MASTER EN SCIENCES INFORMATIQUES

L'administration des systèmes distribués. Approche de la gestion OSI

Deghorain, Hugues

Award date:
1992

Awarding institution:
Universite de Namur

Facultés Universitaires Notre-Dame de la Paix, Namur
Institut d'Informatique
Année académique 1991-1992

L'administration des systèmes distribués
Approche de la gestion OSI

Hugues DEGHORAIN

Promoteur : Jean Ramaekers

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

Rue Grandgagnage, 21, B - 5000 NAMUR (BELGIQUE)

Résumé

La recherche en matière de systèmes informatiques permettra sans doute un jour aux entreprises de disposer de systèmes pleinement distribués. Pour ce type de systèmes, la dispersion des composants matériels et logiciels est invisible aux utilisateurs. A l'heure actuelle, les systèmes distribués proposés se situent plutôt à un niveau intermédiaire entre les réseaux et les systèmes pleinement distribués. Tout comme pour les réseaux, la part de la gestion dans le fonctionnement des systèmes distribués est importante. Elle assure une qualité de service requise pour jouir pleinement des avantages que ces systèmes procurent.

Dans un premier temps, nous introduirons les notions de réseaux et de systèmes distribués pour établir le cadre de l'adminstration. Après avoir présenté les concepts de l'administration des systèmes distribués, nous décrirons la gestion de systèmes normalisée par l'ISO. Nous terminerons par la présentation d'une solution de gestion proposée par un constructeur : ISM de Bull.

Abstract

Research in computer science will certainly afford the enterprises to use fully distributed systems. For this type of systems, the distribution of hardware and software components is invisible from the users's point of view. At the present time, proposed distributed systems are rather on an intermediate level between networks and fully distributed systems. As for the networks, management takes an important part in the work of distributed systems. It ensures the quality of service required for an entire use of the advantages of such systems.

First, we introduce the notions of networks and distributed systems. It sets up a framework for the management. After the presentation of concepts related to distributed systems management, we describe the systems management standardized by ISO. We finish this work by a description of a vendor's management solution : ISM proposed by Bull.

Remerciements

Je tiens à remercier toutes les personnes qui, de près ou de loin, ont contribué à l'élaboration de ce mémoire.

Tout d'abord, Monsieur Jean Ramaekers, mon promoteur, grâce à qui j'ai pu effectuer mon stage chez Bull.

Ensuite, l'équipe ISMDEV de Bull - Louveciennes qui m'a encadré durant ce stage. Et plus particulièrement Messieurs Philippe Marin, Louis Ricbourg et Denis Attal pour les portes qu'ils m'ont ouvertes, Dang-Koa N'guyen, Alain Brunet, Jean-François Matrin, Serge Lassabe, Philippe Grégoire et tous les autres aussi, pour leur accueil spontané.

J'aimerais également marquer toute ma gratitude à Joël Hubin pour ses conseils et ses encouragements dans le suivi de ce mémoire.

Je ne voudrais pas terminer ces remerciements sans penser à tous ceux qui m'ont aidé durant ces derniers mois, de même qu'à mes parents sans qui rien ne serait arrivé.

Table des matières

Introduction

Introduction

L'information est une des clés dont doivent disposer les entreprises pour atteindre un certain niveau de compétitivité. Dans leur quête pour obtenir toujours plus d'informations, les entreprises ont été amenées à se munir de moyens informatiques. Les évolutions technologiques qu'a connues l'informatique ont permis non seulement de gérer les informations, mais en plus, de disposer d'un moyen pour les échanger. Des recherches, surtout en milieu universitaire, tentent de concevoir de nouveaux systèmes : les systèmes distribués. Leur conception, bien plus que celle d'un simple moyen de communication, permettrait aux utilisateurs d'accéder directement à l'information, comme si tout le système lui était dédié.

Dans le chapitre 1, nous serons amenés à nous familiariser avec les systèmes distribués et leurs concepts. Il s'agira tout d'abord de les situer sur une échelle des systèmes informatiques. Nous aborderons les notions de réseaux dont nous présenterons les caractéristiques et les différents types. Avec la notion de systèmes ouverts, nous aurons l'occasion d'aborder la partie concernant les communications dans les systèmes distribués.

Pour que tous les moyens informatiques mis en œuvre dans les systèmes distribués servent de manière optimale aux entreprises, il est primordial de s'assurer de leur bon fonctionnement. Le chapitre 2 nous permettra d'établir le cadre général de l'administration des systèmes distribués. Nous y insisterons sur la nécessité de disposer d'outils d'administration pour de tels systèmes. Nous donnerons les objectifs de la gestion, ses fonctions ainsi que l'approche à adopter face aux problèmes de complexité des systèmes à gérer.

La communication est un élément de base dans les systèmes distribués. L'ISO - International Standards Organization - propose un modèle de référence à partir duquel on peut bâtir une architecture de communication. Le chapitre 3 se concentrera sur la gestion d'un tel système de communication. Nous aborderons pour cela la gestion OSI - Open Systems Interconnection - normalisée par l'ISO. Nous terminerons ce chapitre par une brève présentation de l'approche de la gestion adoptée par le consortium de constructeurs regroupés sous le nom de NM Forum - Network Management Forum.

Dans le chapitre 4, nous aurons l'occasion de découvrir une solution de gestion proposée par un constructeur; en l'occurrence, ISM de Bull. Nous pourrons ainsi voir

comment peut être une plate-forme d'administration de systèmes distribués se conformant aux recommandations du NM Forum et, par la même occasion, aux normes de l'ISO.

Le chapitre suivant présentera le résultat de notre stage effectué chez Bull dans le département développant ISM. Il s'agira d'une application de gestion de systèmes UNIX permettant de contrôler l'état des disques et les utilisateurs connectés aux différentes machines reliées au sein d'un réseau. Cela nous permettra de reprendre, par un exemple, différentes notions introduites au cours des chapitres précédents.

Pour terminer, dans la conclusion,, nous ferons ressortir les notions importantes en matière de systèmes distribués et de leur administration.

Chapitre 1

Les systèmes distribués

Chapitre 1 : Les systèmes distribués

1.1 Introduction

Les systèmes informatiques ont subi ces dernières années de profonds changements. La volonté des utilisateurs tend à se départir des systèmes centralisés qui les lient à un constructeur particulier. Ils désirent, maintenant avoir plus d'indépendance dans leurs choix informatiques. La réalité leur permet à l'heure actuelle de répondre à cette attente. En effet, les systèmes distribués ou répartis ouvrent une nouvelle conception de la stratégie informatique d'une entreprise. La répartition des ressources au travers d'un réseau permet aux entreprises d'adopter le concept de down-sizing, à la mode ces derniers temps, qui consiste à privilégier plusieurs petits éléments informatiques plutôt qu'un gros système centralisé.

Nous aborderons les différentes étapes qui vont des systèmes centralisés aux systèmes distribués. Avant de présenter les systèmes distribués proprement dits, nous verrons les notions qui caractérisent les réseaux.

1.2 Echelle des systèmes

Un système réparti ou système distribué - que nous appellerons parfois environnement distribué - peut être vu comme le sommet d'une échelle sur laquelle les systèmes informatiques se placent en fonction de leur degré de centralisation. L'échelon le plus bas est occupé par les systèmes centralisés alors que les systèmes distribués prennent place sur la barre la plus élevée. Entre les deux extrêmes, une étape intermédiaire est constituée des réseaux.

Comme le souligne [TANEN89], "sans les logiciels, un ordinateur n'est qu'un morceau de métal unique". Pour cette raison, nous ne nous concentrerons pas uniquement sur les composants physiques des systèmes mais nous aborderons également la facette logicielle dont l'élément fondamental est le système d'exploitation. Celui-ci permet de contrôler les ressources du système tout en fournissant une base sur laquelle les développeurs construiront leurs applications.

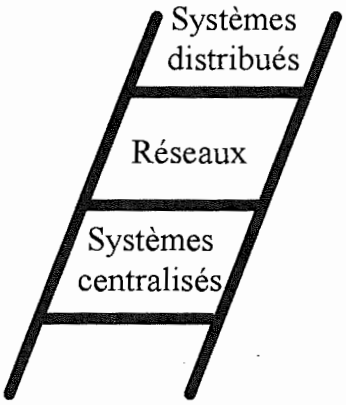


Figure 1.1 : échelle des systèmes

1.2.1 Systèmes centralisés

Le premier échelon est constitué par le niveau où la centralisation de ses différents composants - hardware et software - est la plus élevée. Tout est dans la même machine. Un système d'exploitation unique gère l'ordinateur. Un exemple de système centralisé qui vient en premier à l'esprit est sans nul doute les PC avec un système d'exploitation du style de MS-DOS. D'autres systèmes beaucoup plus gros peuvent être également centralisés même si des utilisateurs répartis géographiquement peuvent y accéder via des terminaux. Le logiciel permettant d'exploiter un ordinateur central est conçu pour cette machine uniquement. C'est le cas par exemple de MULTICS.

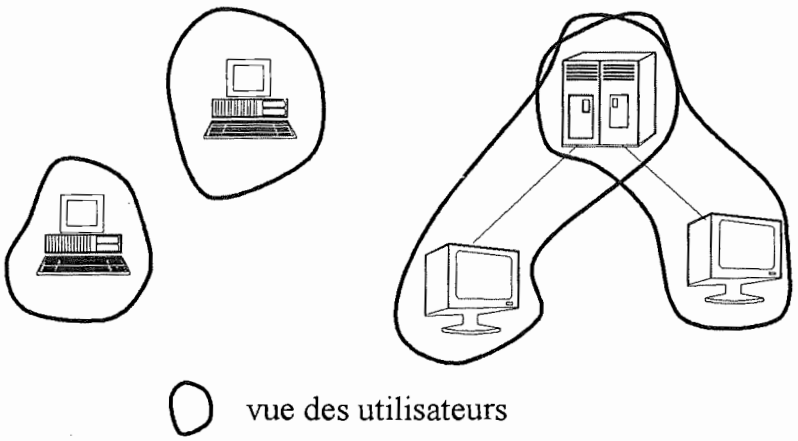


Figure 1.2 : vue des utilisateurs de systèmes centralisés

1.2.2 Réseaux

L'échelon intermédiaire sur l'échelle des systèmes est celui sur lequel se placent les réseaux d'ordinateurs. Typiquement, la configuration d'un réseau est un ensemble d'ordinateurs autonomes - par exemple des ordinateurs personnels - avec des serveurs d'imprimantes et des serveurs de fichiers interconnectés pour échanger des informations au travers d'un réseau local. D'autres types de réseaux que les réseaux locaux peuvent être pris en compte - comme nous le verrons au point 1.3.2.

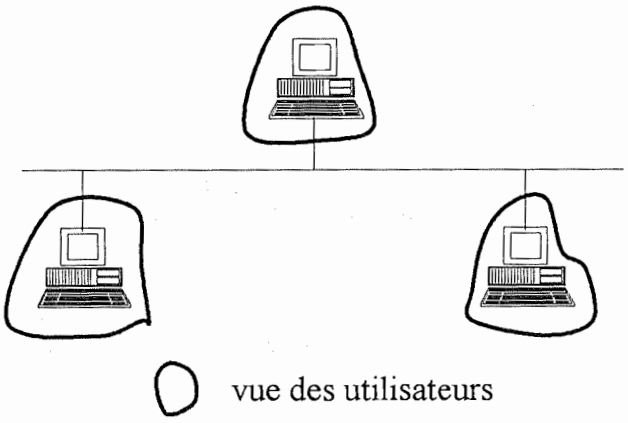


Figure 1.3 : vue des utilisateurs d'un réseau

Etant donné que les machines au sein d'un réseau sont autonomes, elles disposent chacune de leur propre système d'exploitation comme dans le cas des systèmes centralisés. Le système d'exploitation de ces machines comporte cependant un composant supplémentaire pour permettre à ces dernières de communiquer avec d'autres. Il s'agit du système de communications qui par les différents modules qui le composent fournit un chemin d'accès entre les utilisateurs de machines différentes. [BEAUQ90] nous dit que "par chemin d'accès, il faut entendre la suite des fonctions qui rendent possible non seulement la connexion physique de deux utilisateurs terminaux, mais également la communication d'informations, en dépit d'erreurs éventuelles et de différences de formats et de vitesse". Un utilisateur peut très bien être un programme d'application. Ce système de communications permet donc à une machine de s'ouvrir à d'autres. L'ISO, comme nous l'aborderons au point 1.3.3, définit des normes à suivre pour développer un tel système de communication.

1.2.3 Systèmes distribués

Au niveau le plus élevé de l'échelle, nous avons les systèmes distribués. C'est à ce niveau que la distribution des composants du système est la plus élevée. On disperse à la fois le hardware, le software ainsi que le contrôle de ceux-ci. La frontière entre les systèmes distribués et les réseaux d'ordinateurs est très faible. Il est vrai que d'un point de vue matériel, ces deux concepts sont semblables. Tous deux sont constitués de machines autonomes interconnectées. La distinction fondamentale, comme le fait remarquer [TANEN90], concerne la répartition en ordinateurs autonomes. Dans le cas des systèmes distribués, celle-ci est totalement transparente aux utilisateurs. Cette légère différence amène une confusion que l'on retrouve dans de nombreux ouvrages où les termes de systèmes distribués et de réseaux sont employés pour désigner une même idée - voir par exemple [MAEKA87], p.177.

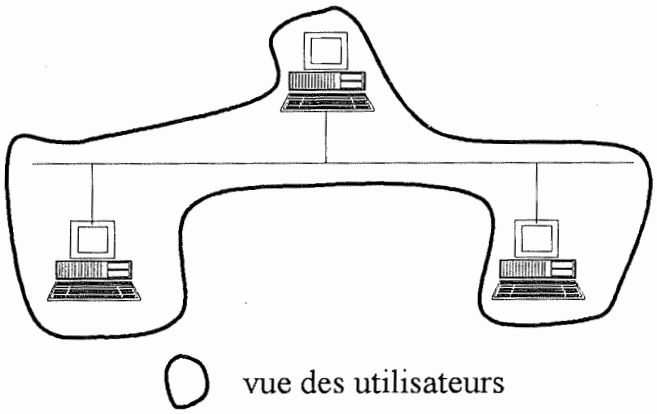


Figure 1.4 : vue des utilisateurs d'un système distribué

Cette transparence consiste à cacher l'utilisation de plusieurs processeurs pour donner l'impression à l'utilisateur du système qu'il travaille sur une seule machine. [TANEN85] parle de machine virtuelle monoprocesseur. Aussi, si nous pouvons dire sur quelle machine nous travaillons, c'est que nous ne sommes pas dans un système distribué.

1.3 Réseaux

Nous venons de voir que la frontière entre les réseaux et les systèmes distribués est très faible. Le réseau sert de base, d'ossature aux systèmes répartis. Nous allons nous pencher maintenant sur les caractéristiques des réseaux, les différents types que l'on rencontre le plus couramment ainsi que des solutions en matière d'interconnexion des machines.

1.3.1 Caractéristiques des réseaux

Une des manières pour caractériser les réseaux est d'en donner les objectifs. Dans la littérature, nombre d'auteurs - [SLOMA87], [WAND84], par exemple - ne font pas de différences entre les objectifs des systèmes distribués et ceux des réseaux. Cela renforce la difficulté à différencier ces deux systèmes informatiques ainsi que la confusion dont nous avons déjà parlé au point 1.2.3.

Les objectifs des réseaux sont le partage des ressources, la fiabilité, la réduction des coûts, l'augmentation de la puissance de calcul et la facilité d'évolution.

- Un des objectifs les plus importants et peut-être celui auquel nous pensons le premier est le ***partage des ressources***. Les réseaux, en effet, permettent à différents utilisateurs, quelle que soit leur situation géographique et au même titre que d'autres utilisateurs, d'accéder à des périphériques comme des imprimantes laser, à des fichiers d'utilisation commune - en matière de réservation de billets d'avion, par exemple. [TANEN80] compare cet objectif à la fin de la "tyrannie de la géographie".

- Il peut arriver que des éléments du réseau soient défectueux. Les services offerts par ces éléments et les données qu'ils contenaient sont alors inutilisables. Mais les réseaux permettent d'assurer une certaine ***fiabilité*** quant à l'utilisation des ressources. En dupliquant les fichiers sur plusieurs machines, par exemple, la disponibilité de ceux-ci, même en cas de panne de l'une des machines, est assurée. Il en est de même au niveau du hardware et des unités centrales. Si l'une tombe en panne, les tâches dont elle était responsable peuvent être prises en charge par une autre.

- Les évolutions technologiques en matière de micro-électronique ont permis d'offrir des ordinateurs à des prix plus petits tout en augmentant les performances. La réduction des coûts est donc un objectif supplémentaire des réseaux d'ordinateurs. Le rapport prix/performance des micro-ordinateurs est meilleur que celui des gros systèmes. Si ces derniers sont dix fois plus rapides que les premiers, leur coût est mille fois plus élevé. Mettre en réseau des machines telles que des ordinateurs personnels - PC - devient dès lors plus avantageux que d'adopter un gros système centralisé. Partager des périphériques onéreux - imprimantes couleurs postscript, par exemple - permet également de réduire les coûts au minimum nécessaire.

- Un autre objectif des réseaux est l'augmentation de la ***puissance de calcul***. On pourrait dire que cet objectif n'est pas spécifique aux réseaux puisqu'un système centralisé multi-processeurs augmente également la puissance comparativement à une machine mono-processeur. Mais des problèmes spécifiques à ce type de système - l'allocation de ressources critiques comme la mémoire centrale, les processeurs - peuvent être contournés dans un réseau d'ordinateurs. Ainsi, afin d'augmenter les performances, les calculs peuvent être répartis entre les différents sites du réseau, selon la charge de ceux-ci.

- Lorsqu'un système d'exploitation a été conçu dès le départ pour tourner dans un réseau d'ordinateurs il est aisé pour lui de prendre en compte l'évolution des éléments du réseau. La ***facilité d'évolution*** constitue un objectif supplémentaire. Pour des raisons d'amélioration de performances, le responsable d'un réseau peut être amené à remplacer certains de ses éléments. Ainsi remplacera-t-il l'ordinateur serveur du courrier électronique dont les performances sont dépassées par rapport aux besoins de l'entreprise. Il en est de même pour le serveur d'impression qui, parce que l'on a ajouté une imprimante doit pouvoir l'utiliser.

1.3.2 Types de réseaux

Deux types de réseaux reviennent souvent dans la littérature. Ce sont les réseaux locaux - LAN -et les réseaux à longue portée - WAN. Un troisième type est apparu voici quelques années : le réseau métropolitain - MAN. Cette classification des réseaux permet d'établir une hiérarchie de l'architecture de réseau. La figure 1.5 en illustre un exemple.

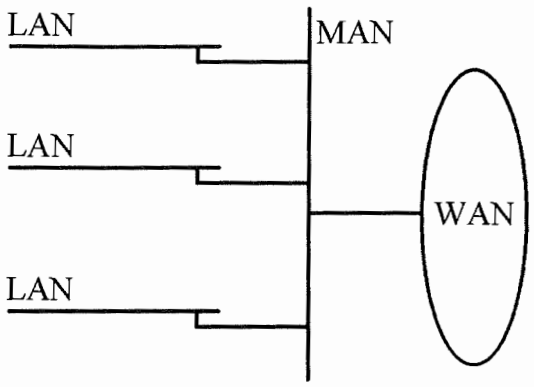


Figure 1.5 : hiérarchie de l'architecture de réseau

1.3.2.1 Réseaux à longue portée

Les réseaux à longue portée, plus communément appelés WAN - Wide Area Network - s'étendent sur une distance qui peut atteindre des milliers de kilomètres. Ce sont donc eux qui sont utilisés à un niveau mondial.

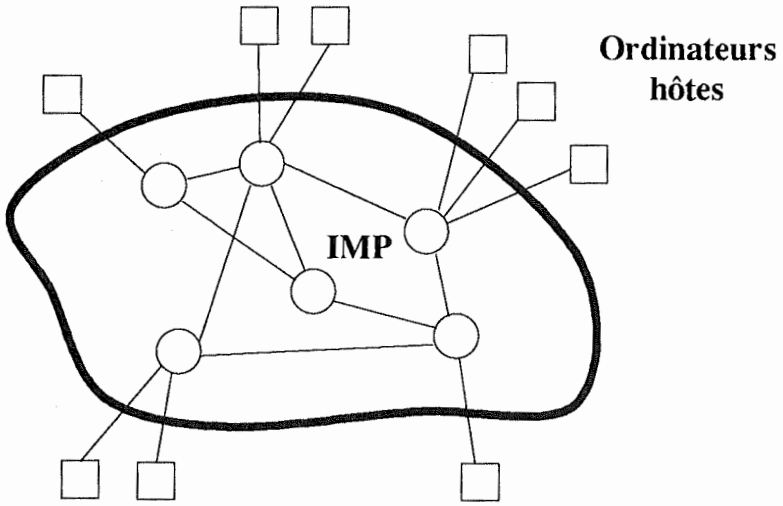


Figure 1.6 : réseau à longue portée

Ils reposent presque tous, sur un système de transmission "store-and-forward" appelé aussi commutation. Un tel réseau est constitué d'un ensemble de nœuds interconnectés. Ces nœuds sont des ordinateurs spécialisés dans la transmission des données. [TANEN90] parle de processeur d'interface de message - IMP dans ARPANET. Les IMP stockent les messages qu'ils reçoivent avant de les réexpédier vers un autre nœud lorsque la ligne est libre. De proche en proche, le message parcourt le chemin qui le mène au destinataire.

Il y a trois types de commutations :

- La **commutation de circuits** qui établit un chemin physique entre les machines désirant communiquer. Ce chemin est conservé pendant toute la durée de la communication.

- La **commutation par messages**. Les informations à transférer sont envoyées en bloc. La longueur de ces messages est variable.

- La **commutation par paquets**. Dans ce cas, les messages échangés sont découpés en paquets de longueur fixe. Ce type de commutation est le plus courant dans le principe du "store-and-forward".

Comme topologies pour ce type de réseaux, on retrouve l'étoile, l'anneau, l'arbre, les maillages régulier et irrégulier ou l'intersection d'anneaux.

Les WAN utilisent, dans la majorité des cas, les réseaux publics tels que le réseau DARPA aux Etats-Unis, ou, plus proche de nous, TRANSPAC en France.

1.3.2.2 Réseaux locaux

Le deuxième type de réseaux est le réseau local ou LAN - Local Area Network. C'est ce type qui est le plus répandu sur le marché. Il est souvent présent dans les organisations pour lesquelles la connexion de leurs ordinateurs sur un réseau est nécessaire.

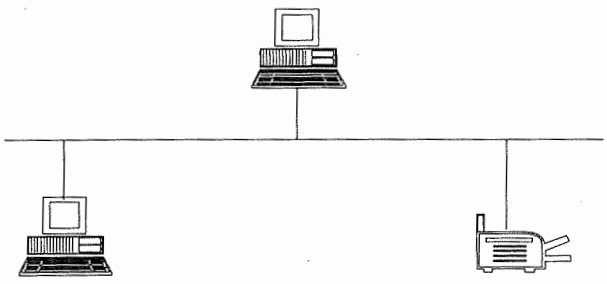


Figure 1.7 : réseau local

L'étendue géographique n'excède pas quelques kilomètres et s'en tient aux limites d'un bâtiment. Contrairement aux WAN, il a des débits de transmission très élevés. De nouvelles technologies telles que les FDDI vont dans ce sens.

Ce type d'architecture de réseau utilise la plupart du temps la diffusion. Les nœuds de communication sont dans les machines hôtes. Les messages sont envoyés sur le réseau de communication partagé par tous les hôtes. Ceux-ci les reçoivent et testent s'ils leur sont destinés.

Les réseaux de diffusion peuvent être de type bus, anneau.

1.3.2.3 Réseaux métropolitains

Depuis quelques années, on entend parler du Metropolitan Area Network - MAN - , réseau métropolitain ou urbain à mi-chemin entre le LAN et le WAN. Il a pour objectif de desservir une étendue géographique plus importante que celle couverte par les réseaux locaux - une ville, par exemple - tout en utilisant les techniques adaptées aux LAN. Il permet d'interconnecter des réseaux locaux tout en gardant le haut débit des communications. Il sert alors en quelque sorte d' "épine dorsale" à l'intégration de réseaux locaux [PUJOL92].

1.3.3 Systèmes ouverts

Pour pouvoir communiquer entre elles, les machines ont besoin de parler un même langage. L'ISO - International Standards Organization - avec son modèle de référence de base [ISO7498] fournit une architecture pour le développement des normes OSI - Open System Interconnection - qui ont pour but de permettre la communication et la coopération entre tous les systèmes informatiques pour réaliser une tâche commune. On obtient en suivant ces normes définies par l'ISO ce que l'on appelle un système ouvert. A cette fin, elles établissent un ensemble de règles. Le modèle de base introduit la notion de structuration en sept couches. Ce modèle en couche s'appuie sur le concept de protocole. Chaque couche fournit un ensemble de services. Ceux-ci sont utilisés par les couches de niveaux supérieurs.

La motivation à privilégier ce type de modèle est d'obtenir une certaine indépendance entre les couches. Ainsi, il est possible de modifier les services et les fonctions propres à une couche sans modifier les couches adjacentes.

1.3.3.1 Concepts élémentaires

Le modèle OSI a défini des concepts élémentaires concernant le fonctionnement des différentes couches et de leurs interactions.

Ainsi, dans une couche (N), des entités (N) réalisent des fonctions spécifiques à cette couche. Les entités de niveau (N) interagissent non seulement avec les entités de niveau (N+1) et (N-1) mais également avec les entités (N) homologues situées dans un système ouvert distant. L'ensemble des règles qui permettent à deux systèmes de communiquer s'appelle un protocole.

1.3.3.2 Couches du modèle OSI

Sept couches composent la pile OSI. Nous allons aborder maintenant une présentation rapide de celles-ci.

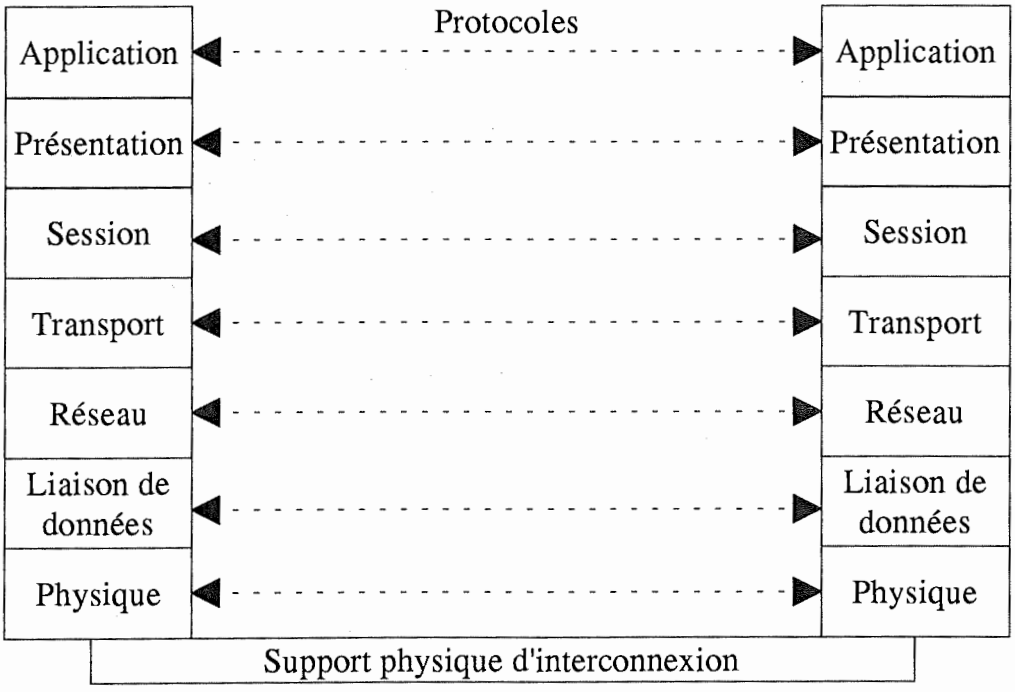


Figure 1.8 : le modèle en couches OSI

- Niveau 1 : la **couche physique**

  La couche physique, située directement au dessus du support physique d'interconnexion s'occupe des transmissions de bits entre deux sites. Elle fournit les moyens mécaniques, électriques, fonctionnels et procéduraux pour établir, maintenir et libérer les connexions physiques. Une telle connexion peut concerner de multiples systèmes intermédiaires relayant la transmission des bits dans la couche physique.

- Niveau 2 : la **couche liaison de données**

  La couche liaison de données est responsable de l'intégrité de l'information envoyée au travers d'un moyen physique. Celui-ci peut comporter du "bruit" et donc modifier le flux des données qui le traversent. Un mécanisme, fourni par le niveau 2 de la pile OSI, permet de détecter les erreurs et les corriger.

- Niveau 3 : la **couche réseau**

  Le niveau 3 a pour mission d'assurer le transfert correct de l'information à travers le réseau. Il s'occupe donc du routage des messages. Il est également habilité à résoudre les problèmes issus de l'interconnexion de réseaux hétérogènes.

- Niveau 4 : la **couche transport**

  La couche transport a été adoptée dans le but de masquer aux couches supérieures les détails à propos des réseaux touchés par la transmission d'informations les concernant. Il est le noeud entre ce que l'on appelle les couches orientées applications et les couches orientées communications, respectivement les niveaux 5 à 7 et 1 à 4.

- Niveau 5 : la **couche session**

  La couche session fournit aux entités de présentation les moyens nécessaires à la synchronisation et à l'organisation du dialogue. Elle offre donc comme service l'établissement, le maintien et la libération d'une connexion ainsi que le contrôle des interactions entre entités de présentation.

- Niveau 6 : la **couche présentation**

  La portée de la couche présentation se situe au niveau de la syntaxe des informations échangées par les entités d'applications. Cette couche est particulièrement importante dans un environnement hétérogène. "C'est un intermédiaire indispensable pour une compréhension commune de la syntaxe des documents qui sont transportés sur le réseau" [PUJOL92]. L'ISO a normalisé une syntaxe abstraite pour le langage syntaxique permettant cette compréhension. Il s'agit de l'Abstract Syntax Notation One (ASN.1) [ISO8824]. ASN.1 définit trois sortes de types : les types standards tels Boolean, Integer, Bit String, Octet String, Null, Sequence of, Set, Set of, Choice, Selection, Tagged, Any; les types Character String et les types utiles comme la date et l'heure.

- Niveau 7 : la **couche application**

  La dernière couche du modèle OSI est la couche application. Elle permet aux processus d'applications d'accéder à l'environnement OSI. L'échange d'information entre les processus d'applications s'effectue via les entités de cette couche. Sa préoccupation première est la sémantique des informations. Nous allons nous attarder quelque peu sur cette couche dans le chapitre 3 consacré à la gestion OSI.

[TANEN90] propose un approfondissement de ces couches en y consacrant un chapitre pour chacune d'entre elles.

1.3.4 Environnement TCP/IP

Alors que l'OSI constitue un standard *de jure*, il existe un environnement qui, par l'ampleur de son utilisation, est devenu un standard *de facto*. Il s'agit de TCP/IP. L'architecture TCP/IP, bien que définie pour la défense américaine est utilisée à l'heure actuelle par bon nombre d'institutions dont l'informatique est fortement mise en réseau.

Le DOD - Department of Defense -, au début des années 1970 décida de définir sa propre architecture de communication afin d'unifier l'interconnexion des innombrables machines qu'il utilise. Cette architecture peut être découpée en quatre couches.

- Correspondante aux niveaux 1 et 2 du modèle OSI, une **couche d'accès au réseau** - Network Interface Layer - définit l'interface entre les couches supérieures et le réseau sous-jacent. Il permet de cacher les détails propres aux types de réseaux. Ainsi, les différentes applications sur des machines distantes communiquent sans se soucier des caractéristiques du réseaux.

- La couche réseau appelée aussi **couche Internet** est responsable du routage de l'information et de l'interconnexion de réseaux Le protocole le plus important de cette couche est l'Internet Protocol - IP. Il fournit un service d'envoi de datagrammes. Le datagramme est l'unité d'information de la couche Internet.

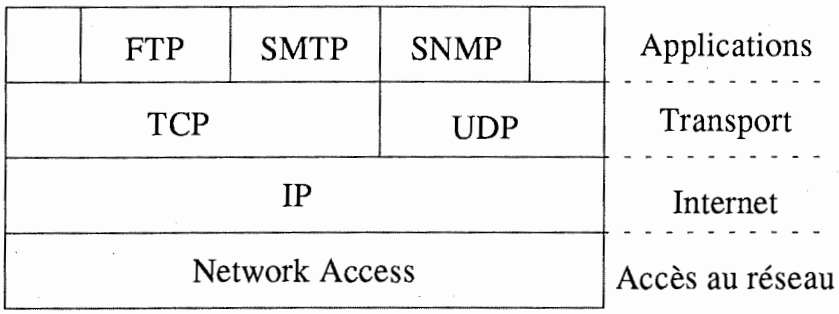


Figure 1.9 : couches TCP/IP

- L'assemblage et le désassemblage des datagrammes est du ressort de TCP - **Transmission Control Protocol**. Ce protocole de niveau 4 fournit un service de transport fiable entre deux systèmes. Un deuxième protocole, UDP - **User**

Datagram Protocol - fournit un service parallèle à TCP. A la différence de ce dernier, UDP est non fiable et en mode non connecté.

- La couche application de l'architecture TCP/IP est constituée d'un ensemble de protocoles offrant des services aux utilisateurs. Ainsi, FTP - **File Transfer Protocol** - permet l'échange de fichiers entre ordinateurs; SMTP - **Simple Mail Transfer Protocol** - offre un service de courrier électronique; SNMP - **Simple Network Management Protocol** - est un utilitaire d'administration de réseaux qui fournit des services d'échanges d'informations de gestion entre les constituants du réseau et la station d'administration - nous pouvons nous reporter à [HEYVA91] dont le mémoire porte sur SNMP.

1.4 Systèmes distribués

1.4.1 Caractéristiques des systèmes distribués

Si l'on retient ce que dit [TANEN90] lorsqu'il affirme qu'un système distribué est un cas particulier de réseau dont le système d'exploitation distribué assure la transparence du système, en plus des objectifs des réseaux, nous devons ajouter comme objectifs aux systèmes distribués le concept-clé de transparence.

La transparence est définie comme la possibilité de voir un système unifié et non comme un ensemble d'objets indépendants. [ANSA87] a défini huit types de transparence :

- **Transparence d'accès** : accéder à des ressources s'effectue toujours de la même manière. Qu'un fichier soit local ou éloigné, l'opération pour y accéder est toujours identique. Typiquement, la manière de nommer une ressource ne variera pas suivant son emplacement.

- **Transparence de localisation** : il n'est pas nécessaire de connaître la localisation physique d'une ressource pour pouvoir l'atteindre.

- **Transparence de concurrence** : plusieurs utilisateurs peuvent se partager simultanément des données sans qu'il y ait d'effets pervers.

- **Transparence de duplication** : afin d'améliorer les performances du système, il peut y avoir plusieurs instances de ressources - des fichiers par exemple - sans que cela soit visible aux utilisateurs.

- **Transparence aux pannes** : des erreurs dues au hardware ou au software sont cachées aux applications. Cette forme de transparence permet donc aux applications de terminer leur tâches sans se rendre compte des problèmes existants.

- **Transparence de migration** : le déplacement de ressources à l'intérieur du système n'affecte en rien le déroulement des travaux en cours. Déplacer un fichier trop encombrant d'un disque vers un autre pour des raisons de performances ne changera en rien l'exécution des tâches.

- **Transparence de performance** : elle permet au système d'être reconfiguré dynamiquement suivant les variations de la charge.

- **Transparence d'échelle** : elle permet l'extension ou la modification du système sans modifier sa structure.

Pour expliquer les caractéristiques des systèmes distribués, nous allons présenter des problèmes auxquels doivent faire face ces systèmes. Ces problèmes existent également dans les réseaux d'ordinateurs. Cela permettra de les comparer afin de mieux en cerner les différences. Il s'agit des systèmes de fichiers, de la protection de l'accès aux ressources et de l'exécution des processus.

1.4.1.1 Systèmes de fichiers

Un des premiers problèmes auxquels sont confrontés les systèmes distribués est l'accès aux fichiers. Dans le cas de machines indépendantes, des systèmes de fichiers gèrent celui-ci pour les fichiers locaux à la machine. Lorsque l'on interconnecte ces machines autonomes, nous obtenons comme nous l'avons dit au point 1.2.2 un réseau d'ordinateurs. [TANEN85] explique comment a évolué la politique adoptée pour tendre vers la transparence d'accès aux fichiers et de localisation de ceux-ci.

La première solution, qui finalement n'en est pas une, est de ne rien faire du tout. Pour accéder à un fichier sur une machine distante, l'utilisateur doit tout d'abord le copier sur sa propre machine. Cela s'effectue grâce à un programme de transfert de fichiers dont un exemple connu est le programme uucp d'UNIX. Nous pouvons remarquer qu'il n'y a pas de transparence d'accès puisque pour accéder à un fichier éloigné, il faut d'abord le copier. Il n'y a pas de transparence de localisation non plus, l'utilisateur désirant accéder à un fichier doit également savoir sur quelle machine il se trouve.

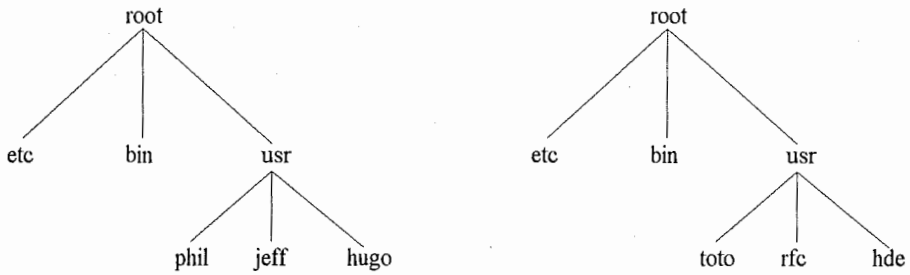


Figure 1.10 : systèmes de fichiers séparés

La seconde étape est de créer un super-répertoire virtuel de tous les systèmes de fichiers. Dans ce cas-ci, l'accession à un fichier se fait de manière transparente. Atteindre un fichier sur une machine distante s'effectue en utilisant les mêmes commandes que pour un fichier local. Par contre la transparence de localisation n'est pas encore réelle. Il est toujours nécessaire de connaître l'emplacement physique d'un fichier pour l'atteindre. Le système de fichiers de Newcastle Connection en est un exemple. La figure 1.11 illustre le super-répertoire virtuel reprenant les répertoires principaux des systèmes de fichiers de machines distantes.

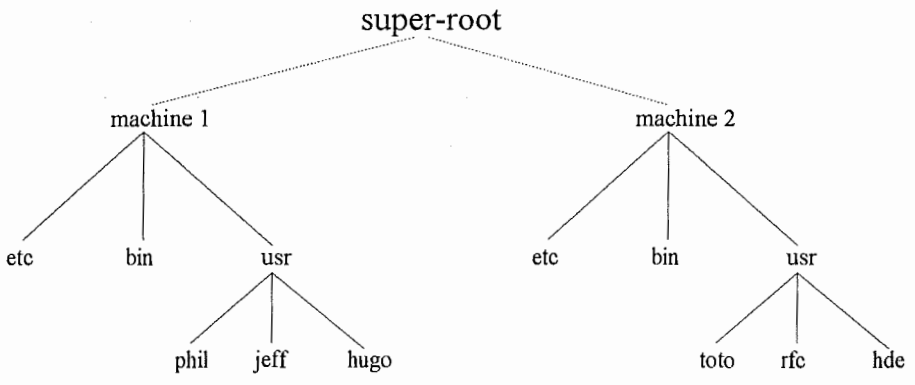


Figure 1.11 : super-répertoire virtuel de systèmes de fichiers distants

Dans les systèmes distribués, l'approche qui est retenue est un système de fichiers unique portant sur l'ensemble du système. Le système distribué LOCUS correspond à cette approche. Les transparences d'accès et de localisation sont toutes les deux bien réelles puisque tout se passe comme si l'on se trouvait devant un seul système de fichiers, comme dans un système centralisé.

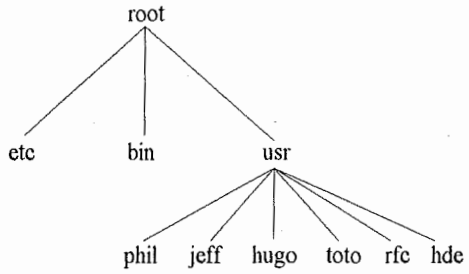


Figure 1.12 : système de fichiers unique

De plus, toute opération de contrôle des fichiers à des fins de performances - transparence de performance -, par exemple, est prise en charge par le système d'exploitation. C'est lui qui décide du déplacement de fichiers - transparence de migration - ou de leur copie -transparence de duplication. A la différence, les réseaux obligent un contrôle manuel non pris en charge automatiquement par le système lui-même.

1.4.1.2 Protection de l'accès

Accéder à une ressource demande de disposer de droits particuliers - droits en écriture pour modifier un fichier, en lecture pour le consulter, etc. - dont la gestion peut s'étendre d'une technique triviale à une plus complexe. Cela correspond à l'évolution entre le réseau et le système distribué. Ainsi, si l'on garde toujours l'exemple des fichiers nous pouvons aborder les solutions au problème de la protection des ressources.

La première solution, la plus facile, mais également la moins transparente pour l'utilisateur, est de demander à celui-ci de se connecter à une machine avant d'accéder à un fichier qui lui est local. A la connexion, l'utilisateur est identifié par le système. Le contrôle de l'accès aux fichiers peut alors être effectué. Cette méthode n'est pas du tout pratique.

L'étape suivante qui ôte l'obligation de se connecter au préalable d'une opération sur un fichier est de permettre à tous les utilisateurs d'atteindre un fichier distant en tant

qu'invité. Mais dans ce cas, seuls les fichiers publics peuvent être accédés. Ce n'est pas une solution à prendre en compte puisqu'elle restreint fortement l'utilité de l'accès à distance.

Une solution plus intelligente est de faire une correspondance entre les identifiants des utilisateurs. Ainsi, le système d'exploitation d'une machine possède une table de correspondance des utilisateurs reprenant pour chacune des machines interconnectées par le réseau les identifiants de leurs utilisateurs et l'identifiant local. Cela pose des problèmes de gestion cohérente de l'ensemble des tables.

Dans les systèmes distribués, l'identifiant d'un utilisateur est unique si bien qu'il n'y a pas de correspondance à effectuer pour accéder à une machine distante.

Pour résumer ce problème de la protection de l'accès aux ressources, nous pouvons dire que dans le cas des réseaux d'ordinateurs, chaque machine autonome possède un système d'identifiants d'utilisateurs qui lui est propre, alors que dans les cas des systèmes distribués, l'identifiant d'un utilisateur est valable à l'échelle du système.

1.4.1.3 Exécution des processus

Dans un système distribué, la localisation de l'exécution des programmes n'est pas connue de l'utilisateur. C'est au système d'exploitation de choisir les processeurs qui seront alloués pour l'exécution des processus. Ce choix s'effectue en fonction, par exemple, de la charge des processeurs. Il choisira celui dont la charge est la moins grande. Cela peut également se faire en fonction des données qui sont utilisées. Il exécutera ainsi le processus sur la machine sur laquelle est physiquement situé le fichier à atteindre.

Dans le cas d'un réseau, les processus tournent habituellement en local, c'est-à-dire sur la machine à partir de laquelle ils ont été lancés. Pour utiliser le processeur d'une autre machine, il faut lancer une commande spéciale du style "remote mac2 prog" - demande d'exécution du programme "prog" sur la machine "mac2".

Ici aussi, les systèmes distribués doivent assurer une transparence quant à la localisation de l'exécution des traitements. C'est une différence supplémentaire avec les réseaux.

1.4.2 Types de systèmes distribués

Un système distribué est un ensemble d'éléments informatiques disposant d'un processeur et reliés entre eux par un moyen de communication. Ce dernier est le support de l'interaction entre les différents processeurs. En ce, ils ne sont pas fort différents des réseaux d'ordinateurs. Nous avons dit que ce qui les différenciait était la vue que l'utilisateur a de ce système. Il ne s'agit plus d'un ensemble de machines interconnectées mais d'une seule machine virtuelle. Si l'on utilise le moyen de communication - même s'il n'apparaît plus dans cette notion de machine unique - pour classifier les sytèmes informatiques répartis nous pouvons en faire ressortir deux types : les architectures fortement couplées et celles qui le sont faiblement.

1.4.2.1 Systèmes distribués fortement couplés

Dans le cas où le moyen de communication est une mémoire commune, on est en présence d'une architecture fortement couplée. Cette mémoire partagée peut être adressée directement par tous les processeurs en présence.

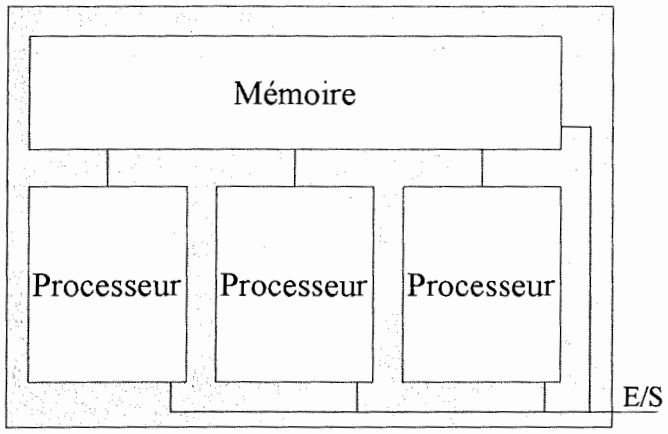


Figure 1.13 : système distribué fortement couplé

Nous ne nous attarderons pas plus longtemps à ce type de systèmes distribués. Nous retiendrons simplement qu'il s'agit d'une architecture multi-processeurs souvent destinée à des techniques de parallélisme.

Cette notion de parallélisme comme nous le fait remarquer [GABAS92] n'est pas à confondre avec celle de distribution - ou de répartition. La répartition de traitements

"consiste à les faire exécuter par la machine la mieux adaptée". Le parallélisme implique plutôt la notion d'exécution de traitements simultanée. Il peut arriver, cependant que la répartition de traitements entraîne un parallélisme de ceux-ci.

1.4.2.2 Systèmes distribués faiblement couplés

Des systèmes distribués faiblement couplés sont différents des précédents par le fait qu'il n'y a pas de mémoire commune. Le lieu commun entre les différents processeurs n'est plus la mémoire mais un moyen de communication. Ce moyen de communication peut servir à classifier les systèmes faiblement couplés en deux catégories : les architectures mono-machine et les architectures multi-machines.

Dans les architectures ***mono-machine***, les différents processeurs communiquent entre eux via un moyen de communication à très haut débit, généralement un bus. La figure 1.14 illustre ce type de système distribué.

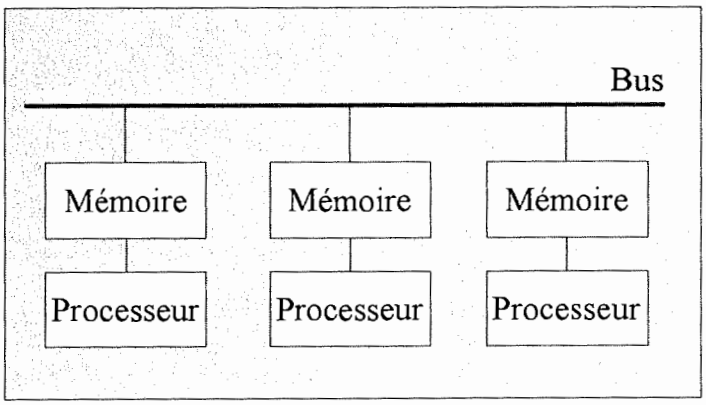


Figure 1.14 : système distribué faiblement couplé mono-machine

Dans le cas des systèmes distribués faiblement couplés ***multi-machines***, le moyen de communication est un réseau. Les machines disposent chacune de leur(s) propre(s) processeur(s), de leur propre mémoire et de leur propre interface d'entrée-sortie. La figure 1.15 donne un exemple de système distribué faiblement couplé dans lequel les machines communiquent entre elles via un réseau local.

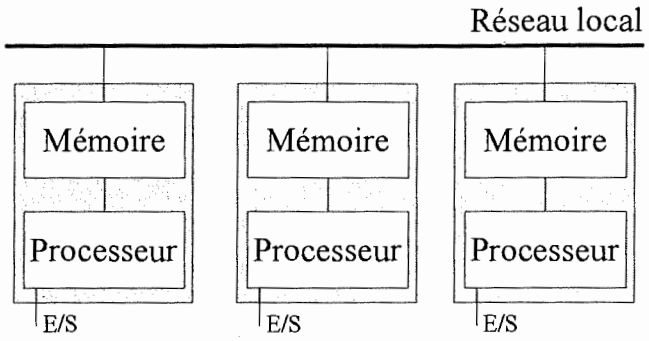


Figure 1.15 : système distribué faiblement couplé multi-machines

Nous nous en tiendrons à ce dernier type de systèmes distribués. Cela ne veut pas dire cependant que les autres systèmes présentés, à architecture mono-machine - ou multi-processeurs - ne constituent pas des systèmes distribués à part entière. Des systèmes d'exploitation distribués tels que Chorus ou Mach sont capables de gérer aussi bien des systèmes multi-processeurs que des systèmes multi-machines. De plus, un amalgame de ces deux types de systèmes peut exister. Au sein d'une organisation multi-machines, les machines peuvent très bien être de type multi-processeurs. C'est au système d'exploitation à prendre en compte ces situations.

Nous pouvons présenter l'architecture des systèmes distribués faiblement couplés multi-machines en présentant trois modèles. Nous verrons, comme [COULO89], le modèle stations de travail/serveurs, le modèle pool de processeurs et le modèle intégré.

- Dans le ***modèle stations de travail/serveurs***, le système distribué est composé de stations de travail et de serveurs reliés entre eux par un réseau. La figure 1.16 en illustre un exemple. Les stations de travail sont des ordinateurs mono-utilisateurs disposant d'un processeur puissant, d'une mémoire, d'un écran haute-résolution et parfois d'un disque. Ce sera par exemple un PC disposant d'un processeur 80486 avec 8 MB de mémoire centrale. Des machines plus puissantes, les serveurs, offrent des services à ces stations de travail. Ces services vont de l'accès aux périphériques - comme les imprimantes - à l'accès aux données partagées - on parlera de serveurs de fichiers comme NFS. D'autres serveurs offrent des fonctionnalités remplaçant celles d'un système d'exploitation centralisé - serveurs d'authentification entre autres.

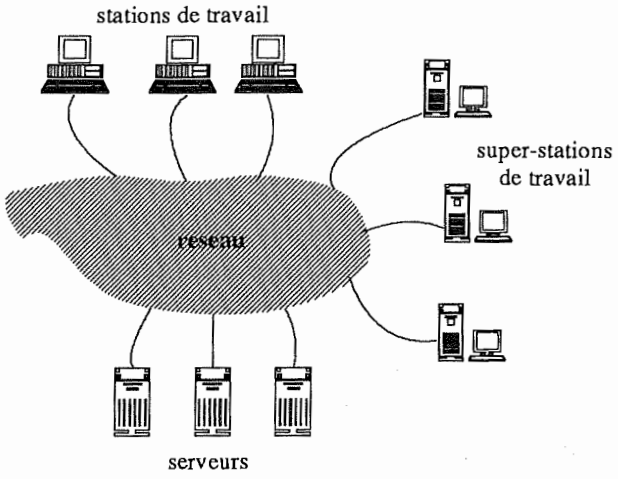


Figure 1.16 : exemple de modèle stations de travail/serveurs

- Comme nous pouvons le voir dans la figure 1.17, le modèle ***pool de processeurs*** ne comporte plus de stations de travail comme dans le modèle précédent. Au contraire, elles ont été remplacées par des terminaux reliés au réseau via un concentrateur. Les terminaux accèdent aux processeurs via le réseau. Les pools de processeurs sont des ordinateurs - mini ou micro - sans terminaux.

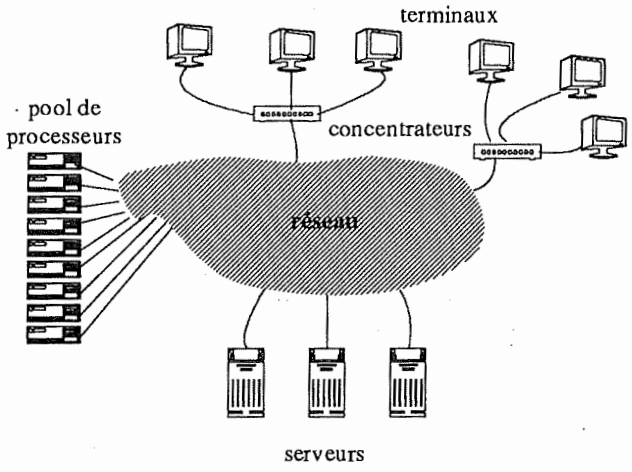


Figure 1.17 : modèle pool de processeurs

Les terminaux et pools de processeurs ne possèdent pas de mémoires de stockages. C'est donc des serveurs de fichiers qui ont la charge de gérer les fichiers. D'autres serveurs sont également attachés à ce modèle. Il s'agira de serveurs de noms, d'impression, etc.. Un serveur de processeur alloue à

l'utilisateur d'un terminal un processeur en fonction de ses besoins. Après cette attribution, un serveur charge dans le processeur un programme - souvent un système d'exploitation. L'utilisateur peut alors interagir avec le processeur. Un exemple de système distribué s'appuyant sur ce modèle est le Cambridge Distributed Computing System.

Des systèmes distribués tels qu'Amœba [MULLE90] combinent ces deux modèles. On parlera alors de systèmes hybrides. La figure 1.18 en montre un exemple. Chaque utilisateur dispose d'une station de travail et, en plus, peut disposer de processeurs comme dans le modèle que nous venons de présenter.

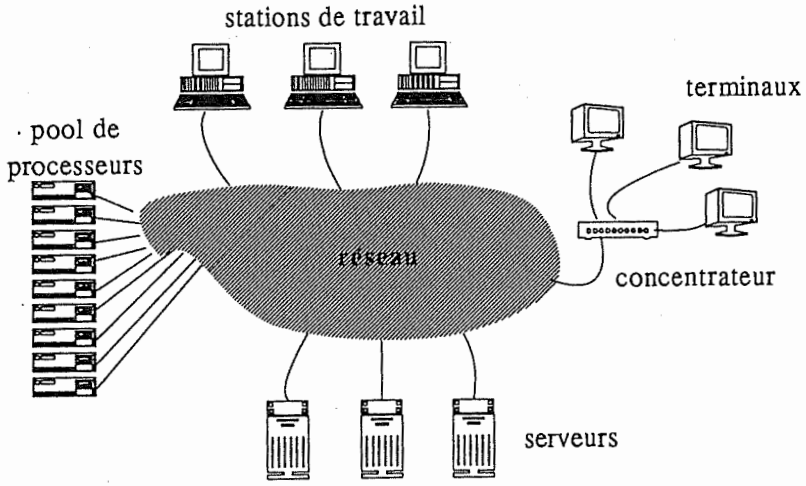


Figure 1.18 : exemple de système hybride

- Le dernier modèle de système distribué faiblement couplé multi-machines est le ***modèle intégré***. Un système comme Locus entre dans cette catégorie. Dans ce type de système distribué, le système d'exploitation est distribué sur toutes les machines. C'est grâce à celui-ci que le système distribué apparaît comme une unité. Les processus sont distribués automatiquement par le système d'exploitation, le système de nommage permettant d'accéder aux données est global.

1.4.3 Composants logiciels d'un système distribué

Ce qui différencie un système distribué d'un réseau comme nous l'avons déjà dit est la transparence. L'utilisateur voit un ensemble d'éléments comme un tout. Afin d'atteindre

cette transparence, il est nécessaire de disposer d'un système d'exploitation gérant les problèmes inhérents à cette spécificité.

Le système d'exploitation d'un vrai système distribué doit être pensé dès le départ en ayant en tête sa finalité. Il doit faire face, bien entendu, aux problèmes que l'on retrouve dans les autres types de systèmes, à savoir la gestion des systèmes de fichiers, la gestion des périphériques, la gestion de la sécurité - authentification, contrôle d'accès,... -, gestion de la mémoire, des processeurs, gestion des processus.

Alors que dans les systèmes centralisés les fonctions du système d'exploitation - OS, Operating System - sont effectuées dans ce que l'on appelle le noyau de l'OS, dans un système d'exploitation distribué, le noyau est allégé et ne comporte plus que des fonctions minimales locales telles que la création de processus, l'accès à la mémoire et aux périphériques ainsi que les fonctions utiles pour la communication locale entre les processus - IPC, interprocess communication. Les autres fonctions autrefois remplies par le noyau deviennent des applications que l'on appelle des services. D'autres applications utilisent ces services. On les appelle les clients. Ces différentes applications peuvent être distribuées physiquement. Nous pourrions avoir une machine dédiée au service s'occupant de la gestion des fichiers, une autre gérant les impressions, etc.. C'est ce que l'on appelle les serveurs. Il faut donc que les processus clients tournant sur une machine puissent communiquer avec les processus serveurs tournant sur d'autres. A cette fin, il faut disposer d'un système de communication. Deux systèmes peuvent être adoptés : celui se basant sur le modèle OSI dont nous avons déjà parlé plus haut au point 1.3.3 et celui basé sur les appels de procédures à distance - RPC, Remote Procedure Call - et s'appuyant plutôt sur l'environnement TCP/IP. Nous n'expliquerons pas ici les mécanismes des RPC - voir [NILLE91]. De plus, comme le montre [DANTH89], il est possible d'adopter une solution reprenant la technique des RPC en utilisant l'OSI. La figure 1.19 résume les composants logiciels d'un système d'exploitation distribué.

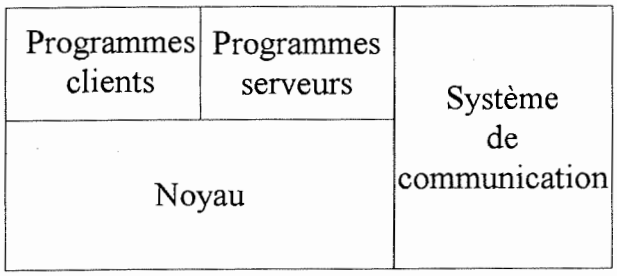


Figure 1.19 : système d'exploitation distribué

1.5 Conclusion

Les évolutions technologiques qui ont caractérisé cette dernière décennie en matière d'informatique ont permis de passer des systèmes informatiques centralisés aux réseaux d'ordinateurs. Ces progrès permettent maintenant de porter nos espoirs sur des systèmes distribués. Il existe différents types de systèmes distribués. Nous pouvons les reprendre en les illustrant par un arbre - figure 1.20. Notez que des croisements entre les branches de cet arbre peuvent exister. Ils ne sont pas représentés ici pour des raisons de clarté.

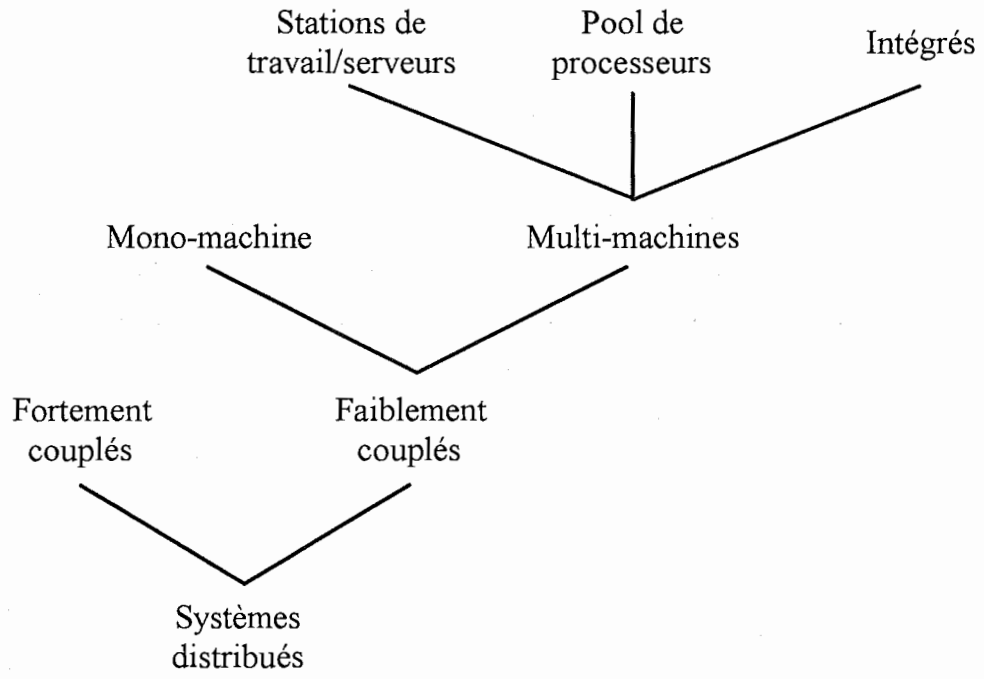


Figure 1.20 : arbre des systèmes distribués

Les différents types sont les systèmes fortement couplés correspondant à des machines multi-processeurs partageant une mémoire commune et les systèmes faiblement couplés ayant plusieurs processeurs. Ces derniers se subdivisent en fonction du moyen de communication entre les processeurs. S'il s'agit d'un bus, nous avons un système mono-machine. Si par contre, c'est un réseau, le système est de type multi-machines. Ces derniers systèmes sont fortement basés sur les réseaux. La différence est la visibilité du système qu'ont les utilisateurs. Alors que dans un réseau chacun est conscient de la multiplicité des composants, dans un système distribué, l'utilisateur voit tous ces composants comme un tout. Il a l'impression de travailler sur une seule machine. Cette transparence est assurée par le système d'exploitation qui assure le fonctionnement du système. Elle se retrouve à différents niveaux. Nous citerons les plus importants - ceux pour lesquels la recherche a le

plus avancé. Ce sont les systèmes de fichiers, l'accès aux données et l'exécution des processus. Tout comme dans les réseaux, des besoins de communiquer entre les différentes machines existent. Nous retiendrons le modèle proposé par l'OSI pour l'implémentation d'un système de communication. Dans le monde informatique, la différence entre les réseaux et les systèmes distribués semble difficilement perceptible si bien que l'on emploie parfois le terme de système distribué dès que le système a une architecture de réseau. Dans les chapitres qui suivent, consacrés à l'administration, il est possible que la notion employée pour exprimer un système distribué se rapproche plus du réseau. Cela est dû au fait que l'administration des systèmes distribués telle qu'elle est présentée dans la littérature ou par les constructeurs, se rapporte plus en réalité aux réseaux d'ordinateurs. Toutefois, il y a très peu de véritables systèmes pleinement distribués qui sont utilisés. Il y a, dès lors, une tendance à ce que des systèmes de réseaux offrant des facilités proches de celles des systèmes distribués soient qualifiés de distribués par leur constructeur.

Chapitre 2

Administration d'environnements distribués

Chapitre 2 : Administration d'environnements distribués

2.1 Introduction

Le cadre de travail étant maintenant établi, nous allons nous consacrer à la présentation de la gestion de ces systèmes. L'environnement distribué qui les caractérise amène les concepteurs de solutions de gestion à penser celles-ci en terme d'intégration.

Nous insisterons tout d'abord sur la nécessité de gérer les systèmes distribués. Nous nous pencherons ensuite sur les objectifs de l'administration ainsi que sur les fonctions permettant de les atteindre. Le point suivant s'attardera à la présentation de la structure de la gestion de systèmes distribués, à ses composants et à la notion de domaine qui revêt une importance particulière dans celle-ci. Nous terminerons ce chapitre par une présentation de la solution idéale quant à l'implémentation de ceux-ci : l'administration intégrée.

2.2 Nécessité d'une gestion

Par ce que nous venons de présenter dans le chapitre 1, il est aisé de percevoir les facettes affirmant l'importance de disposer d'outils pour administrer les environnements distribués. Reprenons-les cependant pour souligner la complexité d'une telle gestion, comme le fait [SLOMA90].

- Les différents composants d'un système distribué, qu'il s'agisse des applications, des moyens de communications ou des autres ressources, devront fonctionner sur des systèmes informatiques appartenant à un certain nombre d'organisations différentes devant parfois se soumettre à des législations différentes.

- Une forme d'hétérogénéité réalisée par une diversité des composants rend la gestion d'un système non triviale. La variété des machines à administrer se caractérise non seulement par leurs différences conceptuelles (il peut s'agir de micro-ordinateurs, de mini, de mainframes, etc.) mais également par leurs fonctionnalités (elles fournissent des services différents allant d'un simple service journalier à des services compliqués tels que des bases de données distribuées.

- La multiplicité des fournisseurs de ces composants et services constitue un second niveau d'hétérogénéité. Ceci rend difficile une vision intégrée de l'administration.

- Le fait que les composants et services d'un système à administrer soient largement distribués physiquement ne contribue pas à rendre aisée la réalisation de cette tâche. Il est en effet difficile d'obtenir les informations de manière uniforme afin de prendre des décisions fondées sur la situation réelle du système ou d'effectuer des opérations de gestion cohérentes avec celle-ci.

- La taille et la complexification des systèmes distribués fait apparaître des problèmes d'échelle. Le nombre important d'éléments à administrer rend la gestion impossible si l'on s'en tient à traiter chacun de ceux-ci séparément. Il devient absolument nécessaire d'effectuer un regroupement des éléments.

2.3 Objectifs de la gestion

La distribution des différents éléments d'un système réparti donne un caractère de très haut niveau à la communication entre les différentes ressources le composant. Les objectifs de la gestion des systèmes distribués s'en ressent. Il faut donc établir un parallèle plus qu'implicite entre la gestion de ce type de systèmes et la gestion de réseaux.

L'administration de systèmes englobe l'ensemble des moyens mis en œuvre pour atteindre ces objectifs : [NOBLO91], [MILLE91]

- offrir aux utilisateurs du système une **qualité de service**. Cela se réalise en assurant une certaine disponibilité des composants du système, en gardant un temps de réponse raisonnable, et ce, quels que soient les changements apportés au système.

- permettre l'**évolution** du système en incluant de nouvelles fonctionnalités.

- remplir les fonctions de la **partie opérationnelle** du système. L'administration est appliquée suivant une politique qui spécifie une stratégie, c'est-à-dire un plan des actions à entreprendre pour atteindre les objectifs de l'entreprise. Afin de

réaliser ce plan, la politique spécifie également une tactique. En soutien à cette tactique un fonctionnement opérationnel est défini. C'est à ce niveau que se situe l'administration.

Les domaines d'intervention de la gestion des systèmes sont le réseau et ses composants, les plates-formes hôtes ainsi que les composants logiciels locaux et distribués.

2.4 Fonctions de gestion

L'administration des systèmes distribués restera toujours dans cet état complexe si un mécanisme de partitionnement n'est pas pris en compte dans le champ de ses fonctionnalités. L'OSI, dans [ISO 7498-4], a classifié les fonctions nécessaires à la gestion de systèmes en cinq aires fonctionnelles : les gestions de la configuration, des fautes, des performances, de la sécurité et de la comptabilité.

2.4.1 Gestion de la configuration

Comme nous le fait remarquer [ADAMS91], cette aire fonctionnelle est très importante. Preuve en est qu'elle établit l'inventaire des ressources à gérer. A partir de cette fonctionnalité, toutes les autres sont possibles.

Les fonctionnalités de la gestion de la configuration reprennent les fonctions d'installation des composants hardware ou software du système. Ainsi, elles permettent la mise en service d'un composant, son initialisation et sa suppression. Elles offrent également une fonction de contrôle pour vérifier, sur demande ou de manière régulière, l'état de différents aspects.

A partir de cet inventaire et des connexions réelles des éléments du point de vue physique, il est possible, entre autres applications, de construire graphiquement la topologie du système. Celui-ci ainsi représenté à l'écran peut servir de carte à partir de laquelle l'administrateur voyage sans quitter son bureau pour effectuer des opérations de gestion.

Une part importante de la gestion de la configuration est la gestion des noms. Elle permet à un utilisateur d'accéder à une ressource en la nommant de manière symbolique. Les mécanismes les plus populaires d'association des noms aux objets, selon [MAZDA91],

sont les techniques des "pages blanches" et des "pages jaunes" par lesquels les utilisateurs peuvent connaître les valeurs associées à un nom ou le nom d'un objet à partir de certains attributs qui lui sont associés.

2.4.2 Gestion des fautes

La gestion des fautes est une aire de l'administration des systèmes qui revêt une importance particulière en ce sens qu'elle contribue à la réalisation des objectifs d'une manière primordiale. En effet, que serait la qualité du service si des éléments du système ne sont plus opérants ou sont défectueux ? Un élément seul, dans un état défectueux, peut entraîner une indisponibilité parfois totale du système. Pensons simplement à ce qu'il adviendrait si une machine supportant un serveur de fichier devenait non fiable. Il est donc important de disposer d'outils permettant de faire face à de telles situations.

Afin de répondre à cette nécessité, la gestion des fautes offre des fonctions de surveillance, d'alarme, de localisation et de détermination des pannes.

Par les fonctions de surveillance, l'administrateur d'un système peut détecter des pannes en s'enquérant de l'état des composants de celui-ci. On parle alors du polling.

Un autre modèle permet à l'administrateur de prendre en compte des événements non sollicités. Ceux-ci sont connus grâce aux alarmes reçues des composants intéressés par l'incident. Pour disposer d'un tel mécanisme, les composants doivent être suffisamment "intelligents" pour faire connaître leurs défauts. On voit dès lors l'importance du polling pour les ressources ne répondant pas à cette condition.

Par un système de journaux, les alarmes sont stockées afin de permettre à l'administrateur de les prendre en compte en temps voulu. Les personnes responsables de la gestion du système ne connaissent pas toutes les pannes existantes, seules les plus importantes lui sont révélées. Des outils automatiques de réparation des erreurs permettent d'alléger leur part de travail. C'est le cas par exemple des systèmes experts.

L'interaction avec l'aire fonctionnelle précédente, la gestion de la configuration, est très forte. Suite à une panne, l'administrateur du système est amené à déterminer les actions curatives. Celles-ci peuvent être le remplacement ou l'élimination du composant défectueux. Une utilisation des fonctionnalités de la gestion de la configuration est nécessaire pour prendre en compte ces changements.

2.4.3 Gestion des performances

La gestion des performances a pour objectif d'offrir à l'administrateur des fonctionnalités de "monitoring" pour suivre la performance du système. Des informations statistiques (charge, débit, nombre de processus, nombre d'entrées/sorties, etc.) concernant ces composants sont rassemblées et sont utilisées pour prendre des décisions en terme de planning à long terme ou de prévoir des événements à court terme.

Ces données statistiques sont stockées dans des journaux afin de pouvoir être exploitées. Ces derniers permettent d'établir l'historique des caractéristiques des composants et ainsi voir les tendances qui en découlent.en terme de performance. [KIESEL91] parle de "longterm statistics" et de "trend analysis".

Afin d'améliorer les performances du système, l'administrateur doit être capable d'effectuer des tests par "tuning" en changeant certains paramètres des composants concernés par cette opération d'optimisation.

Ici également, la gestion de la configuration du système est nécessaire pour permettre une modification de celle-ci en vue d'améliorer les performances.

2.4.4 Gestion de la sécurité

La sécurité, parce que l'aspect critique de certaines ressources est primordial, revêt un caractère essentiel. L'administration des systèmes distribués doit donc prendre en compte la gestion de la sécurité et offrir des outils allant dans ce sens.

Ceux-ci sont par exemple la distribution des clés de chiffrement, le chiffrement des données, les contrôles d'accès aux ressources, la gestion de ces accès, l'audit des activités des utilisateurs. Des journaux de ces activités, à titre d'exemples les connexions, l'accès à des fichiers critiques, sont maintenus et peuvent être examinés.

2.4.5 Gestion de la comptabilité

La comptabilité est utile pour pouvoir disposer d'informations concernant la charge des différentes ressources du système. A partir de ces charges, l'administrateur peut vérifier le bon usage du système. Cela consiste, par exemple, à s'assurer que les ressources sont partagées équitablement entre les utilisateurs, qu'aucun de ceux-ci ne se trouve dans une situation de monopole vis-à-vis des autres.

A partir de la charge, l'administrateur doit dès lors en connaître le coût. Ces deux facettes de la comptabilité doivent pouvoir être réparties entre les différents utilisateurs du système.

La facturation de l'utilisation du système entre également en ligne de compte dans la gestion de la comptabilité. Il est intéressant pour un utilisateur de connaître la répartition de ses coûts d'utilisation du système.

La gestion des limites d'utilisation du système est une fonctionnalité intéressante pour l'administrateur. Les limites budgétaires, fixées en accord ou non avec chacun des utilisateurs, peuvent entraîner,une fois atteintes, des accès réduits aux ressources payantes.

2.5 Structures de l'administration

Nous venons de voir, dans la partie précédente, une réponse au problème posé par la complexité de l'administration des systèmes distribués. Une solution supplémentaire, comme en parle [SLOMA90], repose sur une autre forme de partitionnement. Celle-ci est basée sur la répartition des responsabilités de gestion. La structure de l'administration peut en découler. Sloman en différencie quatre types. Il s'agit de la structure organisationnelle, la structure physique, la structure au niveau des services et la structure consécutive aux frontières liées à la sécurité.

2.5.1 Structure organisationnelle

La première des structures pouvant servir de base à une modélisation de la gestion est la structure organisationnelle de l'entreprise. "Quand des organisations indépendantes coopèrent au sein d'un système distribué, les frontières de responsabilité correspondront

avec les frontières organisationnelles" [SLOMA90]. La structure adoptée pour la gestion du stystème doit également mettre en évidence l'indépendance des organisations.

Cependant, en raison de la coopération au sein du système, il est nécessaire que soient permises des interactions entre les différents administrateurs attachés aux organisations.

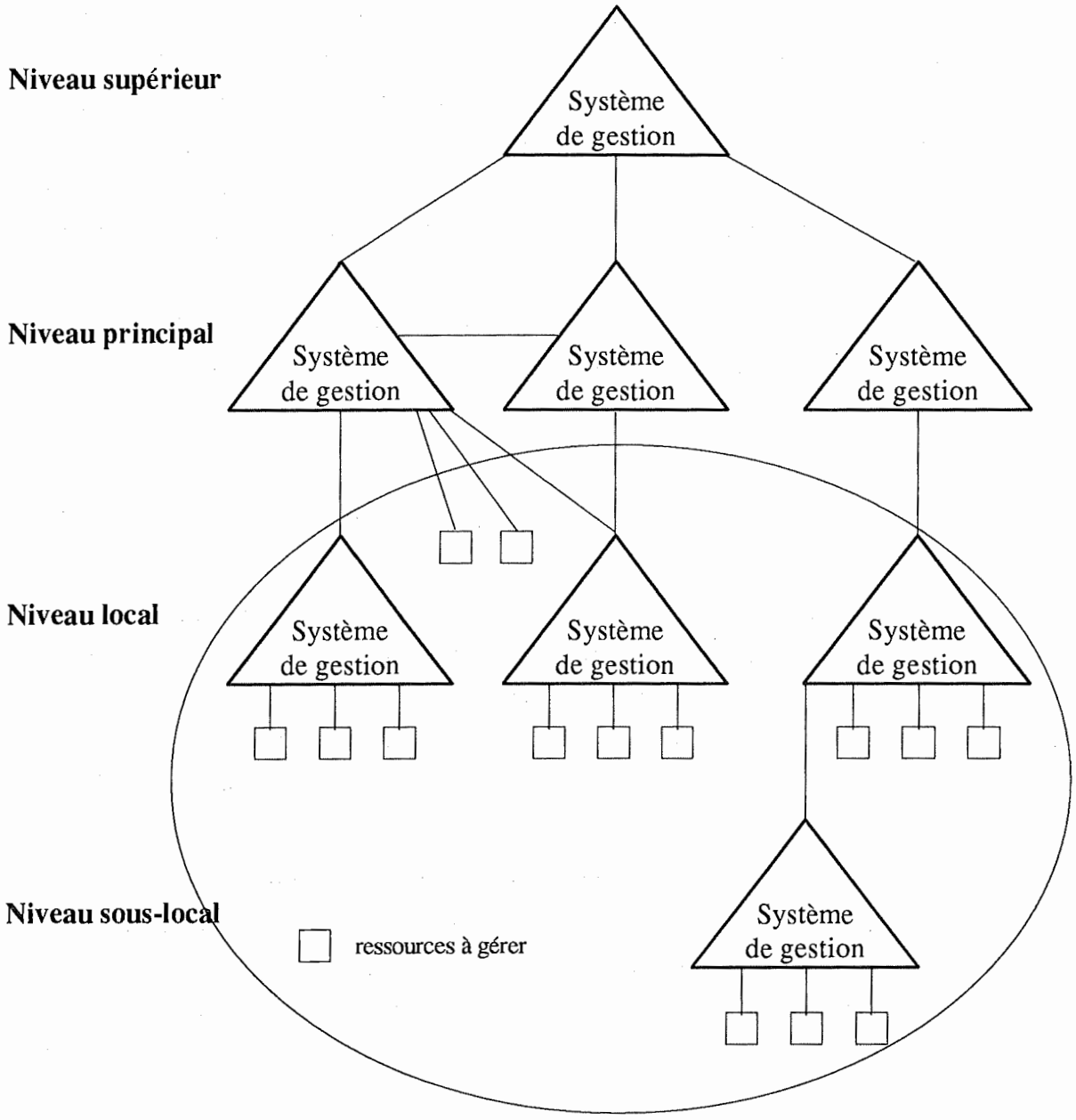


Figure 2.1 : niveaux de gestion

A un niveau inférieur d'indépendance et de coopération, au sein d'une seule entreprise, l'organisation souvent hiérarchique se retrouve dans la structure de la solution de

gestion du système. Ainsi, comme le montre la figure 2.1, différents niveaux de gestion peuvent interagir. Par exemple, trois niveaux (supérieur, principal et local) correspondraient respectivement à la direction générale, un niveau régional, un site. Le niveau local peut encore se décomposer en niveaux sous-locaux équivalents aux départements existants sur un site.

Le niveau supérieur de gestion devrait être capable d'intégrer tous les systèmes d'administration. Des constructeurs tels que AT&T avec son produit Accumaster ou IBM et NetView proposent de tels intégrateurs.

Les administrateurs du niveau principal ont souvent une tâche de surveillance des systèmes de gestion des niveaux locaux. Ils exécutent également des activités de gestion en rapport avec des éléments de plus haut niveau du système distribué, ayant des fonctions entre les différents sites locaux.

Les systèmes de gestion aux niveaux local et sous-local sont responsables de l'administration plus fine des composants du système distribué au niveau local tels que les réseaux locaux.

Ces activités de gestion peuvent être comparées en parallèle à trois niveaux de décisions : au niveau supérieur, les décisions stratégiques ou à long-terme allant de un jour à un mois; les décisions tactiques ou à court terme caractérisant le niveau principal - de une heure à un jour - et, finalement, les décisions immédiates ou opérationnelles, situées au niveau local - de une minute à une heure [MILLE91].

2.5.2 Structure physique

Comme nous avons eu l'occasion de le voir dans le chapitre 1, les systèmes distribués sont basés sur des réseaux interconnectés - LAN, MAN ou WAN. [SLOMA90] propose la contrainte physique et géographique comme second type de structure pouvant influencer le partitionnement des responsabilités de gestion. Des fonctionnalités tels que la gestion des fautes tirent pleinement parti d'une pareille structure.

Strutt parle d' "îlots de gestion" au sein d'un même réseau. Chacun de ceux-ci ayant la responsabilité d'un ensemble de composants indépendamment des autres. Toutefois des aspects du système sont partagés. C'est, pour un LAN, les moyens de communication tels que le câblage ou des éléments comme les "bridges". Pour les WAN, ce sont les connexions

entre sites. Il y a donc toujours "un besoin pour les administrateurs de communiquer" pour interagir dans le domaine de la gestion [STRUT91].

2.5.3 Structure des services

Nous avons vu qu'une des caractéristiques des systèmes distribués est leur modularité en terme de service. [SLOMA90] reprend cette idée pour introduire un partitionnement de la gestion en fonction d'une structure liée aux services. Ces services - communication, serveur de fichiers, courrier électronique, etc. - peuvent avoir des administrateurs différents, chacun spécialisé dans le cadre de son service.

2.5.4 Structure liée à la sécurité

La dernière structure à partir de laquelle Sloman propose un partitionnement des systèmes distribués se base sur les frontières liées à la sécurité des différents composants du système. Ces limites établissent des ensembles de ressources auxquelles peut accéder une classe particulière d'utilisateurs.

2.6 Modèle de gestion

Ce qui suit va nous permettre de cerner quels sont les différents éléments entrant dans la composition d'un modèle de gestion de systèmes distribués. Ces composants, repris dans la figure 2.2, sont un système administrateur échangeant des messages qui permettent des interactions avec le système géré pour accéder aux objets gérés. Ces différents éléments comme nous le verrons peuvent être regroupés logiquement dans des domaines de gestion.

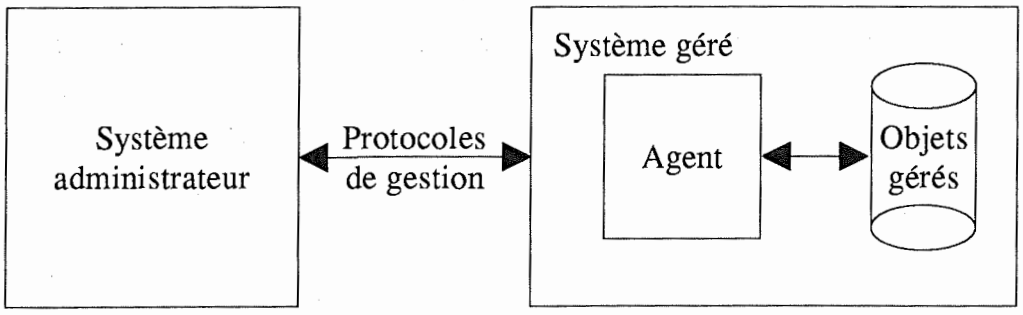


Figure 2.2 : modèle de gestion

2.6.1 Système administrateur

Le système administrateur, comme nous le dit [CELIA90], consiste en une combinaison d'éléments humain, software et hardware. Les administrateurs du système constituent l'élément humain, les parties hardware et software sont les solutions qui offrent les outils de gestion.

[SLOMA90] propose une différenciation entre quatre types d'éléments au sein du système administrateur : l'administrateur humain, les entités de présentation, les fonctions de gestion et les entités de transformation. La figure 2.3 illustre ces distinctions.

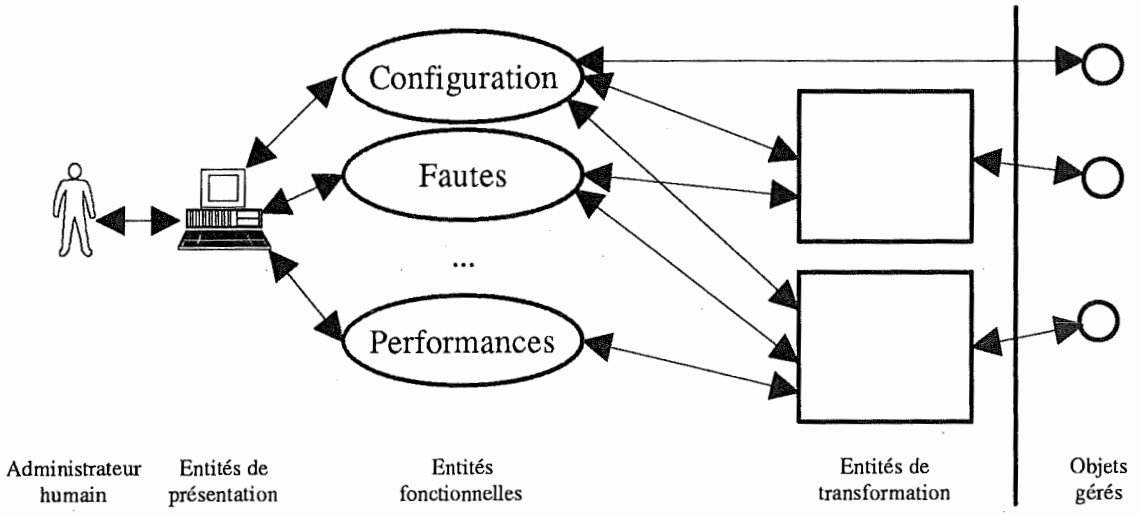


Figure 2.3 : les types d'éléments d'un système administrateur

- L'**administrateur humain** est la personne qui, selon une politique de gestion, utilise le système administrateur.

- Les **entités de présentation** fournissent une interface à l'administrateur pour accéder aux fonctions de gestion. Si cette interface a tendance à être de plus en plus graphique pour assurer un certain confort d'utilisation, elle peut cependant être textuelle.

- Les **entités fonctionnelles** sont les solutions software qui correspondent aux fonctions de gestion de configuration, des fautes, des performances, de la sécurité et de la comptabilité présentées au point 2.4.

- Les **entités de transformation** permettent à un système administrateur de dialoguer avec des objets gérés selon une interface de communication commune. Cette dernière est ce que l'OSI/NM Forum [NMFOR90] appelle l'interface d'interopérabilité. Bull dans son produit de gestion intégrée de systèmes ISM/Manager nomme ces entités sous le terme d'agents intégrateurs [MILLE91]. Nous aborderons ces deux notions dans les chapitres suivant.

2.6.2 Système géré

Le système géré consiste en un agent et des objets gérés. Les agents sont, en quelque sorte, des extensions du système administré dans les systèmes gérés [POTTE91].

Les agents exécutent les opérations de gestion demandées par le système gérant. Ces opérations portent sur les objets gérés. Lesquels peuvent émettre des notifications qui seront relayées par l'agent vers l'administrateur.

Un agent peut être également considéré comme une entité de transformation.Un système administrateur effectue des opérations de gestion sur des objets appartenant à un système de gestion auquel il ne peut accéder directement. Un agent sur un système intermédiaire sert de relais pour ces requêtes. Il traduit alors les messages dans un format compréhensible par le système cible. Dans un même ordre d'idée, la réponse à cette requête suit le chemin inverse.

Le second composant des systèmes gérés est ce qu'il est convenu d'appeler la Management Information Base ou en plus court MIB. Cette MIB est un ensemble d'objets gérés. Ce sont ces objets que nous allons aborder maintenant.

2.6.3 Objets gérés

L'approche générale à adopter en terme d'information de gestion est une approche orientée-objet. Un objet géré est la représentation d'une ressource réelle. Cette **abstraction** est la vue administrative que l'on peut avoir d'une ressource du système distribué. Dans le cas d'un gateway, par exemple, un objet le représentant est un ensemble de tables de routage. Les ressources peuvent être représentées par plusieurs objets. Le même exemple du gateway peut l'illustrer. Le gateway est un ordinateur qui est capable d'offrir d'autres services que le routage. Ces services sont eux-mêmes représentés par des objets de gestion.

Chaque objet est défini par des attributs. De plus, il peut accepter des opérations qui lui seront appliquées et envoyer des notifications lors de certains événements. Nous verrons plus loin, au point 3.4.3, la norme adoptée par l'OSI en matière de définition d'objets de gestion.

En fait, l'objet géré est une convention concernant la connaissance d'une information de gestion entre le système géré et le système gérant.En regard de cette connaissance de gestion que les systèmes administrateurs effectuent des opérations de gestion. Ces opérations sont de différents types. Elles portent soit sur les attributs d'un objet, soit sur l'objet lui-même. Pour les premières, il s'agira de prendre connaissance de leurs valeurs ou de modifier celles-ci. Pour les secondes, l'opération agira en tant que créateur ou destructeur d'objets. [SLOMA90] explique ces opérations en terme d'interactions avec les objets de gestion. Nous reviendrons sur ces notions dans le chapitre portant sur la gestion OSI puisque celle-ci se base également sur la notion d'objet.

Afin de transmettre les opérations de gestion venant du gestionnaire et de leur retourner les réponses provenant des systèmes gérés, le système de gestion met en œuvre des protocoles de gestion.

2.6.4 Protocoles

Afin d'accéder à distance à l'information de gestion sise dans les systèmes gérés, les systèmes gérants doivent disposer d'un moyen de communication transportant leurs requêtes. Cela s'applique également pour l'envoi des notifications dans le sens machines gérées - administrateur. Les protocoles décrivent les procédures qui permettent le transfert de l'information de gestion entre l'agent et l'administrateur. L'ISO, dans [ISO7498], définit un protocole comme étant un "ensemble de règles et de formes - sémantiques et syntaxiques - déterminant les caractéristiques de communications des entités (N) lorsqu'elles effectuent des fonctions (N)". Les entités (N), dans le cas qui nous intéresse, sont des entités de gestion situées dans une couche équivalente au niveau application de l'architecture OSI. Les fonctions sont les opérations de gestion que veulent effectuer ces entités ou les notifications qu'elles veulent envoyer.

A côté des protocoles liés à des constructeurs tels que le Protocole d'Echange Administratif propre à l'architecture DSAC - DSAC/AEP - de Bull, deux protocoles émergent en temps que standards. Le premier, standard *de facto*, est le Simple Network Management Protocol - SNMP - [RFC1157] qui est lié à l'environnement Internet TCP/IP.

Le second, standard *de jure* quant à lui, est le Common Management Information Protocol - CMIP - proposé par l'ISO [ISO 9596].

Nous aborderons plus en profondeur, dans le chapitre consacré à l'OSI, les concepts relatifs à CMIP.

2.6.5 Domaines

Nous avons déjà eu l'occasion de parler de la nécessité de partitionner les systèmes distribués en vue de décomplexifier leur gestion. [ROBIN88] a introduit la notion de domaine comme moyen à la gestion de systèmes. Cette notion implique que l'on ne considère plus les objets individuellement mais plutôt par groupes correspondant à une politique de gestion commune. Ce sont ces groupes que l'on appelle domaines. [SLOMA89] décrit ce concept.

Un domaine est donc constitué d'un ensemble d'objets sur lesquels l'administrateur peut exécuter des opérations de gestion afin de les contrôler. Ces opérations sont spécifiées par des règles d'accès. Pour qu'un objet soit gérable, il doit se trouver à l'intérieur d'un domaine. Un domaine étant lui-même considéré comme un objet, il peut être membre d'un autre domaine. Il est alors un sous-domaine [MOFFE91].

[SLOMA90] donne comme exemples de domaines :

- Des stations de travail connectées à un réseau local sous la responsabilité d'un administrateur responsable de sa maintenance.

- Un sous-ensemble de ces stations gérées par un scheduler leur attribuant des travaux lorsqu'elles sont inoccupées.

- Un ensemble de fichiers dans un répertoire gérés par le propriétaire de ce répertoire.

- Un ensemble d'objets entrant dans le cadre de la gestion de la sécurité.

Dans un même ordre d'idées, [STRUT91] définit un domaine par "la représentation d'une sphère d'intérêts à propos d'un ensemble d'objets gérés de la part d'un administrateur".

Comme il nous le fait remarquer, la notion de domaine peut s'appliquer à différents points de vue - regroupement d'objets de gestion pour des raisons de sécurité, pour des raisons de politiques communes les concernant, etc. - tous ayant le même objectif : le partionnement, le "diviser pour mieux régner".

[MOFFE91] explique également quelles sont les opérations qu'il juge particulièrement importantes pour la compréhension des domaines. Il s'agit de la création et de la destruction d'un objet ainsi que l'inclusion et le retrait d'un objet dans un domaine. Il faut noter qu'un objet doit pouvoir exister dans plusieurs domaines de gestion. L'inclusion et le retrait d'objets dans un domaine ne change en rien les relations qu'ils peuvent avoir avec d'autres domaines.

[SLOMA90] ajoute une autre opération à celles précitées. Il s'agit de pouvoir obtenir la liste des objets associés à un domaine.

Comme nous l'avons déjà expliqué, une des conséquences du partitionnement d'un système distribué est l'établissement de frontières de responsabilités. Celles-ci doivent être réglées par des droits d'accès aux ressources ainsi regroupées. Comme la notion de domaine de gestion établit elle aussi des aires de responsabilité pour les gestionnaires du système, il est important d'établir des règles d'accès à ces domaines.

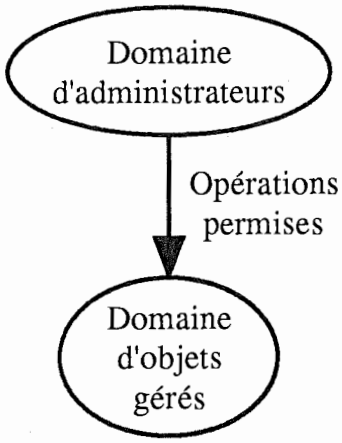


Figure 2.4 : règles d'accès

[SLOMA90] définit l'objet des règles d'accès comme étant la spécification de "la politique de contrôle de l'accès en termes des opérations que peuvent exécuter un ensemble d'utilisateurs sur un ensemble d'objets". Comme le montre la figure 2.4, il s'agit de définir les

relations entre un domaine gérant et un domaine d'objets gérés, cela au travers des opérations qui sont permises.

L'OSI parle également de cette notion de domaine. Dans sa perspective organisationnelle, "la gestion OSI peut être distribuée administrativement à travers des domaines de gestion et des systèmes de gestion à l'intérieur d'un domaine" [KLERE88].

2.7 Administration intégrée

Afin de répondre à la prolifération des applications de gestion, les constructeurs commencent à proposer des systèmes de gestion intégrée. Cette voie correspond mieux aux besoins de gestion.

Un tel système combine par intégration dans un même environnement les différentes fonctionnalités de gestion pouvant provenir de divers fournisseurs. Les utilisateurs peuvent visualiser l'état du système géré à partir d'une interface utilisateur commune à toutes ces fonctionnalités.

Un système de gestion intégrée est utile dans le cas d'une administration hiérarchisée telle qu'elle fut présentée au point 2.5. Il devient alors, en quelque sorte, un gestionnaire des gestionnaires.

"Le résultat final de la gestion intégrée peut être la réduction des niveaux de qualifications requis pour les administrateurs, la réduction des erreurs dans la gestion de multiples réseaux et un accroissement de la flexibilité de la gestion" [CELIA90].

2.7.1 Intégrations

L'intégration en question se situe à quatre niveaux : l'interface utilisateur, les fonctions de gestion, les objets de gestion et les protocoles de gestion [CELIA90].

- L'interface utilisateur doit être unique. Dans ce sens, elle doit permettre d'accéder à l'ensemble des fonctionnalités offertes par le système de gestion. Il est préférable que cette interface soit graphique pour offrir à l'utilisateur une certaine facilité d'emploi. L'état de l'art en matière d'écrans graphiques

s'appuyant sur un système de fenêtres est offert par des solutions telles que OSF/Motif.

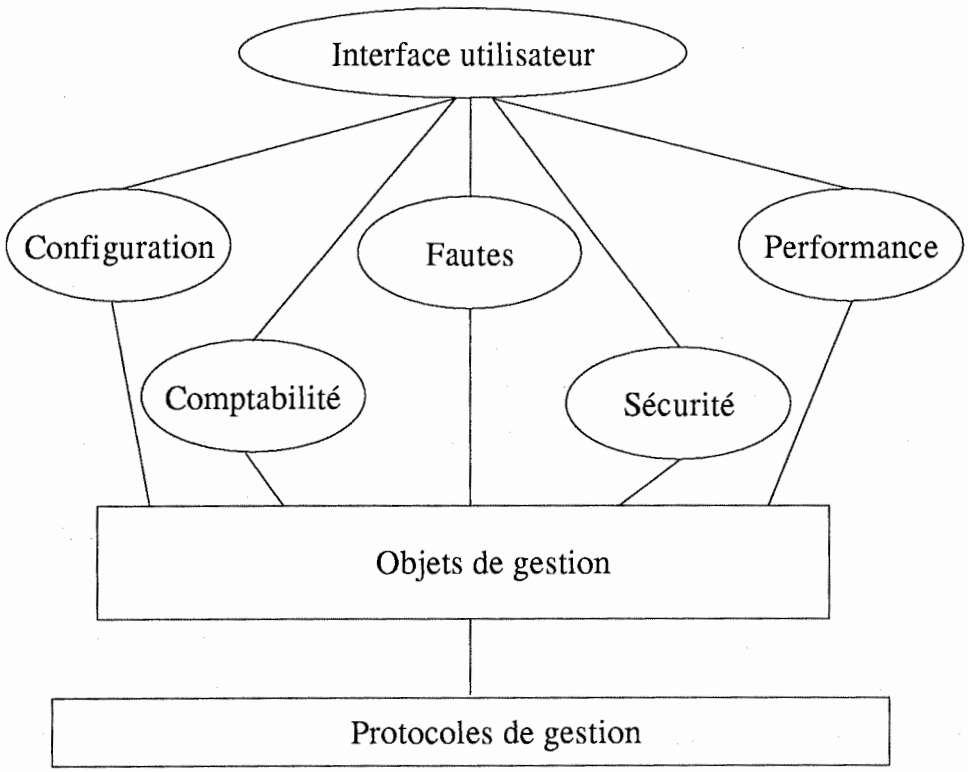


Figure 2.5 : gestion intégrée

[CARTE91] cite quatre caractéristiques importantes que doit avoir une solution de gestion sur le plan de la représentation graphique du système . Il s'agit de la capacité à représenter une carte hiérarchique du système, la capacité à représenter dans cette carte des éléments non-gérables du système, la capacité à représenter des cartes physiques aussi bien que des cartes logiques et la capacité à créer les icônes qui illustrent les éléments du système.

La capacité à représenter une carte hiérarchique du système est très importante dans la mesure où il est fort difficile - la confusion serait en effet trop grande - de représenter tous les éléments du système sur un seul écran. La solution est de hiérarchiser cet ensemble d'éléments. Cela permet en outre d'obtenir des vues correspondant à des niveaux ou des branches différents du réseau.

Lorsque la carte représente non seulement des éléments gérés mais également des éléments du système qui ne le sont pas, comme le dit [CARTE91],

"l'administrateur est capable de déterminer de visu tous les éléments sur le chemin d'une erreur. Utilisant cette information, il est plus aisé d'isoler la source d'un problème".

Non seulement la représentation logique du système est importante, mais la représentation physique revêt également une part importante dans les facilités offertes à l'administrateur. En effet, dans le cas de grands systèmes, le souhait peut être émis de situer rapidement un élément défectueux. Une carte représentant graphiquement le plan d'un bâtiment par étage et les divers éléments du système distribué répartis sur cet étage, par exemple, est le bienvenu pour accélérer le temps d'intervention et ainsi diminuer la mise hors service des éléments problématiques.

Pour représenter les cartes ci-dessus, il paraît intéressant d'utiliser des librairies d'icônes. Ces icônes permettent de visualiser de manière uniforme les différents types d'éléments au sein de ces cartes. Lors de l'introduction de nouveaux éléments au sein du système, l'administrateur disposant de ces librairies pourra choisir le dessin lui convenant. Si aucun ne correspond à l'élément, un outil de création d'icônes lui permettra de la dessiner lui-même.

- Les applications de gestion doivent fournir des outils couvrant les cinq aires fonctionnelles décrites plus haut au point 2.4, la gestion de la configuration, la gestion des fautes, la gestion des performances, la gestion de la comptabilité et la gestion de la sécurité. L'utilisation de ces outils doit se baser sur une même méthode. Passer d'un outil à un autre ne doit pas donner l'impression à l'utilisateur qu'il change d'environnement. L'utilisation d'un vocabulaire unique commun à tous les types de fonctions de gestion peut y contribuer.

- En ce qui concerne les objets de gestion, il est important que leur définition, leur structure, leur mécanisme de nommage soient cohérents entre eux. Il ne faut pas oublier qu'un système de gestion intégrée peut être la solution de gestion d'un niveau supérieur dans la structure hiérarchique du système distribué, dès lors, il est amené à gérer des objets appartenant à des systèmes gérés différents. Il faut donc employer un schéma de nommage commun de manière à pouvoir accéder à un objet, sans ambiguïté, où qu'il soit dans le système.

Les définitions des objets et de leurs relations entre eux peuvent varier d'un système à un autre. L'intégrateur fournit alors une traduction automatique entre les objets gérés des différents systèmes. Il doit pouvoir maintenir également les relations entre les objets.

- Un système de gestion dans le rôle d'intégrateur devra dialoguer avec d'autres systèmes de gestion ne provenant pas toujours d'un même constructeur. Utiliser des protocoles de gestion standards est une solution afin de réaliser l'intégration de l'administration [BRINS88]. En effet, la gestion n'est plus réduite par des interfaces propriétaires propres aux vendeurs. Au contraire, elle permet de fournir des solutions de gestion fonctionnant dans un environnement hétérogène.

2.7.2 Différentes approches

Trois approches principales fournissent l'intégration des outils de gestion. Il s'agit de l'approche par intégrateur, par traducteur et par standards [NOBLO91], [CELIA90].

- Intégrateurs : cette approche consiste à ajouter un "super-système" afin d'intégrer différents outils d'administration. Accumaster d'AT&T est dans cette ligne de produits.

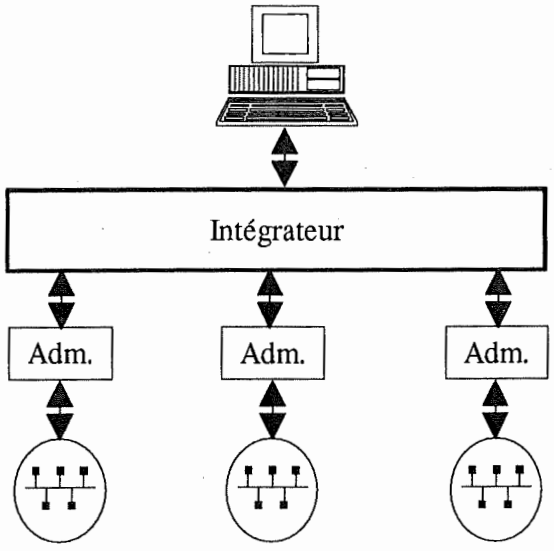


Figure 2.6 : approche par intégrateur

- Traducteurs : dans cette approche, des systèmes de gestion traduisent les informations et les fonctions de gestion dans un format compréhensible par un système d'administration propriétaire. Celui-ci peut donc gérer les premiers en plus de son propre système.C'est le cas d'un produit tel que NetView d'IBM.

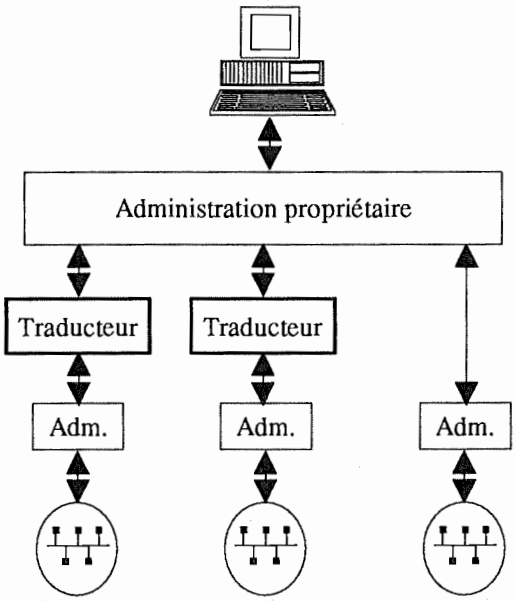


Figure 2.7 : approche par traducteur

- Standards : l'approche "standards" se base sur les normes afin que les différentes administrations puissent dialoguer entre elles. Des produits se conformant aux propositions faites par l'OSI/NM Forum par exemple offrent cette possibilité d'interopérabilité.

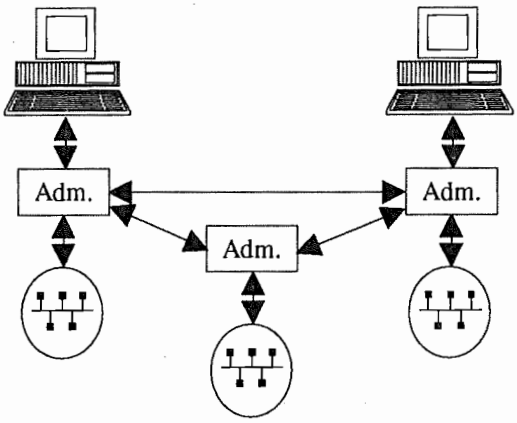


Figure 2.8 approche par les standards

2.8 Conclusion

De plus en plus, l'exigence des utilisateurs de réseaux et de systèmes distribués va se porter sur la gestion de ceux-ci. La large gamme des produits qui les composent occasionne une nécessité de connaissance de plus en plus étendue des problèmes qui peuvent en empêcher un fonctionnement adéquat.

Une qualification accrue, beaucoup plus étendue qu'auparavant, entraîne en termes de coûts salariaux et de fonctionnement une charge non négligeable. L'intégration des services est une réponse à un rabotement de ces coûts. De plus, elle permet d'avoir une vue cohérente du système et de ses composants en matière d'administration.

Les concepteurs de systèmes de gestion - AT&T, DEC, Bull, HP, IBM pour ne citer qu'eux - l'ont compris. Les offres de produits allant dans ce sens apparaissent de plus en plus sur le marché. Mais un problème demeure si l'hétérogénéité de la gestion n'est pas prise en compte dans la conception d'un tel produit : l'interopérabilité entre ces différents systèmes. Les standards internationaux se trouvent au carrefour de ces exigences.

Chapitre 3

La gestion OSI

Chapitre 3 : La gestion OSI

3.1 Introduction

Avec le chapitre 1, nous avons eu l'occasion d'introduire les notions se rapportant au modèle OSI et à la structuration en couches normalisée par l'organisme international. La dernière de ces couches est la couche application. C'est par elle que l'utilisateur d'un système "s'introduit" dans le monde OSI. Elle revêt une importance toute particulière dans le cadre de la gestion OSI puisque c'est à ce niveau que se situe ce que nous appellerons plus loin la gestion de systèmes qui permet d'administrer l'ensemble d'un système OSI. [BOUNE90] nous dit même qu' "il est primordial de comprendre le rôle de la couche application lorsque l'on souhaite utiliser l'OSI pour offrir des services dans un environnement distribué et ouvert."

Nous aborderons donc tout d'abord cette présentation de la gestion OSI par un aperçu de la couche application. Nous insisterons spécialement sur les points essentiels pour la suite, à savoir la gestion OSI en elle-même.

3.2 La couche application

Lors de l'élaboration de cette couche, l'OSI a retenu la nécessité de faire ressortir dans son architecture à la fois des éléments communs et d'autres plus spécifiques. Les premiers rassemblent les fonctions nécessaires à toute application. C'est ce que l'on appelle les éléments de services d'application communs - CASE, Common Application Service Element. Les seconds sont les SASE - Specific Application Service Element. Ils sont inévitables du fait de la diversité des applications implémentées dans un système. La norme [ISO9545] consacrée à la structure de la couche application - ALS, Application Layer Structure - détermine quelles sont les fonctions communes.

3.2.1 Processus d'application et entités de gestion

Les concepts de base définis pour la couche application sont le processus d'application - AP, Application Process - et la notion d'entité d'application - AE, Application Entity.

Un **processus d'application** regroupe les fonctions qui sont nécessaires à la réalisation d'une application. Ce processus est en fait l'utilisateur le plus élevé du modèle OSI.

Tout dans une application ne concerne pas la communication entre systèmes. Or, c'est justement cette facette des applications qui intéresse l'OSI. L'organisme international de normalisation a donc défini une notion supplémementaire, celle d'**entité d'application**. Il s'agit de la partie des processus d'application dédiée aux communications entre systèmes. C'est entre AE que ces communications s'établissent. Comme nous le montre la figure 3.1, il peut y avoir plus d'une AE par AP.

L'AE est en quelque sorte l'intersection entre les applications et le monde OSI - voir figure 3.1.

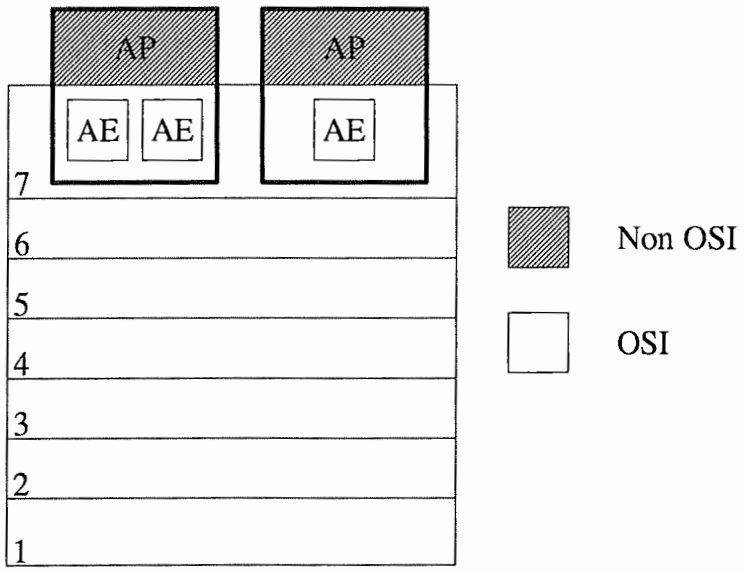


Figure 3.1 : processus d'application, entités d'application et le monde OSI

3.2.2 Eléments de services d'application

Pour pouvoir dialoguer entre elles, les entités d'application font appel aux éléments de service d'application - ASE, Application Service Element - que nous avons introduits plus haut. C'est ainsi que les AE contiennent des ASE dont certaines sont indispensables à l'instauration d'associations entre les AE homologues.

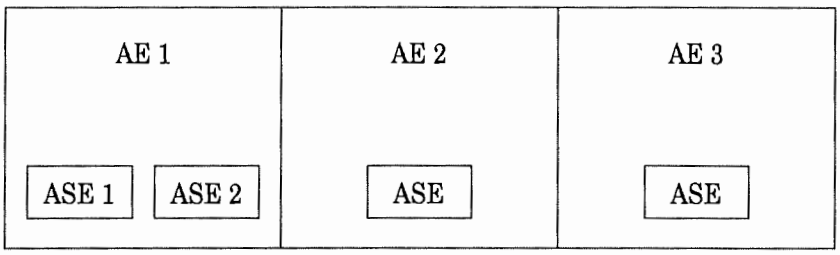


Figure 3.2 : entités d'application et éléments de service d'application.

Voici les principaux éléments de service d'application normalisés par l'ISO :

- L'**ACSE** - Association Control Service Element - [ISO8649] [ISO8650] est l'élément de service de base. En effet, il a pour but la gestion des connexions. Il permet d'établir, de libérer et d'abandonner une association. Les quatre services offerts par ACSE sont :

  - A-ASSOCIATE pour établir une association.

  - A-RELEASE afin de la libérer.

  - A-U-ABORT dans le cas d'un abandon non ordonné de la part de l'utilisateur du service.

  - A-P-ABORT dans le cas où c'est l'ACSE pourvoyeur du service lui-même qui abandonne l'association.

- **CCRSE** - Commitment Concurrency and Recovery Service Element - qui permet de maintenir dans un état cohérent les bases de données réparties ou dupliquées.

- **RTSE** - Reliable Transfer Service Element - permet le transfert fiable d'unités de données du protocole d'application. Les autres ASE utilisent cet ASE pour assurer des transferts complets et sécurisés.

- **ROSE** - Remote Operation Service Element - [ISO9072-1] [ISO9072-2] qui a pour but de permettre l'exécution d'opérations sur un site distant. Les services offerts par ROSE sont :

  - RO-INVOKE pour la demande de l'exécution,

  - RO-RESULT pour la réponse de résultat positif,

  - RO-ERROR pour une réponse de résultat négatif,

  - RO-REJECT-U, rejet par l'utilisateur de la requête ou de la réponse,

  - RO-REJECT-P, rejet de la requête ou de la réponse de la part de l'élément de service commun ROSE pourvoyeur du service.

- **MHS** - Message Handling Systems - qui offre des services qui permettent d'effectuer de la messagerie électronique en mode non connecté.

- **DS** - Directory Service -, un annuaire qui permet de répertorier les équipements et les éléments adressables.

- **FTAM** - File Transfer, Access and Management - qui offre les fonctionnalités pour le transfert de fichiers ainsi que leur manipulation à distance.

- **VT** - Virtual Terminal - pour une présentation normalisée d'un terminal connecté au réseau.

Les autres ASE sont **DTAM** - Document Transfer, Access and Management -, **ODA** - Office Document Architecture -, **ODIF** - Office Document Interchange Format -, **JTM** - Job Transfer and Manipulation -, **RDA** - Remote Database Access -, **MMS** - Manufacturing Messaging Service.

3.2.3 Objet d'association unique

Comme nous l'avons déjà dit, une entité d'application peut contenir plusieurs éléments de service d'application. Leurs activités sont coordonnées dans le cadre d'associations avec d'autres ASE distants.

Dans le cas d'une instance unique de communication, l'ISO a défini deux concepts pour la gestion de cette coordination. Il s'agit de la notion d'objet d'association unique - SAO, Single Association Object - et de SACF - Single Association Control Function.

Un SAO contrôle la communication entre deux ASE durant tout le temps d'existence de l'association. Il est composé d'ASE qui, comme le dit [BOUNE90], "peuvent représenter n'importe quelle combinaison des ASEs que peut fournir l'entité d'application qui utilise une instance de ce SAO".

Dans le point précédent, nous insistions sur l'importance d'ACSE dans la gestion des associations. C'est pour cette raison que cet élément de service est présent dans chacun des SAO. Grâce à lui, deux SAO distants pourront communiquer entre eux.

Le fonctionnement des différentes ASE à l'intérieur d'un SAO est soumis à un séquencement déterminé par une fonction SACF. La figure 3.3 propose une représentation des différents composants d'un objet d'association unique.

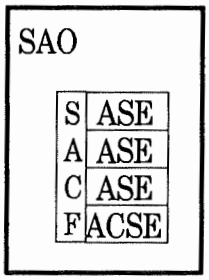


Figure 3.3 : objet d'association unique

3.2.4 Associations multiples

Dans une entité d'application, plusieurs SAO peuvent coexister et collaborer entre-eux dans l'exécution d'une tâche. Afin de coordonner cette coopération, une fonction MACF - Multiple Association Control Function - gère ces SAO. La figure 3.4 illustre les différentes parties d'une entité d'application en tenant compte de cette MACF.

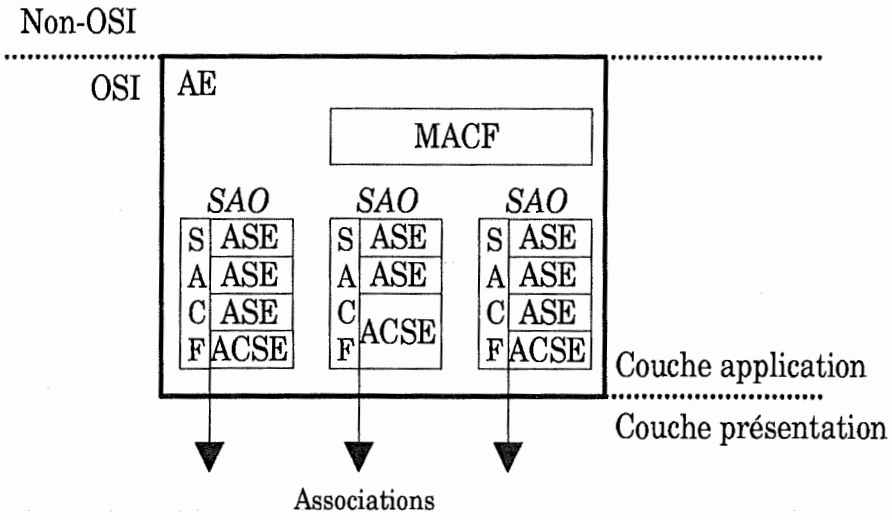


Figure 3.4 : structure interne d'une entité d'application.

3.3 Cadre architectural de la gestion OSI

A partir du modèle de référence de base, l'OSI élabore l'ensemble des normes relatives aux différents domaines des systèmes ouverts. C'est le cas de la norme concernant la gestion OSI [ISO7498-4]. Elle porte le nom de "Cadre architectural pour la gestion OSI". Ce document sert de base pour l'élaboration de toutes les normes relatives à la gestion OSI. L'ISO ne s'intéressant qu'aux moyens de communiquer entre deux systèmes ouverts, seuls les aspects reprenant des échanges protocolaires sont traités.

3.3.1 Les niveaux de gestion OSI

L'OSI a défini trois niveaux de gestion pour lesquels il y a des échanges d'informations. Les trois types d'échanges administratifs sont la gestion de systèmes, la gestion de couche (N) et l'opération de couche (N). Ce sont eux que nous verrons ci-après.

Dans chacun de ces niveaux de gestion, les échanges s'effectuent entre des entités homologues dans des systèmes ouverts distants.

3.3.1.1 La gestion de systèmes

La gestion de systèmes est la méthode conseillée et privilégiée pour échanger de l'information de gestion entre des systèmes ouverts. Elle permet de gérer l'ensemble des couches du modèle OSI. Pour cette raison, il est indispensable que les sept couches soient opérationnelles. Les échanges de ce niveau de gestion s'effectuent, dans la couche application, entre entités d'application pour la gestion-système - SMAE, System Management Application Entity. Ces échanges se font grâce aux protocoles d'application spécifiques - SMAP, System Management Application Protocol.

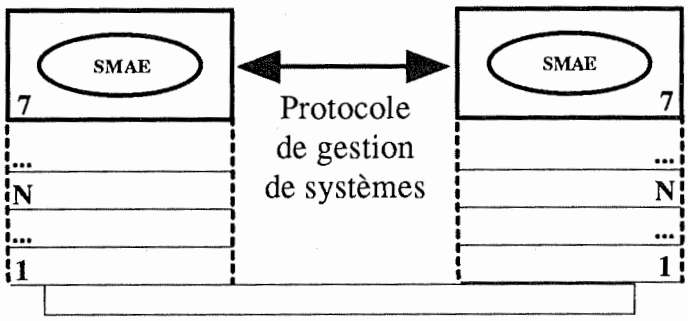


Figure 3.5 : la gestion de systèmes

3.3.1.2 La gestion de couche (N)

Le second niveau de gestion OSI est la gestion de couche (N) - (N) LM, (N) Layer Management. Il représente les échanges nécessaires à la gestion d'une couche particulière d'un système ouvert. La gestion de couche (N) permet le contrôle de plusieurs échanges de données sur plusieurs instances de communications. Pour assurer les échanges de gestion d'une couche (N), des entités de gestion de la couche (N) sur des systèmes ouverts distants mettent en œuvre un protocole spécifique pour la gestion. NCMS - Network Connection Management Subprotocol - est un exemple de protocoles de gestion de la couche transport.

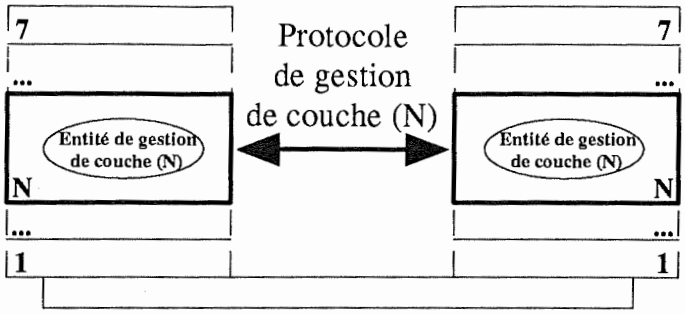


Figure 3.6 : gestion de couche (N)

3.3.1.3 L'opération de couche (N)

Les opérations de couche (N) - (N) Layer Operation - représentent le niveau d'échange d'informations de gestion le plus bas. Ces données administratives concernent les ressources de la couche (N). Mais, à la différence de la gestion de couche (N), l'opération de couche (N) porte sur une seule instance de communication. Les opérations de ce niveau de gestion ne sont pas transportées par des protocoles spécifiques mais bien par les protocoles normaux associés aux couches. C'est ainsi que le protocole X.25 permet de véhiculer des informations comptables telles que la tarification.

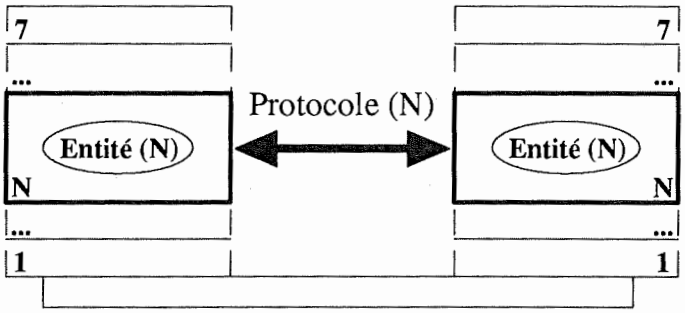


Figure 3.7 : opération de couche (N)

Nous allons par la suite nous attarder au niveau de gestion qui nous intéresse le plus, à savoir la gestion de systèmes. Elle est d'ailleurs la seule à avoir des normes qui lui sont relatives. Mais auparavant, nous allons aborder des concepts communs aux trois types de gestion présentés. Il s'agit de la notion d'aires spécifiques et de connaissance de gestion. La première a déjà été quelque peu abordée lors de la précédente présentation de l'administration des systèmes distribués.

3.3.2 Les aires spécifiques

Comme nous l'avons déjà dit au chapitre 2 - point 2.4 -, l'OSI a défini une série de cinq types d'opérations de gestion, les SMFA - System Management Functional Area. Ces SMFA sont la gestion des fautes, la gestion de la configuration, la gestion des performances, la gestion de la comptabilité, et la gestion de la sécurité.

3.3.2.1 La gestion des fautes

La gestion des fautes fournit les fonctions qui permettent à l'administrateur de détecter des problèmes dans le réseau et dans l'environnement OSI. Elles permettent aussi d'identifier et, dans la mesure du possible, de corriger les erreurs rencontrées. A cette fin, la gestion des fautes fournit les moyens aux systèmes gérés d'émettre des notifications d'erreurs à son administrateur. Les fautes notifiées sont reprises dans un journal qui pourra être examiné et manipulé. Des tests de diagnostics peuvent par ailleurs être effectués.

3.3.2.2 La gestion de la configuration

Les fonctions de gestion de la configuration permettent d'exercer un contrôle sur la configuration du système. Elles fournissent les capacités pour recueillir des informations concernant les systèmes ouverts, fournir des données à ceux-ci afin de préparer l'initialisation, le lancement, le maintien et l'arrêt des services d'interconnexion. Afin de remplir ces capacités, elles permettent de positionner les paramètres du système ouvert pour une utilisation courante, d'initialiser et verrouiller les objets gérés, de collecter les données d'état du système ouvert, de recevoir des notifications à propos de changements concernant ces états, de modifier sa configuration et d'associer des noms aux objets gérés. La plupart des fonctionnalités offertes par cette aire fonctionnelle sont mises à la disposition des autres tels que la gestion des fautes.

3.3.2.3 La gestion des performances

La gestion des performances offre à l'administrateur la capacité de suivre le comportement des objets gérés représentant des ressources du système ainsi que l'efficacité des activités de communication. A cette fin, des fonctions permettent de collecter des données statistiques et de maintenir et examiner des journaux d'états du système ouvert.

L'analyse des performances peut être le point de départ pour la prise de décision en matière de changement de configuration et pour le lancement de tests de diagnostics propres à la gestion des fautes.

3.3.2.4 La gestion de la comptabilité

La gestion de la comptabilité répond à l'une des préoccupations majeures de l'administrateur d'un système : connaître les coûts encourus par l'utilisation des ressources. Elle permet d'établir les limites comptables et les charges d'utilisation des ressources. Elle offre également la capacité de connaître les coûts combinés sur de multiples ressources dans le contexte d'une communication donnée.

3.3.2.5 La gestion de la sécurité

L'aire fonctionnelle relative à la sécurité offre à l'administrateur d'un système des facilités pour gérer les services offrant un accès protégé aux ressources. Dans ce but, elle permet de supporter l'authentification, de contrôler et de maintenir les facilités d'autorisation, de supporter la gestion des clés de chiffrement ainsi que le maintien et l'examen des journaux de sécurité.

3.3.2.6 Commentaire

Il est important d'éviter de croire qu'un tel découpage en aires fonctionnelles a une quelconque influence sur l'implémentation d'un système de gestion reprenant ces fonctionnalités. Ce modèle fonctionnel est purement conceptuel.

3.3.3 La base d'informations de gestion

Le dernier concept abordé par l'ISO dans le cadre architectural [ISO7498-4] est la notion de base d'informations de gestion, couramment appelée MIB pour Management Information Base.

Cette MIB reprend l'ensemble des objets de gestion accessibles de l'extérieur dans un système ouvert. Les objets de gestion représentent les ressources de communications dans l'environnement OSI.

L'implémentation interne de la MIB n'est pas normalisée par l'ISO. Le concepteur de la base d'informations peut l'organiser de manière tout à fait personnelle. La portée de la normalisation se concentre uniquement sur la connaissance des objets par un système extérieur.

La définition de cette base d'informations de gestion sera vue plus loin au point 3.4.3 qui lui est consacré.

Avant de poursuivre, nous allons voir quels sont les accès potentiels à cette MIB.

Tout d'abord, notons que deux types d'accès peuvent exister : un accès dans un environnement local et un accès à distance dans un environnement OSI.

Du point de vue local, les agents administratifs, qu'ils soient humains ou logiciels, manipulent la MIB pour la maintenir à jour.

L'accès distant, quant à lui, se fait via les protocoles correspondant aux différents niveaux de gestion que nous avons introduits plus haut. A savoir : les protocoles de gestion de systèmes, les protocoles de gestion de couche (N) et les protocoles (N) normaux.

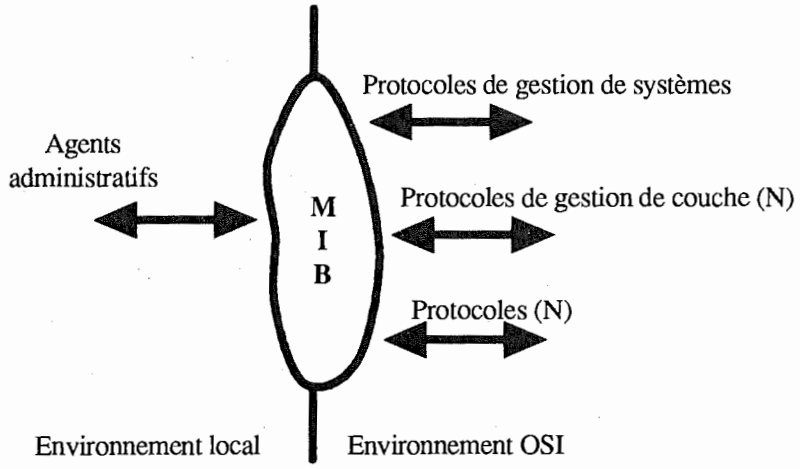


Figure 3.8 : accès à la MIB

Seuls ces derniers accès sont normalisés. L'accès local n'entre pas dans le cadre du travail de l'ISO puisqu'il ne porte pas sur la communication entre des systèmes ouverts distants.

3.4 La gestion de systèmes

3.4.1 Introduction

Nous allons maintenant porter notre attention sur la gestion de systèmes telle que nous l'avons introduite au point 3.3.1.1.

Les normes de l'ISO [ISO9595] et [ISO9596] portent sur la gestion de systèmes. Elles couvrent les aspects de communication entre systèmes distants en ce qui concerne des problèmes communs de gestion. Les normes [ISO 10165-x] concernent quant à elles les fonctions spécifiques de gestion.

Ce sont ces matières qui seront le fil conducteur des pages qui vont suivre.

3.4.2 Architecture

Dans le cadre de la gestion OSI, des échanges d'informations s'effectuent entre un système gérant - que nous appellerons indifférement système administrateur - et un système géré. Ces échanges administratifs se font au sein de ce qu'il est convenu d'appeler un domaine de gestion.

Un domaine de gestion peut être composé par un ensemble de systèmes gérants communiquant soit avec un nouveau domaine de gestion, soit avec des systèmes gérés. Ces derniers se décomposent en objets gérés. La figure 3.9 résume ce modèle organisationnel de la gestion OSI.

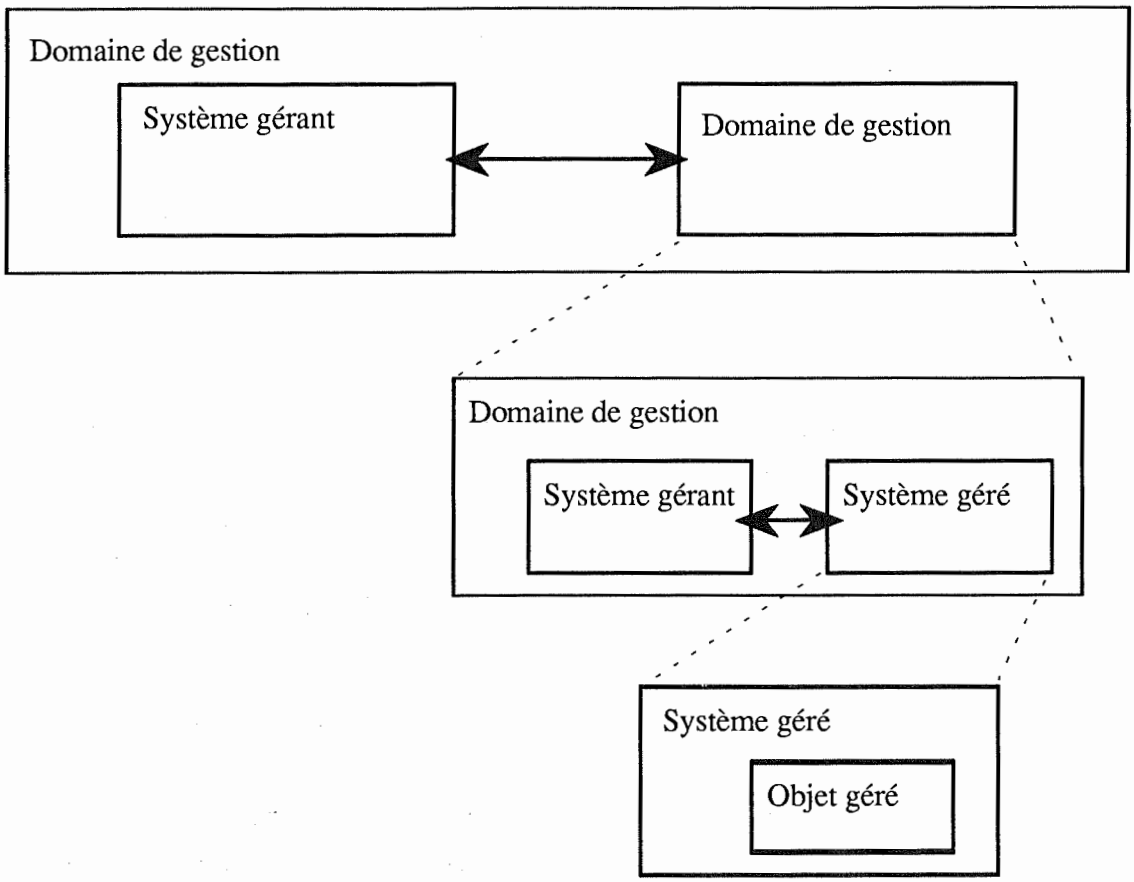


figure 3.9 : domaine de gestion

Dans le domaine de gestion, un processus d'application de gestion échange des informations de gestion avec les processus homologues des systèmes gérés. Ces processus sont appelés des MIS-users parce qu'ils utilisent les services du MIS - Management Information Service - que nous verrons plus loin au point 3.4.4.1. L'un des deux MIS-users, en l'occurence celui qui se trouve dans le système gérant, a le rôle d'administrateur. L'autre, situé sur le système géré a un rôle d'agent. La figure schématise l'architecture de la gestion OSI reprenant les notions de système gérant et système géré appelés ici MIS-users.

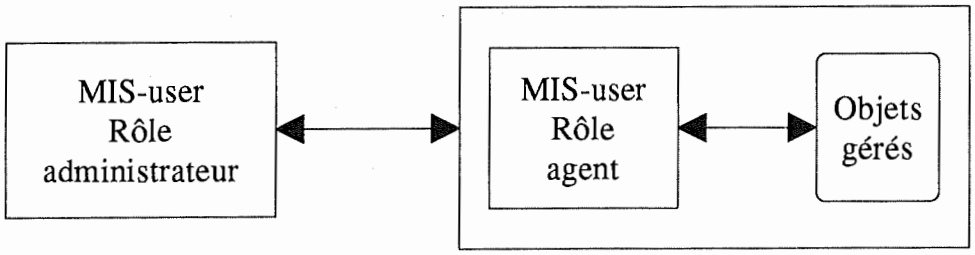


Figure 3.10 : administrateur et agent

Notons que cette répartition des rôles n'est pas fixe et que la mécanique peut être croisée comme nous le montre la figure 3.11.

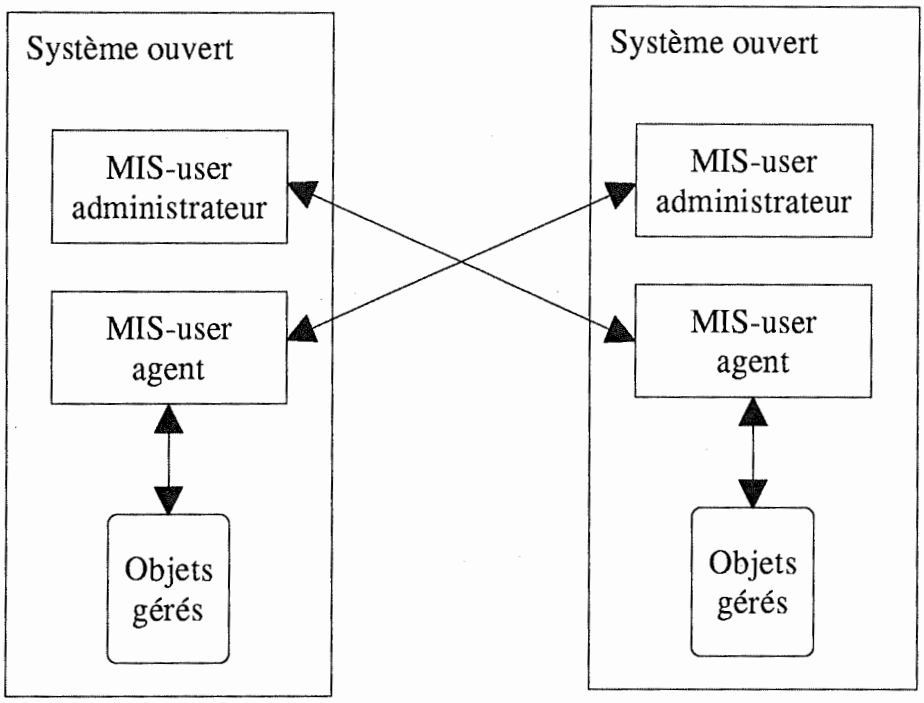


Figure 3.11 : interconnexion de rôles administrateurs et agents

Un MIS-user peut donc être à la fois administrateur et agent.

Un MIS-user dans le rôle d'agent est capable d'effectuer des opérations de gestion sur des objets gérés et d'émettre des notifications à la place des objets gérés. Ces notifications concernent des événements propres à ces objets.

Un MIS-user dans le rôle d'administrateur est, quant à lui, capable de demander des opérations de gestion et de recevoir des notifications.

Nous avons dit que les MIS-user sont des entités d'application de gestion - SMAE, System Management Application Entity. Dans ces SMAE, des éléments de services offrent des services pour la gestion de systèmes. Ce sont :

- SMASE, System Management Application Service Element.

- ACSE, Application Context Service Element.
- CMISE, Common Management Information Service Element.
- ROSE, Remote Operation Service Element.

Deux types d'échanges se situent au niveau des SMAE :

- un type d'échanges est pris en charge par l'élément de service CMISE.
- le second type d'échanges se situe à un niveau supérieur au précédent dans le sens où il lui apporte la compréhension sémantique. Ces échanges sont fournis par l'élément de service SMASE qui utilise les services offerts par CMISE.

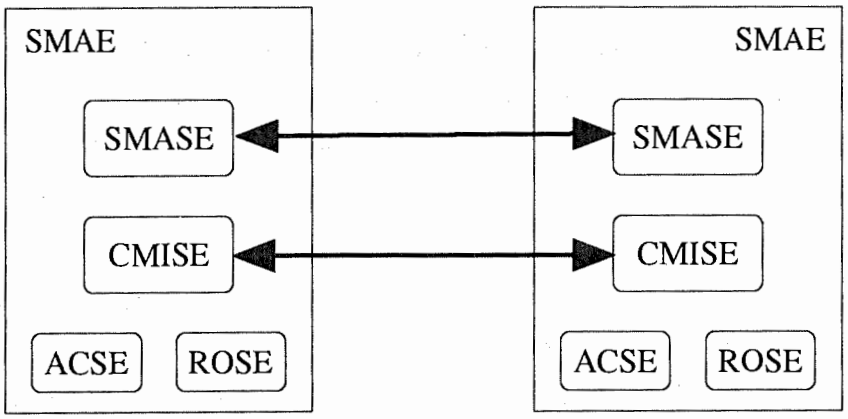


Figure 3.12 : échanges entre deux SMAE.

3.4.3 Connaissance de gestion

L'OSI a défini toute une série de normes relatives aux informations de gestion. Ces normes sont :

- [ISO 10165-1] qui reprend le modèle informationnel de gestion basé sur un modèle orienté-objet

- [ISO 10165-2] qui concerne entre autres la définition des classes d'objets de gestion utilisées par les fonctions de gestion de système que nous aborderons plus tard au point 3.4.5.1

- [ISO 10165-4], communément appelé GDMO - Guidelines for the Definition of Managed Objects - qui contient des directives pour la définition des objets de gestion.

L'ensemble de ces normes constitue ce que l'on appelle la structure de l'information de gestion ou SMI - Structure of Management Information.

Dans le point 3.3.3, nous introduisions la notion de base d'informations de gestion. Nous allons étendre quelque peu cette notion afin de cerner le modèle informationnel de la gestion OSI.

3.4.3.1 Le modèle de l'information de gestion

Dans [ISO 10165-1], l'OSI définit les concepts se rapportant au modèle servant de base à la spécification de l'information de gestion. Pour ce modèle, l'OSI se base sur les principes d'un modèle orienté-objet. Il s'appuie sur le principe d'abstraction. Les objets gérés sont en effet des représentations des ressources physiques et logiques qui doivent être administrées, telles qu'une entité de couche (N), une connexion ou un élément d'un équipement physique de communication. Chaque instance d'objet géré appartient à une classe d'objets gérés spécifiques. C'est de cette notion de classes d'objets dont nous allons nous entretenir maintenant.

a) Classes d'objets gérés

Une classe d'objets de gestion est caractérisée par :

- les **attributs** de l'objet visibles à l'extérieur de son système et donc accessibles à partir d'un système distant. Deux types d'attributs sont identifiés par l'OSI. Il s'agit d'abord des attributs obligatoires pour toute instance d'objet appartenant à cette classe. Le second type reprend les attributs optionnels. Les attributs peuvent être soit simplement valués soit

multi-valués. Ces derniers ont comme valeur un ensemble de membres d'un même type de données.

- les **opérations de gestion** qui peuvent lui être appliquées. On distingue deux types d'opérations de gestion suivant qu'elles s'appliquent à un attribut ou à un objet dans son ensemble. Les opérations de manipulation d'attributs sont les suivantes : lecture de la valeur d'un attribut - GET-ATTRIBUTE-VALUE -, écriture d'une valeur d'un attribut - REPLACE ATTRIBUTE VALUE -, écriture d'une valeur par défaut - REPLACE WITH DEFAULT VALUE -, ajout d'un membre à un attribut multi-valué - ADD MEMBER - et suppression d'un membre d'un attribut multi-valué - REMOVE MEMBER. Les opérations concernant les objets de gestion sont : la création - CREATE -, la destruction d'un objet - DELETE - et la demande faite à un objet d'exécuter une action particulière - ACTION.

- les **comportements** de l'objet suite aux opérations de gestion. Ce comportement consiste en l'effet des opérations sur l'objet, les contraintes de réalisation de ces opérations et la relation entre le comportement de l'objet et la ressource réelle dont il est une abstraction.

- les **notifications** qui peuvent être générées par l'objet, les circonstances d'émission et les informations contenues dans ces notifications.

Les classes sont organisées hiérarchiquement. Plusieurs notions d'arbres en découlent. Ce sont les arbres d'héritage, de contenance et de nommage.

b) L'arbre d'héritage

L'arbre d'héritage est le reflet de la hiérarchie d'héritage des classes d'objets. Celle-ci est organisée sur base de la spécialisation des classes. Le mécanisme de spécialisation permet de définir une nouvelle classe d'objets à partir d'autres existantes. Elle hérite alors de leurs caractéristiques et en ajoute certaines qui lui sont spécifiques. On parle de **sous-classe** pour désigner cette nouvelle classe définie par spécialisation de sa classe supérieure, la **super-classe**. La super-classe de plus haut niveau est appelée "Top". Elle n'a aucune propriété qui lui est propre.

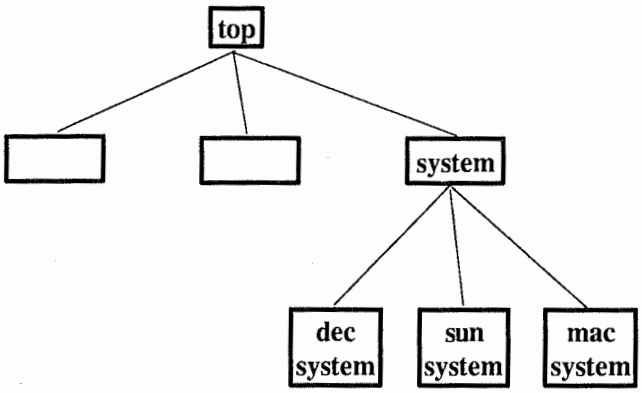


Figure 3.13 : arbre d'héritage

La figure 3.13 illustre un exemple d'arbre d'héritage dans lequel on trouve des spécialisation de l'objet system.

c) L'arbre de contenance

L'arbre de contenance est la représentation de la relation "est contenu dans" des classes d'objets entre elles. Deux types de classes d'objets découlent de cette relation : les classes d'objets **supérieures** qui contiennent les classes d'objets que l'on qualifie d'**inférieures**. Une relation 1-N existe entre les objets gérés. A savoir qu'un objet d'une classe particulière peut contenir plusieurs objets issus ou non de la même classe. Par contre, dans l'autre sens, un objet ne peut être contenu que dans une seule classe. Ces règles donnent à la hiérarchie une importance particulière dans la mesure où elles permettent de nommer sans ambiguïté les différents éléments de l'arbre. C'est ainsi qu'elle a été choisie pour le nommage des instances des objets gérés. C'est le mécanisme de nommage qui sera expliqué dans le point suivant sous le titre "arbre de nommage".

La relation de contenance implique également que l'existence d'un objet géré dépend de l'existence de l'objet le contenant. Ainsi, la disparition d'une instance d'objet entraîne automatiquement la disparition de tous les objets qu'elle contient.

d) L'arbre de nommage

Directement issu de l'arbre de contenance, l'arbre de nommage - que l'on appelle également MIT pour Management Information Tree - est la représentation d'une structure hiérarchique d'objets basés sur la relation du lien de nommage - "name binding". Cette

relation entre des classes d'objets spécifie que le nom d'un objet d'une classe supérieure peut servir à nommer les instances de ces classes inférieures.

Un objet d'une classe inférieure est nommé par la combinaison de :

- le nom de son supérieur
- une information qui identifie de manière unique cet objet dans le contexte de son supérieur. C'est ce que l'on appelle le RDN - Relative Distinguished Name.

Cette combinaison est connue comme étant le DN - Distinguished Name - de l'instance de l'objet géré. La figure 3.14 illustre le mécanisme du nommage au travers de l'arbre de nommage.

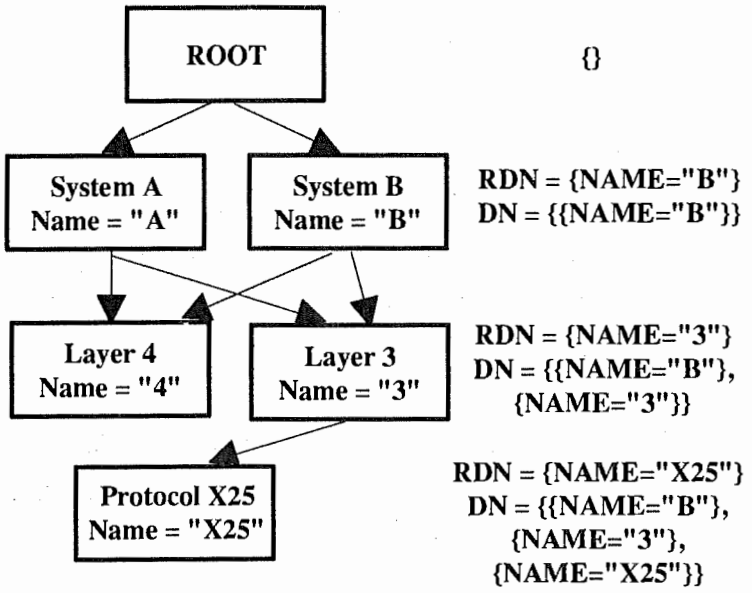


Figure 3.14 : MIT

3.4.3.2 Définition de l'information de gestion

La seconde partie des règles SMI [ISO10165-2] a pour objectif de définir des classes d'objets utilisées par les fonctions de gestion de systèmes. La définition précise des classes est du ressort des normes correspondantes à ces fonctions. [ISO10165-2] sert simplement à référencer les classes dans un même système d'enregistrement.

3.4.3.3 Directives pour la définition des objets de gestion

Ce que l'on a l'habitude d'appeler les GDMO, c'est-à-dire les directives pour la définition des objets de gestion et de leurs caractéristiques, sont spécifiées par la norme [ISO10165-4]. A partir de ces normes les développeurs ou experts chargés de la spécification d'objets de gestion pour une application de gestion particulière peuvent définir ceux-ci de manière cohérente. Cette norme est particulièrement utile dans ce sens qu'elle permet une compatibilité dans la définition des objets avec les autres standards de gestion OSI.

Pour ce faire l'ISO a défini un ensemble de formulaires - "templates". Le langage syntaxique abstrait ASN.1 [ISO8824] est utilisé afin d'écrire ces formulaires. L'annexe 1 reprend ces templates ainsi que les conventions syntaxiques utiles pour comprendre le formalisme d'ASN.1.

La définition des objets de gestion consiste en une énumération de :

- la position de la classe d'objet dans l'arbre d'héritage,

- "packages" qui spécifient un ensemble d'éléments caractérisant une classe d'objets de gestion. Ces caractéristiques sont les comportements, les attributs, les opérations et les notifications. Il y a deux sortes de "packages" : les "packages" obligatoires - "mandatory" - et les conditionnels - "conditional".

- un identifiant unique utilisé dans les communications de gestion. Tout ce qui est définit en suivant les GDMO est référencé de manière unique et entre dans un arbre d'enregistrement. La figure 3.15 illustre ce mécanisme d'enregistrement pour des objets définis par l'OSI/NM Forum.

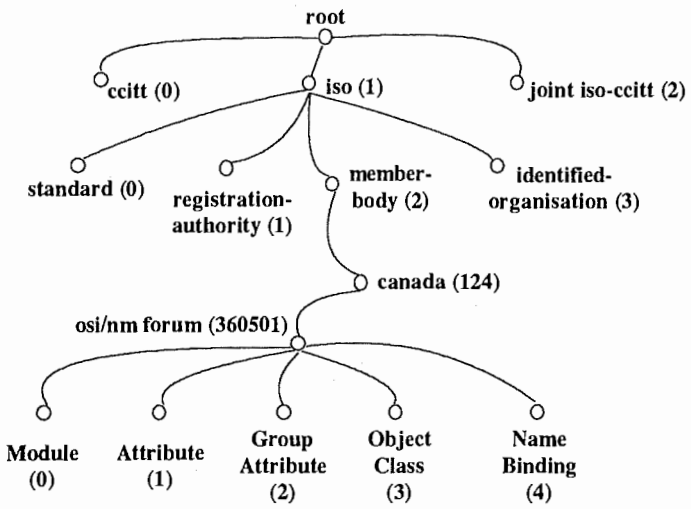


Figure 3.15 : arbre d'enregistrement

3.4.4 L'élément de service CMISE

L'élément de service Common Management Information Service Element - CMISE - est défini dans [ISO 9595] et [ISO 9596]. Ces deux normes portent respectivement sur CMIS et CMIP. CMISE fournit la base des mécanismes de communication pour des opérations et des notifications de gestion. Ainsi, il offre des services communs de gestion aux SMASE. Nous verrons par après quels sont ces services ainsi que les protocoles supportant ces services.

3.4.4.1 Le service commun CMIS

CMIS, Common Management Information Service, est l'ensemble des services communs pour la gestion OSI fournis par l'élément de service d'application CMISE. Ces services fournissent les moyens qui permettront de véhiculer des informations de gestion entre les SMAE des systèmes ouverts distants, entre un administrateur et un agent.

Les services, définis dans [ISO 9595] sont au nombre de sept. Pour chacun d'eux, il peut y avoir jusqu'à quatre types de primitives ainsi que des notions de sélection d'objets et de synchronisation. Les services sont de deux types : les services de notification et d'opération. Ces derniers permettent de connaître des informations de gestion, de les mettre à jour, de les créer ou de les détruire. Ils permettent également d'exécuter des opérations plus complexes que celles précitées.

Service de notification : M-EVENT-REPORT est un service de notification qui permet à un système agent de signaler à un système administrateur un événement survenu sur un objet géré. Ce service est, selon la demande de son utilisateur, en mode confirmé ou non.

Service de consultation : M-GET est un service d'opération de consultation qui permet à une SMAE dans un système administrateur d'obtenir des informations à propos d'objets gérés issus de la MIB d'un système distant. Ce service est toujours confirmé.

Service d'annulation de GET : M-CANCEL-GET est utilisé par un administrateur pour demander l'annulation d'un précédent M-GET. Cela occasionne l'arrêt du flot d'informations qui lui arrive. Ce service est toujours confirmé.

Service de mise à jour : M-SET est le service de mise à jour. Il permet à un système administrateur de demander à un agent la modification d'informations à propos d'objets gérés se trouvant dans la MIB de ce système. Ce service peut être confirmé ou non.

Service de création : M-CREATE est le service de création qui permet à un système administrateur de demander à un système agent la création d'une instance d'objets gérés stockés dans la MIB de ce système. Ce service est toujours confirmé.

Service de destruction : M-DELETE est le service de destruction permettant à un administrateur de demander à un agent la suppression d'une instance d'un objet géré de la MIB de ce dernier. Ce service est toujours confirmé.

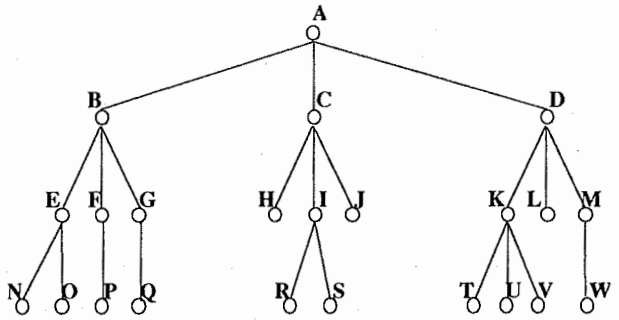
Service d'opérations complexes : M-ACTION est un service qui offre la possibilité à un système de demander à un autre système l'exécution d'une action. Cette action porte sur des objets gérés de la MIB de ce système. Elle est précisée dans les paramètres de la primitive de service. Grâce à ce service, il est permis de demander des opérations plus complexes que celles présentées dans les services ci-dessus. Le mode du service M-ACTION est confirmé ou non.

L'annexe 2 reprend pour chacun de ces services les paramètres qui lui sont associés. Elle montre également une comparaison entre les opérations sur les objets et les services CMIS.

CMIS fournit également trois services supplémentaires : un mécanisme de réponses multiples, les mécanismes de filtre et de profondeur de sélection et la synchronisation.

Le mécanisme de **réponses multiples** permet à un système de recevoir plusieurs réponses suite à une demande de service qu'il a produit. Lorsque le système envoie les réponses, il lie celles-ci logiquement. Ainsi, le demandeur du service initial reconnaît les réponses de manière non ambigüe. Ce mécanisme est disponible pour les services introduits ci-dessus exceptés le service d'annulation de GET - M-CANCEL-GET - et le service de notification - M-EVENT-REPORT. Les réponses multiples sont fort utiles dans le cas où une demande de service concerne plus d'un objet de gestion. Ceci implique qu'il soit possible d'atteindre plus d'un objet de gestion en une opération de gestion. Les mécanismes suivants en découlent.

Les mécanismes de **filtre et de profondeur de sélection** permettent à un système d'indiquer que l'opération qu'il demande s'applique à plus d'un objet de gestion. Le filtre permet par assertions logiques de n'atteindre que des objets de la base d'information de gestion vérifiant ces dernières. Couplée au mécanisme de profondeur de sélection, une opération de gestion peut ne concerner qu'un sous-arbre du MIT. La figure 3.16 montre des exemples de sélection d'objet par réduction de la profondeur de sélection.



Base = B, niveau = (Base) -> Espace = {B}
Base = A, niveau = (1° niveau) -> Espace = {B,C,D}
Base = C, niveau = (Base + 1° niveau) -> Espace = {C,H,I,J}
Base = D, niveau = (Base + fils) -> Espace = {D,K,L,M,T,U,V,W}

Figure 3.16 : sélection d'objet

La réduction de l'espace de choix se fait à partir d'un objet de base correspondant à la racine du sous-arbre du MIT. Plusieurs niveaux de profondeur de sélection peuvent être considérés. Soit l'objet de base seul, l'espace des objets auxquels vont s'adresser les opérations se réduit donc à la racine du sous-arbre; soit les descendants jusqu'à un certain

niveau; soit l'objet de base et tous ses descendants jusqu'à un niveau déterminé; soit l'objet de base et tous ses descendants, l'espace des objets correspond alors au sous-arbre entier.

Lorsque plusieurs objets sont concernés par une opération de gestion, synchroniser les opérations sur les différents objets devient nécessaire. L'ISO a défini deux modes de synchronisation : le type "**atomic**" et le type "**best effort**".

Dans le cas du type "atomic", il y a vérification pour s'assurer que toute l'opération est faisable pour chacun des objets sélectionnés. Si cela s'avère correct, alors toutes les opérations sont exécutées. Si par contre au moins un objet ne répond pas à cette nécessité, aucune opération ne sera prise en compte.

Par défaut, le type de synchronisation est le "best effort". Contrairement au précédent, il consiste à exécuter toutes les opérations possibles.

La synchronisation dont parle l'ISO, comme nous pouvons le remarquer, ne concerne pas l'ordre dans lequel les opérations sont effectuées. Ceci est du ressort de l'implémentation locale non normalisée.

3.4.4.2 Le protocole commun CMIP

Le protocole commun CMIP - Common Management Information Protocol - décrit dans [ISO9596] a pour objectif de fournir les moyens grâce auxquels les utilisateurs des services CMIS pourront effectuer des échanges de gestion entre eux.

Comme nous avons pu le constater, CMIS n'offre pas de service de gestion d'association. L'ISO, en effet, préconise à cet effet, l'utilisation de l'élément de service ACSE et plus particulièrement les services A-ASSOCIATE pour l'établissement d'une association entre des utilisateurs homologues de services CMIS : A-RELEASE pour la terminaison normale d'une association et A-P-ABORT pour une terminaison brutale de l'association

CMIP utilise les services offerts par l'élément de service ROSE - Remote Operation Service Element dans le but de véhiculer les dits échanges de type question-réponse. Pour les requêtes de services, le transfert s'effectuera sous la forme d'un RO-INVOKE. Les réponses à ces requêtes utilisent le service RO-RESULT si la réponse est positive, RO-

ERROR dans le cas contraire. Il arrive que des unités de données soient incorrectes à leur réception. Un RO-REJECT-U est alors utilisé pour envoyer la notification de l'erreur rencontrée.

La syntaxe utilisée pour la spécification des éléments de protocole CMIP et ses paramètres est également ASN.1. L'annexe 3 concerne la spécification en ASN.1 des unités de données du protocole commun CMIP. Elle reprend également une table de correspondance entre les primitives de CMISE et les opérations de CMIP.

3.4.5 Les fonctions spécifiques de la gestion de systèmes

Pour répondre au besoin des cinq aires fonctionnelles de gestion - SMFA -, l'OSI définit des fonctions spécifiques de gestion - SMF, System Management Function. Ces fonctions sont définies dans les normes ISO 10164-x. Chacune d'entre elles est utilisatrice des services offerts par CMISE.

Les fonctions spécifiques de gestion actuellement normalisées - c'est-à-dire à l'état d'IS - sont la gestion des objets [ISO10164-1], la gestion des états [ISO10164-2], la gestion des relations [ISO10164-3], la gestion des erreurs [ISO10164-4], la gestion des notifications d'événements [ISO10165-5], la gestion des journaux [ISO10164-6] et la gestion des alarmes de sécurité [ISO10164-7].

3.4.5.1 Gestion des objets

La fonction de gestion des objets a été développée afin d'apporter des services spécifiques de manipulations d'objets appartenant à la MIB. Elle permet ainsi aux autres fonctions spécifiques de ne pas se préoccuper de cet aspect de la gestion de systèmes. Les services spécifiés sont :

- La notification de création et de destruction d'objets de gestion
- La notification de changements apportés à un attribut d'un objet de gestion - par ajout, retrait ou remplacement.

De plus, [ISO10164-1] définit des services "pass-through" directement associés à CMIS. Ces services n'apportent pratiquement pas de valeur ajoutée aux services offerts par CMISE.

- La création d'objets de gestion - PT-CREATE.
- La destruction d'objets de gestion - PT-DELETE.
- L'exécution d'actions sur des objets de gestion - PT-ACTION.
- La modification d'attributs - PT SET.
- La lecture d'attributs - PT-GET.
- Les notifications - PT-EVENT-REPORT.

3.4.5.2 Gestion des états

L'état de gestion d'un objet géré représente la condition à un moment donné de la disponibilité et de l'opérabilité de la ressource qui lui est associée. Un objet géré possède un attribut particulier qui permet de déterminer cet état. La fonction de gestion des états [ISO10164-2] s'occupe de ces changements d'états.

Trois sortes d'attributs d'états sont définis : l'état opérationnel, l'état d'utilisation et l'état administratif. La valeur de l'état de gestion est fonction de celle des trois états précédents.

- L'**état opérationnel** illustré par la figure 3.17 peut avoir deux valeurs : disponible - ENABLED - et indisponible - DISABLED. La première signifie que la ressource est utilisable. La seconde indique que la ressource associée à l'objet de gestion n'est pas disponible pour une raison indépendante de la gestion du système. Cela pourrait être par exemple un modem qui n'est plus utilisable à cause d'un court-circuit.

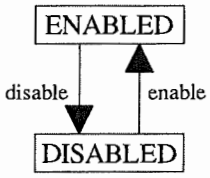


Figure 3.17 : état opérationnel d'un objet de gestion

- La figure 3.18 décrit l'**état d'utilisation**. Les deux valeurs associées à cet état sont : inactif - IDLE - lorsque la ressource n'est pas utilisée; actif - ACTIVE - pour signaler que la ressource décrite est utilisée mais, cependant, a suffisamment de capacités pour accepter un utilisateur supplémentaire; et enfin occupé - BUSY - si la ressource est utilisée au maximum de ses capacités. Cet état n'a de raison d'être que si l'état opérationnel a la valeur ENABLED

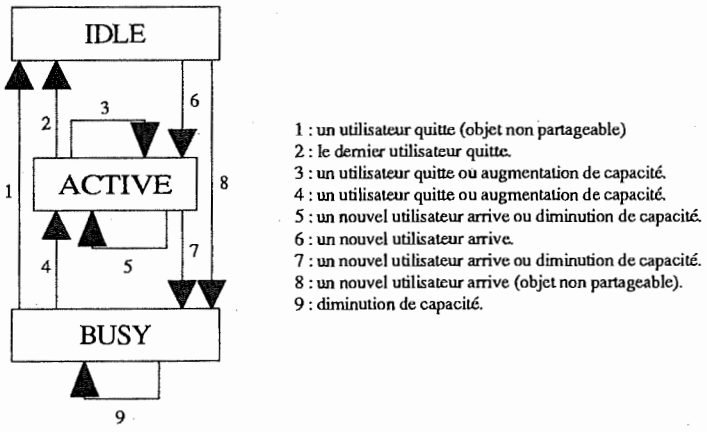


Figure 3.18 : état d'utilisation d'un objet de gestion

Cet état peut être utilisé dans une application de gestion qui contrôle les éléments d'un système. Ainsi, la couleur de la représentation d'une machine peut être associée à l'état d'utilisation de celle-ci - vert pour l'état IDLE, orange pour ACTIVE, rouge pour BUSY.

- L'**état administratif**, ainsi que nous pouvons le voir dans le diagramme de transition de la figure 3.19, passe par trois valeurs possibles : bloqué - LOCKED - auquel cas l'utilisation de la ressource n'est pas possible suite à une

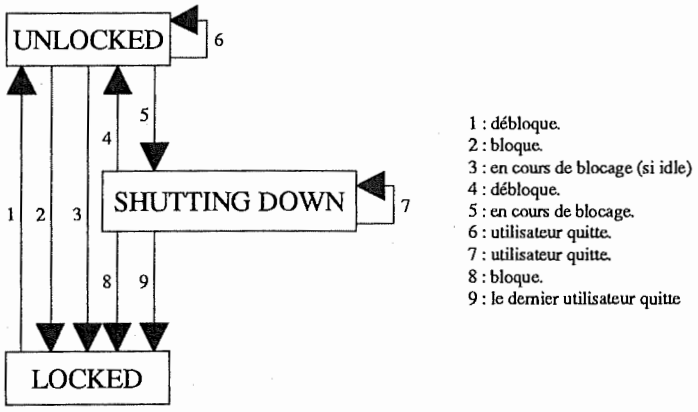


Figure 3.19 : état administratif d'un objet de gestion

action de gestion; non bloqué - UNLOCKED - qui permet d'utiliser la ressource à des fins de gestion; en cours de blocage - SHUTTING DOWN -, la ressource ne peut plus être utilisée par de nouveaux utilisateurs. Seules les utilisations en cours sont permises.

Cet état est utile lorsque une opération de gestion doit être exécutée sur des objets gérés. Quand c'est le cas, il est nécessaire de bloquer l'utilisation d'une ressource afin de ne pas perturber l'opération de gestion.

3.4.5.3 Gestion des relations

Les relations entre les ressources d'un système ouvert sont importantes car elles permettent de connaître l'évolution de son fonctionnement. Elles permettent de voir l'incidence d'une partie d'un système sur une partie. En plus des relations de contenance dont nous avons déjà parlé au point 3.4.3.1, il y a quatre autres catégories qui peuvent être définies. Ce sont les relations entre homologues, de service, de secours et de groupe.

- La **relation de service** qui relie des objets de gestion est une relation du type utilisateur et fournisseur de service correspondant aux deux rôles la caractérisant - service provider et service user.

- La **relation entre homologues** permet à les objets de gestion de communiquer entre eux. Les entités concernées par cette relation jouent toutes les deux le rôle d'homologue - peer - ayant des fonctions similaires.

- La **relation de secours** identifie des objets de secours. Ceux-ci seront amenés à remplacer un objet indisponible. L'objet remplaçable est l'objet primaire tandis que les objets de secours sont les objets secondaires.

- La **relation de groupe** permet de regrouper des objets de gestion du même type ou partageant des caractéristiques communes. Elle indique qu'un objet en possède un autre. Au sein de cette relation, deux rôles distinguent les objets qui la composent. D'abord le possesseur et ensuite le membre.

Des services de manipulations des relations et de leurs informations sont également spécifiés par [ISO10164-3]. Ce sont :

- La **création** d'une relation.

- La **destruction** d'une relation.

- La **modification** d'une relation.

- La **détermination des relations** d'un objet de gestion.

- La **détermination des objets** de gestion composant une relation.

- La **notification** en cas de création, de destruction et de modification de relations.

3.4.5.4 Gestion des erreurs

Par [ISO10164-4], l'ISO aborde la gestion des erreurs qui peuvent survenir dans un système. Lors de l'apparition de celles-ci, il convient, par notifications, de les signaler ainsi que de transmettre des informations qui leur sont relatives. Cela permet de prendre des décisions de gestion afin de ne pas trop perturber le fonctionnement du système.

Cinq catégories d'alarmes englobent l'ensemble des problèmes qui peuvent entraver la bonne marche du système. Les problèmes qui s'y rapportent sont les suivants :

- Les problèmes qui se rapportent à la **communication** tels que les erreurs de construction des trames, de transmission de données, de pertes de signal, etc.

- Les problèmes liés à la dégradation de la **qualité de service** - taux de transmissions élevés, augmentation du temps de réponse, par exemple.

- Les erreurs de **traitement** associées à des problèmes de stockage, d'incompatibilité entre des versions de logiciels, ...

- Des anomalies issues des **équipements** comme par exemple des chutes de tension dans le système électrique d'un terminal.

- Les problèmes d'**environnement** concernant la détection d'humidité, d'une température trop basse ou encore la détection de fumée.

Les informations supplémentaires - seuils, jauges, compteurs - utiles pour mieux cerner les occurences d'erreurs sont repris en paramètres de la notification calquée sur le service CMIS M-EVENT-REPORT.

3.4.5.5 Gestion des notifications d'événement

La norme [ISO10164-5] définit une fonction qui permet à un administrateur de contrôler la transmission des notifications d'événements des objets de gestion indépendamment de leur définition. La norme définit également deux classes d'objets de

gestion : le discriminateur et une sous-classe de celui-ci, le discriminateur de rapports d'événements - EFD, Event Forwarding Discriminator.

Le discriminateur permet à un système de sélectionner les opérations et les rapports d'événements concernant des objets de gestion. La sélection est basée sur un filtre et sur d'autres critères contenus dans l'objet.

L'EFD agit sur des rapports potentiels d'événements qui sont constitués par des informations qui doivent être envoyées dans la notification.

La figure 3.20 illustre le modèle de gestion des notifications d'événements.

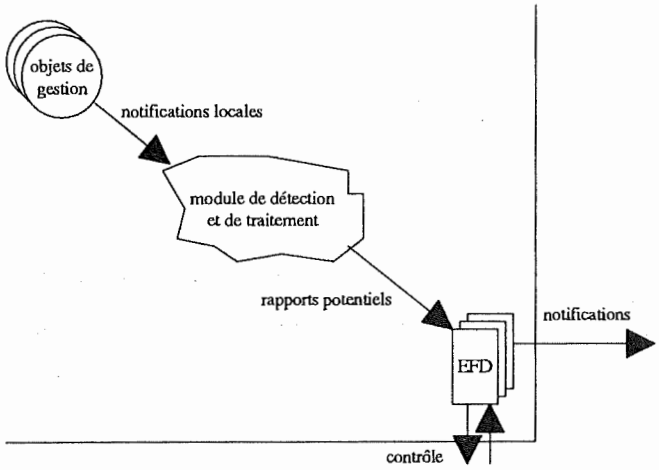


Figure 3.20 : modèle de la gestion des notifications d'événements.

Les notifications issues des objets de gestion sont traitées par un module de détection et de traitement. Ce dernier construit les rapports potentiels d'événements qui sont alors transmis à tous les EFD. Lorsque les valeurs des attributs contenus dans ces rapports répondent à des critères définis par un administrateur, ils sont envoyés par notifications aux destinataires spécifiés dans l'EFD. Ces destinataires peuvent être des journaux.

Un administrateur peut également contrôler ces EFD par des opérations de gestion.

3.4.5.6 Gestion des journaux

[ISO10164-6] fournit un modèle pour enregistrer les rapports d'événements émis par les EFDs et les notifications locales. Deux classes d'objets sont également définies par cette norme : le log et le log record.

Le log est une classe d'objets qui modélise des ressources pour l'enregistrement des log records, les unités d'informations qui le composent.

Le modèle de contrôle des journaux - log - est sensiblement le même que pour les notifications d'événements. Un traitement des notifications locales construit des log records potentiels. Ces derniers sont filtrés de la même manière que les EFD filtrent les rapports d'événements. Les log records une fois filtrés sont stockés et peuvent être consultés par les administrateurs via des protocoles de gestion ou d'autres services tels que le transfert de fichiers.

3.4.5.7 Gestion des alarmes de sécurité

Par [ISO10164-7], l'ISO définit le mécanisme de notification des alarmes de la sécurité de la gestion. Elle permet de faire connaître toutes les occurences d'attaques, de mauvaises opérations de service et d'événements relatifs à la sécurité. Ce mécanisme de notifications est calqué sur le service CMIS M-EVENT-REPORT dont il redéfinit les paramètres. Ceux-ci permettent de connaître le type et la cause de l'alarme, sa sévérité, son origine, l'utilisateur et le fournisseur du service incriminé.

3.5 OSI/NM Forum

En Juillet 1988, plus d'une soixantaine de vendeurs et de fournisseurs informatiques tels que Bull, Hewlett-Packard, IBM et DEC pour ne citer qu'eux, se sont regroupés en un consortium appelé OSI/Network Management Forum. Ils sont plus de cent à l'heure actuelle. Cela montre l'importance qui est accordée à cette matière.

Les buts du Forum sont d'accélérer l'interopérabilité des gestions de réseaux OSI, d'améliorer les standards, de définir une architecture qui permette d'intégrer les

administrations des constructeurs, de spécifier les protocoles ainsi que les objets à gérer, de déterminer les services d'applications et de décrire les méthodes d'implémentation.

OSI/NM Forum est particulièrement intéressant dans la mesure où il ouvre les portes du monde réel à l'OSI.

En fait, il ne se limite pas aux standards de l'ISO mais tient compte également des recommandations du CCITT - Comité Consultatif International Télégraphique et Téléphonique. Avant de présenter l'architecture et les concepts de NM Forum, il faut avoir à l'esprit que contrairement à l'ISO qui définit des standards de gestion de systèmes OSI et au CCITT qui émet des propositions sur les façons d'administrer des réseaux de télécommunications, OSI/NM Forum propose des solutions se basant sur ces standards pour gérer n'importe quel type de réseaux.

NM Forum a défini un modèle architectural qui permet de cerner les différents composants du problème de la gestion de réseaux ainsi que les relations entre ceux-ci. Avant de voir ces différents éléments, nous allons présenter les objectifs de ce modèle.

3.5.1 Objectifs du modèle

L'architecture de l'OSI/NM Forum a été définie afin de répondre à cinq objectifs.

- ***Intéroperabilité :*** "Le but premier de l'architecture est de fournir un cadre qui permettra à des produits et des services d'administration de réseaux de différents fournisseurs de travailler ensemble pour gérer les communications et les réseaux d'ordinateurs" [FORUM04].

  Il est important de pouvoir identifier quelles sont les données qui seront échangées ainsi que la manière selon laquelle l'échange s'effectuera. L'interopérabilité assure que des systèmes de conception différente puissent dialoguer de manière compréhensible. Un produit doit pouvoir gérer un autre produit et vice-versa.

- ***Modèle commun*** : "Une fois que les systèmes d'administration de réseau interagissent, il est nécessaire que chaque système comprenne quelles sont les

fonctions de gestion supportées ou requises par les autres, et les caractéristiques des composants physiques ou logiques sur lesquels ils agissent" [FORUM04].

Le modèle permettant cette compréhension est un modèle orienté-objet.

- ***Liberté d'implémentation*** : l'architecture doit permettre une certaine liberté d'implémentation. Différents constructeurs peuvent avoir, pour des raisons techniques ou économiques, adopté une implémentation de cette architecture. Au delà de toute considération technique, du point de vue du marché, il est bon de garder une concurrence entre les constructeurs.

- ***Flexibilité*** : "L'architecture doit être assez flexible pour être appliquée efficacement à la gestion de grands ou de petits réseaux de communications constitués d'une diversité de types d'équipements fournissant une variété de services." [FORUM04].

  Cet objectif doit permettre de gérer un réseau au-delà de toutes les contraintes liées à la diversité de ses composants. Un système d'administration doit être capable de gérer des types de réseaux allant du simple réseau local auquel sont connectées quelques stations de travail jusqu'au réseau public touchant des millions d'utilisateurs.

- ***Alignement avec ISO et CCITT***. Comme nous l'avons déjà introduit précédemment, l'architecture proposée par l'OSI/NM Forum doit respecter les standards de l'OSI et les recommandations du CCITT. Elle doit donc s'aligner à l'architecture de gestion de l'ISO ainsi qu'à celle du CCITT. Il peut cependant exister des différences tant que celles-ci n'entravent par l'objectif d'interopérabilité.

3.5.2 Composants de l'architecture générale

- ***Eléments gérés***. Les éléments gérés sont des ressources physiques ou logiques qui doivent être administrées. Ce sont des ressources du réseau de communication et des ressources des systèmes - des ordinateurs, des multiplexeurs, des processeurs de communication, par exemple.

- ***Solution de gestion***. Les utilisateurs désirant gérer ces éléments font appel à des solutions de gestion. Une solution de gestion est donc l'ensemble des systèmes, procédures et facilités utilisés par une organisation pour la gestion de son réseau. Cette partie de l'administration n'entre pas dans le giron des propositions d'implémentation du Forum. Pour répondre à l'objectif de liberté d'implémentation, il s'agit d'une boîte noire assurant une liberté aux constructeurs et une diversité dans les caractéristiques des produits qu'ils offrent.

  Dans les solutions de gestion, lorsqu'il ne s'agit pas de solutions sans interface utilisateur, le NM/FORUM intègre également leurs utilisateurs. Ceux-ci, en effet, jouent un rôle clé de par le fait qu'ils contribuent à résoudre les problèmes rencontrés après en avoir pris connaissance via l'interface utilisateur.

- ***CME***. Un CME - Conformant Management Entity - est un système de gestion qui est conforme aux standards adoptés par le Forum. Il permet donc à d'autres CME d'accéder aux données de gestion.

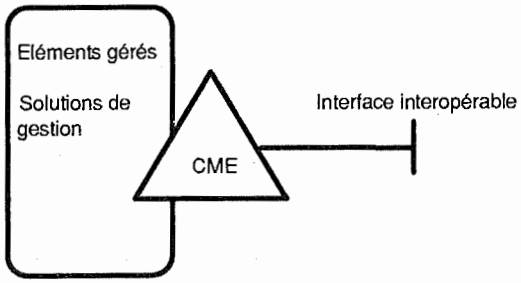


Figure 3.21 : CME

Un CME seul ne sert à rien. [EMBRY91] compare un CME isolé à une main tentant d'applaudir. Il faut dès lors au moins deux instances pour effectuer des activités de gestion interopérables. L'un des deux CME jouera le rôle de l'administrateur et le second sera l'agent tout comme les MIS-users dans l'architecture de gestion de système de l'OSI définie au point 3.4.2. Ils peuvent également exercer les deux rôles.

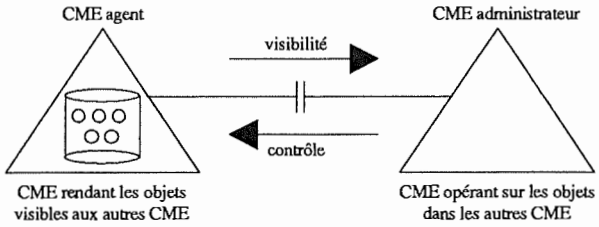


Figure 3.22 : CME agent et administrateur

Les objets de gestion, dont l'approche orientée-objet suit celle de l'ISO, constituent la MIB au sein des CME. Le CME dans le rôle de l'agent les rend visibles aux autres CME et permet donc à ceux-ci agissant dans le rôle de l'administrateur d'effectuer des opérations de gestion pour contrôler les ressources représentées par ces objets. Pour la définition des objets de gestion, NM Forum suit le GDMO de l'ISO. Ces objets peuvent représenter non seulement des éléments gérés mais ils peuvent également avoir été définis uniquement à des fins de gestion. Alors que l'OSI n'a défini à l'heure actuelle que des objets de ce dernier type, NM Forum en a défini du premier. Ils sont repris en annexe 4.

- ***Réseau de gestion***. Le réseau de gestion est un réseau qui met en relation les différents CMEs afin de leur permettre de communiquer entre eux. L'OSI/NM Forum le différencie du réseau de communication bien qu'il puisse l'utiliser.

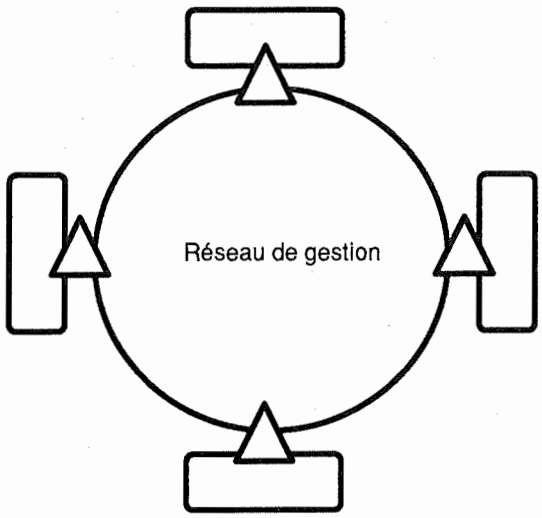


Figure 3.23 : le réseau de gestion

- ***Interface interopérable***. L'interface interopérable est l'ensemble des protocoles, procédures, formats et sémantiques des messages utilisés dans le cadre de l'administration de réseau. Cette interface est appelée l'interface P+M - Protocoles + Messages. Elle utilise un ensemble de protocoles de communications défini par l'ISO pour échanger des messages de gestion.

L'annexe 4 reprend le profil des protocoles de communications adoptés par le Forum. Nous nous en tiendrons dans cette présentation à la couche application illustrée par la figure 3.24 ci-dessous.

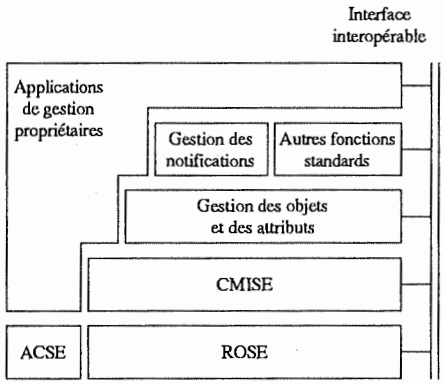


Figure 3.24 : couche application selon l'OSI/NM Forum

Afin d'assurer les associations et l'exécution d'opérations distantes entre CME, le Forum propose l'utilisation de l'élément de service ACSE et de ROSE conformément à l'ISO. CMISE assure l'échange des messages de gestion ainsi que les mécanismes propres aux services de gestion tels que définis au point 3.4.4.

Dans [FORUM02], le Forum définit des fonctions de gestion d'objets génériques et de leurs attributs basées sur CMIS. S'appuyant sur elles, des fonctions de gestion de notifications génériques ont également été spécifiées. Pour l'heure, d'autres fonctions de gestion ont été proposées. Il s'agit de fonctions relatives à la gestion de la configuration et à la gestion des fautes. D'autres fonctions sont en chantier.

Afin d'assurer l'interopérabilité, les applications de gestion développées par les constructeurs utilisent ces fonctions retenues par le NM Forum.

3.5.3 Relation avec l'OSI

Nous avons vu que l'architecture proposée est fortement basée sur celle exprimée dans [ISO7498-4] puisque le NM Forum suit les standards de l'ISO. La figure 3.10 représentait l'architecture de la gestion ISO. Dans ce modèle, les objets de gestion, du point de vue de l'administrateur, apparaissent derrière l'agent.

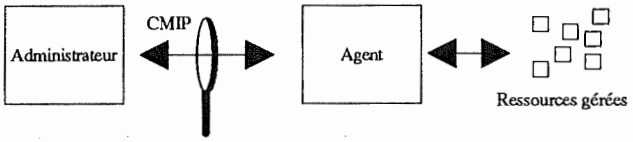


Figure 3.25 : objets gérés et le concept de loupe

Le Forum préfère, quant à lui, dire que les objets de gestion sont également une caractéristique de l'interface entre les systèmes de gestion interopérants. L'interopérabilité est réalisée en fournissant une vue des objets de gestion à l'interface interopérable. La figure 3.25 représente cette idée d'objets de gestion fournissant une vue des ressources avec une "loupe" assurant un regard orienté-objet des ressources gérées [FORUM02].

La figure 3.26 illustre cette conception des objets de gestion en intégrant les notions de CME et d'interface P+M. Elle insiste également sur la symétrie des systèmes de gestion se conformant aux spécifications du Forum.

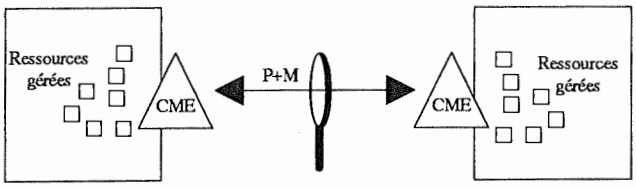


Figure 3.26 : loupe et symétrie des CME.

3.6 Conclusion

L'approche de l'OSI en matière de gestion de système est intéressante à retenir en vue d'une implémentation. On retrouve la technique du partionnement du problème à aborder, du système à administrer.

Ainsi, la notion de domaines de gestion se retrouve dans l'architecture de la gestion de systèmes standardisée par l'ISO.

L'approche orienté-objet, également retenue, permet elle aussi d'adopter des moyens d'implémentation répondant à un besoin d'abstraction et d' "information hiding" de la part des concepteurs de solutions de gestion.

Par contre, on pourrait reprocher à ces standards de ne pas avoir défini d'objets de gestion autres que des objets propres aux fonctions spécifiques de gestion. Ce sont de tels objets qu'attendent d'abord les utilisateurs d'un système de gestion. Ils représentent en effet les ressources réelles à administrer. L'OSI/NM Forum répond, en attendant une prise de position par l'ISO elle-même, à cette attente en proposant des objets de cette sorte.

Il est intéressant de voir l'attrait des constructeurs aux normes de gestion OSI. La formation et le travail de l'OSI/NM Forum en témoignent. Ceci nous amène à dire que le futur en matière de gestion de systèmes passera sans doute inévitablement par l'adoption des standards définis par l'ISO.

Chapitre 4

Integrated System Management de Bull

Chapitre 4 : Integrated System Management de Bull

4.1 Introduction

De plus en plus de constructeurs proposent des solutions pour administrer les systèmes distribués. Bull se place dans cette lignée. D'autres constructeurs tels que DEC avec son EMA - Enterprise Managemement Architecture - ou IBM et son offre Netview sont comparables au produit développé chez Bull, à savoir l'Integrated System Management - ISM. Dans ce chapitre, nous allons voir ce que peut être un CME dont parle l'OSI/NM Forum. Cette présentation porte sur une version non encore commercialisée d'ISM. Il se peut dès lors que l'on retrouve quelques contradictions par rapport à la version actuellement sur le marché.

4.2 Présentation générale

Bull a développé une architecture distribuée qui permet l'implémentation d'applications au sein d'un environnement distribué. Il assure la transparence qui caractérise les systèmes distribués vue au chapitre 1. La figure 4.1 nous montre les différentes briques de cette architecture appelée DCM - Distributed Computing Model. ISM en fait partie en tant que responsable de la gestion. Il permet d'administrer les ressources des systèmes distribués - système d'exploitation, utilisateurs, applications, réseaux, etc.

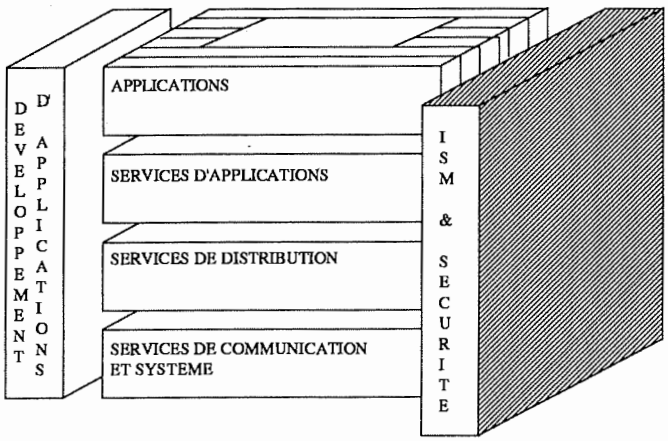


Figure 4.1 : Distributed Computing Model

Le modèle d'ISM est conforme au modèle de référence de l'ISO et aux normes qui s'y rapportent. De plus, l'implémentation d'ISM suivra les travaux de l'OSI/NM Forum.

4.3 Architecture d'ISM

Comme dans l'OSI, nous trouvons les notions d'agent et d'administrateur. Elles correspondent également aux CME du NM Forum. L'architecture d'ISM est donc constituée de deux parties. D'une part, nous avons l'ISM Manager et d'autre part, les ISM Agents. Ils dialoguent entre eux via des protocoles de gestion.

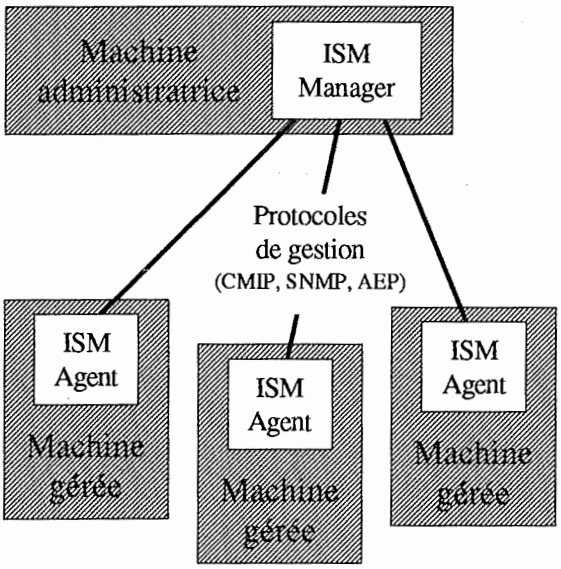


Figure 4.2 : architecture d'ISM

4.4 ISM Manager

Un système distribué, comme nous l'avons introduit au chapitre 2, peut être partitionné en domaines, à des fins administratives. Ces domaines sont susceptibles d'être conformes à des standards de communication et d'administration différents. Afin d'être capable de gérer ces domaines comme un tout, il faut dès lors tenir compte de cet aspect dans la conception du système administrateur. L'ISM Manager doit être capable de supporter des protocoles d'administration différents. Le premier qui nous vient à l'esprit est celui défini par l'ISO. ISM l'adopte en suivant les propositions du NM Forum. D'autres protocoles de gestion sont également pris en compte par ISM afin d'assurer une

interopérabilité avec d'autres systèmes ne répondant pas aux critères spécifiés par le NM Forum. Ces protocoles sont SNMP pour le monde TCP/IP et DSAC/AEP pour les environnements propriétaires développés par Bull.

La figure 4.3 propose la représentation des différents composants de l'ISM Manager dans lequel certains éléments tiennent compte de la contrainte expliquée ci-dessus.

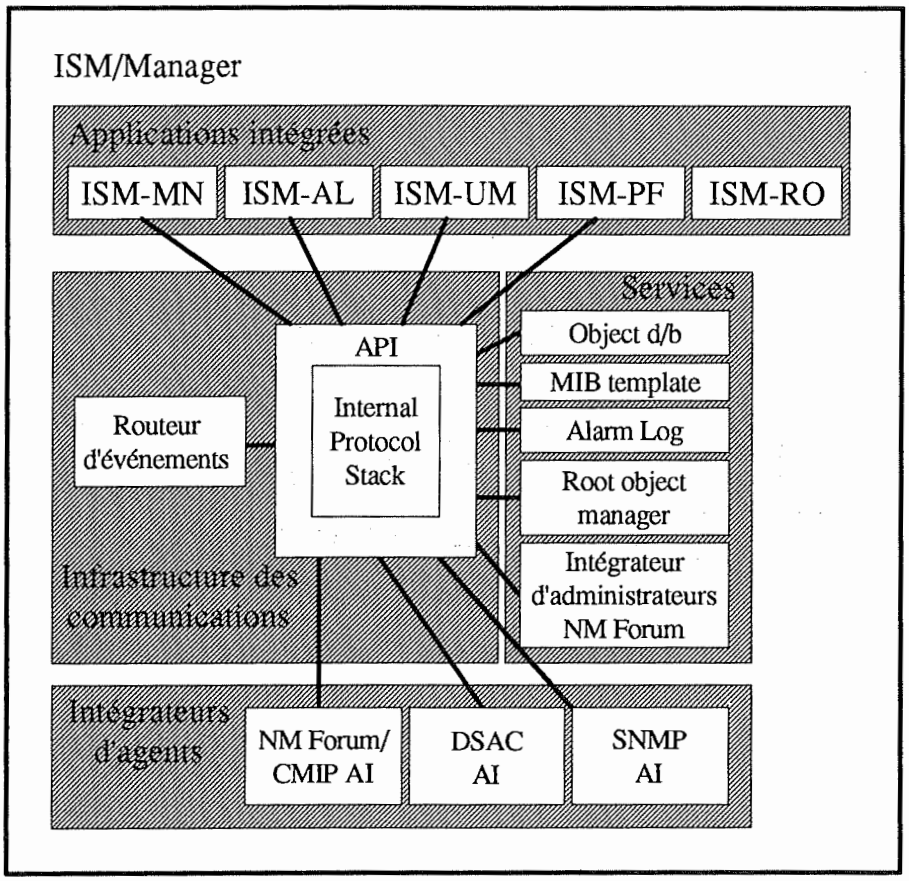


Figure 4.3 : ISM Manager

Les différents éléments qui constituent l'administrateur dans ISM sont les applications intégrées, les services, l'infrastructure des communications et les intégrateurs d'agents . C'est uniquement au moyen de ces derniers que le système administrateur - gérant - accède aux autres systèmes de gestion administrateurs ou administrés - gérés. Nous les verrons aux points 4.6 et suivants après avoir présenté la notion de MIB dans ISM.

4.5 ISM MIB

Toutes les classes d'objets gérés de la MIB d'ISM sont définies selon les GDMO du NM Forum. Elles couvrent toutes les classes d'objets nécessaires pour gérer l'ensemble du système distribué. Elles sont donc définies pour représenter des composants locaux du système - logiciels, périphériques, etc. -, les utilisateurs, les composants des communications - interfaces, circuits, etc. -, les abstractions du système distribué - vendeurs, localisation des composants - et les objets propres à la gestion - alarmes, logs,... .

De plus, il faut tenir compte du fait qu'ISM supporte plusieurs protocoles d'administration. La MIB devra donc contenir les classes d'objets gérés conformes à SNMP, DSAC et NM Forum/CMIP.

L'ISM MIB est un ensemble de sous-MIB appelées des MIBlets. Chacune de celles-ci est un sous-arbre de la MIB et est attachée directement à la racine. L'instance de l'objet géré attachée à la racine est appelée la rootlet.

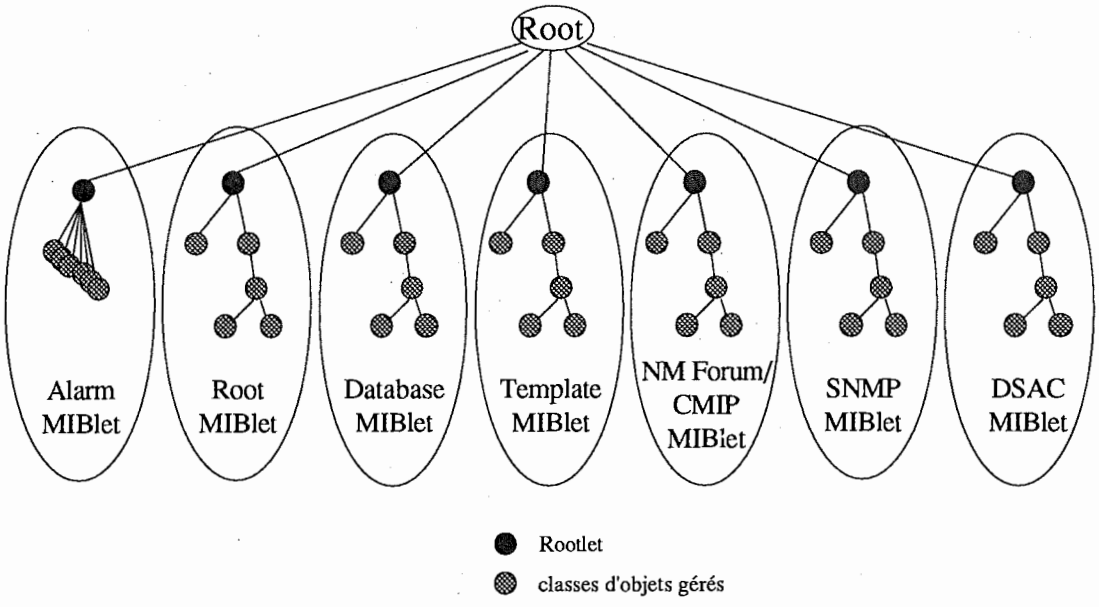


Figure 4.4 : ISM MIB

Ces MIBlets pour être connues du système doivent être instanciées auprès de l'ISM Manager. Les composants d'ISM qui se chargent de cet instanciation sont appelés des

gestionnaires d'objets. Ce sont les intégrateurs d'agents ou les services qui ont ce rôle. Ainsi, l'intégrateur d'agents NM Forum/CMIP instancie-t-il la MIBlet NM Forum/CMIP qui contient des objets CMIP définis par le NM Forum.

Malgré cette division en MIBlets, la MIB d'ISM est vue comme une seule entité. Une application peut exécuter des opérations sur une instance d'un objet n'importe où dans la MIB sans avoir besoin de connaître le gestionnaire d'objets qui s'en occupe.

4.6 Applications intégrées

Les applications constituent la partie d'ISM Manager qui dispose d'une interface utilisateur. Elles peuvent être cataloguées de différentes manières : les applications de base, les applications complémentaires, les applications locales, les applications génériques et les applications spécifiques.

Les **applications de base** sont celles qui font partie intégrante d'ISM Manager et qui utilisent l'infrastructure de ce dernier. Elles s'exécutent sur la machine de gestion.

Les **applications complémentaires** quoique tournant sur la machine de gestion comme les précédentes, ne sont pas intégrées à ISM Manager. Ce sont des applications qui sont utilisées par le personnel de gestion mais qui n'utilisent pas les fonctionnalités offertes par ISM Manager. Ce sont par exemple des tableurs utilisés pour faire des statistiques de gestion.

Les **applications locales** ne tournent pas, quant à elles, sur la machine de gestion mais bien sur d'autres dans le réseau géré.

Les **applications génériques** sont celles qui peuvent gérer tous les objets de la MIB d'ISM. Elles connaissent la syntaxe des classes d'objets mais pas leur sémantique. Elles offrent des fonctionnalités qui sont indépendantes du sous-système à gérer. Ajouter des nouvelles classes d'objets à gérer n'oblige donc pas à modifier ces applications.

Enfin, les **applications spécifiques**, contrairement aux applications génériques sont celles qui agissent sur certaines classes d'objets uniquement. Elles connaissent à la fois la syntaxe et la sémantique de celles-ci.

Les applications intégrées à ISM sont des applications de base. De plus, elles sont soit génériques - ISM Monitor (ISM-MN), ISM Alarm (ISM-AL), ISM Remote Operation (ISM-RO) -, soit spécifiques - ISM Performance (ISM-PF), ISM User Management (ISM-UM). La figure 4.5 ci-dessous les reprend dans un tableau.

Applications intégrées	
génériques	ISM-MN ISM AL ISM-RO
spécifiques	ISM-PF ISM-UM

Figure 4.5 : applications intégrées à ISM

Nous allons maintenant les passer en revue.

4.6.1 ISM Monitor

ISM Monitor est l'application principale d'ISM. Elle entre dans le cadre de la gestion de la configuration. Elle permet de construire et d'afficher des images du système à administrer. Un mécanisme d'enchaînement d'images permet d'examiner différents niveaux de détails de ce système. Chaque image est constituée de symboles graphiques représentant des instances d'objets gérés. Chaque symbole graphique peut avoir une image qui lui est associée et représentant un niveau de détail plus fin. Il peut également avoir un tableau de bord de graphiques représentant la valeur d'un attribut de l'instance de l'objet qui lui est associé. Ce sont par exemple des compteurs, des graphes, des VU-mètres.

ISM Monitor permet également d'obtenir des informations sous forme de tables des attributs d'une instance d'objets. A partir de cette table, l'administrateur peut demander de balayer la MIB en affichant les attributs d'une autre instance.

L'état courant d'un objet - défini par l'état opérationnel, par exemple - est directement visible à l'écran par le symbole dont la couleur change en fonction de cet état.

Il est possible de contrôler le système en mettant à jour les valeurs de certains attributs à travers les tables les reprenant. Ainsi, un administrateur pourra changer le nom de la personne à contacter pour obtenir des informations concernant une machine particulière. Il lui est possible, également, de créer, copier ou effacer des instances de certains objets.

Après avoir sélectionné un objet, l'administrateur peut demander de lancer une autre application se rapportant à cet objet. Il pourra par exemple demander le lancement de l'application responsable du contrôle des alarmes pour une passerelle au sein d'un réseau.

Les applications des clients peuvent également être appelées à partir d'ISM Monitor.

4.6.2 ISM Alarm

ISM Alarm est une application de gestion de fautes définie par l'OSI. Elle permet la détection et l'affichage d'alarmes en temps réels ou non.

Les alarmes sont affichées textuellement sous forme de matrice. En sélectionnant une ligne correspondant à une alarme, il est possible de demander des détails la concernant. Une table s'affiche alors avec l'ensemble des attributs qui s'y rapportent.

ISM Alarm permet d'effectuer certaines actions sur une alarme. Elle offre entre autres la possibilité de ne pas afficher une alarme à moins d'une demande contraire. Des filtres peuvent également être définis pour sélectionner une partie seulement des alarmes.

4.6.3 ISM Performance

ISM Performance suit les principes ISO pour la gestion des performances. Elle est utilisée pour afficher des indicateurs de performance comme les compteurs d'alarmes associés au objets gérés. L'évolution de ces compteurs peut être représentée graphiquement, en temps réel - par un thermomètre, par exemple. Elle permet également d'établir les seuils pour la création des alarmes. Les seuils définissent des niveaux au-delà desquels une alarme doit être émise. Nous pouvons donner comme exemple le cas de la gestion des disques. L'administrateur peut définir comme seuil d'alarmes un taux de remplissage des disques de 95 %.

4.6.4 ISM Remote Operation

ISM offre à l'administrateur la possibilité de se connecter à distance sur une machine en cliquant sur l'icône la représentant dans ISM Monitor. ISM Remote Operation est l'application qui offre cette fonctionnalité. Les systèmes distants auxquels elle permet d'accéder par émulation de terminal sont des systèmes GCOS6, GCOS7, GCOS8 - propres à Bull -, UNIX ou Datanet.

Une fois connecté, l'administrateur peut lancer une application de gestion locale au système atteint.

4.6.5 ISM User Management

ISM User Management, comme son nom l'indique, permet la gestion des utilisateurs d'un système distribué. Le rôle principal de cette application est de faciliter la tâche de l'administrateur en matière d'identification et d'enregistrement des utilisateurs des différents services du système distribué. Elle s'occupe de maintenir de manière uniforme la connaissance des utilisateurs et de leurs droits d'accès aux services.

4.7 Outils de développement

En plus de ces applications intégrées à ISM, Bull offre des outils de développement permettant de programmer ses propres applications de gestion. Le langage qui est fourni s'appelle ISM Management Language - ISM-ML

Ainsi, un client désirant créer sa propre application de gestion utilisera-t-il les fonctionnalités offertes par ce langage. Il s'agit d'un langage interprété qui fait penser très fortement au langage LISP.

ISM-ML intègre différentes interfaces. La figure 4.6 les illustre. Il s'agit des interfaces d'accès aux librairies X/Motif et GO pour l'affichage graphique, SQL pour l'accès aux bases de données, CMIS pour les services d'administration. L'intégration de ces interfaces forme une interface de programmation - API en raccourci. Il est possible d'étendre les fonctions de base offertes par ISM-ML en implémentant les nouvelles fonctionnalités en C.

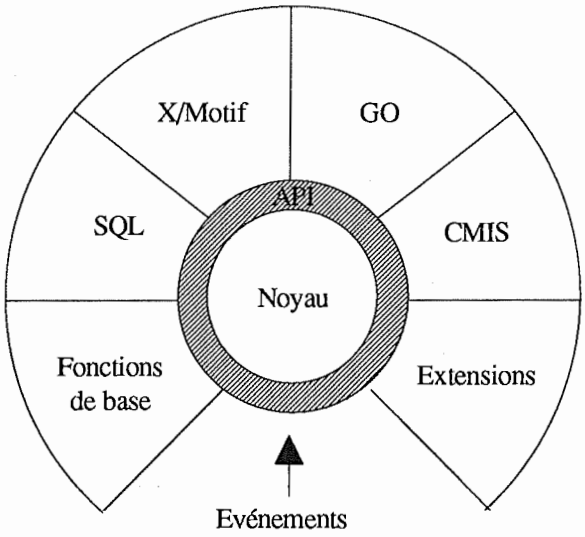


Figure 4.6 : composants d'ISM-ML

ISM-ML est un langage de très haut niveau. Il permet par exemple d'afficher une valeur représentée graphiquement en une seule ligne de code. Il réagit à des événements dans la même philosophie que X-Windows.

4.8 Infrastructure des communications

Tous les composants d'ISM Manager - applications, services, intégrateurs d'agents - communiquent entre eux via l'infrastructure de communications. Cette dernière supporte la distribution des composants à travers plusieurs machines. Elle comporte les services suivant : une interface de programmation, l'Internal Protocol Stack, le routeur d'événements.

4.8.1 Interface de programmation

Les services CMIS - M-GET, M-SET, M-CREATE, M-DELETE, M-CANCEL-GET, M-ACTION et M-EVENT-REPORT - sont fournis aux composants d'ISM Manager par une interface de programmation.Cette interface de programmation est accessible via le langage ISM-ML présenté ci-dessus dont nous avons vu qu'un des composant permettait d'avoir accès aux services CMIS.

4.8.2 Internal Protocol Stack

Le protocole qui permet les communications entre les différents composants d'un ISM/Manager, que celui-ci soit sur une seule machine ou distribué sur plusieurs est CMIP.

4.8.3 Routeur d'événements

L'infrastructure des communications s'occupe du routage des événements pour les autres composants de l'ISM Manager. Une application ou un service peut utiliser un discriminateur pour filtrer les notifications venant d'un autre gestionnaire d'objet.

Les gestionnaires d'objets envoient les notifications au routeur d'événements qui les transfère selon les discriminateurs qui leur sont associés.

4.9 Services

ISM Manager fournit un ensemble de services commun aux applications. Ces services sont l'Object Database Service, le MIB Template Service, l'Alarm Log Service, le Root Object Manager et l'intégrateur d'administrateurs NM Forum.

4.9.1 Object Database Service

Le service "Object Database" tient à jour une MIBlet appelée la Database MIBlet . Les classes d'objets de celle-ci sont locales à l'ISM Manager et n'ont pas de contrepartie directe dans le monde réel. En d'autres mots, il n'y a pas d'agent qui lui est liée.

Les classes d'objets que ce service gère sont utilisées pour représenter le système distribué du client en fonction des choix de celui-ci. Ce sera par exemple la topologie de son réseau. La MIBlet se base sur [FORUM06] dont il retient les classes d'objets gérés suivantes :

- Network
- Computer System
- Equipment
- Facility
- Function
- Processing Entity
- Circuit
- LAN MAC Bridge

- Location
- Vendor
- Provider
- Service
- Customer
- Contact
- Manufacturer
- Note

4.9.2 MIB Template Service

Etant donné que les applications génériques doivent être capables de supporter une grande variété de classes d'objets allant de celles prédéfinies par l'OSI/NM Forum à d'autres spécifiques aux utilisateurs, il est vivement déconseillé de programmer "en dur" la définition de ces classes d'objets. Le MIB Template Service fournit la solution pour que les applications génériques connaissent la description syntaxique de n'importe quelle classe d'objets.

Le contenu de chaque classe est décrit par un formulaire - un template. C'est à partir de ces formulaires que les applications connaissent la définition de classes. Ces formulaires suivent le document du NM Forum concernant la spécification des objets, [FORUM06].

4.9.3 Alarm Log Service

L'Alarm Log Service est un gestionnaire d'objets qui s'occupe des alarmes. Il maintient une MIBlet, l'Alarm MIBlet qui contient des classes d'objets basées sur NM Forum. Il s'agit des classes Alarm Log et Alarm Record. Cette dernière étant une instance de la première.

Les Alarm Record sont reçus des gestionnaires d'objets en tant que notifications et stockés dans l'Alarm Log. Les applications - typiquement celles concernant la gestion des fautes comme ISM Alarm - peuvent dès lors accéder aux enregistrements de ces alarmes grâce aux services offerts par l'Alarm Log Service.

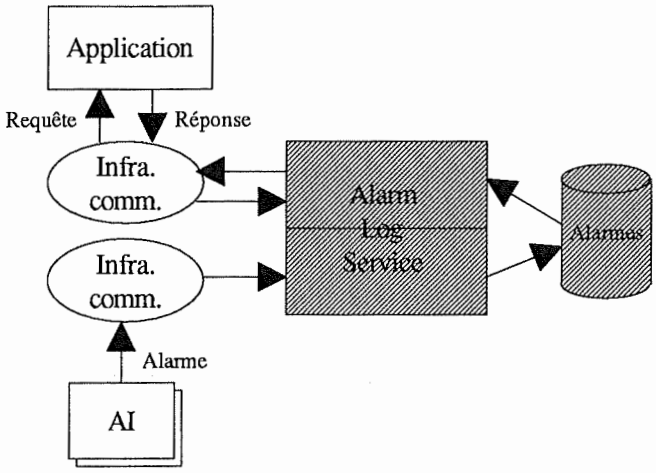


Figure 4.7 : Alarm Log Service

4.9.4 Root Object Manager

Nous avons vu au point 4.5 que la MIB d'ISM était constituée d'un ensemble de sous-MIB, les MIBlets attachées à sa racine par les rootlets.

Les applications d'ISM voient cette MIB comme un tout sans se préoccuper de savoir à quels gestionnaires d'objets revient la responsabilité d'une instance d'objets gérés particulière. Le Root Object Manager connaît la localisation de chacun des gestionnaires d'objets et est responsable du routage des commandes vers ceux-ci.

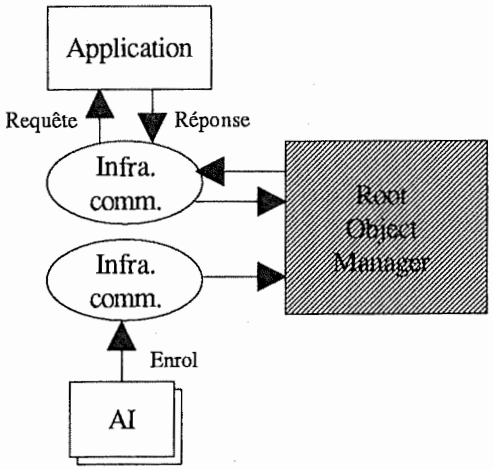


Figure 4.8 : Root Object Manager

Il maintient pour cela une table des rootlets auxquels est associée l'adresse de leur gestionnaire - qui est également celui des objets de sa MIBlet. L'infrastructure des communications utilise donc cette table pour diriger les requêtes des utilisateurs des services CMIS - applications ou services - vers le gestionnaire d'objets approprié.

La rootlet et sa MIBlet sont accessibles une fois qu'ils ont été rendus visibles dans l'ISM MIB. Alors seulement les applications pourront demander l'exécution d'opérations de gestion sur les objets gérés. Pour cela, les gestionnaires d'objets par une notification - appelée un "enrol" -, enregistre la rootlet auprès du Root Object Manager.

4.9.5 L'intégrateur d'administrateurs NM Forum

L'intégrateur d'administrateurs NM Forum permet à ISM d'être administré par un administrateur conforme à NM Forum - un CME dans le rôle administrateur. ISM agit donc comme un CME agent.

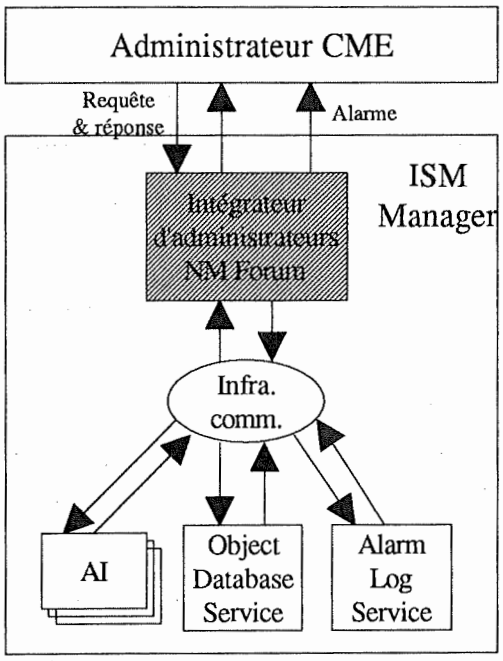


Figure 4.9 : intégrateur d'administrateurs NM Forum

Ce service est implémenté pour la communication entre différents ISM Managers et entre un ISM Manager et un système de gestion provenant d'un autre constructeur agissant comme un super gestionnaire.

4.10 Intégrateurs d'agents

La philosophie adoptée dans la conception d'ISM pourrait être comparée à celle d'un système d'exploitation tel qu'UNIX. Dans ce dernier, lorsque qu'il s'agit de contrôler un nouvel équipement, on installe ce que l'on appelle un driver. Dans ISM lorsqu'un nouveau sous-système doit être administré et qu'il ne correspond pas à ce que l'ISM Manager connaît, l'administrateur installe un driver pour la gestion de ce sous-système. Bull l'appelle un intégrateur d'agent - Agent Integrator, AI.

Un AI supporte un ensemble de services - CREATE, DELETE, GET, SET, ALARM, ENROL - et implémente ces services en utilisant ceux transportés par le protocole de gestion propre au sous-système qu'il administre.

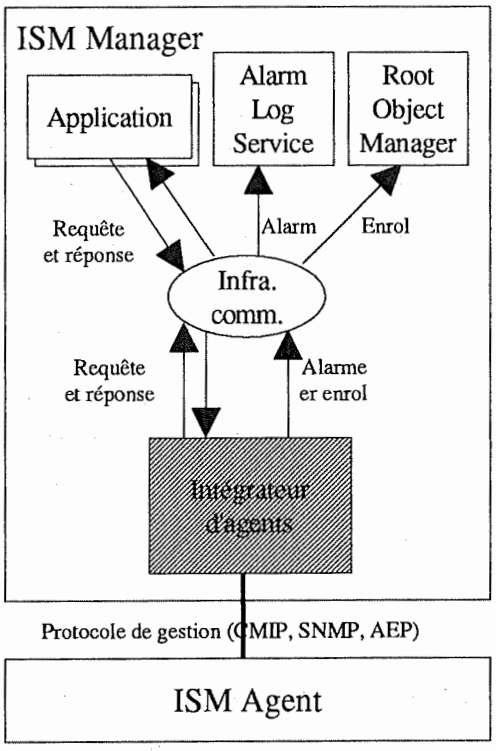


Figure 4.9 : intégrateur d'agents

ISM offre trois intégrateurs d'agent correspondant aux sous-systèmes Internet utilisant le protocole SNMP, aux sous-systèmes DSA propres à Bull dont le protocole de gestion est DSAC/AEP et enfin aux sous-systèmes conformes à OSI et plus particulièrement aux propositions du NM Forum. Nous nous en tiendrons pour la suite à ces derniers.

Les fonctions des AI sont :

- le **multiplexage** pour permettre à différentes applications et services de communiquer avec différents agents

- la **traduction des protocoles**. Ils traduisent le protocole de gestion du sous-système en celui interne à ISM et vice-versa.

- la **traduction de la MIB** pour traduire les classes d'objets du sous-système à administrer en classes d'objets conformes à celles d'ISM.

- la **génération d'alarmes**. Les alarmes peuvent provenir de différentes sources. Tout d'abord, celles issues des agents des sous-systèmes. L'AI les traduit dans un format ISM et les transmet au service de gestion des alarmes - Alarm Log Service. Enfin les alarmes reconnues par l'AI lui-même - lorsqu'il détecte qu'un agent ne fonctionne plus, par exemple.

- la **découverte de rootlet** pour détecter l'existence d'une nouvelle rootlet et générer une notification d'enrol.

- l'**augmentation de la MIB**. Cette fonction permet d'ajouter des attributs supplémentaires aux classes d'objets gérés.

- toutes autres **fonctions spécifiques** au sous-système comme par exemple le calcul statistique de performance ou de comptabilité.

Dans le cas de l'intégrateur d'agent NM Forum/CMIP, la traduction des objets n'est pas nécessaire puisque ceux-ci sont déjà conformes au format d'ISM. Il peut donc supporter les classes d'objets gérés conformes aux standards ISO pour la définition d'objets. Il s'agit des objets

- standards OSI

- NM Forum

- d'un ISM Manager et rendus visibles par l'intégrateur d'administrateur NM Forum d'un autre ISM Manager

- privés supportés par un agent NM Forum/CMIP.

La figure 4.11 illustre une configuration type d'ISM utilisant CMIP. L'intégrateur d'agent NM Forum/CMIP utilise une pile ISO complète pour atteindre l'agent.

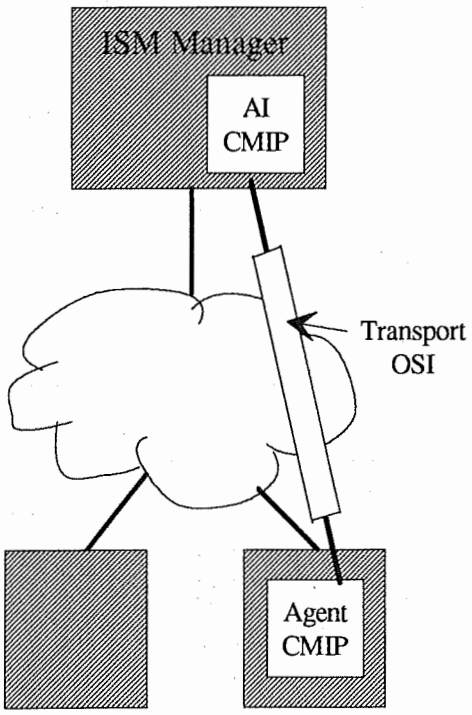


Figure 4.11 : exemple de configuration CMIP.

4.11 ISM Agents

Sur chaque machine dans le système distribué tourne un agent responsable du contrôle de cette machine. Il est le pendant de l'ISM Manager duquel il dépend.

Les ISM Agents sont responsables de rendre les objets gérés visibles aux ISM Managers. Ils envoient également des notifications à ces derniers en utilisant le protocole de gestion qui lui est propre. De plus, ils effectuent les opérations de gestion demandées par les ISM Managers.

Il y a trois types d'agents implémentés dans ISM : les agents SNMP, les agents DSAC et les agents NM Forum/CMIP. L'architecture de ces agents est ouverte de telle sorte qu'elle permet d'introduire un nouvel objet à gérer sans modifier le code existant.

La figure 4.12 illustre cette architecture et la communication entre l'ISM Manager et l'ISM Agent NM Forum/CMIP via le protocole CMIP et l'intégrateur d'agent NM Forum/CMIP.

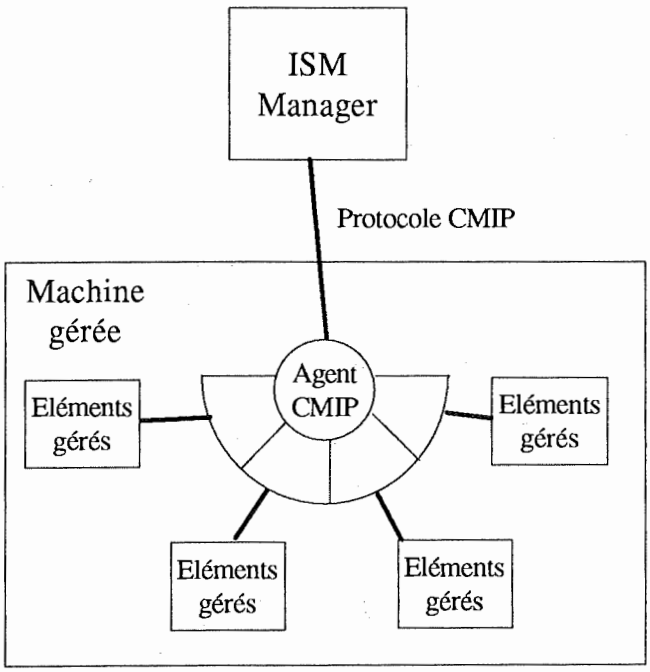


Figure 4.12 : agent CMIP

4.12 Conclusion

ISM de Bull offre aux administrateurs des systèmes distribués des fonctionnalités fort intéressantes. Ainsi l'adoption de trois protocoles de gestion donne à ce produit un atout non négligeable. CMIP de l'ISO est le standard qui devrait ressortir à l'avenir. SNMP pour le monde TCP/IP l'est à l'heure actuelle. Ceci permet de sélectionner les protocoles de gestion en fonction des éléments du systèmes à administrer. Ainsi, les machines nécessitant des fonctions de gestion sophistiquées utiliseront-elles CMIP, alors que les autres n'en ayant pas le besoin ou la puissance pour le supporter se baseront sur SNMP plus simple. De plus, le protocole et les services internes à ISM Manager sont ceux de l'ISO.

La structuration de l'administration d'un système distribué, par les intégrateurs d'administration NM Forum/CMIP rejoint le partitionnement de celui-ci pour en simplifier la gestion.

Du point de vue de l'implémentation et de l'extensibilité des applications et des sous-systèmes à gérer, la technique du driver, l'intégrateur d'agents, permet de ne pas perturber la plate-forme existante. Le client désirant ajouter des applications à l'ISM Manager dispose d'outils de développement de haut niveau avec ISM-ML et l'interface de programmation qui permet d'accéder aux services offerts par CMIS. S'il veut étendre le champ d'administration à des sous-systèmes non encore gérables - typiquement non conformes à NM Forum/CMIP, SNMP ou DSAC/AEP, des outils lui sont également proposés pour faciliter l'implémentation de l'intégrateur d'agents correspondant ainsi que l'agent.

Par contre, la technique de l'intégrateur d'agents oblige le client à implémenter deux piles de protocoles. D'une part une pile OSI pour la communication avec les autres composants de l'ISM Manager via l'infrastructure des communications. D'autre part une pile correspondant à l'environnement du nouveau sous-système à gérer - SNA d'IBM, par exemple.

Chapitre 5

UniXView

Chapitre 5 : UniXView

5.1 Introduction

Durant notre stage effectué chez Bull, nous avons eu l'occasion de cotoyer une équipe développant l'application de gestion présentée dans le chapitre précédent : ISM. L'application qu'il nous fut demandé de réaliser avait pour trait l'administration de système UNIX. Il s'agissait plus spécialement de permettre à un administrateur de connaître l'état des disques - entendez par là systèmes de fichiers - des machines ainsi que les utilisateurs qui étaient connectés à ces différentes machines. Cela revenait à reproduire les commandes "who" et "df". Cette application était en quelque sorte une application complémentaire spécifique dans les termes qui ont été définis au point 4.6.

Les matières que cela permettait d'aborder étaient les réseaux et les communications, l'administration de systèmes distribués, UNIX, le protocole d'administration SNMP, le langage C, OSF/Motif - standard de présentation graphique adopté par l'OSF - Open Software Foundation - et tournant au dessus de X/Windows - et le langage ISM-ML - voir point 4.6.6.

5.2 Cadre de l'application

Avant d'aborder les mécanismes sous-jacents à UniXView, nous allons établir un cadre de travail. Nous fixerons pour cela le type de réseau et ses caractéristiques.

Tout d'abord, comme l'illustre la figure 5.1, le système est de type faiblement couplé multi-machines. Il est constitué de différentes stations de travail sur lesquels tournent un OS. En l'occurence, dans le cadre de notre stage, il s'agissait de l'UNIX de SCO - Santa Cruz Operation. Il y avait également de plus grosses machines serveurs. Ne tournant pas avec SCO, nous ne les intégrons pas à notre exemple.

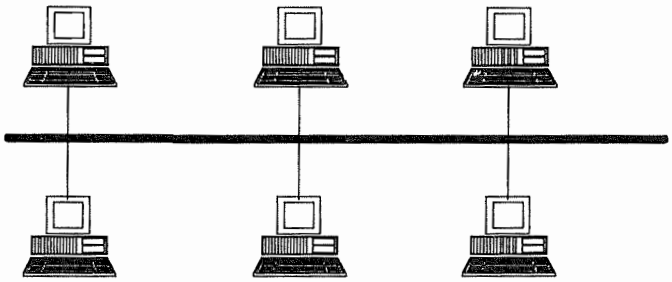


Figure 5.1 : exemple de système admininistré par UniXView

5.3 Présentation de l'application

La fenêtre principale de l'application UniXView permet de charger des images définies par l'administrateur représentant le système à gérer. En fait, les images sont des fichiers construits à partir d'ISM-MN. A la demande de l'utilisateur pour charger une image, un boîte de dialogue - figure 5.2 - permet de sélectionner le fichier correspondant.

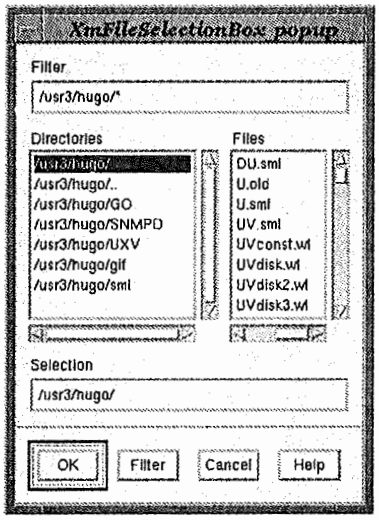


Figure 5.2 : boîte de dialogue de sélection de fichiers

La figure 5.3 représente cet écran principal dans lequel on a chargé l' image d'un réseau et de quelques machines.

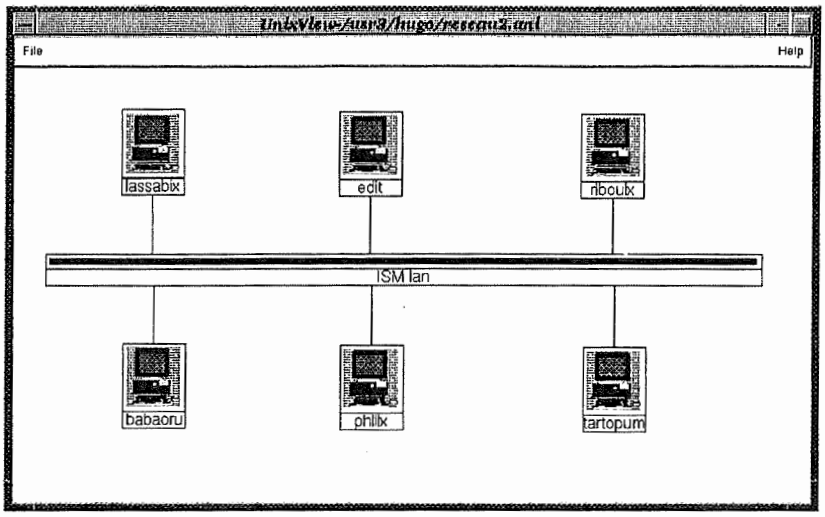


Figure 5.3 : écran principal d'UniXView

A partir de cet écran, l'administrateur peut sélectionner, au moyen de la souris, une machine. Lors de la sélection, un menu se déroule et propose les deux fonctionnalités d'UniXView : obtenir de l'information sur les systèmes de fichiers et sur les utilisateurs. L'utilisateur choisit alors l'opération désirée.

5.3.1 Systèmes de fichiers

Si l'utilisateur choisit d'obtenir de l'information sur les différents systèmes de fichiers montés sur la machine sélectionnée, une nouvelle fenêtre s'affiche. Cette fenêtre est illustrée par la figure 5.4. Dans cet exemple, l'utilisateur avait sélectionné la machine appelée "lassabix". Sur la fenêtre, apparaissent les systèmes de fichiers disponibles pour l'utilisateur de "lassabix". Ils sont représentés par des disques plus ou moins remplis selon le pourcentage de capacité utilisée. A ce pourcentage sont associées des couleurs. Si le taux est de moins de 50 %, le disque est vert; jaune s'il est compris entre 50 et 90 %; rouge lorsque le taux de remplissage dépasse 90 %.

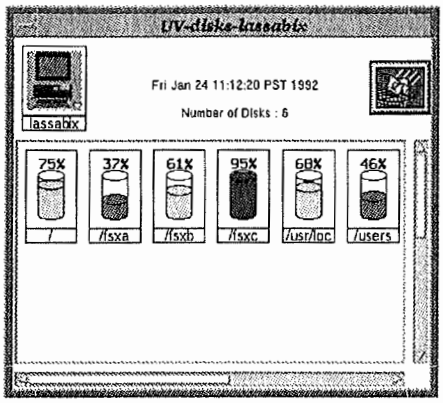


Figure 5.4 : écran des systèmes de fichiers

En cliquant sur l'icône représentant un système de fichiers particulier, l'utilisateur d'UniXView demande des informations supplémentaires à son propos. La figure 5.5 est la fenêtre résultant de cette demande.

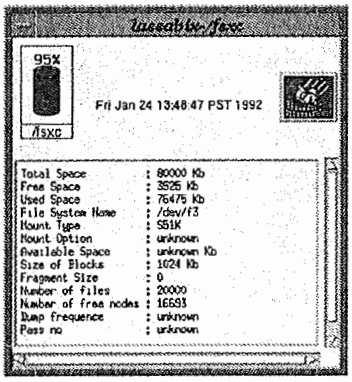


Figure 5.5 : informations sur un système de fichier

Les données supplémentaires sont :

- la capacité totale exprimée en kilo-byte - total space.
- la capacité libre exprimée en kilo-byte - free space.
- la capacité utilisée exprimée en kilo-byte - used space.
- le nom du système de fichiers - file system name.
- le type de montage du système de fichiers - NFS, par exemple - - mount type.

- la taille des blocs -size of blocks.
- la taille de la fragmentation du disque - fragment size.
- le nombre de fichiers - number of files.
- le nombre de nœuds disponibles - number of free nodes.

Les autres informations - mount option, available space, dump frequence, pass no - sont spécifiques à l'UNIX de Bull, BOS2.

5.3.2 Utilisateurs

Si par contre l'administrateur désire obtenir de l'information sur les utilisateurs connectés à une machine, il obtient une fenêtre telle que celle représentée par la figure 5.6. Dans l'exemple, on remarque que "Jeff" s'est connecté deux fois à la machine dont le nom est "tartopum" et le "root", une fois.

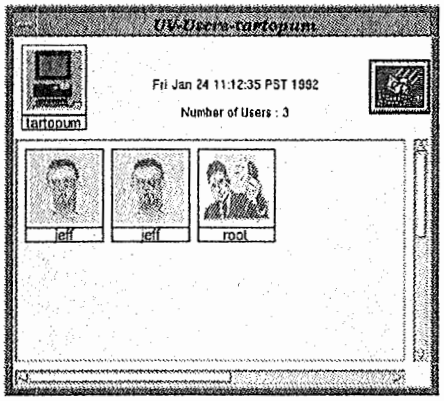


Figure 5.6 : écran des utilisateurs connectés

De la même manière que pour les systèmes de fichiers, l'utilisateur peut obtenir des données sur un utilisateur connecté. Un écran - figure 5.7 - reprend les informations suivantes :

- l'identifiant de l'utilisateur dans le fichier /etc/inittab - Inittab ID.
- le nom de la console sur laquelle l'utilisateur est connecté - Device Name.
- le répertoire principal de l'utilisateur - Home Directory.

- le shell de l'utilisateur - Shell.
- le numéro identifiant l'utilisateur - User ID.
- le numéro identifiant le groupe de l'utilisateur - Group ID.
- l'identifiant du shell process de l'utilisateur - Process ID.
- l'heure et la date à laquelle s'est connecté l'utilisateur - Login Time.

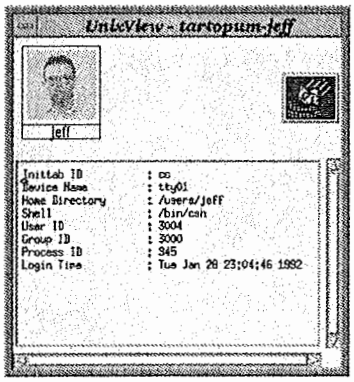


Figure 5.7 : informations sur un utilisateur

5.4 Mécanismes de fonctionnement

L'application UniXView est constitué de différentes parties. Nous avons tout d'abord l'interface graphique qui a été présentée au point 5.3. Cette partie a été codée en ISM-ML. La seconde partie est une extension de l'agent SNMP. sur les machines administratrices. Notons que cette application, pour des raisons d'avancement dans le développement d'ISM n'a pu utiliser le framework de l'ISM Manager. Une solution a été adoptée pour accéder à l'agent directement à partir de l'application. Notons que malgré cette déficience, nous avons intégré UniXView à ISM Monitor. Ce dernier servant alors simplement d'interface avec l'utilisateur. A l'heure actuelle, UniXView doit avoir été implémenté en utilisant intégralement ISM Manager.

5.4.1 Architecture d'UniXView

La figure 5.8 schématise l'architecture d'UniXView. Dans la machine administratrice, nous avons l'application UniXView. Celle-ci dialogue avec les agents SNMP au moyen du

protocole de gestion SNMP. Les agents sont sur les machines gérées. Ils accèdent à la MIB pour effectuer des opérations de gestion demandée par l'utilisateur d'UniXView.

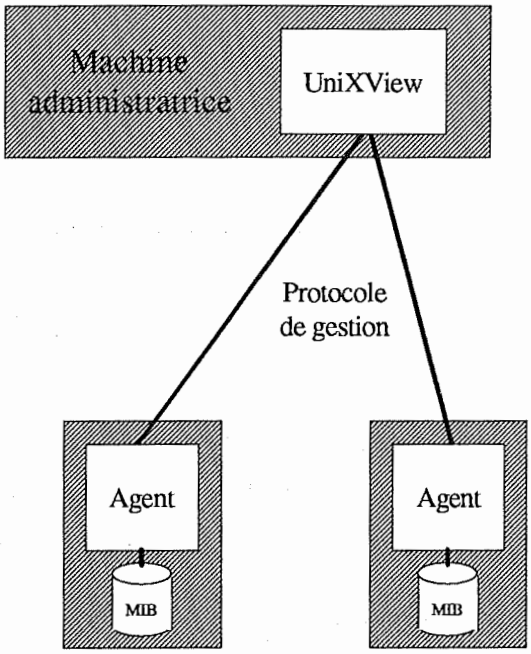


Figure 5.8 : architecture d'UniXView

Pour montrer le fonctionnement d'UniXView, nous allons nous concentrer sur l'une des deux fonctionnalités que cette application offre : le contrôle des systèmes de fichiers. Nous avons vu qu'à partir d'une machine représentée à l'écran, l'administrateur peut obtenir des informations sur les systèmes de fichiers montés sur cette machine. Lorsqu'il en fait la demande, une fenêtre avec l'ensemble des systèmes de fichiers est affichée.

Nous allons présenter les mécanismes d'UniXView en utilisant les notions de la gestion OSI. Cela nous pemettra de nous rappeler le concept de MIB - point 5.4.2 -, les notions de services CMIS et de protocole CMIP - point 5.4.3 - en expliquant le déroulement d'une requête.

5.4.2 Définitions des objets

Comme nous nous limitons à la partie concernant les systèmes de fichiers, nous ne nous occuperons pas des autres objets de la MIB. Afin de définir l'objet "file-system" qui nous intéresse ici, nous nous baserons sur les formulaires spécifiés par les GDMO de l'ISO.

```
file-system          MANAGED OBJECT CLASS
      DERIVED FROM {top};
      CHARACTERIZED BY     fs-package;
REGISTERED AS   {unixview-objectClass 1};
```

Figure 5.9 : définition de la classe d'objets "file-system"

Le comportement et les attributs de la classe "file-system" sont définis par le formulaire illustré par la figure 5.10.

```
fs-package          PACKAGE
      BEHAVIOUR      Lors d'un GET, il faut lancer l'application permettant d'aller
                     chercher l'information sur les systèmes de fichiers puis
                     compléter les valeurs des attributs en fonctions des résultats
      ATTRIBUTES     file-sys-name  GET,
                     mount-name     GET,
                     mount-type     GET,
                     total-space    GET,
                     free-space     GET,
                     size-blocks    GET,
                     frg-size       GET,
                     nb-files       GET,
                     free-nodes     GET;
REGISTERED AS   {unixview-package 1};
```

Figure 5.10 : définition du package de "file-system"

Chaque attribut est défini par un formulaire différent. Il donne entre autres le type de l'attribut. Pour l'attribut "mount-name", une ligne supplémentaire permet de spécifier le mécanisme de filtre. Dans ce cas-ci, le filtre se fera par une relation d'égalité entre la valeur de l'attribut "mount-name" et la valeur spécifiée dans la requête émise par l'application UniXView.

```
file-sys-name          ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      CHARACTER STRING;
REGISTERED AS    {unixview-attribute 1};

mount-name             ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      CHARACTER STRING;
        MATCHES FOR      EQUALITY;
REGISTERED AS    {unixview-attribute 2};

mount-type             ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      CHARACTER STRING;
REGISTERED AS    {unixview-attribute 3};

total-space            ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      INTEGER;
REGISTERED AS    {unixview-attribute 4};

free-space             ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      INTEGER;
REGISTERED AS    {unixview-attribute 5};

size-blocks            ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      INTEGER;
REGISTERED AS    {unixview-attribute 6};

frg-size               ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      INTEGER;
REGISTERED AS    {unixview-attribute 7};

nb-files         ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      INTEGER;
REGISTERED AS    {unixview-attribute 8};

free-nodes             ATTRIBUTE
        WITH ATTRIBUTE SYNTAX      INTEGER;
REGISTERED AS    {unixview-attribute 9};
```

Figure 5.11 : définitions des attributs

Nous voyons dans ce cas que la seule opération autorisée est le GET. Ceci est dû au fait que l'administrateur utilisant UniXView ne fait que des demandes d'informations.

Pour la simplicité de l'exemple, nous n'avons pas tenu compte du fait que la classe d'objet "file-system" peut avoir des relations avec d'autres classes d'objets. Au contraire,

nous l'avons fait directement dépendre de la classe top. La figure 5.12 illustre la partie de l'arbre d'héritage portant sur "file-system".

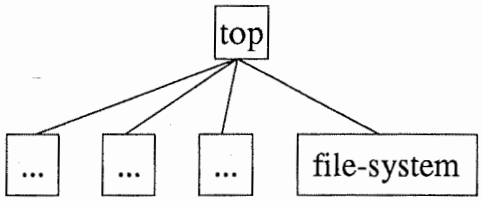


Figure 5.12 : arbre d'héritage pour "file-system"

5.4.3 Déroulement d'une requête

Nous allons maintenant voir quels sont les mécanismes sous-jacents à une requête de la part de l'utilisateur. Admettons qu'il clique au moyen de la souris sur l'icône représentant la machine appelée "lassabix". Il choisit, au déroulement du menu, la fonctionnalité portant sur les systèmes de fichiers. Ceci va amener l'application UniXView à faire une demande de service à CMISE. Cette demande de service est un M-GET portant sur l'objet "file-system". Suivons la demande d'information à propos des systèmes de fichiers de "lassabix" comme l'illustre la figure 5.13.

UniXView dans le rôle d'utilisateur de CMIS, effectue un M-GET-req qui est transporté par le protocole CMIP m-Get-req. L'agent, également utilisateur des services CMIS sur le système distant recevra une indication du M-GET - M-GET-ind. Il consultera sa MIB pour savoir comment répondre à cette requête. Il lancera alors le programme spécifié dans le BEHAVIOUR du formulaire définissant le package "fs-package". Ce programme va lire le fichier "/etc/mnttab" duquel il obtiendra deux informations :

- le chemin d'accès correspondant au répertoire des systèmes de fichiers.
- le nom des systèmes de fichiers.

Grâce à ces données, il est possible, par l'appel-système "statfs", d'obtenir d'autres informations correspondant aux attributs de la classe d'objets "file-system" :

- mount-name
- mount-type
- total-space
- free-space
- size-blocks
- frg-size
- nb-files
- free-nodes

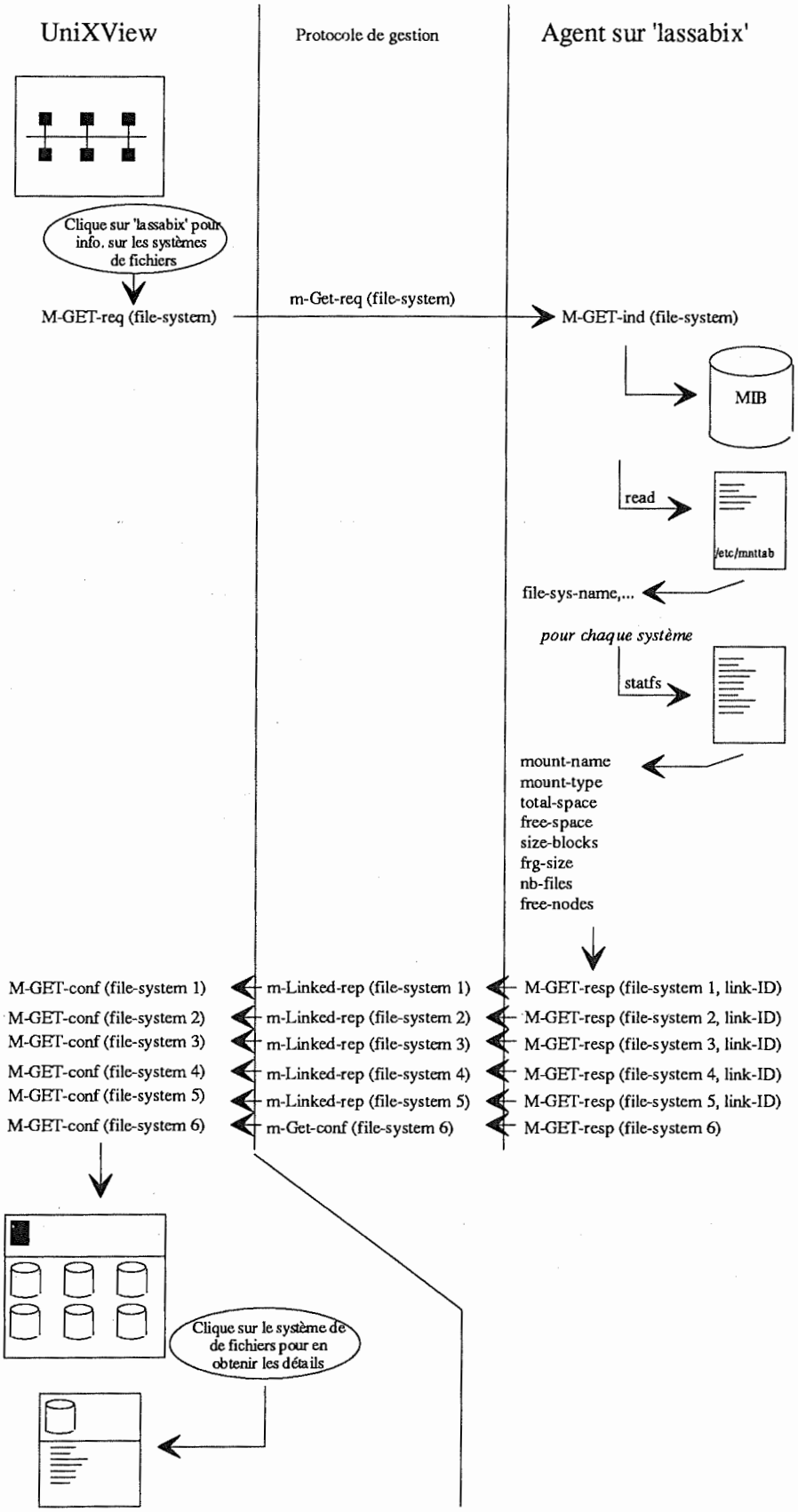


Figure 5.13 : déroulement d'une demande d'informations sur les systèmes de fichiers

Une fois ces opérations effectuées, l'agent fait une demande de services à CMISE. Ces services sont des M-GET multiples pour répondre à la requête d'UniXView. Un paramètre permet de lier ces différentes réponses. Chacun des M-GET-resp est transporté par le protocole CMIP grâce à un m-Linked-Reply-req. Lorsque UniXView reçoit les différentes confirmations à son M-GET, il peut afficher la nouvelle fenêtre illustrée à la figure 5.4.

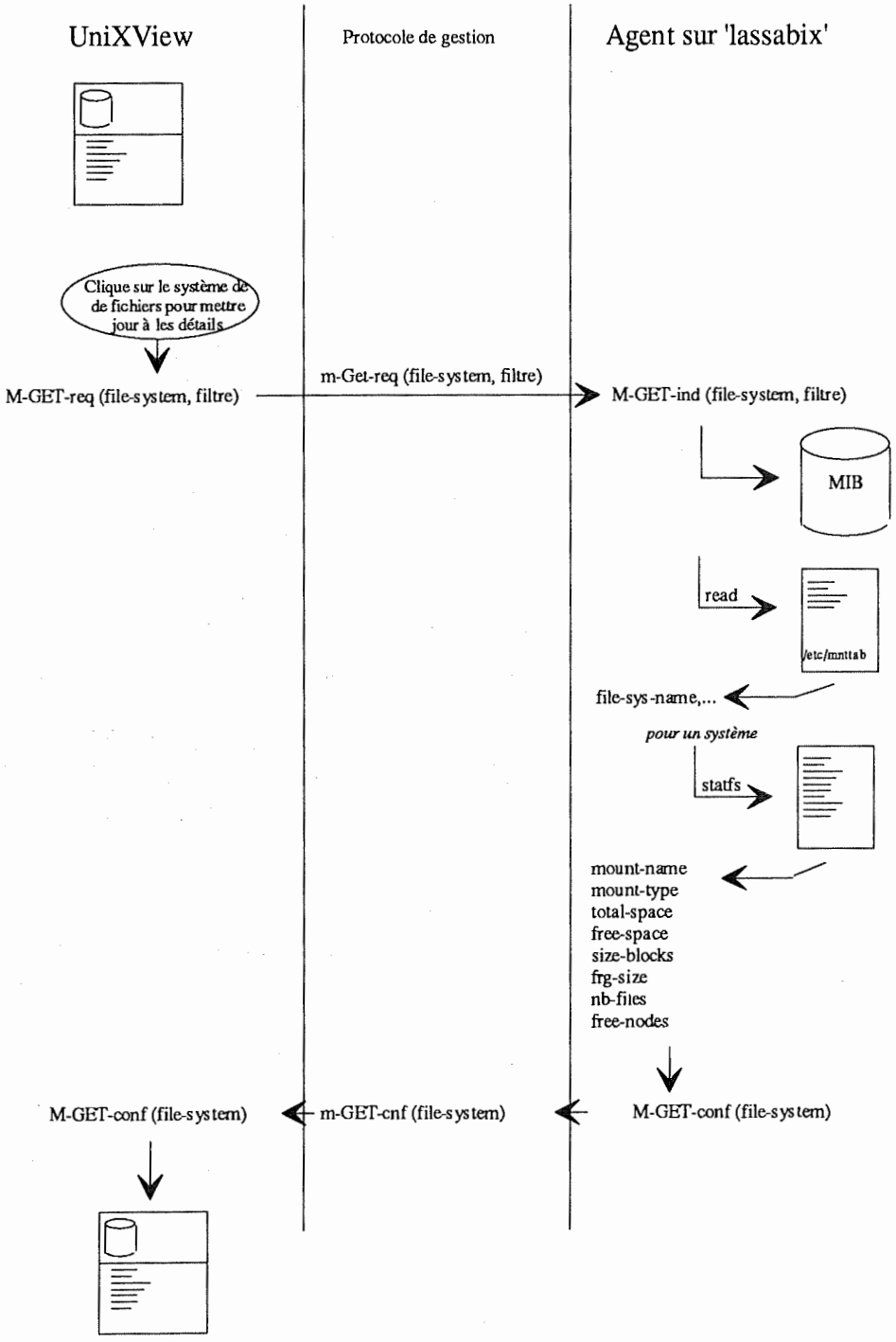


Figure 5.14 : déroulement d'une mise-à-jour

A ce niveau, l'utilisateur peut demander un complément d'informations sur un système de fichiers spécifique. Ces informations sont déjà connues d'UniXView. Il ne lui reste plus qu'à afficher la fenêtre - voir figure 5.5. Si l'administrateur désire mettre-à-jour les informations , il le fait en cliquant sur l'icône représentant le système de fichiers dans le coin supérieur gauche. La figure 5.14 illustre les mécanismes qui découlent d'une telle demande. Ils sont similaires à ceux expliqués ci-dessus. Deux différences cependant : l'utilisation d'un filtre dans la demande de service M-GET et l'absence de réponses multiples. Le filtre portera sur la valeur de l'attribut "mount-name" qui doit être équivalent à celui du système de fichiers concerné par la requête. De plus, il n'y a plus de M-GET-resp liées entre elles puisqu'il n'y a plus qu'une seule instance de "file-system" correspondante à la demande de mise-à-jour.

Conclusion

Conclusion

Au fil des années, les recherches en matière de systèmes informatiques ont permis aux utilisateurs de ces systèmes de passer des systèmes centralisés aux réseaux. L'étape suivante devrait être les systèmes distribués. Les différents types de systèmes distribués que nous avons eu l'occasion de présenter vont de la machine multi-processeurs, partageant une mémoire commune ou non, à une architecture multi-machines se basant fortement sur les réseaux. Alors que les réseaux permettaient d'échanger des informations, les systèmes distribués font bien plus que cela, ils cachent la multiplicité des composants qui le constituent. C'est ce que l'on appelle la transparence. Là est la clé des systèmes distribués. Les utilisateurs de ces systèmes n'ont pas besoin de connaître l'emplacement des ressources - données, processeurs, etc. - pour y accéder. C'est le système d'exploitation distribué qui permet cette transparence. A la différence des OS classiques, il doit prendre en compte la distribution des éléments hardware et software. Il doit également gérer la communication entre les différentes machines pour permettre à des processus de coopérer entre eux de manière invisible à l'utilisateur. Cette contrainte nécessite un système de communications. L'ISO en propose un modèle : le modèle OSI qui permet d'interconnecter des systèmes ouverts dans le but de réaliser une tâche commune. Il faut cependant noter que la différence entre réseaux et systèmes distribués est souvent confuse. Nous sommes encore bien loin des systèmes pleinement distribués, offrant une transparence totale. La recherche avance, cependant, et ouvre de nouveaux horizons en la matière. Ce que l'on propose, à l'heure actuelle, comme systèmes distribués relèvent plutôt de réseaux munis de fonctionnalités permettant de répondre à certains niveaux de transparence - les systèmes de fichiers en sont les premiers bénéficiaires.

Un système distribué n'échappe pas à la règle : pour fournir des services de manière cohérente, il doit être géré. Le travail de l'administrateur n'est pas une mince affaire au vu de la quantité des composants de ce type de système. Cela va des utilisateurs aux applications en passant par la communication, les processeurs, les imprimantes, etc.. Une solution pour attaquer un tel problème est le partitionnement. Cette méthode permet de réduire les composants à administrer et donc à décomplexifier la tâche. Mais réduire en sous-niveaux de gestion ne veut pas dire qu'il ne faut pas offrir aux administrateurs des solutions intégrant l'ensemble des fonctions de gestion. Une administration intégrée permet de réduire la diversité de ces fonctions et offre une vue cohérente du système.

La gestion des communications n'est pas une mince affaire et l'ISO définit là aussi des normes à suivre pour permettre l'administration de systèmes OSI. Les concepts de gestion OSI proposent également l'idée de partitionnement des systèmes pour les

administrer. Elle adopte également une approche orientée-objet pour la définition des problèmes de gestion. C'est ainsi qu'elle spécifie les règles à suivre pour définir, par la notion d'objets, les éléments que l'on doit gérer dans un système. Elle définit également les services communs à toutes activités de gestion de systèmes - CMIS - ainsi que le protocole pour transporter ces services aux systèmes distants - CMIP. De plus en plus, les constructeurs se rendent compte de la nécessité de suivre de tels standards dans la conception des systèmes de gestion qu'ils proposent. Le consortium OSI/NM Forum, qui est constitué de constructeurs justement, propose un profil de normes OSI pour l'administration de réseaux. Il répond également aux interrogations dans certains domaines non couverts par l'ISO. C'est ainsi qu'il a défini des objets de gestion nécessaires à l'administration de réseaux alors que l'ISO n'en est pas encore à ce stade. Il insiste plus particulièrement sur la notion d'interface interopérable. Cet interface est le point de rencontre entre deux systèmes de gestion qu'il appelle CME. L'interface est en fait l'ensemble des protocoles et messages conformes aux normes de l'ISO que le Forum a adoptées dans son profil.

Des produits comme ISM de Bull suivent les propositions du NM Forum. Ce produit d'administration intégrée de systèmes distribués privilégie une ouverture vers des éléments du système à gérer non propriétaires - c'est-à-dire non spécifiques à Bull. C'est ainsi, qu'il permet d'administrer des environnements TCP/IP grâce au protocole d'administration SNMP, des systèmes conformes à l'OSI par CMIP et des systèmes propres à Bull via le protocole DSAC/AEP. Une approche originale - et qui nous paraît intéressante - lors du design d'ISM a été de permettre à tout utilisateur d'étendre la portée de ce produit de gestion à d'autres systèmes - SNA, par exemple - sans devoir modifier ISM. L'extension d'ISM se fait à la manière d'UNIX : par ajout de "drivers". C'est ce que Bull appelle les intégrateurs d'agents. Grâce à des outils de développements - dont le langage ISM-ML -, l'administrateur peut également intégrer à ISM des applications de gestion spécifiques à ses problèmes de gestion.

Les solutions de gestion entièrement conformes à CMIP actuellement sur le marché sont très rares. Même ISM ne l'est pas encore. Ce que nous avons présenté à son sujet concerne une version à venir. Les constructeurs ont préféré tout d'abord s'en tenir au protocole SNMP de l'Internet Activity Board - IAB - responsable de TCP/IP. Cette politique est de bonne augure quand on sait qu'il existe très peu de systèmes OSI. Mais la tendance à migrer de TCP/IP vers OSI - ou à coupler les deux systèmes - se dessine . L'avenir gardera très certainement une place à l'OSI. Le NM Forum en témoigne. La gestion OSI trouvera alors pleinement sa raison d'être.

Références bibliographiques

Références bibliographiques

[ADAMS91] ADAMS Elisabeth K., Global Commonality in User Requirements, in : KRISHNAN I., ZIMMERS W. (eds), *Integrated Network Management II*, Elsevier Science Publishers B.V. (North-Holland), IFIP, 1991, pp. 171-181

[ATTAL91] ATTAL Denis, *Introduction to ISM Architecture*, Bull S.A., 1991.

[BEAUQ90] BEAUQUIER Joffroy, BÉRARD Béatrice, *Systèmes d'exploitation, concepts et algorithmes*, Mc Graw-Hill, Paris, 1990.

[BOUNE90] BOUNEMRA K, *Normalisation des réseaux, la couche application et la gestion de réseaux*, Eyrolles, Paris, 1990.

[BRINS88] BRINSFIELD J.G., GILBERT W.E., Unified Network Management Architecture - Putting It All Together, in : *AT&T Technology*, vol. 3, n° 2, 1988, pp. 6-17.

[CARTE91] CARTER H. Elston, DIA Januario P., Evaluating Network Management Systems : Criteria and Observations, in : KRISHNAN I., ZIMMERS W. (eds), *Integrated Network Management II*, Elsevier Science Publishers B.V. (North-Holland), IFIP, 1991, pp. 213-223.

[CELIA90] CELIA A. Joseph, KURUDI H. Muralidhar, Integrated Network Management in an Enterprise Environment, in : *IEEE Network Magazine*, vol. 4, n° 4, Juillet 1990, pp. 7-13.

[COULO89] COULOURIS George F., DOLLIMORE Jean, *Distributed systems : concepts and design*, Addison-Wesley Publishing Company, Workingham, 1989.

[DANTH89] DANTHINE André A.S., Communications support for distributed systems, OSI versus special protocols, in : RITTER G.X. (eds), *Information Processing*, Elsevier Science Publishers B.V., North-Holland, 1989.

[EMBRY90] EMBRY Jock, MANSON Peter, MILHAM Dave, An Open Network Management Architecture : OSI/NM Forum Architecture and Concepts, in : *IEEE Network Magazine*, vol. 4, n° 4, july 1990, pp. 14-22.

[EMBRY91] EMBRY Jock, MANSON Peter, MILHAM Dave, Interoperable Network Management : OSI/NM Forum architecture and concepts, in : KRISHNAN I., ZIMMER W. (eds), *Integrated Network Management II*, Elsevier Science Publishers B.V. (North-Holland), IFIP, 1991, pp. 29-44.

[EMSLE91a] EMSLEY Ian, *ISM Global High Level Design Document*, Revision 2, Bull S.A., 1991.

[EMSLE91b] EMSLEY P., GREGOIRE P., *Uniview, External Interface Specification*, Revision 7, Bull S.A., 1991.

[FORUM02] OSI/NETWORK MANAGEMENT FORUM, *Application Services*, Forum 002, issue 1.1, June 1990.

[FORUM04] OSI/NETWORK MANAGEMENT FORUM, *Forum Architecture*, Forum 004, issue 1.0, January 1990.

[FORUM05] OSI/NETWORK MANAGEMENT FORUM, *Forum Glossary*, Forum 005, issue 1.0, January 1990.

[FORUM06] OSI/NETWORK MANAGEMENT FORUM, *Forum Library of Managed Object Classes, Name Bindings and Attributes*, Forum 006, issue 1.1, June 1990.

[GABAS92] GABASSI Michel, DUPOUY Bertrand, *L'informatique répartie sous UNIX*, Eyrolles, Paris, 1992.

[HEYVA91] HEYVAERT Didier, *Network Management Systems, The management of TCP/IP networks*, Facultés Universitaires Notre-Dame de la Paix, Institut d'Informatique, Namur, 1991.

[ISO10040] ISO/IEC 10040, *Information technology - Open Systems Interconnection - Systems Management Overview,* International Standard Organisation, 1991.

[ISO10164-1] ISO/IEC 10165-1, *Information technology - Open Systems Interconnection - Systems Management - Part 1 : Object Management Function,* International Standard Organisation, 1991.

[ISO10164-2] ISO/IEC 10165-2, *Information technology - Open Systems Interconnection - Systems Management - Part 2 : State Management Function,* International Standard Organisation, 1991.

[ISO10164-3] ISO/IEC 10165-3, *Information technology - Open Systems Interconnection - Systems Management - Part 3 : Attributes for representing Relationships,* International Standard Organisation, 1991.

[ISO10164-4] ISO/IEC 10165-4, *Information technology - Open Systems Interconnection - Systems Management - Part 4 : Alarm Reporting Function,* International Standard Organisation, 1991.

[ISO10164-5] ISO/IEC 10165-5, *Information technology - Open Systems Interconnection - Systems Management - Part 5 : Event Reporting Management Function,* International Standard Organisation, 1991.

[ISO10164-6] ISO/IEC 10165-1, *Information technology - Open Systems Interconnection - Systems Management - Part 6 : Log Control Function,* International Standard Organisation, 1991.

[ISO10164-7] ISO/IEC 10165-7, *Information technology - Open Systems Interconnection - Systems Management - Part 7 : Security Alarm Reporting Function,* International Standard Organisation, 1991.

[ISO10165-1] ISO/IEC 10165-1, *Information Processing Systems - Open Systems Interconnection, Structure of Management Information - Part 1 : Management Information Model,* International Standard Organisation, 1991.

[ISO10165-2] ISO/IEC 10165-2, *Information Processing Systems - Open Systems Interconnection, Structure of Management Information - Part 2 : Definition of Management Information,* International Standard Organisation, 1991

[ISO10165-4] ISO/IEC 10165-4, *Information Processing Systems - Open Systems Interconnection, Structure of Management Information - Part 4 : Guidelines for the Definition of Managed Objects,* International Standard Organisation, 1991

[ISO7498] ISO/IEC 7498, *Information Processing Systems - Open Systems Interconnection, Basic Reference Model*, International Standards Organization, 15 octobre 1984.

[ISO7498-4] ISO/IEC 7498-4, *Information Processing Systems - Open Systems Interconnection, Basic Reference Model - Part 4 : Management Framework*, International Standards Organization, 15 novembre 1989.

[ISO8649] ISO/IEC 8649, *Information Processing Systems - Open Systems Interconnection, Service Definition for the Association Control Service Element*, International Standard Organisation, 1987.

[ISO8650] ISO/IEC 8650, *Information Processing Systems - Open Systems Interconnection, Protocol Specification for the Association Control Service Element*, International Standard Organisation, 1987.

[ISO8824] ISO/IEC 8824, *Information technology - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1),* International Standard Organisation, 1990.

[ISO9072-1] ISO/IEC 9072-1, *Information Processing Systems - Text Communication, Remote Operations - Part 1 : Model, notation and Service Definition*, International Standard Organisation, 1989.

[ISO9072-2] ISO/IEC 9072-2, *Information Processing Systems - Text Communication, Remote Operations - Part 2 : Protocol Specification*, International Standard Organisation, 1989.

[ISO9545] ISO/IEC 9545, *Information Processing Systems - Open Systems Interconnection, Application Layer Structure*, International Standard Organisation, 1989.

[ISO9595] ISO/IEC 9595, *Information technology - Open Systems Interconnection - Common Management Information Service Definition*, International Standard Organisation, 1991.

[ISO9596] ISO/IEC 9596, *Information technology - Open Systems Interconnection - Common Management Information Protocol Specification*, International Standard Organisation, 1991.

[KIESEL91] KIESEL W., DEIRETSBACHER K.H., Communications Network for Manufacturing Applications (CNMA) - A European Initiative for Network Management in Industrial Communication, in : KRISHNAN I., ZIMMERS W. (eds), *Integrated Network Management II*, Elsevier Science Publishers B.V. (North-Holland), IFIP, 1991, pp. 245-257.

[KLERE88] KLERER S. Mark, The OSI Management Architecture : an Overview, in : *IEEE Network*, vol 2, n° 2, mars 1988, pp. 20-29.

[MAZDA91] MAZDA Fraidoon, Structured network management, in : *Communications international*, vol. 8, n° 12, December 1991, pp. 53-58.

[MILLE91] MILLER Paul, *ISM Global Features Specification*, Revision 0.49, Bull S.A., 1991.

[MILLE92] MILLER Paul, *ISM, Integrated System Management*, Bull S.A., 1992.

[MOFFE91] MOFFETT Jonathan D., SLOMAN Morris S., Delegation of Authority, in : KRISHNAN I., ZIMMERS W. (eds), *Integrated Network Management II*, Elsevier Science Publishers B.V. (North-Holland), IFIP, 1991, pp. 595-606.

[MULLE90] MULLENDER S.J., VAN ROSSUM G., TANENBAUM A.S., VAN RENESSE R., VAN STAVEREN H., Amœba : a distributed operating system for the 1990s, in : *IEEE Computer*, vol. 23, n° 5, mai 1990, pp. 44-53.

[NILLE91] NILLES Romain, *Remote Procedure Call - Théorie et implémentation*, Facultés Universitaires Notre-Dame de la Paix, Institut d'Informatique, Namur, 1991.

[NMFOR90] OSI/NETWORK MANAGEMENT FORUM, *Forum Architecture - Forum 004*, Issue 1.0, Janvier 1990.

[NOBLO91] NOBLOT C., ANGOULMINE Nazim, *La gestion de réseaux*, Bull, 1991.

[POTTE91] POTTER Duncan, The need for network management, in : *Computer Communications*, vol. 14, n° 2, mars 1991, pp. 121-125.

[PUJOL92] PUJOLLE Guy, *Télécommunications et réseaux*, Eyrolles, Paris, 1992.

[RFC 1157] CASE Jeffrey D., FEDOR Mark S., SCHOFFSTALL Martin L., DAVIN James R., *A Simple Network Management Protocol (SNMP)*, Request For Comments 1157, DDN Network Information Center, SRI International, mai 1990.

[ROBIN88] ROBINSON D.C., SLOMAN M.S., Domain Based Access Control for Distributed Computing Systems, in : *Software Engineering Journal*, vol. 3, n° 5, septembre 1988, pp 161-170.

[SLOMA89] SLOMAN M.S., MOFFETT J.D., Domain Management for Distributed Systems, in : MEANDZIJA, WESTCOTT (eds), *Proceedings of the IFIP Symposium on Integrated Network Management*, Boston, USA, Mai 1989, North-Holland, pp. 505-516.

[SLOMA90] SLOMAN Morris, MOFFETT Jonathan, *Managing Distributed Systems*, notes préparatives, Imperial College, Department of Computing, London, 10 août 1990.

[STRUT91] STRUTT Colin, Dealing With Scale In An Enterprise Management Director, in : KRISHNAN I., ZIMMERS W. (eds), *Integrated Network Management II*, Elsevier Science Publishers B.V. (North-Holland), IFIP, 1991, pp. 577-593.

[TANEN85] TANENBAUM A.S., VAN RENESSE Robbert, Distributed Operating Systems, in : *Computing Surveys*, vol. 17, n° 4, ACM, New-York Décembre 1985.

[TANEN89] TANENBAUM A., *Les systèmes d'exploitation, conception et mise en œuvre*, InterEditions, Paris, 1989.

[TANEN90] TANENBAUM A., *Réseaux, architectures, protocoles, applications*, InterEditions, Paris, 1990.

Glossaire

Glossaire

ACSE	Association Control Service Element
AE	Application Entity
AEP	Administrative Exchange Protocol
AI	Agent Integrator
ALS	Application Layer Structure
AP	Application Process
API	Application Programmatic Interface
ASE	Application Service Element
ASN.1	Abstract Syntax Notation One
AT&T	American Telegraph and Telephone
CASE	Common Application Service Element
CCITT	Comité Consultatif International Télégraphique et Téléphonique
CCRSE	Commitment Concurrency and Recovery Service Element
CME	Conformant Management Entity
CMIP	Common Management Information Protocol
CMIS	Common Management Information Service
CMISE	Common Management Information Service Element
DCM	Distributed Computing Model
DEC	Digital Equipment Corporation
DN	Distinguished Name
DS	Directory Service
DSA	Distributed System Architecture
DSAC	Distributed Systems Administration and Control
DTAM	Document Transfer, Access and Management
EFD	Event Forwarding Discriminator
EMA	Enterprise Management Architecture
FDDI	Fiber Distributed Data Interface
FTAM	File Transfer, Access and Management
FTP	File Transfer Protocol
GDMO	Guidelines for the Definition of Managed Objects
HP	Hewlett - Packard
IBM	International Business Machine
IMP	Interface Message Processor
IP	Internet Protocol
IPC	Interprocess Communication
ISM	Integrated System Management
ISM-AL	ISM Alarm
ISM-ML	ISM Management Language
ISM-MN	ISM Monitor
ISM-PF	ISM Performance
ISM-RO	ISM Remote Operation
ISM-UM	ISM User Management
ISO	International Standards Organization

JTM	Job Transfer and Manipulation
LAN	Local Area Network
LM	Layer Management
MACF	Multiple Association Control Function
MAN	Metropolitan Area Network
MHS	Message Handling System
MIB	Management Information Base
MIS	Management Information Service
MIT	Management Information Tree
MMS	Manufacturing Message Service
MS-DOS	Microsoft - Disk Operating System
NCMS	Network Connection Management Subprotocol
NFS	Network File System
NM Forum	Network Management Forum
ODA	Office Document Architecture
OS	Operating System
OSF	Open Software Foundation
OSI	Open Systems Inteconnection
OSI/NM Forum	OSI/Network Management Forum
P+M	Protocoles + Messages
PC	Personnal Computer
RDA	Remote Database Access
RDN	Relative Distinguished Name
ROSE	Remote Operation Service Element
RPC	Remote Procedure Call
RTSE	Reliable Transfer Service Element
SACF	Single Association Control Function
SAO	Single Association Object
SASE	Specific Application Service Element
SCO	Santa Cruz Operation
SMAE	System Management Application Entity
SMAP	System Management Application Protocol
SMASE	System Management Application Service Element
SMF	System Management Function
SMFA	System Management Functional Area
SMI	Structure of Management Information
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VT	Virtual Terminal
WAN	Wide Area Network

Annexes

Annexe 1

Formulaires pour la définition des objets gérés

Conventions syntaxiques

- [ ] délimite un champ optionnel.
- [ ]* champ pouvant apparaître zero ou plusieurs fois.
- <> indique un champ à remplacer dans chaque instance.
- | séparateur de champ alternatif (opérateur OU)
- ; marque la fin d'une construction
- Les mots clé sont en majuscules.
- Le champ "label" est composé de caractères alphabétiques, numériques, "-", "/" et commence par une minuscule.

Formulaire de définition des classes d'objets gérés

```
<class-label>  MANAGED OBJECT CLASS
        [DERIVED FROM   <class-label> [,<class-label>]*;
        ]
        [CHARACTERIZED BY     <package-label> [,<package-label>]*;
        ]
        [CONDITIONAL PACKAGES
                <package-label>       PRESENT IF condition-definition
                [,<package-label>     PRESENT IF condition-definition]*;
        ]
REGISTERED AS   object-identifier;

condition-definition   →      delimited-string
-- "delimited-string" : use of a natural language text or formal description techniques.
```

Formulaire de définition d'un package

```
<package-label>      PACKAGE
      [BEHAVIOUR       <behaviour-definition-label>
                       [,<behaviour-definition-label>]*;
      ]
      [ATTRIBUTES      <attribute-label> propertylist [<parameter-label>]*
                       [,<attribute-label> propertylist [<parameter-label>]*]*;
      ]
      [ATTRIBUTE GROUPS    <group-label> [<attribute-label>]*
                           [,<group-label> [<attribute-label>]*]*;
      ]
      [ACTIONS     <action-label> [<parameter-label>]*
                   [,<action-label> [<parameter-label>]*]*;
      ]
      [NOTIFICATIONS   <notification-label> [<parameter-label>]*
                       [,<notification-label> [<parameter-label>]*]*;
[REGISTERED AS   object-identifier];

propertylist   →     [REPLACE-WITH-DEFAULT]
                     [DEFAULT-VALUE value-specifier]
                     [INITIAL VALUE    value-specifier]
                     [PERMITTED VALUES     type-reference]
                     [REQUIRED VALUES      type-reference]
                     [get-replace]
                     [add-remove]
value-specifier →    value-reference | DERIVATION RULE <behaviour-definition-label>
get-replace    →     GET | REPLACE | GET-REPLACE
add-remove     →     ADD | REMOVE | ADD-REMOVE
```

Formulaire de définition d'un paramètre

```
<parameter-label>    PARAMETER
      CONTEXT    context-type;
      syntax-or-attribute-choice;
      [BEHAVIOUR       <behaviour-definition-label>
                       [,<behaviour-definition-label>]*;
      ]
[REGISTERED AS   object-identifier];
context-type   →     context-keyword |
                     ACTION-INFO |
                     ACTION-REPLY |
                     EVENT-INFO |
                     EVENT-REPLY |
                     SPECIFICERROR
context-keyword      →     type-reference <identifier>
syntax-or-attribute-choice     →     WITH SYNTAX type-reference; |
                                     ATTRIBUTE <attribute-label>
```

Formulaire pour la définition d'un groupe d'attributs

```
<group-label> ATTRIBUTE GROUP
        [GROUP ELEMENTS        <attribute-label> [,<attribute-label>]*;
        ]
        [FIXED;
        ]
        [DESCRIPTION       delimited-string;
        ]
REGISTERED AS    object-identifier;
```

Formulaire pour la définition d'un comportement

```
<behaviour-definition-label>  BEHAVIOUR
        DEFINED AS          delimited-string;

--      The "delimited-string" gives a definition of the behaviour of a managed object class
        or of the associated attributes, actions or notifications.
--      May be documented by use of a naturel language text or formal description
        techniques.
```

Formulaire pour la définition d'une action

```
<action-label> NOTIFICATION
        [BEHAVIOUR          <behaviour-definition-label>
                            [,<behaviour-definition-label>]*;
        ]
        [PARAMETERS         <parameter-label> [,<parameter-label>]*;
        ]
        [WITH INFORMATION SYNTAX          type-reference;
        ]
        [WITH REPLY SYNTAX    type-reference;
        ]
REGISTERED AS    object-identifier;
```

Formulaire pour la définition d'une notification

```
<notification-label>   NOTIFICATION
        [BEHAVIOUR          <behaviour-definition-label>
                            [,<behaviour-definition-label>]*;
        ]
        [PARAMETERS         <parameter-label> [,<parameter-label>]*;
        ]
        [WITH INFORMATION SYNTAX            type-reference
                [AND ATTRIBUTES IDS    <field-name> <attribute-label>
                                       [,<field-name> <attribute-label>]*
                ];
        ]
        [WITH REPLY SYNTAX     type-reference;
        ]
REGISTERED AS     object-identifier;
```

Annexe 2

Paramètres des procédures CMIS

Extraits de [ISO9595]

8.2 Management notification service

The M-EVENT REPORT service is used by a CMISE-service-user to report an event to a peer CMISE-service-user. It is defined as a confirmed and a non-confirmed service.

8.2.1 M-EVENT-REPORT parameters

Table 4 lists the parameters for this service.

Table 4 — M-EVENT-REPORT parameters

Parameter Name	Req/Ind	Rsp/Conf
Invoke identifier	M	M(=)
Mode	M	—
Managed object class	M	U
Managed object instance	M	U
Event type	M	C(=)
Event time	U	—
Event information	U	—
Current time	—	U
Event reply	—	C
Errors	—	C

8.2.1.1 Invoke identifier

This parameter specifies the identifier assigned to the notification. It can be used to distinguish this notification from other notifications or operations that the CMISE-service-provider may have in progress.

8.2.1.2 Mode

This parameter specifies the mode requested for the operation. The possible values are

— confirmed;

— non-confirmed.

8.2.1.3 Managed object class

This parameter specifies the class of the managed object in which the event occurred. It may be included in any confirmation.

8.2.1.4 Managed object instance

This parameter specifies the instance of the managed object in which the event occurred. It may be included in any confirmation.

8.2.1.5 Event type

This parameter specifies the type of event being reported. It may be included in the success confirmation and shall be included if the event reply parameter is included.

8.2.1.6 Event time

This parameter contains the time of generation of the event.

8.2.1.7 Event information

This parameter contains information that the invoking CMISE-service-user is able to supply about the event.

8.2.1.8 Current time

This parameter contains the time at which the response was generated. It may be included in the success confirmation.

8.2.1.9 Event reply

This parameter contains the reply to the event report. It may be included in the success confirmation.

8.2.1.10 Errors

This parameter contains the error notification for the operation. It shall be included in the failure confirmation. The following errors may occur

— duplicate invocation: the invoke identifier specified was allocated to another notification or operation;

— invalid argument value: the event information value specified was out of range or otherwise inappropriate;

— mistyped argument: one of the parameters supplied has not been agreed for use on the association between the CMISE-service-users;

— no such argument: the event information specified was not recognized;

— no such event type: the event type specified was not recognized;

— no such object class: the class of the specified managed object was not recognized;

— no such object instance: the instance of the specified managed object was not recognized;

— processing failure: a general failure in processing the notification was encountered;

— resource limitation: the notification was not processed due to resource limitation;

— unrecognized operation: the operation is not one of those agreed between the CMISE-service-users.

8.2.2 M-EVENT-REPORT procedures

8.2.2.1 The invoking CMISE-service-user reports an event to a performing CMISE-service-user by issuing an M-EVENT-REPORT request primitive to the CMISE-service-provider.

8.2.2.2 If the CMISE-service-provider accepts the request, then it issues an M-EVENT-REPORT indication primitive to the performing CMISE-service-user. Otherwise, the CMISE-service-provider rejects the request and the following procedures do not apply.

8.2.2.3 In the confirmed mode, the performing CMISE-service-user reports acceptance or rejection of the M-EVENT-REPORT request primitive by issuing an M-EVENT-REPORT response primitive to the CMISE-service-provider.

8.2.2.4 In the confirmed mode, the CMISE-service-provider issues an M-EVENT-REPORT confirm primitive to the invoking CMISE-service-user.

8.3 Management operation services

8.3.1 M-GET service

The M-GET service is used by a CMISE-service-user to retrieve attribute values from a peer CMISE-service-user. It is defined as a confirmed service. This service may be cancelled by an invocation of the M-CANCEL-GET service.

8.3.1.1 M-GET parameters

Table 5 lists the parameters for this service.

Table 5 — M-GET parameters

Parameter Name	Req/Ind	Rsp/Conf
Invoke identifier	M	M
Linked identifier	—	C
Base object class	M	—
Base object instance	M	—
Scope	U	—
Filter	U	—
Access control	U	—
Synchronization	U	—
Attribute identifier list	U	—
Managed object class	—	C
Managed object instance	—	C
Current time	—	U
Attribute list	—	C
Errors	—	C

8.3.1.1.1 Invoke identifier

This parameter specifies the identifier assigned to the operation. It can be used to distinguish this operation from other notifications or operations that the CMISE-service-provider may have in progress.

Each response shall have a unique invoke identifier; the final response shall have an invoke identifier equal to that of the invoke identifier provided in the indication primitive.

8.3.1.1.2 Linked identifier

If multiple replies are to be sent for this operation, then this parameter specifies the identification that is provided by the performing CMISE-service-user when those replies are returned. The linked identifier shall have the same value as that of the invoke identifier provided in the indication primitive.

8.3.1.1.3 Base object class

This parameter specifies the class of the managed object that is to be used as the starting point for the selection of managed objects to which the filter (when supplied) is to be applied.

8.3.1.1.4 Base object instance

This parameter specifies the instance of the managed object that is to be used as the starting point for the selection of managed object to which the filter (when supplied) is to be applied.

8.3.1.1.5 Scope

This parameter indicates the subtree, rooted at the base managed object, which is to be searched. The levels of search that may be performed are

— the base object alone;

— the n^{th} level subordinates of the base object;

— the base object and all of its subordinates down to and including the n^{th} level;

— the base object and all of its subordinates.

The default scope is the base object alone.

8.3.1.1.6 Filter

This parameter specifies the set of assertions that defines the filter test to be applied to the scoped managed object(s). Multiple assertions may be grouped using the logical operators AND, OR and NOT. Each assertion may be a test for equality, ordering, presence, or set comparison. Assertions about the value of an attribute are evaluated according to the matching rules associated with the attribute syntax. If an attribute value assertion is present in the filter and that attribute is not present in the scoped managed object, then the result of the test for that attribute value assertion shall be evaluated as FALSE. The managed object(s) for which the filter test evaluates to TRUE is(are) selected for the application of the operation. If the filter is not specified, all of the managed objects included by the scope are selected.

8.3.1.1.7 Access control

This parameter is information of unspecified form to be used as input to access control functions.

8.3.1.1.8 Synchronization

This parameter indicates how the invoking CMISE-service-user wants the M-GET operations synchronized across the selected object instances. Two ways of synchronizing a series of operations are defined

— Atomic: All managed objects selected for the operation are checked to ascertain if they are able to successfully perform the operation. If one or more is not able to successfully perform the operation, then none perform it, otherwise all perform it.

— Best effort: All managed objects selected for the operation are requested to perform it.

If this parameter is not supplied, best effort synchronization is performed. If the base managed object alone is selected for the operation, this parameter (if present) is ignored.

8.3.1.1.9 Attribute identifier list

This parameter contains a set of attribute identifiers for which the attribute values are to be returned by the performing CMISE-service-user. If this parameter is omitted, all attribute identifiers are assumed. The definitions of the attributes are found in the specification of the managed object class.

8.3.1.1.10 Managed object class

If the base object alone is specified, then this parameter is optional; otherwise it shall specify the class of the managed object whose attribute values are returned. It may be included in any confirmation.

8.3.1.1.11 Managed object instance

If the base object alone is specified, then this parameter is optional; otherwise it shall specify the instance of the managed object whose attribute values are returned. It may be included in any confirmation.

8.3.1.1.12 Current time

This parameter contains the time at which the response was generated. It may be included in the success confirmation.

8.3.1.1.13 Attribute list

This parameter contains the set of attribute identifiers and values that are returned by the performing CMISE-service-user. It shall be included in the success confirmation.

8.3.1.1.14 Errors

This parameter contains the error notification for the operation. It shall be included in the failure confirmation. The following errors may occur

— access denied: the requested operation was not performed for reasons pertinent to the security of the open system;

— class instance conflict: the specified managed object instance is not a member of the specified class;

— complexity limitation: the requested operation was not performed because a parameter was too complex;

— duplicate invocation: the invoke identifier specified was allocated to another notification or operation;

— get list error: one or more attribute values were not read for one of the following reasons

— access denied: the requested operation was not performed for reasons pertinent to the security of the open system;

— no such attribute: the identifier for the specified attribute or attribute group was not recognized.

The attribute values that could be read are returned;

— invalid filter: the filter parameter contains an invalid assertion or an unrecognized logical operator;

— invalid scope: the value of the scope parameter is invalid;

— mistyped argument: one of the parameters supplied has not been agreed for use on the association between the CMISE-service-users;

— no such object class: the class of the specified managed object was not recognized;

— no such object instance: the instance of the specified managed object was not recognized;

— operation cancelled: the operation was cancelled by an M-CANCEL-GET operation, and no further attribute values will be returned by this invocation of the M-GET service;

— processing failure: a general failure in processing the operation was encountered;

— resource limitation: the operation was not performed due to resource limitation;

— synchronization not supported: the type of synchronization specified is not supported;

— unrecognized operation: the operation is not one of those agreed between the CMISE-service-users.

8.3.1.2 M-GET procedures

8.3.1.2.1 The invoking CMISE-service-user requests a performing CMISE-service-user to retrieve attribute value(s) by issuing an M-GET request primitive to the CMISE-service-provider.

8.3.1.2.2 If the CMISE-service-provider accepts the request, then it issues an M-GET indication primitive to the performing CMISE service-user. Otherwise, the CMISE-service-provider rejects the request and the following procedures do not apply.

8.3.1.2.3 If the operation cannot be performed, then the performing CMISE-service-user rejects the M-GET request by issuing an M-GET response primitive with the appropriate error code. In this case, the following procedures do not apply.

8.3.1.2.4 If only one response is to be generated, then procedures [§§ l] 8.3.1.2.5, 8.3.1.2.6 and 8.3.1.2.7 shall be ignored.

8.3.1.2.5 The performing CMISE-service-user retrieves the requested attribute value(s) and generates a response which includes results and/or error notifications. The linked identifier shall be present in the service primitive, with its value to be set equal to the invoke identifier of the indication primitive, and the managed object class and instance shall be present.

8.3.1.2.6 The CMISE-service-provider issues an M-GET confirm primitive to the invoking CMISE-service-user which shall include the linked identifier and managed object class and instance.

8.3.1.2.7 Procedures [§§ l] 8.3.1.2.5 and 8.3.1.2.6 shall be repeated until all replies have been completed.

8.3.1.2.8 The completion of the response is indicated by the performing CMISE-service-user issuing an M-GET response primitive which shall not contain the linked identifier, and, if there were linked responses generated by procedures [§§ l] 8.3.1.2.5 and 8.3.1.2.6, which shall not contain the attribute list.

8.3.1.2.9 The CMISE-service-provider issues an M-GET confirm primitive to the invoking CMISE-service-user, completing the M-GET operation.

8.3.1.3 M-CANCEL-GET service

The M-CANCEL-GET service is used by an invoking CMISE-service-user to request the cancellation of a previously requested and currently outstanding invocation of the M-GET service by a peer CMISE-service-user. It is defined as a confirmed service.

8.3.1.3.1 M-CANCEL-GET parameters

Table 6 lists the parameters for this service.

Table 6 — M-CANCEL-GET parameters

Parameter Name	Req/Ind	Rsp/Conf
Invoke identifier	M	M(=)
Get invoke identifier	M	—
Errors	—	C

8.3.1.3.1.1 Invoke identifier

This parameter specifies the identifier assigned to the operation. It can be used to distinguish this operation from other notifications or operations that the CMISE-service-provider may have in progress.

8.3.1.3.1.2 Get invoke identifier

This parameter specifies the identifier assigned to the previously requested and currently outstanding M-GET operation.

8.3.1.3.1.3 Errors

This parameter contains the error notification for the operation. It shall be included in the failure confirmation. The following errors may occur

— duplicate invocation: the invoke identifier specified was already allocated to another notification or operation;

— mistyped operation: the get invoke identifier parameter did not refer to an M-GET operation;

— no such invoke identifier: the get invoke identifier parameter was not recognized;

— processing failure: a general failure in processing the operation was encountered;

— resource limitation: the operation was not performed due to resource limitation;

— unrecognized operation: the operation is not one of those agreed between the CMISE-service-users.

8.3.1.3.2 M-CANCEL-GET procedures

8.3.1.3.2.1 The invoking CMISE-service-user requests a performing CMISE-service-user to cancel a previously requested and currently outstanding M-GET operation by issuing an M-CANCEL-GET request primitive to the CMISE-service-provider.

8.3.1.3.2.2 If the CMISE-service-provider accepts the request, then it issues an M-CANCEL-GET indication primitive to the performing CMISE-service-user. Otherwise, the CMISE-service-provider rejects the request and the following procedures do not apply.

8.3.1.3.2.3 If the operation cannot be performed, then the performing CMISE-service-user rejects the M-CANCEL-GET request by issuing an M-CANCEL-GET response primitive with the appropriate error code. In this case, the following procedures do not apply.

8.3.1.3.2.4 The performing CMISE-service-user cancels the outstanding M-GET operation and issues an M-GET response primitive which shall contain the operation cancelled error and which shall not contain the managed object class and managed object instance parameters, and an M-CANCEL-GET response primitive to the CMISE-service-provider.

8.3.1.3.2.5 If there are any M-GET response primitives issued by the performing CMISE-service-user after the invoking CMISE-service-user issues the M-CANCEL-GET request primitive, but before the performing CMISE-service-user receives the M-CANCEL-GET indication primitive, then the invoking CMISE-service-user shall receive M-GET confirm primitives for those M-GET response primitives.

8.3.1.3.2.6 The CMISE-service-provider issues an M-GET confirm primitive to the invoking CMISE-service-user which shall include the operation cancelled error indication, completing the M-GET operation, and an M-CANCEL-GET confirm primitive to the invoking CMISE-service-user, completing the M-CANCEL-GET operation.

8.3.2 M-SET service

The M-SET service is used by an invoking CMISE-service-user to request the modification of attribute values by a peer CMISE-service-user. It is defined as a confirmed and a non-confirmed service.

8.3.2.1 M-SET parameters

Table 7 lists the parameters for this service.

Table 7 — M-SET parameters

Parameter Name	Req/Ind	Rsp/Conf
Invoke identifier	M	M
Linked identifier	—	C
Mode	M	—
Base object class	M	—
Base object instance	M	—
Scope	U	—
Filter	U	—
Access control	U	—
Synchronization	U	—
Managed object class	—	C
Managed object instance	—	C
Modification list	M	—
Attribute list	—	U
Current time	—	U
Errors	—	C

8.3.2.1.1 Invoke identifier

This parameter specifies the identifier assigned to the operation. It can be used to distinguish this operation from other notifications or operations that the CMISE-service-provider may have in progress.

Each response shall have a unique invoke identifier; the final response shall have an invoke identifier equal to that of the invoke identifier provided in the indication primitive.

8.3.2.1.2 Linked identifier

If multiple replies are to be sent for this operation, then this parameter specifies the identification that is provided by the performing CMISE-service-user when those replies are returned. The linked identifier shall have the same value as that of the invoke identifier provided in the indication primitive.

8.3.2.1.3 Mode

This parameter specifies the mode requested for the operation. The possible values are

— confirmed;

— non-confirmed.

8.3.2.1.4 Base object class

This parameter specifies the class of the managed object that is to be used as the starting point for the selection of managed objects t which the filter (when supplied) is to be applied.

8.3.2.1.5 Base object instance

This parameter specifies the instance of the managed object that is to be used as the starting point for the selection of managed object to which the filter (when supplied) is to be applied.

8.3.2.1.6 Scope

This parameter indicates the subtree, rooted at the base managed object, which is to be searched. The levels of search that may b performed are

— the base object alone;

— the n^{th} level subordinates of the base object;

— the base object and all of its subordinates down to and including the n^{th} level;

— the base object and all of its subordinates.

The default scope is the base object alone.

8.3.2.1.7 Filter

This parameter specifies the set of assertions that defines the filter test to be applied to the scoped managed object(s). Multipl assertions may be grouped using the logical operators AND, OR and NOT. Each assertion may be a test for equality, ordering, presence or set comparison. Assertions about the value of an attribute are evaluated according to the matching rules associated with the attribut syntax. If an attribute value assertion is present in the filter and that attribute is not present in the scoped managed object, then th result of the test for that attribute value assertion shall be evaluated as FALSE. The managed object(s) for which the filter test evaluate to TRUE are selected for the application of the operation. If the filter is not specified, all of the managed objects included by the scop are selected.

8.3.2.1.8 Access control

This parameter is information of unspecified form to be used as input to access control functions.

8.3.2.1.9 Synchronization

This parameter indicates how the invoking CMISE-service-user wants the M-SET operations synchronized across the selected objec instances. Two ways of synchronizing a series of operations are defined

— Atomic: All managed objects selected for the operation are checked to ascertain if they are able to succes fully perform th operation. If one or more is not able to successfully perform the operation, then none perform it, otherwise all perform it.

— Best effort: All managed objects selected for the operation are requested to perform it.

If this parameter is not supplied, best effort synchronization is performed. If the base managed object alone is selected for th operation, this parameter (if present) is ignored.

8.3.2.1.10 Managed object class

If the base object alone is specified, then this parameter is optional; otherwise it shall specify the class of the managed object whose attribute values were modified. It may be included in any confirmation.

8.3.2.1.11 Managed object instance

If the base object alone is specified, then this parameter is optional; otherwise it shall specify the instance of the managed object whose attribute values were modified. It may be included in any confirmation.

8.3.2.1.12 Modification list

This parameter contains a set of attribute modification specifications, each of which contains

—attribute identifier: the identifier of the attribute or attribute group whose value(s) is/are to be modified. An attribute group identifier shall only be specified when the set to default modify operator is specified;

—attribute value: the value(s) to be used in the modification of the attribute. The use of this parameter is defined by the modify operator. This parameter is optional when the set to default modify operator is specified and if supplied, shall be ignored;

—modify operator: the way in which the attribute value(s) (if supplied) is/are to be applied to the attribute. The possible operators are

— replace: the attribute value(s) specified shall be used to replace the current value(s) of the attribute;

— add values: the attribute value(s) specified shall be added to the current value(s) of the attribute. This operator shall only be applied to a set-valued attribute and shall perform a set union (in the mathematical sense) between the current value(s) of the attribute and the attribute value(s) specified. Value(s) specified in the attribute value parameter which are already in the current value(s) of the attribute shall not cause an error to be returned;

— remove values: the attribute value(s) specified shall be removed from the current value(s) of the attribute. This operator shall only be applied to a set-valued attribute and shall perform a set difference (in the mathematical sense) between the current value(s) of the attribute and the attribute value(s) specified. Value(s) specified in the attribute value parameter which are not in the current value(s) of the attribute shall not cause an error to be returned;

— set to default: when this operator is applied to a single-valued attribute, the value of the attribute shall be set to its default value. When this operator is applied to a set-valued attribute, the value(s) of the attribute shall be set to their default value(s) and only as many values as defined by the default shall be assigned. When this operator is applied to an attribute group, each member of the attribute group shall be set to its default value(s). If there is no default value defined, the "invalid operation" error shall be returned.

The modify operator is optional, and if it is not specified, the replace operator shall be assumed.

8.3.2.1.13 Attribute list

This parameter contains a set of attributes (one for each of the attribute identifiers specified in the modification list), each with its value(s) following the modification. It may be included in the success confirmation.

8.3.2.1.14 Current time

This parameter contains the time at which the response was generated. It may be included in the success confirmation.

8.3.2.1.15 Errors

This parameter contains the error notification for the operation. It shall be included in the failure confirmation. The following errors may occur

— access denied: the requested operation was not performed for reasons pertinent to the security of the open system;

— class instance conflict: the specified managed object instance is not a member of the specified class;

— complexity limitation: the requested operation was not performed because a parameter was too complex;

— duplicate invocation: the invoke identifier specified was allocated to another notification or operation;

— invalid filter: the filter parameter contains an invalid assertion or an unrecognized logical operator;

— invalid scope: the value of the scope parameter is invalid;

— mistyped argument: one of the parameters supplied has not been agreed for use on the association between the CMISE-service-users.

— no such object class: the class of the specified managed object was not recognized;

— no such object instance: the instance of the specified managed object was not recognized;

— processing failure: a general failure in processing the operation was encountered;

— resource limitation: the operation was not performed due to resource limitation;

— set list error: one or more attribute values were not modified for one of the following reasons

— access denied: the requested operation was not performed for reasons pertinent to the security of the open system;

— invalid attribute value: the attribute value specified was out of range or otherwise inappropriate;

— invalid operator: the modify operator specified is not recognized;

— invalid operation: the modify operator specified may not be performed on the specified attribute;

— no such attribute: the identifier for the specified attribute was not recognized.

The attribute values that could be modified, were modified;

— synchronization not supported: the type of synchronization specified is not supported;

— unrecognized operation: the operation is not one of those agreed between the CMISE-service-users.

8.3.2.2 M-SET procedures

8.3.2.2.1 The invoking CMISE-service-user requests a performing CMISE-service-user to modify attribute value(s) by issuing an M-SET request primitive to the CMISE-service-provider.

8.3.2.2.2 If the CMISE-service-provider accepts the request, then it issues an M-SET indication primitive to the performing CMISE-service-user. Otherwise, the CMISE-service-provider rejects the request and the following procedures do not apply.

8.3.2.2.3 In the non-confirmed mode the following procedures shall be ignored.

8.3.2.2.4 If the operation cannot be performed, then the performing CMISE-service-user rejects the M-SET request by issuing an M-SET response primitive with the appropriate error code. In this case, the following procedures do not apply.

8.3.2.2.5 If only one response is to be generated, then procedures [§§ I] 8.3.2.2.6, 8.3.2.2.7 and 8.3.2.2.8 shall be ignored.

8.3.2.2.6 The performing CMISE-service-user modifies the requested attribute value(s) and generates a response which includes results and/or error notifications. The linked identifier shall be present in the service primitive, with its value to be set equal to the invoke identifier of the indication primitive, and the managed object class and instance shall be present.

8.3.2.2.7 The CMISE-service-provider issues an M-SET confirm primitive to the invoking CMISE-service-user which shall include the linked identifier and managed object class and instance.

8.3.2.2.8 Procedures [§§ I] 8.3.2.2.6 and 8.3.2.2.7 shall be repeated until all replies have been completed.

8.3.2.2.9 The completion of the response is indicated by the performing CMISE-service-user issuing an M-SET response primitive which shall not contain the linked identifier, and, if there were linked responses generated by procedures [§§ I] 8.3.2.2.6 and 8.3.2.2.7, which shall not contain the attribute list.

8.3.2.2.10 The CMISE-service-provider issues an M-SET confirm primitive to the invoking CMISE-service-user, completing the M-SET operation.

8.3.3 M-ACTION service

The M-ACTION service is used by a CMISE-service-user to request a peer CMISE-service-user to perform an action on managed object(s). It is defined as a confirmed and a non-confirmed service.

8.3.3.1 M-ACTION parameters

Table 8 lists the parameters for this service.

Table 8 — M-ACTION parameters

Parameter Name	Req/Ind	Rsp/Conf
Invoke identifier	M	M
Linked identifier	—	C
Mode	M	—
Base object class	M	—
Base object instance	M	—
Scope	U	—
Filter	U	—
Managed object class	—	C
Managed object instance	—	C
Access control	U	—
Synchronization	U	—
Action type	M	C(=)
Action information	U	—
Current time	—	U
Action reply	—	C
Errors	—	C

8.3.3.1.1 Invoke identifier

This parameter specifies the identifier assigned to the operation. It can be used to distinguish this operation from other notification or operations that the CMISE-service-provider may have in progress.

Each response shall have a unique invoke identifier; the final response shall have an invoke identifier equal to that of the invok identifier provided in the indication primitive.

8.3.3.1.2 Linked identifier

If multiple replies are to be sent for this operation, then this parameter specifies the identification that is provided by the performin CMISE-service-user when those replies are returned. The linked identifier shall have the same value as that of the invoke identifi provided in the indication primitive.

8.3.3.1.3 Mode

This parameter specifies the mode requested for the operation. The possible values are

— confirmed;

— non-confirmed.

8.3.3.1.4 Base object class

This parameter specifies the class of the managed object that is to be used as the starting point for the selection of managed objects which the filter (when supplied) is to be applied.

8.3.3.1.5 Base object instance

This parameter specifies the instance of the managed object that is to be used as the starting point for the selection of managed objec to which the filter (when supplied) is to be applied.

8.3.3.1.6 Scope

This parameter indicates the subtree, rooted at the base managed object, which is to be searched. The levels of search that may performed are

— the base object alone;

— the n^{th} level subordinates of the base object;

— the base object and all of its subordinates down to and including the n^{th} level;

— the base object and all of its subordinates.

The default scope is the base object alone.

8.3.3.1.7 Filter

This parameter specifies the set of assertions that defines the filter test to be applied to the scoped managed object(s). Multiple assertions may be grouped using the logical operators AND, OR and NOT. Each assertion may be a test for equality, ordering, presence, or set comparison. Assertions about the value of an attribute are evaluated according to the matching rules associated with the attribute syntax. If an attribute value assertion is present in the filter and that attribute is not present in the scoped managed object, then the result of the test for that attribute value assertion shall be evaluated as FALSE. The managed object(s) for which the filter test evaluates to TRUE are selected for the application of the operation. If the filter is not specified, all of the managed objects included by the scope are selected.

8.3.3.1.8 Managed object class

If the base managed object alone is specified, then this parameter is optional; otherwise it shall specify the class of the managed object that performed the action. It may be included in any confirmation.

8.3.3.1.9 Managed object instance

If the base object alone is specified, then this parameter is optional; otherwise it shall specify the instance of the managed object that performed the action. It may be included in any confirmation.

8.3.3.1.10 Access control

This parameter is information of unspecified form to be used as input to access control functions.

8.3.3.1.11 Synchronization

This parameter indicates how the invoking CMISE-service-user wants the M-ACTION operations synchronized across the selected object instances. Two ways of synchronizing a series of operations are defined

— Atomic: All managed objects selected for the operation are checked to ascertain if they are able to successfully perform the operation. If one or more is not able to successfully perform the operation, then none perform it, otherwise all perform it.

— Best effort: All managed objects selected for the operation are requested to perform it.

If this parameter is not supplied, best effort synchronization is performed. If the base managed object alone is selected for the operation, this parameter (if present) is ignored.

8.3.3.1.12 Action type

This parameter specifies a particular action that is to be performed. It may be included in the success confirmation and shall be included if the action reply parameter is included.

8.3.3.1.13 Action information

This parameter specifies extra information when necessary to further define the nature, variations, or operands of the action to be performed. The syntax and semantics of the parameter depend upon the action requested.

8.3.3.1.14 Current time

This parameter contains the time at which the response was generated. It may be included in the success confirmation.

8.3.3.1.15 Action reply

This parameter contains the reply to the action. It may be included in the success confirmation.

8.3.3.1.16 Errors

This parameter contains the error notification for the operation. It shall be included in the failure confirmation. The following errors may occur

— access denied: the requested operation was not performed for reasons pertinent to the security of the open system;

— class instance conflict: the specified managed object instance is not a member of the specified class;

— complexity limitation: the requested operation was not performed because a parameter was too complex;

— duplicate invocation: the invoke identifier specified was allocated to another notification or operation;

— invalid argument value: the action information value specified was out of range or otherwise inappropriate;

— invalid filter: the filter parameter contains an invalid assertion or an unrecognized logical operator;

— invalid scope: the value of the scope parameter is invalid;

— mistyped argument: one of the parameters supplied has not been agreed for use on the association between the CMISE-service-users;

— no such action: the action type specified was not supported;

— no such argument: the action information specified was not supported;

— no such object class: the class of the specified managed object was not recognized;

— no such object instance: the instance of the specified managed object was not recognized;

— processing failure: a general failure in processing the operation was encountered;

— resource limitation: the operation was not performed due to resource limitation;

— synchronization not supported: the type of synchronization specified is not supported;

— unrecognized operation: the operation is not one of those agreed between the CMISE-service-users.

8.3.3.2 M-ACTION procedures

8.3.3.2.1 The invoking CMISE-service-user requests a performing CMISE-service-user to perform an action on a set of managed objects by issuing an M-ACTION request primitive to the CMISE-service-provider.

8.3.3.2.2 If the CMISE-service-provider accepts the request, then it issues an M-ACTION indication primitive to the performing CMISE-service-user. Otherwise, the CMISE-service-provider rejects the request and the following procedures do not apply.

8.3.3.2.3 In the non-confirmed mode the following procedures shall be ignored.

8.3.3.2.4 If the operation cannot be performed, then the performing CMISE-service-user rejects the M-ACTION request by issuing an M-ACTION response primitive with the appropriate error code. In this case, the following procedures do not apply.

8.3.3.2.5 If only one response is to be generated, then procedures [§§ I] 8.3.3.2.6, 8.3.3.2.7 and 8.3.3.2.8 shall be ignored.

8.3.3.2.6 The performing CMISE-service-user applies the action to the managed object and generates a response which includes a result or an error notification. The linked identifier shall be present in the service primitive, with its value to be set equal to the invoke identifier of the indication primitive, and the managed object class and instance shall be present.

8.3.3.2.7 The CMISE-service-provider issues an M-ACTION confirm primitive to the invoking CMISE-service-user which shall include the linked identifier and managed object class and instance.

8.3.3.2.8 Procedures [§§ I] 8.3.3.2.6 and 8.3.3.2.7 shall be repeated until all replies have been completed.

8.3.3.2.9 The completion of the response is indicated by the performing CMISE-service-user issuing an M-ACTION response primitive which shall not contain the linked identifier, and, if there were linked responses generated by procedures [§§ I] 8.3.3.2.6 and 8.3.3.2.7, which shall not contain the action result.

8.3.3.2.10 The CMISE-service-provider issues an M-ACTION confirm primitive to the invoking CMISE-service-user, completing the M-ACTION operation.

8.3.4 M-CREATE service

The M-CREATE service is used by an invoking CMISE-service-user to request a peer CMISE-service-user to create a new managed object instance, complete with its identification and the values of its associated management information, and simultaneously to register its identification. It is defined as a confirmed service.

8.3.4.1 M-CREATE parameters

Table 9 lists the parameters for this service.

Table 9 — M-CREATE parameters

Parameter Name	Req/Ind	Rsp/Conf
Invoke identifier	M	M(=)
Managed object class	M	U
Managed object instance	U	C
Superior object instance	U	—
Access control	U	—
Reference object instance	U	—
Attribute list	U	C
Current time	—	U
Errors	—	C

8.3.4.1.1 Invoke identifier

This parameter specifies the identifier assigned to the operation. It can be used to distinguish this operation from other notifications or operations that the CMISE-service-provider may have in progress.

8.3.4.1.2 Managed object class

This parameter specifies the class of the new managed object instance which is to be created by the performing CMISE-service-user. The performing CMISE-service-user assigns to the new managed object instance, a set of attribute values as specified by the definition of its class. If the reference object instance field is not supplied by the invoking CMISE-service-user, those attributes whose values are not assigned in the attribute list field are assigned a set of default values as specified by the definition of the new managed object's class. This parameter may be included in any confirmation.

8.3.4.1.3 Managed object instance

This parameter specifies the instance of the managed object which is to be registered by the performing CMISE-service-user. If neither the managed object instance nor the superior object instance parameters is supplied by the invoking CMISE-service-user, then the performing CMISE-service-user assigns a value to this identification of instance. This parameter may be included in the success confirmation and shall be included if it is not supplied by the invoking CMISE-service-user.

8.3.4.1.4 Superior object instance

This parameter identifies the existing managed object instance which is to be the superior of the new managed object instance. If this parameter is supplied by the invoking CMISE-service-user, then the managed object instance parameter shall not be supplied.

8.3.4.1.5 Access control

This parameter is information of unspecified form to be used as input to access control functions.

8.3.4.1.6 Reference object instance

When this parameter is supplied by the invoking CMISE-service-user, it must specify an existing instance of a managed object, called the reference object, of the same class as the managed object to be created. Attribute values associated with the reference object instance become the default values for those not specified by the attribute list parameter.

8.3.4.1.7 Attribute list

When this parameter is supplied by the invoking CMISE-service-user, it contains a set of attribute identifiers and values that the performing CMISE-service-user is to assign to the new managed object instance. These values override the values for the corresponding attributes derived from either the reference object (if the reference object instance field is supplied) or the default value set specified in the definition of the managed object's class. When returned by the performing CMISE-service-user, this parameter contains the complete list of all attribute identifiers and values that were assigned to the new managed object instance. It may be included in the success confirmation.

8.3.4.1.8 Current time

This parameter contains the time at which the response was generated. It may be included in the success confirmation.

8.3.4.1.9 Errors

This parameter contains the error notification for the operation. It shall be included in the failure confirmation. The following error may occur

— access denied: the requested operation was not performed for reasons pertinent to the security of the open system;

— class instance conflict: the specified managed object instance may not be created as a member of the specified class;

— duplicate invocation: the invoke identifier specified was allocated to another notification or operation;

— duplicate managed object instance: the new managed object instance value supplied by the invoking CMISE-service-user was already registered for a managed object of the specified class;

— invalid attribute value: an attribute value specified was out of range or otherwise inappropriate;

— invalid object instance: the object instance name specified implied a violation of the naming rules;

— missing attribute value: a required attribute value was not supplied, and a default value was not available;

— mistyped argument: one of the parameters supplied has not been agreed for use on the association between the CMISE-service-users;

— no such attribute: an attribute specified was not recognized;

— no such object class: the class of the specified managed object was not recognized;

— no such object instance: the instance of the specified superior managed object was not recognized;

— no such reference object: the reference object instance parameter was not recognized;

— processing failure: a general failure in processing the operation was encountered;

— resource limitation: the operation was not performed due to resource limitation;

— unrecognized operation: the operation is not one of those agreed between the CMISE-service-users.

8.3.4.2 M-CREATE procedures

8.3.4.2.1 The invoking CMISE-service-user requests the creation and registration of a new managed object instance by issuing an M-CREATE request primitive to the CMISE-service-provider.

8.3.4.2.2 If the CMISE-service-provider accepts the request, then it issues an M-CREATE indication primitive to the performing CMISE-service-user. Otherwise, the CMISE-service-provider rejects the request and the following procedures do not apply.

8.3.4.2.3 The performing CMISE-service-user creates and registers the new managed object instance or rejects the M-CREATE request, and issues an M-CREATE response primitive to the CMISE-service-provider.

8.3.4.2.4 The CMISE-service-provider issues an M-CREATE confirm primitive to the invoking CMISE-service-user.

8.3.5 M-DELETE service

The M-DELETE service is used by an invoking CMISE-service-user to request a peer CMISE-service-user to delete managed object instance(s) and to deregister their identification. It is defined as a confirmed service.

8.3.5.1 M-DELETE parameters

Table 10 lists the parameters for this service.

Table 10 — M-DELETE parameters

Parameter Name	Req/Ind	Rsp/Conf
Invoke identifier	M	M
Linked identifier	—	C
Base object class	M	—
Base object instance	M	—
Scope	U	—
Filter	U	—
Access control	U	—
Synchronization	U	—
Managed object class	—	C
Managed object instance	—	C
Current time	—	U
Errors	—	C

8.3.5.1.1 Invoke identifier

This parameter specifies the identifier assigned to the operation. It can be used to distinguish this operation from other notifications or operations that the CMISE-service-provider may have in progress.

Each response shall have a unique invoke identifier; the final response shall have an invoke identifier equal to that of the invoke identifier provided in the indication primitive.

8.3.5.1.2 Linked identifier

If multiple replies are to be sent for this operation, then this parameter specifies the identification that is provided by the performing CMISE-service-user when those replies are returned. The linked identifier shall have the same value as that of the invoke identifier provided in the indication primitive.

8.3.5.1.3 Base object class

This parameter specifies the class of the managed object that is to be used as the starting point for the selection of managed objects to which the filter (when supplied) is to be applied.

8.3.5.1.4 Base object instance

This parameter specifies the instance of the managed object that is to be used as the starting point for the selection of managed objects to which the filter (when supplied) is to be applied.

8.3.5.1.5 Scope

This parameter indicates the subtree, rooted at the base managed object, which is to be searched. The levels of search that may be performed are

— the base object alone;

— the n^{th} level subordinates of the base object;

— the base object and all of its subordinates down to and including the n^{th} level;

— the base object and all of its subordinates.

The default scope is the base object alone.

8.3.5.1.6 Filter

This parameter specifies the set of assertions that defines the filter test to be applied to the scoped managed object(s). Multiple assertions may be grouped using the logical operators AND, OR and NOT. Each assertion may be a test for equality, ordering, presence, or set comparison. Assertions about the value of an attribute are evaluated according to the matching rules associated with the attribute syntax. If an attribute value assertion is present in the filter and that attribute is not present in the scoped managed object, then the result of the test for that attribute value assertion shall be evaluated as FALSE. The managed object(s) for which the filter test evaluates to TRUE are selected for deletion. If the filter is not specified, all of the managed objects included by the scope are selected.

8.3.5.1.7 Access control

This parameter is information of unspecified form to be used as input to access control functions.

8.3.5.1.8 Synchronization

This parameter indicates how the invoking CMISE-service-user wants the M-DELETE operations synchronized across the selected object instances. Two ways of synchronizing a series of operations are defined

— Atomic: All managed objects selected for the operation are checked to ascertain if they are able to successfully perform the operation. If one or more is not able to successfully perform the operation, then none perform it, otherwise all perform it.

— Best effort: All managed objects selected for the operation are requested to perform it.

If this parameter is not supplied, best effort synchronization is performed. If the base managed object alone is selected for the operation, this parameter (if present) is ignored.

8.3.5.1.9 Managed object class

If the base object alone is specified, then this parameter is optional; otherwise it shall specify the class of the managed object which was deleted. It may be included in any confirmation.

8.3.5.1.10 Managed object instance

If the base object alone is specified, then this parameter is optional; otherwise it shall specify the instance of the managed object which was deleted. It may be included in any confirmation.

8.3.5.1.11 Current time

This parameter contains the time at which the response was generated. It may be included in the success confirmation.

8.3.5.1.12 Errors

This parameter contains the error notification for the operation. It shall be included in the failure confirmation. The following errors may occur

— access denied: the requested operation was not performed for reasons pertinent to the security of the open system;

— class instance conflict: the specified managed object instance is not a member of the specified class;

— complexity limitation: the requested operation was not performed because a parameter was too complex;

— duplicate invocation: the invoke identifier specified was allocated to another operation;

— invalid filter: the filter parameter contains an invalid assertion or an unrecognized logical operator;

— invalid scope: the value of the scope parameter is invalid;

— mistyped argument: one of the parameters supplied has not been agreed for use on the association between the CMISE-service-users;

— no such object class: the class of the specified managed object was not recognized;

— no such object instance: the instance of the specified managed object was not recognized;

— processing failure: a general failure in processing the operation was encountered;

— resource limitation: the operation was not performed due to resource limitation;

— synchronization not supported: the type of synchronization specified is not supported;

— unrecognized operation: the operation is not one of those agreed between the CMISE-service-users.

8.3.5.2 M-DELETE procedures

8.3.5.2.1 The invoking CMISE-service-user requests a performing CMISE-service-user to delete managed object(s) by issuing an M-DELETE request primitive to the CMISE-service-provider.

8.3.5.2.2 If the CMISE-service-provider accepts the request, then it issues an M-DELETE indication primitive to the performing CMISE-service-user. Otherwise, the CMISE-service-provider rejects the request and the following procedures do not apply.

8.3.5.2.3 If the operation cannot be performed, then the performing CMISE-service-user rejects the M-DELETE request by issuing an M-DELETE response primitive with the appropriate error code. In this case, the following procedures do not apply.

8.3.5.2.4 If only one response is to be generated, then procedures [§§ l] 8.3.5.2.5, 8.3.5.2.6 and 8.3.5.2.7 shall be ignored.

8.3.5.2.5 The performing CMISE-service-user deletes the specified managed object and generates a response. The linked identifier shall be present in the service primitive, with its value to be set equal to the invoke identifier of the indication primitive, and the managed object identifier shall be present.

8.3.5.2.6 The CMISE-service-provider issues an M-DELETE confirm primitive to the invoking CMISE-service-user which shall include the linked identifier and managed object identifier.

8.3.5.2.7 Procedures [§§ l] 8.3.5.2.5 and 8.3.5.2.6 shall be repeated until all replies have been completed.

8.3.5.2.8 The completion of the response is indicated by the performing CMISE-service-user issuing an M-DELETE response primitive which shall not contain the linked identifier, and, if there were linked responses generated by procedures [§§ l] 8.3.5.2.5 and 8.3.5.2.6, which shall not contain the delete result.

8.3.5.2.9 The CMISE-service-provider issues an M-DELETE confirm primitive to the invoking CMISE-service-user, completing the M-DELETE operation.

Comparaison entre les opérations sur les objets et les services CMIS

MODELE INFORMATIONNEL		*CMISE*	
Opération / Notification		Service	Mode
Opérations	Get attribute value	M–GET M–CANCEL–GET	confirmé confirmé
	Replace attribute value Replace with default value Add member Remove member	M–SET	confirmé / non–confirmé
	Create	M–CREATE	confirmé
	Delete	M–DELETE	confirmé
	Action	M–ACTION	confirmé / non–confirmé
Notification		M–EVENT–REPORT	confirmé / non–confirmé

Annexe 3

Unités de données CMIP

Extraits de [ISO9596]

The protocol is described in terms of Common Management Information Protocol Data Units exchanged between the peer CMISEs. The PDUs are specified using ASN.1 and the Remote Operations Protocol OPERATION and ERROR external macros defined in [Recommendation X.219 [8] / ISO/IEC 9072-1].

```
-- Common Management Information Protocol (CMIP)

CMIP-1 {joint-iso-ccitt ms(9) cmip(1) modules(0) protocol(3)}
DEFINITIONS ::= BEGIN

-- Remote Operations definitions

IMPORTS OPERATION, ERROR FROM Remote-Operation-Notation {joint-iso-ccitt remoteOperations(4) notation(0)}

-- Remote Operations Service definitions

InvokeIDType FROM Remote-Operations-APDUs {joint-iso-ccitt remoteOperations(4) apdus(1)}

-- Directory Service definitions

DistinguishedName, RDNSequence FROM InformationFramework {joint-iso-ccitt ds(5) modules(1) informationFramework(1)};

-- CMISE operations

-- in the following operations, the argument type is mandatory in the corresponding ROSE APDU

-- Action operations (M-ACTION)

m-Action OPERATION
  ARGUMENT ActionArgument
  ::= localValue 6

m-Action-Confirmed OPERATION
  ARGUMENT      ActionArgument
  RESULT        ActionResult
  -- this result is conditional; for conditions see {Recommendation X.710 § / ISO/IEC 9595 subclause} 8.3.3.2.9
  ERRORS {      accessDenied, classInstanceConflict, complexityLimitation, invalidScope, invalidArgumentValue,
                invalidFilter, noSuchAction, noSuchArgument, noSuchObjectClass, noSuchObjectInstance,
                processingFailure, syncNotSupported }
  LINKED {      m-Linked-Reply }
  ::= localValue 7

m-CancelGet OPERATION
  ARGUMENT
    getInvokeId     InvokeIDType
  RESULT
  ERRORS {      mistypedOperation, noSuchInvokeId, processingFailure }
  ::= localValue 10

-- Create operation (M-CREATE)

m-Create OPERATION
  ARGUMENT      CreateArgument
  RESULT        CreateResult
  -- this result is conditional; for conditions see {Recommendation X.710 § / ISO/IEC 9595 subclause} 8.3.4.1.3
  ERRORS {      accessDenied, classInstanceConflict, duplicateManagedObjectInstance, invalidAttributeValue,
                invalidObjectInstance, missingAttributeValue, noSuchAttribute, noSuchObjectClass,
                noSuchObjectInstance, noSuchReferenceObject, processingFailure }
  ::= localValue 8
```

```
  -- Delete operation (M-DELETE)

  m-Delete OPERATION
    ARGUMENT      DeleteArgument
    RESULT        DeleteResult
    -- this result is conditional; for conditions see {Recommendation X.710 § | ISO/IEC 9595 subclause} 8.3.5.2.8
    ERRORS {      accessDenied, classInstanceConflict, complexityLimitation, invalidFilter, invalidScope,
                  noSuchObjectClass, noSuchObjectInstance, processingFailure, syncNotSupported }
    LINKED {      m-Linked-Reply }
    ::= localValue 9

  -- Event Reporting operations (M-EVENT-REPORT)

  m-EventReport OPERATION
    ARGUMENT EventReportArgument
    ::= localValue 0

  m-EventReport-Confirmed OPERATION
    ARGUMENT      EventReportArgument
    RESULT        EventReportResult -- optional
    ERRORS {      invalidArgumentValue, noSuchArgument, noSuchEventType, noSuchObjectClass,
                  noSuchObjectInstance, processingFailure }
    ::= localValue 1

  -- Get operation (M-GET)

  m-Get OPERATION
    ARGUMENT      GetArgument
    RESULT        GetResult
    -- this result is conditional; for conditions see {Recommendation X.710 § | ISO/IEC 9595 subclause} 8.3.1.2.8
    ERRORS {      accessDenied, classInstanceConflict, complexityLimitation, getListError, invalidFilter, invalidScope,
                  noSuchObjectClass, noSuchObjectInstance, operationCancelled, processingFailure, syncNotSupported }
    LINKED {      m-Linked-Reply }
    ::= localValue 3

  -- Linked operation to M-GET, M-SET (Confirmed), M-ACTION (Confirmed), and M-DELETE

  m-Linked-Reply OPERATION
    ARGUMENT      LinkedReplyArgument
    ::= localValue 2

-- Set operations (M-SET)

m-Set OPERATION
  ARGUMENT SetArgument
  ::= localValue 4

m-Set-Confirmed OPERATION
  ARGUMENT      SetArgument
  RESULT        SetResult
  -- this result is conditional; for conditions see {Recommendation X.710 § | ISO/IEC 9595 subclause} 8.3.2.2.9
  ERRORS {      accessDenied, classInstanceConflict, complexityLimitation, invalidFilter, invalidScope, noSuchObjectClass,
                noSuchObjectInstance, processingFailure, setListError, syncNotSupported }
  LINKED {      m-Linked-Reply }
  ::= localValue 5

-- CMIS error definitions

-- in the following errors, unless otherwise indicated, the parameter type is mandatory in the corresponding ROSE APDU

accessDenied ERROR
  ::= localValue 2

classInstanceConflict ERROR
  PARAMETER BaseManagedObjectId
  ::= localValue 19

complexityLimitation ERROR
  PARAMETER ComplexityLimitation -- optional
  ::= localValue 20

duplicateManagedObjectInstance ERROR
  PARAMETER ObjectInstance
  ::= localValue 11
```

```
getListError ERROR
  PARAMETER GetListError
  ::= localValue 7

invalidArgumentValue ERROR
  PARAMETER InvalidArgumentValue
  ::= localValue 15

invalidAttributeValue ERROR
  PARAMETER Attribute
  ::= localValue 6

invalidFilter ERROR
  PARAMETER CMISFilter
  ::= localValue 4

invalidObjectInstance ERROR
  PARAMETER ObjectInstance
  ::= localValue 17

invalidScope ERROR
  PARAMETER Scope
  ::= localValue 16

missingAttributeValue ERROR
  PARAMETER SET OF AttributeId
  ::= localValue 18

mistypedOperation ERROR
  ::= localValue 21

noSuchAction ERROR
  PARAMETER NoSuchAction
  ::= localValue 9

noSuchArgument ERROR
  PARAMETER NoSuchArgument
  ::= localValue 14

noSuchAttribute ERROR
  PARAMETER AttributeId
  ::= localValue 5

noSuchEventType ERROR
  PARAMETER NoSuchEventType
  ::= localValue 13

noSuchInvokeId ERROR
  PARAMETER InvokeIdType
  ::= localValue 22

noSuchObjectClass ERROR
  PARAMETER ObjectClass
  ::= localValue 0

noSuchObjectInstance ERROR
  PARAMETER ObjectInstance
  ::= localValue 1

noSuchReferenceObject ERROR
  PARAMETER ObjectInstance
  ::= localValue 12

operationCancelled ERROR
  ::= localValue 23

processingFailure ERROR
  PARAMETER ProcessingFailure -- optional
  ::= localValue 10

setListError ERROR
  PARAMETER SetListError
  ::= localValue 8
```

```
syncNotSupported ERROR
  PARAMETER CMISSync
  ::= localValue 3

-- Supporting type definitions.

AccessControl ::= EXTERNAL

ActionArgument ::= SEQUENCE {  COMPONENTS OF   BaseManagedObjectId,
                               accessControl   [5] AccessControl OPTIONAL,
                               synchronization [6] IMPLICIT CMISSync DEFAULT bestEffort,
                               scope           [7] Scope DEFAULT baseObject,
                               filter          CMISFilter DEFAULT and {},
                               actionInfo      [12] IMPLICIT ActionInfo }

ActionError ::= SEQUENCE {  managedObjectClass     ObjectClass OPTIONAL,
                            managedObjectInstance  ObjectInstance OPTIONAL,
                            currentTime            [5] IMPLICIT GeneralizedTime OPTIONAL,
                            actionErrorInfo        [6] ActionErrorInfo }

ActionErrorInfo ::= SEQUENCE {  errorStatus  ENUMERATED {  accessDenied          (2),
                                                           noSuchAction          (9),
                                                           noSuchArgument        (14),
                                                           invalidArgumentValue  (15) },
                                errorInfo    CHOICE {  actionType      ActionTypeId,
                                                       actionArgument  [0] NoSuchArgument,
                                                       argumentValue   [1] InvalidArgumentValue } }

ActionInfo ::= SEQUENCE {  actionType     ActionTypeId,
                           actionInfoArg  [4] ANY DEFINED BY actionType OPTIONAL }

ActionReply ::= SEQUENCE {  actionType      ActionTypeId,
                            actionReplyInfo [4] ANY DEFINED BY actionType }

ActionResult ::= SEQUENCE {  managedObjectClass     ObjectClass OPTIONAL,
                             managedObjectInstance  ObjectInstance OPTIONAL,
                             currentTime            [5] IMPLICIT GeneralizedTime OPTIONAL,
                             actionReply            [6] IMPLICIT ActionReply OPTIONAL }

ActionTypeId ::= CHOICE {  globalForm  [2] IMPLICIT OBJECT IDENTIFIER,
                           localForm   [3] IMPLICIT INTEGER }

  -- This {Recommendation | part of ISO/IEC 9596} does not allocate any values for localForm. Where this alternative is used, the
  -- permissible values for the integers and their meanings shall be defined as part of the application context in which they are used

Attribute ::= SEQUENCE {  attributeId    AttributeId,
                          attributeValue ANY DEFINED BY attributeId }

AttributeError ::= SEQUENCE {  errorStatus  ENUMERATED {  accessDenied          (2),
                                                          noSuchAttribute       (5),
                                                          invalidAttributeValue (6),
                                                          invalidOperator       (24),
                                                          invalidOperation      (25) },
                               modifyOperator  [2] ModifyOperator OPTIONAL, -- present for invalidOperator & invalidOperation
                               attributeId     AttributeId,
                               attributeValue  ANY DEFINED BY attributeId OPTIONAL -- absent for setToDefault
                            }

AttributeId ::= CHOICE {  globalForm  [0] IMPLICIT OBJECT IDENTIFIER,
                          localForm   [1] IMPLICIT INTEGER }

  -- This {Recommendation | part of ISO/IEC 9596} does not allocate any values for localForm. Where this alternative is used, the
  -- permissible values for the integers and their meanings shall be defined as part of the application context in which they are used

AttributeIdError ::= SEQUENCE {  errorStatus  ENUMERATED {  accessDenied     (2),
                                                            noSuchAttribute  (5) },
                                 attributeId  AttributeId }

BaseManagedObjectId ::= SEQUENCE {  baseManagedObjectClass     ObjectClass,
                                    baseManagedObjectInstance  ObjectInstance }
```

```
CMISFilter ::= CHOICE {  item    [8] FilterItem,
                         and     [9] IMPLICIT SET OF CMISFilter,
                         or      [10] IMPLICIT SET OF CMISFilter,
                         not     [11] CMISFilter }

CMISSync ::= ENUMERATED {  bestEffort  (0),
                           atomic      (1) }

ComplexityLimitation ::= SET {  scope   [0] Scope OPTIONAL,
                                filter  [1] CMISFilter OPTIONAL,
                                sync    [2] CMISSync OPTIONAL }


CreateArgument ::= SEQUENCE {
  managedObjectClass      ObjectClass,
                          CHOICE {  managedObjectInstance   ObjectInstance,
                                    superiorObjectInstance  [8] ObjectInstance } OPTIONAL,
  accessControl           [5] AccessControl OPTIONAL,
  referenceObjectInstance [6] ObjectInstance OPTIONAL,
  attributeList           [7] IMPLICIT SET OF Attribute OPTIONAL }

CreateResult ::= SEQUENCE {  managedObjectClass     ObjectClass OPTIONAL,
                             managedObjectInstance  ObjectInstance OPTIONAL,
                                                    -- shall be returned if omitted from CreateArgument
                             currentTime            [5] IMPLICIT GeneralizedTime OPTIONAL,
                             attributeList          [6] IMPLICIT SET OF Attribute OPTIONAL }

DeleteArgument ::= SEQUENCE {  COMPONENTS OF  BaseManagedObjectId,
                               accessControl      [5] AccessControl OPTIONAL,
                               synchronization    [6] IMPLICIT CMISSync DEFAULT bestEffort,
                               scope              [7] Scope DEFAULT baseObject,
                               filter             CMISFilter DEFAULT and {} }

DeleteError ::= SEQUENCE {  managedObjectClass     ObjectClass OPTIONAL,
                            managedObjectInstance  ObjectInstance OPTIONAL,
                            currentTime            [5] IMPLICIT GeneralizedTime OPTIONAL,
                            deleteErrorInfo        [6] ENUMERATED { accessDenied (2) } }

DeleteResult ::= SEQUENCE {  managedObjectClass     ObjectClass OPTIONAL,
                             managedObjectInstance  ObjectInstance OPTIONAL,
                             currentTime            [5] IMPLICIT GeneralizedTime OPTIONAL }

EventReply ::= SEQUENCE {  eventType      EventTypeId,
                           eventReplyInfo [8] ANY DEFINED BY eventType OPTIONAL }

EventReportArgument ::= SEQUENCE {  managedObjectClass     ObjectClass,
                                    managedObjectInstance  ObjectInstance,
                                    eventTime              [5] IMPLICIT GeneralizedTime OPTIONAL,
                                    eventType              EventTypeId,
                                    eventInfo              [8] ANY DEFINED BY eventType OPTIONAL }

EventReportResult ::= SEQUENCE {  managedObjectClass     ObjectClass OPTIONAL,
                                  managedObjectInstance  ObjectInstance OPTIONAL,
                                  currentTime            [5] IMPLICIT GeneralizedTime OPTIONAL,
                                  eventReply             EventReply OPTIONAL }

EventTypeId ::= CHOICE {  globalForm  [6] IMPLICIT OBJECT IDENTIFIER,
                          localForm   [7] IMPLICIT INTEGER }

  -- This [Recommendation | part of ISO/IEC 9596] does not allocate any values for localForm. Where this alternative is used, the
  -- permissible values for the integers and their meanings shall be defined as part of the application context in which they are used
```

```
FilterItem ::= CHOICE {
  equality              [0] IMPLICIT Attribute,
  substrings            [1] IMPLICIT SEQUENCE OF CHOICE {
                            initialString   [0] IMPLICIT SEQUENCE {
                                                attributeId     AttributeId,
                                                string          ANY DEFINED BY attributeId},
                            anyString       [1] IMPLICIT SEQUENCE {
                                                attributeId     AttributeId,
                                                string          ANY DEFINED BY attributeId},
                            finalString     [2] IMPLICIT SEQUENCE {
                                                attributeId     AttributeId,
                                                string          ANY DEFINED BY attributeId} },
  greaterOrEqual        [2] IMPLICIT Attribute, -- asserted value ≥ attribute value
  lessOrEqual           [3] IMPLICIT Attribute, -- asserted value ≤ attribute value
  present               [4] AttributeId,
  subsetOf              [5] IMPLICIT Attribute, -- asserted value is a subset of attribute value
  supersetOf            [6] IMPLICIT Attribute, -- asserted value is a superset of attribute value
  nonNullSetIntersection [7] IMPLICIT Attribute }

GetArgument ::= SEQUENCE {  COMPONENTS OF   BaseManagedObjectId,
                            accessControl   [5] AccessControl OPTIONAL,
                            synchronization [6] IMPLICIT CMISSync DEFAULT bestEffort,
                            scope           [7] Scope DEFAULT baseObject,
                            filter          CMISFilter DEFAULT and { },
                            attributeIdList [12] IMPLICIT SET OF AttributeId OPTIONAL }

GetInfoStatus ::= CHOICE {  attributeIdError  [0] IMPLICIT AttributeIdError,
                            attribute         [1] IMPLICIT Attribute }

GetListError ::= SEQUENCE {  managedObjectClass     ObjectClass OPTIONAL,
                             managedObjectInstance  ObjectInstance OPTIONAL,
                             currentTime            [5] IMPLICIT GeneralizedTime OPTIONAL,
                             getInfoList            [6] IMPLICIT SET OF GetInfoStatus }

GetResult ::= SEQUENCE {  managedObjectClass     ObjectClass OPTIONAL,
                          managedObjectInstance  ObjectInstance OPTIONAL,
                          currentTime            [5] IMPLICIT GeneralizedTime OPTIONAL,
                          attributeList          [6] IMPLICIT SET OF Attribute OPTIONAL }

InvalidArgumentValue ::= CHOICE {  actionValue   [0] IMPLICIT ActionInfo,
                                   eventValue    [1] IMPLICIT SEQUENCE {
                                                     eventType   EventTypeId,
                                                     eventInfo   [8] ANY DEFINED BY eventType OPTIONAL } }

LinkedReplyArgument ::= CHOICE {  getResult          [0] IMPLICIT GetResult,
                                  getListError       [1] IMPLICIT GetListError,
                                  setResult          [2] IMPLICIT SetResult,
                                  setListError       [3] IMPLICIT SetListError,
                                  actionResult       [4] IMPLICIT ActionResult,
                                  processingFailure  [5] IMPLICIT ProcessingFailure,
                                  deleteResult       [6] IMPLICIT DeleteResult,
                                  actionError        [7] IMPLICIT ActionError,
                                  deleteError        [8] IMPLICIT DeleteError }

ModifyOperator ::= INTEGER {  replace       (0),
                              addValues     (1),
                              removeValues  (2),
                              setToDefault  (3) }

NoSuchAction ::= SEQUENCE {  managedObjectClass  ObjectClass,
                             actionType          ActionTypeId }

NoSuchArgument ::= CHOICE {  actionId   [0] IMPLICIT SEQUENCE {
                                            managedObjectClass  ObjectClass OPTIONAL,
                                            actionType          ActionTypeId },
                             eventId    [1] IMPLICIT SEQUENCE {
                                            managedObjectClass  ObjectClass OPTIONAL,
                                            eventType           EventTypeId } }

NoSuchEventType ::= SEQUENCE {  managedObjectClass  ObjectClass,
                                eventType           EventTypeId }

ObjectClass ::= CHOICE {  globalForm  [0] IMPLICIT OBJECT IDENTIFIER,
                          localForm   [1] IMPLICIT INTEGER }
```

```
-- This [Recommendation | part of ISO/IEC 9596] does not allocate any values for localForm. Where this alternative is used, the
-- permissible values for the integers and their meanings shall be defined as part of the application context in which they are used

ObjectInstance ::= CHOICE {  distinguishedName       [2] IMPLICIT DistinguishedName,
                             nonSpecificForm         [3] IMPLICIT OCTET STRING,
                             localDistinguishedName  [4] IMPLICIT RDNSequence }

-- localDistinguishedName is that portion of the distinguished name that is necessary to unambiguously
-- identify the managed object within the context of communication between the open systems

ProcessingFailure ::= SEQUENCE {  managedObjectClass     ObjectClass,
                                  managedObjectInstance  ObjectInstance OPTIONAL,
                                  specificErrorInfo      [5] SpecificErrorInfo }

Scope ::= CHOICE {  INTEGER {  baseObject      (0),
                               firstLevelOnly  (1),
                               wholeSubtree    (2) },
                    individualLevels  [1] IMPLICIT INTEGER,  -- POSITIVE integer indicates the level to be selected
                    baseToNthLevel    [2] IMPLICIT INTEGER }  -- POSITIVE integer N indicates that the range of levels
                                                              -- (0 - N) is to be selected
-- with individualLevels and baseToNthLevel, a value of 0 has the same semantics as baseObject
-- with individualLevels, a value of 1 has the same semantics as firstLevelOnly

SetArgument ::= SEQUENCE {  COMPONENTS OF  BaseManagedObjectId,
                            accessControl  [5] AccessControl OPTIONAL,
                            synchronization  [6] IMPLICIT CMISSync DEFAULT bestEffort,
                            scope  [7] Scope DEFAULT baseObject,
                            filter  CMISFilter DEFAULT and { },
                            modificationList  [12] IMPLICIT SET OF SEQUENCE {
                              modifyOperator  [2] IMPLICIT ModifyOperator DEFAULT replace,
                              attributeId  AttributeId,
                              attributeValue  ANY DEFINED BY attributeId OPTIONAL -- absent for setToDefault
                                                                                  }

SetInfoStatus ::= CHOICE {  attributeError  [0] IMPLICIT AttributeError,
                            attribute  [1] IMPLICIT Attribute }

SetListError ::= SEQUENCE {  managedObjectClass     ObjectClass OPTIONAL,
                             managedObjectInstance  ObjectInstance OPTIONAL,
                             currentTime  [5] IMPLICIT GeneralizedTime OPTIONAL,
                             setInfoList  [6] IMPLICIT SET OF SetInfoStatus }

SetResult ::= SEQUENCE {  managedObjectClass     ObjectClass OPTIONAL,
                          managedObjectInstance  ObjectInstance OPTIONAL,
                          currentTime  [5] IMPLICIT GeneralizedTime OPTIONAL,
                          attributeList  [6] IMPLICIT SET OF Attribute OPTIONAL }

SpecificErrorInfo ::= SEQUENCE {  errorId    OBJECT IDENTIFIER,
                                  errorInfo  ANY DEFINED BY errorId }

END -- End of CMIP syntax defintions
```

Table de correspondance entre les primitives CMIS et les opérations CMIP

CMIS primitive	Mode	Linked-ID	CMIP operation
M-CANCEL-GET req/ind	confirmed	not applicable	m-Cancel-Get-Confirmed
M-CANCEL-GET rsp/conf	not applicable	not applicable	m-Cancel-Get-Confirmed
M-EVENT-REPORT req/ind	non-confirmed	not applicable	m-EventReport
M-EVENT-REPORT req/ind	confirmed	not applicable	m-EventReport-Confirmed
M-EVENT-REPORT rsp/conf	not applicable	not applicable	m-EventReport-Confirmed
M-GET req/ind	confirmed	not applicable	m-Get
M-GET rsp/conf	not applicable	absent	m-Get
M-GET rsp/conf	not applicable	present	m-Linked-Reply
M-SET req/ind	non-confirmed	not applicable	m-Set
M-SET req/ind	confirmed	not applicable	m-Set-Confirmed
M-SET rsp/conf	not applicable	absent	m-Set-Confirmed
M-SET rsp/conf	not applicable	present	m-Linked-Reply
M-ACTION req/ind	non-confirmed	not applicable	m-Action
M-ACTION req/ind	confirmed	not applicable	m-Action-Confirmed
M-ACTION rsp/conf	not applicable	absent	m-Action-Confirmed
M-ACTION rsp/conf	not applicable	present	m-Linked-Reply
M-CREATE req/ind	confirmed	not applicable	m-Create
M-CREATE rsp/conf	not applicable	not applicable	m-Create
M-DELETE req/ind	confirmed	not applicable	m-Delete
M-DELETE rsp/conf	not applicable	absent	m-Delete
M-DELETE rsp/conf	not applicable	present	m-Linked-Reply

Annexe 4

Classes d'objets définies par l'OSI/NM Forum

Add Value Event Record

Agent Conformant Management Entity

Alarm Record

Attribute Change Event Record

Circuit

Computer System

Connection Oriented Transport Protocol Entity

Connectionless Oriented Network Layer Protocol Entity

Contact

Customer

Deenrol Object Event

Enrol Object Event Record

Equipment

Event Log

Event Record

Event Reporting Sieve

Facility

Function

LAN MAC Bridge

Location

MAC Bridge

Manufacturer

Network

Processing Entity

Provider

Reenrol Object Event Record

Remove Value Event Record

Root

Service

Sieve

Top

Transport Connection

Vendor

Profil des protocoles de NM Forum