interface
ASN.1	: Abstract Syntax Notation One

B

BERT	: Bit Error Rate Tester
BOSX	: Base Operating System for AIX

C

CCITT	: Comité Consultatif International du Télégraphe et du Téléphone (voir sous ITU)
CMIP	: Common Management Information Protocol (ISO)
CMIS	: Common Management Information Service (ISO)

D

DN	: Distinguished Name (ISO)
DoD	: Department of Defense (USA)
DSAC	: Distributed Systems Administration and Control

E

ECMA	: European Computer Manufacturers' Association
EFD	: Event Forwarding Discriminator

F

FDDI	: Fiber Distributed Data Interface

G

GDMO	: Guidelines for the Definition of Managed Objects (ISO)
GMP	: Gcos Management Protocol

I

IAB	: Internet Activities Board
IHM	: Interface Homme-Machine
IOPS	: Input/Output OPerations per Second
IP	: Internet Protocol
ISM	: Integrated Systems Management (Bull)
ISO	: International Standardization Organization
ITU	: International Telecommunication Union (ancien CCITT)

L

LAN	: Local Area Network

M

MIB	: Management Information Base
MIPS	: Million Instructions Per Second
MIT	: Massachusetts Institute of Technology
MOC	: Managed Object Class
MOI	: Managed Object Instance
MTBF	: Mean Time Between Failure
MTTR	: Mean Time To Repair

N

NFS	: Network File System

O

OID	: Object IDentifier
OS	: Operating System
OSI	: Open Systems Interconnection (ISO)

P

PMS : Performance Monitoring Service (Bull/ISM)

R

RDN : Relatif Distinguished Name (ISO)

RISC : Reduced Instruction Set Computer

RFC : Request For Comments (IAB)

RMON : Remote MONitoring (IAB)

ROM : Root Object Manager (Bull/ISM)

S

SAT : Snmp Agent Toolkit (Bull/ISM)

SIGMETRICS: Special Interest Group on Measurement and Evaluation (ACM)

SMI : Supra Manager Interfaces (Bull/ISM)

SML : System Management Language (Bull/ISM)

SNA : Systems Network Architecture (IBM)

T

TCP : Transmission Control Protocol

U

UDP : User Datagram Protocol

Références bibliographiques

Références bibliographiques.

[Ben-Artzi,1990]
BEN-ARTZI A., CHADNA A., WARRIER U.
"Network Management of TCP/IP Networks : Present and Future"
in: IEEE Network Magazine, 7/1990, pages 35-43

[Bodnarchuk,1991]
BODNARCHUK Roberta, BUNT Richard
"A Synthetic Workload Model for Distributed System File Server"
in: Proc. of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, San Diego 1991

[Comer,1991]
COMER Douglas
"Internetworking with TCP/IP : Vol. 1 : Principles, Protocols and Architecture (2nd edition)"
Prentice Hall, 1991

[Coulouris,1988]
COULOURIS George, DOLLIMORE Jean
"Distributed Systems, Concepts and Design"
Addison-Wesley, 1988

[Encyclopedia,1978]
RALSTON Anthony (editor)
"Encyclopedia of Computer Science"
Van Nostrand Reinhold Company, 1978

[Ferrari,1986]
FERRARI Domenico
"Considerations on the Insularity of Performance Evaluation"
in: IEEE Transactions on Software Engineering, vol 12, no 6, 6/1986

[Franta]
FRANTA W.R.
"The Process View Of Simulation."
North Holland, 19??

[Gering,1993]
GERING Michael
"CMIP versus SNMP"
in: Proc. of Integrated Network Management III, Editeurs: Hegering H-G & Yemini Y., Elsevier Science Publishers (North Holland), 1993

[GHLDD,1994]
EMSLEY Ian
"ISM3 : Global High Level Design Document"
Document interne Bull, draft du 17.06.94

[Goldszmidt,1993]
GOLDSZMIDT G., YEMINI Y.
"Evaluating Management Decisions via Delegation"
in: Proc. of Integrated Network Management III, Editeurs: Hegering H-G & Yemini Y., Elsevier Science Publishers (North Holland), 1993

[Haban,1990]
HABAN Dieter, WYBRANIETZ Dieter
"A Hybrid Monitor for Behaviour and Performance Analysis of Distributed Systems"
in: IEEE Transactions on Software Engineering, vol 16, no 2, 2/1990

[Hac,1992]
HAC Anna
"Modeling Distributed File Systems"
in: Performance Evaluation Review Vol. 19(4), 5/1992, pages 22-27

[Hawking,1989]
HAWKING Steven
"Une brève histoire du temps. Du big bang aux trous noirs."
Flammarion, 1989

[Hegering,1993]
HEGERING Heinz-Gerd, YEMINI Yechiam
"Integrated Network Management III (Proceedings)"
North-Holland, 1993

[Hegering,1994]
HEGERING Heinz-Gerd, ABECK Sebastian
"Integrated Network and System Management"
Addison-Wesley, 1994

[Hoffner, 1993]
HOFFNER Y.
"The management of monitoring in object-based distributed systems"
in: Proc. of Integrated Network Management III, Editeurs: Hegering H-G & Yemini Y., Elsevier Science Publishers (North Holland), 1993

[Hofmann,1994]
HOFMANN R., KLAR R., MOHR B., QUICK A., SIEGLE M.
"Distributed Performance Monitoring: Methods, Tools, and Applications"
in: IEEE Transactions on Parallel and Distributed Systems, vol 5, no 6, 6/1994

[ISM,1994
ISM
"Integrated System Management - Overview" (ISM version 3)
Bull, 9/1994

[ISM,1995]
ISM
"Integrated Systems Management"
serveur WWW: http://www.bull.com/Prod/ISM/ , Bull 1995

[Leinwand,1992]
LEINWAND Allan
"Accomplishing Performance Management with SNMP"
in: The Simple Times : The bi-monthly Newsletter of SNMP Technology, Vol. 1 (5), 12/1992

[Molloy,1992]
MOLLOY Michael
"Anatomy of NHFSSTONES Benchmarks"
in: Performance Evaluation Review Vol. 19 (4), 5/1992, pages 28-39

[Neumair,1993]
NEUMAIR B.
"Modeling Resources for Integrated Performance Management"
in: Proc. of Integrated Network Management III, Editeurs: Hegering H-G & Yemini Y., Elsevier Science Publishers (North Holland), 1993

[PMS,1993]
DOUMMAR Raphaël
"ISM-Performance Monitoring Service. External Interface Specification"
Bull/ISM, 1993

[PMS,1994]
DOUMMAR Raphaël, TESSERON Jean-Luc
"ISM-Performance Monitoring Service. Guide d'utilisation."
Bull/ISM, 1994

[Rose,1991]
ROSE Marshall T.
"The Simple Book : An Introduction to Management of TCP/IP-based Internets."
Prentice Hall, 1991

[Rose,1994]
ROSE Marshall T.
"The Simple Book : An Introduction to Internet Management (2nd edition)."
Prentice Hall, 1994

[SAT,1993]
Bull/ISM
"Snmp Agent Toolkit (SAT). Programmers Guide (UNIX)"
Bull, 1993

[Sauer]
SAUER Charles, CHANDY Mani
"Computer Systems Performance Modeling"
Prentice Hall, 19??

[Seitz,1994]
SEITZ Jochen
"Netzwerkmanagement"
International Thomson Publishing GmbH, 1994

[Shneiderman,1984]
SHNEIDERMAN Ben
"Response Time and Display Rate in Human Performance with Computers"
in: ACM Computing Surveys, Vol 16 (3), 9/1984

[Stallings,1993]
STALLINGS William
"SNMP, SNMPv2 and CMIP. The Practical Guide to Network Management Standards"
Addison-Wesley, 1993

[Tanenbaum,1991]
TANENBAUM Andrew
"Réseaux : Architectures, protocoles, applications (2ième tirage)"
InterEditions, 1991

[Tarouco,1989]
TAROUCO Liane
"Intelligent Network Management"
in: Proc. of Integrated Network Management I, Editeurs: Meandzija & Westcott, Elsevier Science Publishers (North Holland), 1989

[Umar,1993]
UMAR Amjad
"Distributed Computing. A Practical Synthesis."
Prentice Hall, 1993

[Vincent,1994]
VINCENT Stefan
"La Gestion et l'Evolution des Systèmes Distribués" (mémoire)
Facultés Universitaires Notre-Dame de la Paix, Institut d'Informatique, 1994

[Waldbusser,1992]
WALBUSSER Steven, NAIR Mohan, HOERTH Mark
"SNMP Management Goes Down To The Wire"
in: Data Communications, 5/1992, pages 111-116

[Waldbusser,1992b]
WALDBUSSER Steven
"How RMON stands to replace the traditional protocol analyser"
in: The Simple Times : The bi-monthly Newsletter of SNMP Technology, Vol. 1 (5), 12/1992

Normes ISO

[ISO 7498]
ISO 7498
OSI Basic Reference Model. Add. 1: Connectionless Model, Add. 2: Security Architecture, Add. 3: Naming and Addressing, Add. 4: Management Framework

[ISO 9595]
ISO 9595
Common Manangement Information Service (CMIS)

[ISO 9596]
ISO 9596
Common Managemen Information Protocol (CMIP)

[ISO 10040]
ISO 10040
OSI Systems Management Overview

[ISO 10164]
ISO 10164
OSI Systems Management

[ISO 10165]
ISO 10165
OSI Structure of Management Information

RFC

[RFC 1155]
ROSE M.T., McCLOGHRIE K.
Structure and identification of management information for TCP/IP-based internets
Request For Comments 1155
IAB, 1990

[RFC 1156]
ROSE M.T., McCLOGHRIE K.
Management Information Base for network management of TCP/IP-based internets
Request For Comments 1156
IAB, 1990

[RFC 1157]
CASE J.D., FEDOR M., SCHOFFSTALL M.L., DAVIN C.
Simple Network Management Protocol (SNMP)
Request For Comments 1157
IAB, 1990

[RFC 1213]
ROSE M.T., McCLOGHRIE K.
Management Information Base for network management of TCP/IP-based internets: MIB-II
Request For Comments 1213
IAB, 1991

[RFC 1215]
ROSE M.T.
Convention for Defining Traps for Use with the SNMP
Request For Comments 1215
IAB, 1991

[RFC 1271]
WALDBUSSER S.
Remote Network Monitoring Management Information Base
Request For Comments 1271
IAB, 1991