

Analyse des MIB en vue d'extraction d'information pour le placement des tâches

Maher Ben Sedrine

▶ **To cite this version:**

Maher Ben Sedrine. Analyse des MIB en vue d'extraction d'information pour le placement des tâches. [Stage] 98-R-283 || ben_sedrine98a, 1998, 64 p. inria-00098716

HAL Id: inria-00098716
https://inria.hal.science/inria-00098716

Submitted on 26 Sep 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.







Mémoire de fin d'études

Pour l'obtention du
Diplôme d'Ingénieur en informatique

Analyse des MIB en vue d'extraction d'information pour le placement des tâches

Réalisé par :

Maher Ben Sedrine

Sous la direction de :

Carlos Gamboa Dos Santos et **Jacques Guyard**

Juillet 1998

Table des matières

1 Introduction **1**

2 Parallélisme **3**
2.1 Introduction 3
2.2 Modèles d'exécution 4
2.3 Architecture matérielle 5
2.3.1 La machine multi-processeurs 6
2.3.2 Les systèmes répartis 6
2.4 Pourquoi choisir un NoW 6
2.5 Problèmes posés 7
2.6 Conclusion 8

3 Placement **9**
3.1 Introduction 9
3.2 Placement statique 10
3.3 Placement dynamique 10
3.4 Outils de placement 11
3.5 Conclusion 12

4 Environnement **13**
4.1 MeDley 13
4.1.1 Programmation par échange de message 13
4.1.2 Le langage MeDley 14
4.1.3 Graphe des communications 15
4.2 Analyse 15
4.3 PlaTo 16
4.3.1 Politique de transfert 16
4.3.2 Politique d'information 16
4.3.3 Politique de localisation 16
4.4 Conclusion 17

5 Gestion des réseaux **19**
5.1 Nécessité de gérer les réseaux 19
5.2 Architecture 19
5.3 L'architecture d'une gestion distribuée 20
5.4 Les proxies 20

5.5 Les fonctions du gestionnaire 21
5.5.1 La supervision 21
5.5.2 Fonction de contrôle 22
5.6 Conclusion 23

6 Le Protocole de gestion SNMP 25
6.1 Architecture 25
6.2 Format SNMP 27
6.2.1 Le message SNMP 27
6.2.2 Les requêtes 28
6.2.3 Le format des requêtes 29
6.3 Les avantages et les inconvénients de SNMP 30
6.3.1 Avantages 30
6.3.2 Inconvénients 30
6.4 MIB 31
6.4.1 Structure de la MIB 31
6.4.2 La MIB standard 31
6.4.3 Les MIB privées 32
6.5 Conclusion 34

7 L'apport des MIB 35
7.1 Besoins 35
7.2 Topologie du réseau 36
7.3 Surveillance 37
7.4 Placement 38
7.5 Démarche pour la découverte de la topologie 40
7.6 Conclusion 42

8 Conclusion 43

Bibliographie 45

A Glossaire 47
A.1 Ethernet 47
A.2 HUB 48
A.3 interface 48

B Introduction à CMIP 49
B.1 Principes de CMIP 49
B.2 Comparaison entre CMIP et SNMP 49

C Les Types ASN.1 51
C.1 Les types simples 51
C.2 Les types prédéfinis 51
C.3 Les types structurés 52
C.4 Les types ASN.1 utilisés dans SNMP 52

D Ordre lexicographique **53**

Table des figures

2.1 Calcul parallèle 3
2.2 Modèles d'exécution proposés par Flynn 5
2.3 Exemple de réseau hétérogène 7

3.1 Équilibrage et partage de charge 11

4.1 Exemple d'application M-SPMD 14

5.1 Hiérarchisation de la gestion 21

6.1 Protocole Gestionnaire/Agents 26
6.2 Architecture de la gestion SNMP 26
6.3 La trame SNMP 27
6.4 Le premier format PDU de SNMPv1 29
6.5 Le format Trap de SNMPv1 29
6.6 La MIB standard 32
6.7 La MIB privée 33

7.1 Découverte de la topologie 38
7.2 Surveillance du réseau 39
7.3 Placement sur le réseau 40

Chapitre 1

Introduction

Les besoins en puissance de calcul deviennent de plus en plus importants, non seulement par les informaticiens, notamment pour les applications de base de données, mais aussi dans le domaine du calcul scientifique, notamment par les chimistes, les physiciens et les mathématiciens. On développe des modèles mathématiques de plus en plus fins ayant une demande plus accrue en puissance de calcul. Bien que les machines séquentielles évoluent rapidement et offrent une puissance croissante de calcul, elles restent incapables de répondre aux besoins de plusieurs applications, de plus s'équiper d'une nouvelles machines pour répondre à un nouveau besoin de calcul n'est pas une solution raisonnable, surtout qu'on a plusieurs machines sous utilisées. En conséquence, une implantation parallèle de ces applications est la solution la plus judicieuse, son principe consiste à décomposer l'application en plusieurs tâches, les exécuter en parallèle sur les processeurs disponibles tout en garantissant la coopération et l'ordonnancement entre les tâches. Pour ce faire, en général on utilise un modèle de programmation basé sur l'utilisation de langages séquentiels (C, C++ ou Fortran) liés à des bibliothèques de communication, comme par exemple MPI[1] ou PVM[2]. Un environnement **MeDley** de spécification des messages échangés entre les différentes tâches d'une application parallèle est développé par l'équipe RESEDAS. L'outil de placement développé est **PlaTo**, il place les tâches de l'application parallèle en tenant compte de la charge des machines et des coûts de communication entre les tâches. PlaTo collabore avec l'outil de gestion de réseau pour déterminer les informations sur le système.
Étant donné que nous travaillons dans le contexte de réseaux locaux, la solution OSI est par conséquent défavorisée. C'est l'approche **SNMP qui est privilégiée, il est fortement déployé pour les équipements du réseau, ce qui facilite la gestion. SNMP gère les ressources du réseau au travers les MIB. En utilisant SNMP un certain nombre d'informations sur la topologie et l'état des ressources est disponible. Se basant sur la politique d'information des algorithmes de placement dynamique, nous distinguons trois niveaux d'informations . Nous essayons d'utiliser les MIB pour avoir ces informations.**
Ce travail entre dans le cadre d'un des axes de recherche de l'équipe RESEDAS, qui fait partie d'une opération du centre Charles Hermite, centre

1. Message Passing Interface
2. Parallel Virtual Machine

Lorrain de compétence en modélisation et calcul à hautes performances. L'un des aspects de ce travail consiste à découvrir la topologie du réseau, l'état de charge des processeurs et des équipements intervenant dans le réseau, l'état des liens de communication et l'impact sur le temps de transfert des messages entre les tâches.

La première partie du mémoire est consacrée principalement à l'état de l'art. Nous commençons par présenter globalement le domaine du parallélisme et les problèmes qui se posent lors de l'utilisation d'un réseau de station. Nous détaillerons au chapitre 3 la problématique du placement. Nous présentons dans le chapitre 4 l'environnement MeDley et PlaTo. Comme notre approche est le placement dynamique nous détaillerons dans le chapitre 5 le domaine de la gestion des réseaux. Nous présentons au chapitre 6 le protocole de gestion SNMP. Au chapitre 7, nous détaillerons l'apport de l'utilisation des MIB pour le placement des tâches. Nous terminerons le mémoire par une conclusion et les perspectives de notre travail.

Chapitre 2

Parallélisme

Nous présentons dans ce chapitre le parallélisme d'un point de vue général, l'architecture matérielle en mettant l'accent sur les systèmes répartis.

2.1 Introduction

Le parallélisme est le fait d'effectuer plusieurs traitements en même temps sur plusieurs ressources. Le but du calcul parallèle est de résoudre les problèmes de façon rapide ou d'exécuter des applications qui ont un très grand volumes de données.

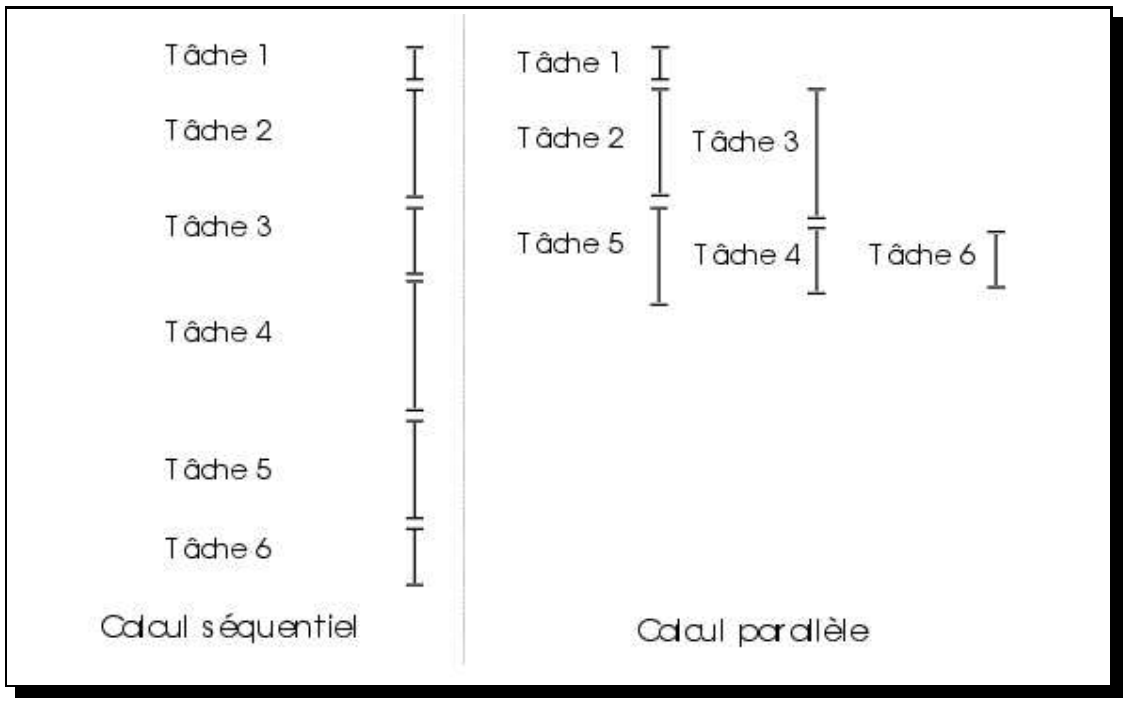


FIG. 2.1 – *Calcul parallèle*

Le parallélisme est la coopération entre tâches communicantes afin de réduire un problème commun. Il peut être envisagé à plusieurs niveaux: dans les systèmes monoprocesseur ou dans les systèmes multi-processeurs. Pour le premier type, l'utilisateur n'a pas à se soucier, le traitement parallèle est assuré par le système d'exploitation et se fait d'une façon transparente. Pour le second type, l'affectation des processeurs aux tâches doit être faite d'une façon explicite. La mise en application de ce dernier se fait par divers moyens:

- plusieurs unités fonctionnelles,
- UAL parallèles,
- fonctionnement simultané des différentes unités,
- multi-programmation et temps partagé.

La difficulté réside au niveau de la diversité des caractéristiques des machines parallèles. La programmation portable est difficile si on veut avoir les mêmes performances. Les machines parallèles ont été développées au début des années soixante dix.

2.2 Modèles d'exécution

Pour distinguer entre les différentes machines parallèles plusieurs modèles ont été proposés, dont celui de Flynn, il les classe selon le flot d'instructions et le flot de données (voir figure 2.2) .

SISD (Single Instruction Single Data) un seul flot de données et un seul flot d'instructions, cette catégorie inclut les machines séquentielles classique. Elles possèdent une seule mémoire centrale reliée à une unique unité de calcul.

MISD (Multiple Instruction Single Data) plusieurs flots d'instructions et un seul flot de données, ceci modélise les architectures *pipe-line*, tels que les calculateurs vectoriels, les architectures à flux ou les grilles systoliques. Ces architectures comprennent une mémoire centrale commune et un processeur divisé en plusieurs étages, chacun chargé d'exécuter une tranche de l'instruction. L'idée du *pipe-line* est de décomposer les opérations en plusieurs étapes consécutives et de les recouvrir.

SIMD (Single Instruction Multiple Data) un seul flot d'instruction et plusieurs flots de données. C'est l'architecture des grilles de processeurs ou les ordinateurs massivement parallèles, composés d'une unité de contrôle distribuant un flot d'instructions à un ou à un ensemble d'unités de calcul connectées par un réseau. Ce modèle s'est évolué en SPMD d'ou la synchronisation se fait sur tout un programme.

MIMD (Multiple Instruction Multiple Data) plusieurs flots d'instructions traitent plusieurs flots de données. C'est le modèle qui regroupe les machines multi-processeurs et les réseaux de stations. Une extension de ce modèle a donné le MPMD(Multiple Program Multiple Data).

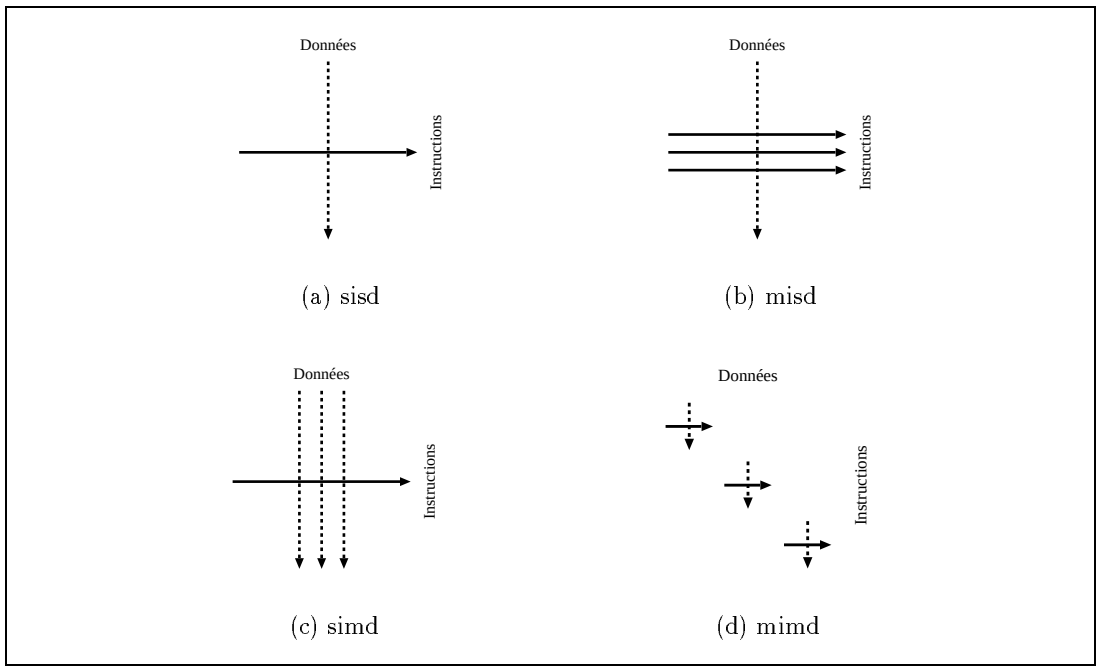


FIG. 2.2 – *Modèles d'exécution proposés par Flynn*

L'approche de Flynn ne tient pas compte de l'organisation de la mémoire par rapport aux processeurs. En fait il y a deux types d'organisation mémoire pour les machines parallèles :

Mémoire partagée (SM) : tous les processeurs accédant à une même mémoire physique, l'échange de données entre tâches se fait par l'intermédiaire de la mémoire et c'est à la charge du programme de faire la synchronisation. Les difficultés d'accès à la mémoire commune limitent le nombre de processeurs.

Mémoire distribuée (DM) : chaque processeur dispose d'une mémoire locale qui lui est propre, l'échange de données se fait explicitement par les messages, ce qui assure en même temps la synchronisation.

2.3 Architecture matérielle

Pour le calcul parallèle on distingue deux supports matériels possibles : les machines parallèles ou multi-processeurs et les systèmes répartis ou machines parallèles virtuelles.

2.3.1 La machine multi-processeurs

C'est une machine constituée de plusieurs processeurs (en général identiques) fortement couplés par des communications et des synchronisations très rapides, en plus de cela la machine dispose d'une capacité mémoire importante. On peut distinguer deux types de machines parallèles : les massivement parallèles à mémoire distribuée et les multi-processeurs symétriques à mémoire partagée.

2.3.2 Les systèmes répartis

C'est un réseau de stations de travail ou NoW[1] éventuellement hétérogène. Les machines sont autonomes, communiquent au travers d'un réseau local et offrent des services de coopération. Cette dernière caractéristique distingue un NoW d'un *networked system*, qui est aussi un agrégat de machines autonomes qui peuvent s'échanger des données généralement sous forme client/serveur. L'utilisateur doit percevoir un système réparti comme une seule machine. Néanmoins les systèmes d'exploitation actuels n'offrent pas la transparence aux utilisateurs et donc un placement explicite des processus est nécessaire. Un système réparti ressemble aux machines massivement parallèles ou MPP[2], mais la différence est que dans un MPP les liens de communication entre les nœuds sont dédiées et donc l'objectif est la performance de calcul sans souci du prix du support de communications. Une autre différence est à noter, en fait le système d'exploitation d'une MPP a la connaissance de l'état de ses nœuds et offre des techniques de gestion des programmes parallèles, alors que dans un NoW, le programmeur doit lui même se soucier des états des nœuds, la charge des liens et le placement des processus. Notons qu'il est possible d'avoir un système réparti constitué de stations de travail et de machines parallèles (voir figure2.3) , cependant la programmation devient plus difficile.

2.4 Pourquoi choisir un NoW

Deux raisons sont derrière l'utilisation des systèmes répartis :

- le besoin en puissance de calcul pousse au calcul parallèle, mais les machines parallèles restent trop chères et le NoW se présente comme une alternative pour traiter un certain nombre d'applications ;
- la deuxième raison et aussi la principale est que dans la plupart des cas les stations sont utilisées en mode interactif et donc restent inexploitées la nuit, pendant les vacances ou les pauses. Une étude a révélé que même pendant les heures de travail, entre 60% et 90% des stations sont inutilisées.

1. Network of Workstations
2. Massive Parallel Processing

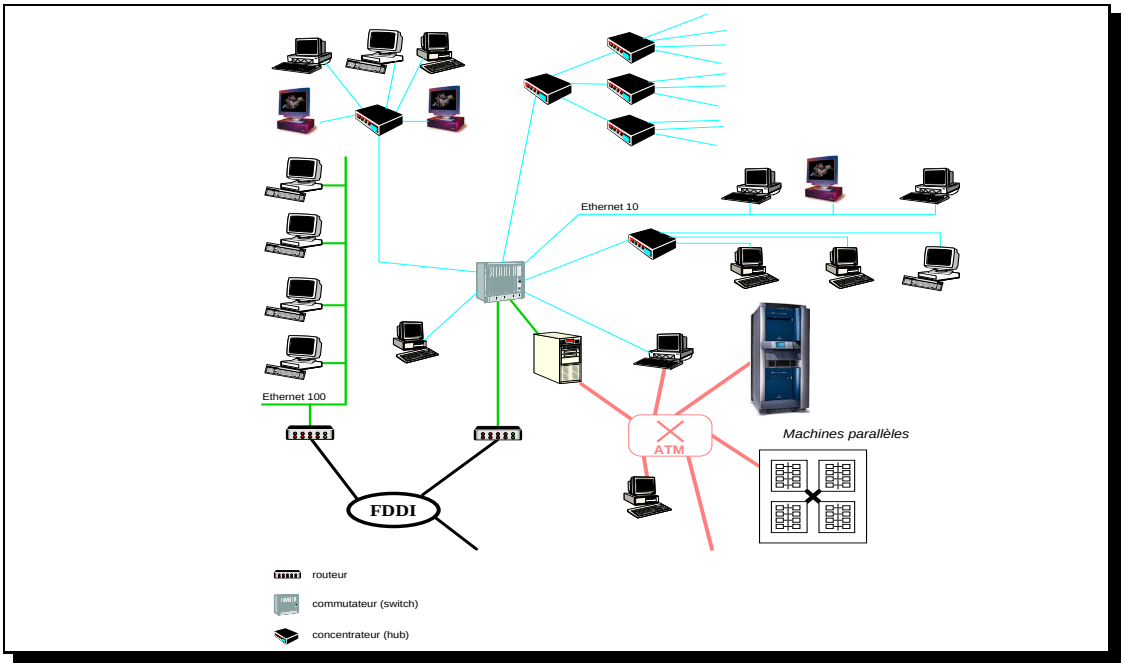


FIG. 2.3 – *Exemple de réseau hétérogène*

Il est donc judicieux d'utiliser ces machines pour un calcul parallèle en maintenant la mission de base des machines, c'est à dire en respectant les performances pour les utilisateurs. Notre travail se place dans le contexte des NoW, le choix s'est clairement basé sur le fait de la disponibilité du réseau, le prix élevé des machines parallèles et la souplesse d'utilisation.

2.5 Problèmes posés

Les NoWs ont participé à l'expansion du parallélisme, en fait il n'était possible que pour ceux qui disposent de machines parallèles. Les NoWs ne vont pas bien-sûr remplacer les machines parallèles. En fait les machines SIMD ou les calculateurs vectorielles offrent des performances inégalables pour des applications spécifiques. En plus de ça, l'utilisation des NoWs n'est pas dépourvue de problèmes; le partage des ressources, l'hétérogénéité, les bibliothèques de communication, le bas niveau du paradigme du message-passing et la tolérance aux pannes. Les différents aspects sont décrits ci après.

– le partage des ressources peut être vu à deux niveaux, d'une part les machines qui ne sont pas forcément dédiées au calcul parallèle et donc éviter de gêner le travail de son utilisateur, d'autre part pour les liens de communication, les applications qui sont fortement communicantes risquent de saturer le réseau ;

- l'hétérogénéité pose aussi des problèmes, en fait si les machines sont d'architectures différentes, les données échangées doivent être encodées afin qu'elles puissent être traitées par la machine de réception, de plus une grande différence entre les puissances de calcul des machines peut mettre certains processus, s'exécutant sur des machines rapides, en une longue attente de résultats des processus qui s'exécutent sur des processeurs lents. Un dernier aspect de l'hétérogénéité, c'est les services et les performances offerts par les liens de communication. Cette hétérogénéité peut être un avantage puisque on peut choisir les machines qui nous offrent la qualité de service dont nous avons besoin ;
- les bibliothèques de communication sont des outils logiciels conçues indépendamment du support et généralement les fonctions sont incorporées dans le code de l'application parallèle. Ceci élève le nombre de couches et par conséquent les appels de fonctions nécessaires à l'échange de données, d'autre part les opérations de communication peuvent baisser les performances ;
- la programmation par échange de message est à la charge de l'utilisateur, donc les risques d'erreurs peuvent être important et la performance peut dégrader ;
- le problème de tolérance aux pannes est l'un des axes de recherche. En fait une étude a prouvé qu'une machine s'arrête en moyenne tous les deux à sept jours, donc les processus qui sont lancés sur cette machines doivent être replacés sur une autre machine, ceci en essayant de ne pas reprendre la tâche en entier.

2.6 Conclusion

Le parallélisme est une source de puissance de calcul, dans le cas d'un NoW on a en plus une meilleure utilisation des ressources. Cependant la diversité des modèles de calcul parallèle rend la tâche de programmation un peu difficile. Un modèle de passage de message est proposé, permettant d'utiliser un NoW comme une machine parallèle virtuelle. Notre problème est de faire le placement des tâches d'une application en tenant compte de l'état courant du système, pour ce faire un outil de gestion de réseau interroge les agents qui sont placés sur les nœuds du réseau.

Chapitre 3

Placement

Dans ce chapitre nous essayons d'introduire les principes de base du placement des tâches, le domaine reste très vaste et fait l'objet de plusieurs sujets de recherche. Le placement des tâches est la condition nécessaire pour garantir la performance des machines parallèles, bien que jusqu'à maintenant il n'y a pas de méthode qui puissent garantir les meilleures performances dans tous les cas.

3.1 Introduction

Dans une machines parallèle on a P processeurs, une application parallèle est composée de plusieurs tâches, le placement consiste à affecter ces T tâches aux P processeurs disponibles, ceci en minimisant le coût global de l'exécution, c'est à dire le coût de calcul et celui de la communication entre les tâches. On a tendance a éloigner les processus pour avoir le moins de temps de calcul et de rapprocher les processus qui ont besoin de se communiquer afin de diminuer la charge sur le réseau. L'affectation des tâches se fait par des algorithmes d'ordonnancement, dans la littérature on distingue deux groupes :

- local : l'affectation des tâches se fait d'une façon locale au niveau de chaque machine, dans ce cas le système d'exploitation prend en charge le partage du processeur ;
- global : l'affectation des tâches se fait sur tout le système avec l'hypothèse que les tâches démarrent immédiatement après le placement, ceci afin d'éviter les relations de précédence, dans le cas contraire on les regroupe et ainsi on évite les contraintes temporelles.

Dans l'ordonnancement global on trouve deux types ; le placement statique et le placement dynamique. Pour le premier l'affectation des tâches au processeurs se fait avant la compilation de l'application. Pour le placement dynamique l'affectation se fait au démarrage et éventuellement à la migration. Ainsi l'affectation évolue en fonction de l'état du système ; la charge des processeur, les pannes et le trafic sur le réseau.

3.2 Placement statique

De point de vue algorithmique c'est plus simple, mais il est restreint à un certain nombre d'applications vérifiant les conditions suivantes :

- architecture figée du système durant toute l'application,
- comportement de l'application connue à l'avance ; le nombre de tâches et les inter-relations,
- les interférences externes lors de l'exécution de l'application sont négligée.

Le problème qui consiste à trouver un placement optimal pour les tâches sur une architecture donnée est dans le cas général NP-complet, c'est à dire qu'il n'existe pas d'algorithme qui résout le problème en un temps borné par une fonction polynômiale de la taille du problème. Donc il est préférable de chercher un algorithme moins complexe qui donne une solution approchée. Il existe dans la littérature plusieurs algorithmes sur le placement statique qu'on peut regrouper sur deux niveaux : les algorithmes optimaux et les algorithmes sous-optimaux. Si les premiers donnent une solution optimale pour le problème, mais ils perdent en temps de calcul, le deuxième type cherche une solution rapidement sans garantir qu'elles soit la meilleure.

3.3 Placement dynamique

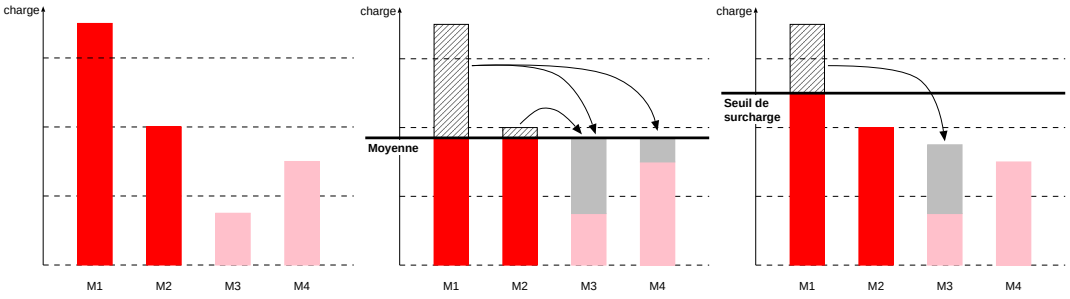
On a essayé de résoudre les problèmes et les contraintes posés par le placement dynamique en adoptant une stratégie dynamique, en fait la décision se fait en tenant compte de l'état courant du système. L'affectation des tâches aux machines se fait au moment où cette tâche doit commencer à s'exécuter et pendant l'exécution en faisant migrer les tâches. La migration se fait de la machine où il y a une forte charge sur le processeur à une autre qui soit moins chargée, ou bien à cause d'une insatisfaction de la qualité de service d'un lien de communication qu'utilise cette tâche. Donc la question au quelle doit répondre l'algorithme de placement est quand faire du placement ou de la migration, pour quelles tâches et sur quelles machines. Pour répondre à ces questions l'opération de placement est décomposée en trois composantes décrites ci-dessous.

Politique d'information : détermine les informations sur les machines, leurs répartitions, les tâches de point de vue volume des données à échanger, temps CPU requis... On ne connaît pas une combinaison des paramètres du système qui reflètent l'état courant du système et son évolution probable, qui soit vérifiable pour tout système. En général, les algorithmes utilisent pour des raisons de simplicité l'indicateur de

charge **UNIX** qui est calculé en fonction de la longueur de la file d'attente des processus actifs. Notons que l'indicateur de charge ainsi obtenu dépend fortement du passé et du présent de la machine. Or, pour le placement c'est la charge future qui est importante. Pour obtenir de meilleurs résultats l'algorithme de placement doit donc anticiper la charge d'une machine après le placement ou la migration sur cette dernière.

Politique de transfert : détermine quand et quelle tâche il faut placer. Suivant l'objectif du système de placement, nous pouvons distinguer les deux cas suivants répondant au quand de la politique de transfert :

1. équilibrage de charge : on essaye de répartir la charge sur tout le système, les décisions de placement peuvent être révisées pour toutes les machines, ce qui entraîne éventuellement un nombre élevé de migration de tâches ;
2. partage de tâches : le système essaye de diminuer la charge d'une machine donnée en affectant une partie de ces processus à une autre. Donc la remise en question de placement se fait d'une façon locale.



(a) Situation de départ (b) Après équilibrage (c) Après partage

FIG. 3.1 – *Équilibrage et partage de charge*

Politique de localisation : c'est la dernière étape dans l'algorithme de placement, elle consiste à déterminer sur quelle machine on place une tâche qu'on veut lancer ou faire migrer.

3.4 Outils de placement

En général dans le placement dynamique , lorsque on tient compte de la communication entre les tâches on utilise le principe de répulsion et d'attraction. En fait, les tâches qui ont besoin de se communiquer ont tendance

à s'exécuter sur la même machine, et contrairement celles qui ont besoin de la même ressource (CPU par exemple) ont tendance à se répulser et donc choisir des machines distinctes pour s'exécuter. Cette approche est utilisée dans plusieurs plates forme de placement. On installe au niveau de chaque machine un agent, qui coopèrent ensemble afin d'équilibrer la charge entre les machines. Afin de favoriser le parallélisme, on multiplie la charge de la machine par un facteur proportionnel à la communication, ainsi les tâches communicantes ne vont pas être pénalisées par le temps partagé, car elles auront tendance à s'exécuter sur les machines les moins chargées.

3.5 Conclusion

Pour le placement de tâches dans le calcul distribué deux approches peuvent être utilisées : le placement statique et le placement dynamique. Le premier impose des contraintes sur le système et l'application, en plus la recherche d'une solution optimale est très coûteuse. Contrairement à cela, le placement dynamique essaye de tenir compte de l'état courant du système, ainsi que l'évolution de l'application. En plus, un outil de placement doit tenir compte de l'hétérogénéité des services et des performances offertes par les réseaux mis en application et aussi de l'hétérogénéité des machines. Pour récupérer ces informations, on a besoin d'un système de gestion de réseau, qui contrôle et supervise l'état du réseau.

Chapitre 4

Environnement

L'équipe Résédas développe un environnement facilitant le développement d'applications parallèles. Nous présentons dans ce chapitre le langage de spécification d'applications parallèles MeDley, nous introduisons par la suite les principes de base du placement des tâches. Nous terminons par présenter l'outil de placement PlaTo développé par l'équipe. Rappelons que nous travaillons sur un système distribué hétérogène où la communication entre processeurs se fait par l'échange de messages. L'utilisateur développe son application parallèle en C, C++ ou Fortran et spécifie les messages dans le langage de description MeDley.

4.1 MeDley

4.1.1 Programmation par échange de message

MeDley est un langage de description d'applications parallèles basé sur la programmation par échange de messages. Cette dernière prend les hypothèses suivantes :

- les données des tâches sont privées,
- la coopération entre les tâches se fait par le biais d'une communication,
- les parties invoquées dans un message doivent faire appel à des primitives spécifiques de communication.

On a deux types de messages :

- message de donnée où l'utilisateur fait appel à des primitives de la bibliothèque de communication,
- message de contrôle ou de service, sont transparents à l'utilisateur et sont utilisés par la bibliothèque pour implanter le protocole (synchronisation, acquittement et notification).

Le modèle supporté par MeDley est M-SPMD (Multiple Single Program Multiple Data) qui est une extension du *MPMD*. En fait on regroupe les tâches de même fonctionnalité par famille et sont ainsi de type *SPMD*, la famille peut contenir de un à plusieurs tâches, le nombre maximale reste à déterminer en fonction de plusieurs facteurs entre autre le nombre de processeurs disponibles.

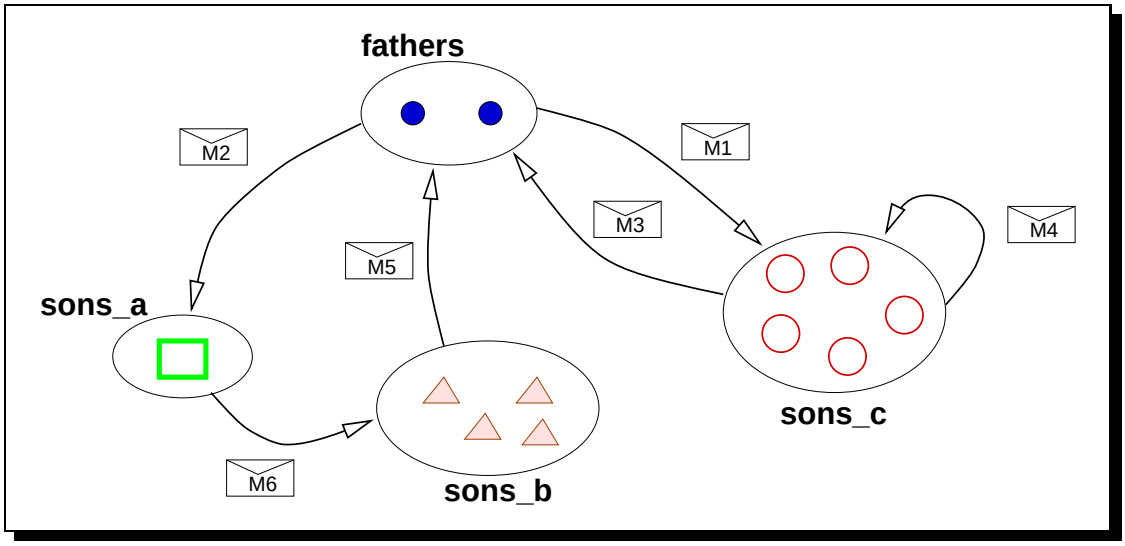


FIG. 4.1 – *Exemple d'application M-SPMD*

4.1.2 Le langage MeDley

S'inspirant du modèle *M-SPMD*, MeDley associe à chaque famille un représentant et décrit les interactions entre les représentants de chaque famille. La partie contrôle indiquant l'ordonnancement des tâches est laissée à l'utilisateur.

Pour chaque représentant de tâche, MeDley associe un module composé de trois parties principales :

- sous-bloc uses déclare les variables locales qui vont être utilisées pour le calcul et les communications,
- sous-bloc sends spécifie les messages à émettre aux tâches, les messages sont constitués des variables du sous-bloc uses,
- sous-bloc receives spécifie les messages de réception des autres tâches, le contenu du message se rapporte aussi aux variables déclarées dans le sous-bloc uses.

Ainsi, on définit dans la partie sends comment construire le message et dans la partie receives comment projeter le message sur les variables locales.

Exemple relatif à la figure 4.1 :

```
Task sons_b connected with setof(fathers), setof(sons_a)
   uses
            {
               float moyenne ;
               int numero ;
               vector <float >[10] note ;
            }
   sends
            {
               M5 to fathers  =sequence {moyenne, numero}
            }
   receives
            {
               M6  from sons_a  = sequence {numero, notes};
            }
```

4.1.3 Graphe des communications

A partir de la spécification des communication, MeDley génère le graphe des communication précisant le nombre de famille et les message à échanger entre les familles. Le graphe généré ne précise pas donc le nombre et la taille des messages à échanger ni les instances de familles qui vont communiquer, car la cardinalité de la famille n'est spécifiée qu'au démarrage de l'application. Un autre problème dans le langage MeDley est l'absence de spécification des structures de contrôle donc l'ordonnancement des tâches ne peut pas être déduit, pour contourner ce problème toutes les tâches sont lancées au démarrage de l'application et les blocages de réception de message assureront les relations de précédence. Mais cette solution a aussi des inconvénients, en fait les tâches qu'on a lancées et sont bloquées à l'attente du premier message vont consommer les ressources, en plus de ça le placement ne tient pas compte de l'ordre séquentiel des tâches pour les placer sur la même machine.

4.2 Analyse

Les informations fournies par MeDley sont insuffisantes pour le placement, en fait on a besoin de connaître le volume de données échangées entre les tâches pour faire le meilleur placement tenant compte des services et de l'état des liens de communication. Pour récupérer cette information on analyse les traces d'exécution de l'application, il y a deux méthodes pour ce faire : la première consiste à pré-exécuter l'application sur un jeu de données réduit et ainsi déduire le poids relatif des communications, la

deuxième méthode consiste à comptabiliser les messages réellement échangés lors d'une exécution sur des données réelles. Le résultat de l'analyse ne peut pas être efficace à cent pour cent, mais c'est une prévision pour améliorer les performances.

4.3 PlaTo

L'environnement sur lequel nous travaillons est non dédié au calcul parallèle, donc la charge du système est constamment perturbée par les lancement ou les terminaisons des processus ainsi que la disponibilité des ressources. L'outil de placement PlaTo proposée par l'équipe a tenu compte de cette condition c'est pour cela PlaTo est dynamique et donc il supporte les trois politiques de placement dynamique.

4.3.1 Politique de transfert

En se basant sur MeDley, les tâches d'une application sont toutes placées en même temps. L'inconvénient majeur de cette approche est le fait de ne pas tenir compte des perturbations dûes au lancement et la terminaison des processus des autres applications ou les pannes de stations ou de liens.

4.3.2 Politique d'information

La politique d'information est également facile à définir. Les informations nécessaires sont d'une part la topologie du réseau comprenant les machines de calcul, les liens de communication et les autre équipements influençant les délais d'acheminement d'un message. Il est donc indispensable de modéliser dans cette topologie la capacité d'un lien de communication, ainsi que les temps de transfert en fonction du volume des données émises. En plus des informations sur la topologie, on a besoin de connaître l'état de charge du système ; les indicateurs de charge des processus ainsi que la charge réseau. La troisième source d'information est le graphe de communication généré par MeDley.

4.3.3 Politique de localisation

Le but est d'affecter les tâches aux machines disponibles tout en ayant les meilleurs performances. La sélection des machines reste difficile puisqu'il faut tenir compte des coûts de la communication. Cette opération est effectuée en deux étapes :

- la première est à la pré-exécution et donc sans tenir compte du volume des données à échanger ;

- la deuxième étape se fait en fonction des réseaux d'interconnexion ainsi que les débits prévus entre les tâches. Dans le cas échéant réservation de la bande passante entre les machines.

4.4 Conclusion

Une application ne peut pas se faire sans tenir compte de l'état courant du système et de ses capacités afin de garantir les performances attendues. Pour cela nous essayons d'utiliser un outil de gestion de réseau pour découvrir la topologie du réseau et l'état de charge des liens de communication et des nœuds du système.

Chapitre 5

Gestion des réseaux

Nous présentons dans ce chapitre l'intérêt et les principes de base de la gestion des réseaux. Nous montrerons également l'apport de la gestion des réseaux pour le placement des tâches dans le calcul parallèle sur un NoW.

5.1 Nécessité de gérer les réseaux

Les besoins de communication et de coopération ont engendré l'expansion des réseaux ; il y a une tendance vers les réseaux complexes supportant plusieurs applications et utilisateurs. Ceci implique que les ressources vont être fortement sollicitées, en plus une augmentation de la probabilité pour qu'un équipement ou une portion du réseau présente un service dégradé ou soit complètement en panne. Donc la gestion des réseaux est primordiale pour l'allocation, l'exploitation, la maintenance et la planification des réseaux. Son objectif est d'assurer un bon fonctionnement des équipements et des services du réseau afin de présenter un service fiable et à moindre coût. La gestion des réseaux s'est réduit pour longtemps a la gestion des équipements, mais maintenant et pour répondre à l'évolution des réseaux, elle ajoute des fonctions de gestion des réseaux en ayant une vision sur la topologie, les services déployés ainsi que les applications qui les utilisent.

5.2 Architecture

Dans un système de gestion de réseau on trouve au minimum une machine qui est le gestionnaire ou le manager, des agents installés sur les équipements à gérer et un protocole arbitrant les interaction entre ces deux entités. Tout équipement du système contient une entité de gestion de réseau (NME)[1] et une interface de gestion. Chaque NME prend en charge de collecter des statistiques sur les communications, stocker localement ces informations et permet de répondre aux requêtes du manager. Les agents sont

1. Network Management Entity

implémentés sur les équipements terminaux ou sur les nœuds de communication (commutateur, HUB, pont...). Un agent a besoin d'une interface de gestion, d'une entité de gestion et bien-sûr une couche de communication. Pour le gestionnaire, on ajoute une entité d'application (NMA)[2] qui est l'interface du gestionnaire avec l'administrateur du réseau.

5.3 L'architecture d'une gestion distribuée

Dans le paragraphe précédent nous avons présenté l'architecture d'un système de gestion où il y a un seul gestionnaire dans le système, cette stratégie a été longuement adoptée, mais avec l'utilisation des systèmes répartis, la gestion distribuée est devenue une nécessité (voir figure 5.1). Ainsi dans un réseau on peut avoir plus qu'un gestionnaire, mais afin d'éviter l'incohérence une hiérarchie est proposée se basant sur deux niveaux de gestion :

- les stations de gestion distribuées (gestionnaires élémentaires) , ont un accès limité pour le contrôle et la supervision du réseau, en général elles se limitent au niveau des ressources gérées ;
- une station de gestion centrale ou serveur de gestion qui a le droit sur tout le réseau, elle peut contrôler les fonctions des autres gestionnaires. On peut avoir plusieurs serveurs de gestion dans le même réseau à condition qu'ils utilisent des MIB et des fonctions différentes. Cette approche apporte plusieurs avantages :
  1. diminuer la charge sur le réseau. les communications de gestion diminuent puisqu'elle se fait localement,
  2. l'ajout de gestionnaire simple.
  3. une tolérance aux pannes puisque la gestion n'est plus centralisée.

5.4 Les proxies

La gestion des réseaux est basée sur la communication entre les agents et le ou les gestionnaires. Les équipements dans un réseau sont rarement de même type et donc une incompatibilité entre les deux. En plus cette architecture impose que chaque équipement dispose d'un agent, or ceci n'est pratique car un agent coûte cher, voir parfois impossible si l'équipement ne supporte pas une partie *soft* comme pour les modems ou les multiplexeurs. Pour contourner ces problèmes, on utilise un proxie qui peut gérer un ou plusieurs équipements et sera intermédiaire entre ces nœuds et le gestionnaire.

2. Network Management Application

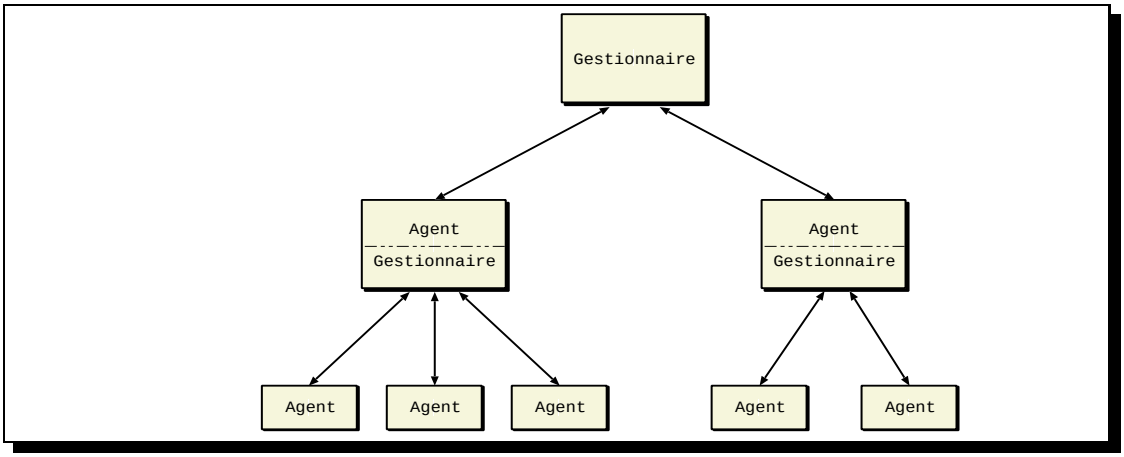


FIG. 5.1 – *Hiérarchisation de la gestion*

5.5 Les fonctions du gestionnaire

Les cinq fonctions de base de la gestion du réseau sont : la performance, les erreurs, les rapports, la configuration et la sécurité. Ces fonctionnalités font toutes appel aux fonctions de contrôle et de supervision, cependant les trois premières se basent surtout sur la fonction de supervision et les deux autres sur le contrôle.

5.5.1 La supervision

Cette fonction prend en charge l'observation et l'analyse de l'état et du comportement des équipements du système et des liens qui les connectent. Pour le faire il y a deux techniques qui sont utilisées pour mettre les données à la disposition du gestionnaire :

- ***Polling*** le gestionnaire interroge les agents sur les variables, l'agent répond en allant chercher l'information dans sa base d'information de gestion,
- ***Event reporting*** l'agent émet un rapport pour le manager.

Performance du réseau : l'une des difficultés pour mesurer la performance du réseau est de sélectionner des indicateurs qui puissent refléter fidèlement son état. En fait le nombre des indicateurs est très élevé, certains sont supportés par les constructeurs, d'autres pas, le sens de quelques indicateurs n'est pas encore bien compris. D'autres demandent beaucoup de temps pour être calculés et donc une fois ces résultats peuvent être disponibles, ils ne reflètent plus l'environnement. Ces indicateurs peuvent être regroupés sous deux catégories :

1. les mesures de services incluent la disponibilité, le temps de réponse et la fiabilité,

2. les mesures de rendement comprenant le taux des événements des applications (par exemple le nombre de sessions ...) et l'utilisation comme le taux de pourcentage d'utilisation d'une ressource donnée pendant une période de temps.

Supervision des erreurs : le but est de localiser et identifier les erreurs afin de lancer les actions de répartition le plutôt possible. Une fois la panne détectée, la fonction établit un rapport, pour un certain type de panne elle émet le rapport au manager, pour d'autre elle garde une copie pour une éventuelle recherche de la part du gestionnaire. La fonction peut anticiper l'erreur en affectant des seuils aux variables, en plus de ça elle assiste pour les tests, comme les tests de connexion, d'intégrité des données. Cependant ceci n'est pas facile pour les environnements complexes, où on trouve trois types de fautes qui sont plus difficiles à contourner :

- des fautes qu'on ne peut pas détecter localement comme le blocage mortel dans une application ou une panne que l'équipement n'enregistre pas ;
- des fautes où on ne peut pas faire la diagnostic par exemple un nœud qui ne répond pas(machine en panne ou défaillance au niveau du protocole des couches inférieures) ;
- des incertitudes dans l'observation, certains indicateurs n'impliquent pas une panne précise, par exemple un manque de réponse d'un équipement peut être dû à une congestion dans son buffer, son timer est défaillant ou le réseau est partitioné.

Donc afin d'améliorer la fiabilité, une interface avec l'utilisateur est nécessaire pour résoudre les pannes et les problèmes qui ne peuvent l'être par le système.

Comptabilité : cette fonction est assurée surtout pour garder trace de l'utilisation des ressources du système pour des raisons de facturation ou de partage.

5.5.2 Fonction de contrôle

Cette fonction de gestion de réseau permet la modification des paramètres du système et de lancer des actions sur des équipements, elle contrôle les mécanismes de sécurité qui sont définis par le système.

1. Contrôle de la configuration La gestion de configuration se charge du suivi de la configuration, des modifications en réponse à des commandes de l'utilisateur ou à d'autres fonctions de gestion et définition d'informations de gestion :

   - description des ressources physiques (routeur, pont, station de travail ...) , logiques (compteur, horloge, circuit virtuel ...) et de

leurs attributs. Ces informations sont stockées prés des ressources et restent accessibles par la station de gestion ;

- modification des valeurs des attributs : le manager doit avoir le droit de modifier certains attributs, certains ne peuvent être modifiée que de l'extérieur comme le nombre de ports sur un routeur. Il doit être possible à l'utilisateur de spécifier certaines valeurs des attributs ;
- définition et modification des inter-relations : permet de définir les relations qui existent entre les composants et les ressources du réseau. L'utilisateur doit être autorisé à modifier, créer ou annuler certaines ;
- initialisation et terminaison des opérations : consiste à vérifier les ressources, les attributs et les commandes utilisateurs.

2. Contrôle de la sécurité Les types de menaces sont la confidentialité, l'intégrité et la disponibilité des informations aux parties autorisées. Les parties qui interviennent sont hardware, software, liens de communication et les données. Les fonctions de sécurité peuvent être décomposées en trois fonctions :

   - maintenir la sécurité des informations : les clés, les droits d'accès, paramètres de sécurité, information d'authentification ...
   - contrôle d'accès aux services : comprend l'authentification, les autorisations et les décisions d'accord ou de refus d'accès aux ressources (comme les codes, les répertoires, les tables de routage, les seuils, les tables de rapport ...)

5.6 Conclusion

La gestion des réseaux permet d'avoir une vision sur l'ensemble du réseau, de gérer les ressources et de faire des statistiques et des prévisions. Le but est d'améliorer le service offert. Sa fonction de supervision permet dans l'approche du placement des tâches parallèles d'avoir des informations sur la disponibilité et l'état de charge des nœuds du réseau. Nous présentons dans le chapitre suivant l'approche SNMP de gestion des réseaux, qui est le protocole le plus utilisé dans les réseaux locaux.

Chapitre 6

Le Protocole de gestion SNMP

Les réseaux deviennent de plus en plus larges, complexes et hétérogènes, d'où un coût élevé pour la gestion. Afin de pouvoir contrôler le coût de la gestion et pouvoir gérer des équipements de types et de constructeurs différents, SNMP[1]a été proposé comme solution pour les environnements TCP/IP. Il a été adopté comme standard par IAB[2] en 1988, par la suite on lui a ajouté en 1991 RMON[3] pour gérer les réseaux locaux(LAN). Il s'est encore évolué pour produire deux protocoles de gestion de réseau SNMPv2 puis SNMPv3. SNMPv2 a été proposé en 1993 et a été révisé en 1996, il supporte les fonctionnalités de la première version et ajoute d'autres surtout de point de vue sécurité. Vue la simplicité et l'interopérabilité entre systèmes hétérogènes, SNMP a été déployé sur une très large proportion d'équipements existant.

6.1 Architecture

Une gestion de réseau dans un environnement TCP/IP utilisant SNMP comme protocole regroupe les éléments suivant : une ou plusieurs stations de gestion, des agents et une base d'information dans chacune d'eux (voir figure 6.1)

Agent il se place sur les nœuds du réseau pour collecter les données. Si un composant n'en possède pas ou il est incompatible avec le manager, on utilise un proxie qui peut gérer un ou plusieurs équipements. Plusieurs constructeurs l'intègrent dans le software UNIX de l'équipement.

MIB la base d'informations de l'agent où sont stockées les données modélisant le réseau. La MIB est ordonnancée lexicographiquement (voir annexe D) .

1. Simple Network Managment Protocol
2. Internet Activity Board
3. Remote network MONitoring

Manager ou gestionnaire, il est installé sur une machine du réseau, son rôle principal est d'interroger les agents pour certaines variables (mécanisme de *polling*). SNMP est basé sur UDP[4], comme UDP est un

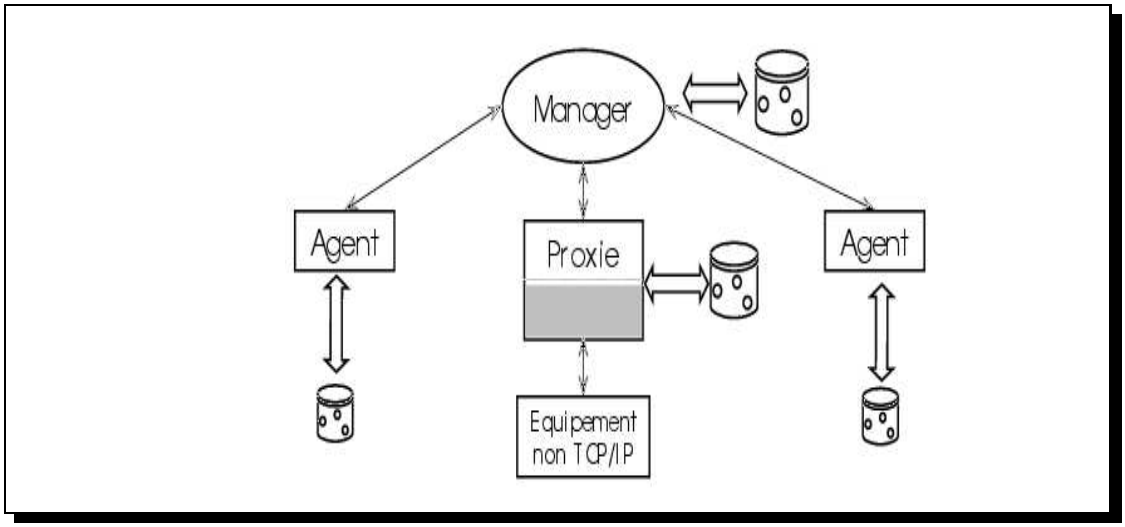


FIG. 6.1 – *Protocole Gestionnaire/Agents*

protocole non connecté, SNMP l'est aussi. Il n'y a pas de connexion continue entre l'agent et le manager et donc le message de l'un ou l'autre peut être perdu. SNMPv1 n'accepte qu'une seule station de gestion dans le réseau. L'administration est donc centralisée d'où des problèmes si le réseau est très large ou le gestionnaire tombe ne panne. SNMPv2 a contourné ces problèmes en admettant plusieurs gestionnaires.

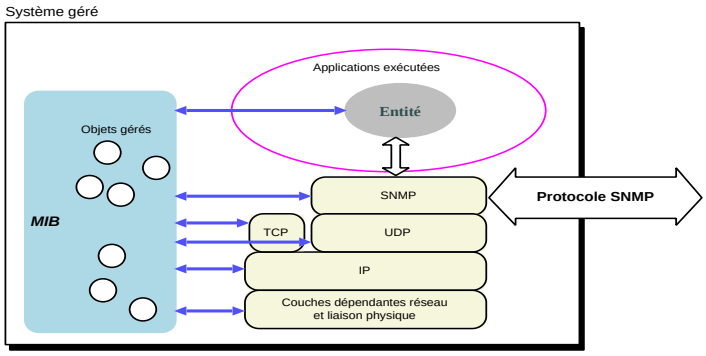


FIG. 6.2 – *Architecture de la gestion SNMP*

4. User Datagram Protocol

6.2 Format SNMP

Dans l'approche SNMP l'échange d'informations entre le gestionnaire et l'agent se fait grâce à des messages SNMP, ils permettent de transporter des requêtes SNMP entre les entités de gestion.

6.2.1 Le message SNMP

Pour qu'une entité SNMP communique avec une autre, elle construit la requête PDU en ASN.1 (voir annexe C) , par la suite elle lui ajoute le nom de la communauté et la version SNMP, en fin il est codé et passé au service de transport.

$SNMP_MESSAGE ::=$
SEQUENCE
{
Version INTEGER
{
$Version - 1(0),$
$Version - 2(1)$
}
Community OCTET STRING ,
Data ANY
}

Le nom de la communauté est le moyen d'authentification du message entre l'agent et les entités de gestion. C'est l'outil de sécurité utilisé par SNMP.Il est comparable à un mot de passe, il doit être entré dans l'agent par un opérateur.

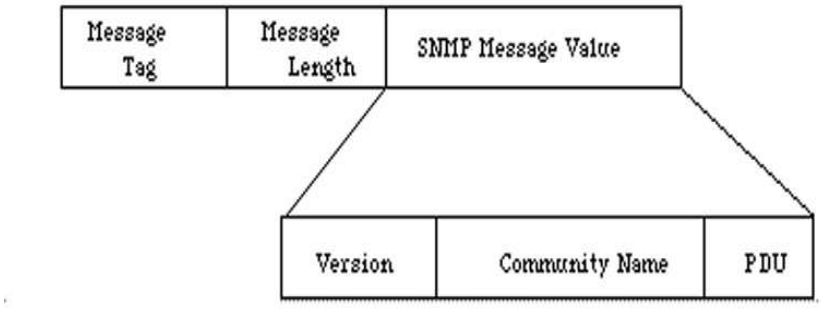


FIG. 6.3 – *La trame SNMP*

6.2.2 Les requêtes

Les requêtes sont de trois types ; de recherche, d'affectation et de notification :

GetRequest une requête émise de la station de gestion vers l'agent pour récupérer la valeur d'une ou de plusieurs variables de la MIB.

GetNextRequest elle permet de chercher la valeur ou les valeurs des variables qui suivent dans l'ordre lexicographique les variables passées en paramètre.

SetRequest le manager demande à l'agent d'affecter des valeurs à une ou plusieurs variables de la MIB, cette requête est atomique et donc si une des affectations est refusée (défaut de variables ou de droit d'accès, ou si la taille de la réponse dépasse les capacités de l'agent) la requête est rejetée.

Trap cette requête permet à l'agent d'informer le gestionnaire d'un problème sur une ressource du réseau (rupture liaison, shut-down ...) .Elle est la seule à ne pas être confirmée.

GetResponse réponse générée par l'agent. Si la requête a réussi elle retourne la ou les valeurs des variables recherchées .

L'accès à la MIB ne se fait qu'au niveau des feuilles, avec une seule requête on ne peut pas récupérer le contenu d'une table ou d'une rangée, ceci simplifie l'implémentation de SNMPv1 mais augmente la charge sur le réseau lorsqu'on a besoin de ces informations. A cet effet, SNMPv2 a proposé la requête GetBulkRequest. Une autre requête InformRequest PDU a été ajoutée à SNMPv2, afin de supporter la gestion multi-gestionnaires, ainsi qu'une révision de la requête Trap.

GetBulkRequest cette requête suit le même principe que GetNext-Request, elle retourne les valeurs des variables qui sont supérieures de point de vue lexicographique à la ou les variables passées en paramètre. La seule contrainte concerne le nombre de variables retournées, limité par la taille du message. Cette requête permet de réduire le trafic sur le réseau.

InformRequest elle permet aux entités SNMPv2 jouant le rôle de gestionnaires d'échanger des informations. L'entité qui reçoit cette requête génère un GetResponse, si la taille maximale d'une réponse de l'entité de réception permet de répondre elle retourne les résultats, sinon elle positionne la réponse à erreur.

SNMPv2-Trap joue le même rôle que la requête Trap de SNMPv1 et en plus, elle utilise le même format des autres requêtes contrairement à celle de SNMPv1 qui a un format propre.

6.2.3 Le format des requêtes

Pour ces cinq requêtes, SNMPv1 a défini deux formats PDU et ceci afin de faciliter la présentation. Le premier format est utilisé pour les requêtes Trap, alors que l'autre est utilisé pour le reste des requêtes. L'un des avantages de SNMPv2 est qu'il utilise le même format pour toutes les requêtes. Chaque requête est identifiée par un numéro qui la distingue des autres, ainsi le gestionnaire n'est pas obligé d'attendre et peut émettre d'autres requêtes. requête elle correspond.

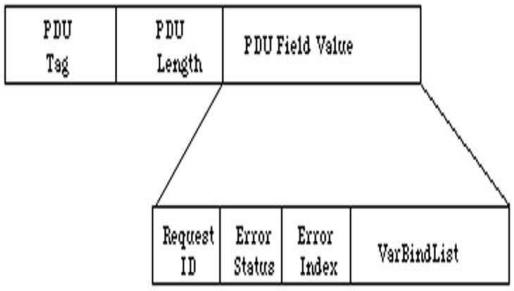


FIG. 6.4 – *Le premier format PDU de SNMPv1*

PDU Tag Type du PDU, GetRequest, GetNextRequest...

Request ID Identité du message.

Error Status Indication d'erreur.

Error Index Pointeur sur l'erreur

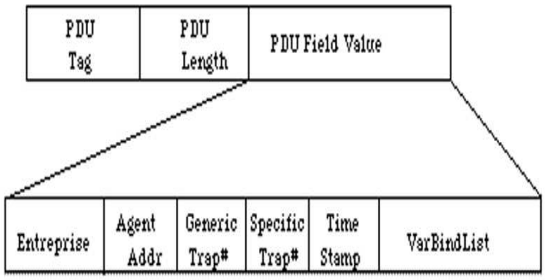


FIG. 6.5 – *Le format Trap de SNMPv1*

Entreprise valeur de l'objet sysObjectId de la MIB de l'agent (notation ASN.1).

Agent Addr valeur de l'objet Network Address de l'agent (adresse IP).

Generic Trap précise la nature de l'événement, par exemple une interface qui tombe en panne ou une ré-initialisation.

Specific Trap identifie l'entrepriseSpecific trap (trap spécifique à l'agent donc non standardisé).

Time Stamp valeur de l'objet sysUpTime de la MIB de l'agent lorsque l'événement s'est produit

VarBindList liste de variables contenant des informations sur le trap

6.3 Les avantages et les inconvénients de SNMP

SNMP est parmi les protocoles de gestion les plus performants, ceci n'empêche pas qu'il ait certains défauts et limitations.

6.3.1 Avantages

- L'avantage majeur de SNMP est sa conception et son installation simples.
- Un autre avantage de SNMP est qu'il est beaucoup répandu. Presque tous les constructeurs de matériel hardware inter-réseaux, conçoivent leurs produits de manière à ce qu'ils supportent SNMP, ainsi il serait facile aux utilisateurs de l'implémenter.
- Enfin, SNMP est basé sur le protocole de transport UDP ce qui nécessite moins de ressources et de connexions simultanées qu'avec TCP. Donc une solution moins coûteuse.

6.3.2 Inconvénients

- Le premier défaut de SNMPv1 est qu'il ne garantit pas la sécurité, puisque des intrus peuvent accéder aux informations véhiculant sur le réseau. En plus ils peuvent provoquer même un shut-down sur certains terminaux. La solution a été en partie résolue dans SNMPv2 qui implémente des mécanismes de sécurité : authentification, protection (cryptage) et contrôle d'accès.
- Le deuxième inconvénient de SNMP est qu'il hérite les défauts de UDP ; il n'y a ni reprise sur erreur ni contrôle de flux. Les requêtes peuvent être perdues, ce qui peut être gênant dans le cas d'un Trap.
- Le manager procède à des interrogations régulières de ses agents : le *polling*.

6.4 MIB

Dans un système de gestion de réseau on a besoin de stocker les informations sur les éléments gérés. Aussi bien pour l'architecture OSI que TCP/IP, cette base de données est connue sous le nom base d'information de données ou MIB (Managment Information Base). Chaque ressource à gérer est représentée par un objet dans la MIB et chaque équipement maintient sa MIB qui reflète l'état des ressources qu'il gère. Ces informations peuvent être structurées sous forme d'une base de données relationnelle, base de données orientée objet dans les environnements OSI, ou sous forme de liste de structures de données et c'est le cas pour SNMP.

6.4.1 Structure de la MIB

La MIB SNMP est représentée sous forme d'arbre hiérarchique, les objets groupés suivant la relation logique, ainsi on trouve le groupe système, interface ...(voir table 6.1) . Les feuilles de l'arbre représentent les objets à gérer, alors que les nœuds pères servent seulement pour le groupement. A chaque objet de la MIB est associé un identifiant en notation ASN.1 de type OBJECT IDENTIFER, ou autrement appelé OID, il est unique dans toute la MIB, sa valeur n'est qu'une séquence d'entiers qui identifie un arc de la MIB. Par exemple, les objets d'administration (se trouvant sous mgmt) leurs OID commence par 1.3.6.1.2 ou iso.org.dod.internet.mgmt. Chaque nœud de l'arbre possède un numéro qui varie de un à n, où n est le nombre de fils sous le nœud père, ainsi la séquence de numéro à partir de la racine à un objet donné est unique. En plus de ça le nœud est caractérisé par un nom unique, les noms sont attribués par ISO et IUT-T[5] ce qui les a rendus standards et partagés par plusieurs types de MIB. Les objets de la MIB SNMP sont décris en utilisant ASN.1, la définition précise le type de l'objet (voir table à l'annexe C.1) son mode d'accès, la plage des valeurs autorisées et ses relations avec les autres objets de la MIB.

6.4.2 La MIB standard

La MIB est une collection de tous les objets que maintient un agent donné. la MIB contient un certain nombre d'informations standards. C'est l'ensemble des informations communes aux nœuds du réseau :c'est la MIB standard (voir figure 6.6) . La MIB standard est décrite dans les RFC[6] Le premier standard utilisé pour la définition des objets d'administration de la MIB standard fut la mib1. Par la suite et vue le

5. ex CCITT
6. Request For Comments

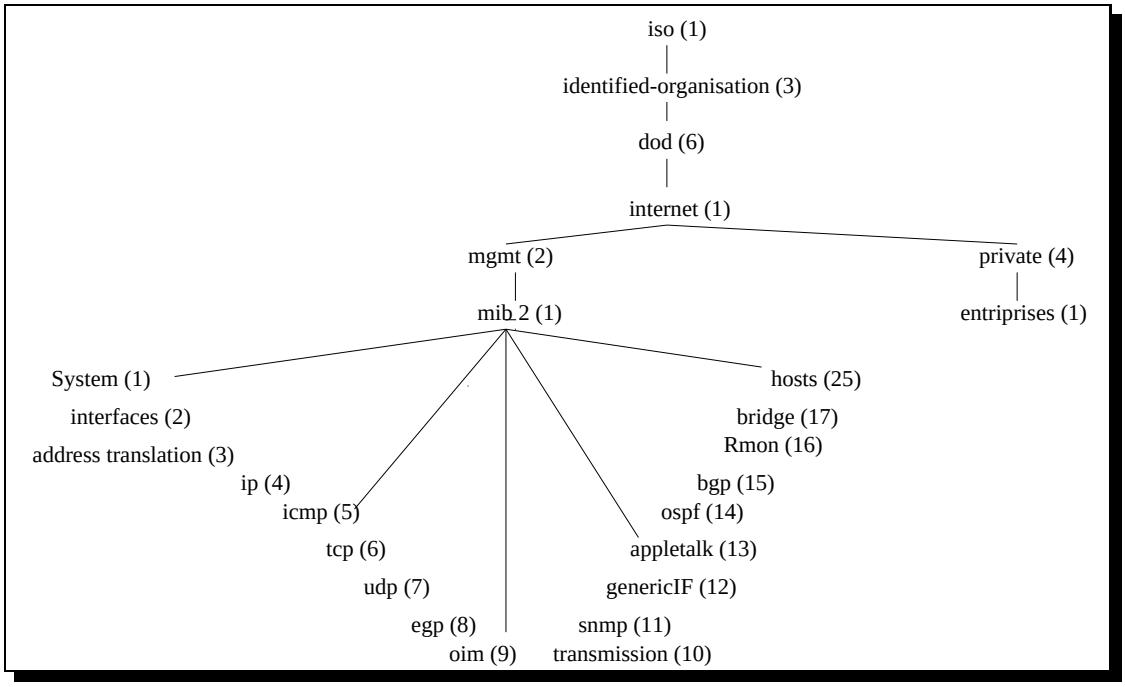


FIG. 6.6 – *La MIB standard*

besoin de gérer d'autres ressources, un second standard fut défini appelé mib2, il a ajouté d'autres objets. Il remplace actuellement la mib1 et utilise son OID 1.3.6.1.2.1. Ces objets définis par la MIB standard, sont parfois insuffisants pour gérer des particularités d'un équipement donné, c'est pour cela on définit la MIB privée.

6.4.3 Les MIB privées

Les MIB privées des sociétés trouvent leurs places sous le nœud Entreprises dans l'arbre de nommage, elle a comme OID 1.3.6.1.4. La MIB privée donne aux constructeurs la possibilité d'ajouter des objets pour gérer d'autres entités et d'autres informations. (voir figure 6.7). Le problème qui se pose est que les vendeurs utilisent trois formats pour décrire les MIB privées :

- Spécification originale de SNMP définie dans RFC1155
- Le nouveau format concis de MIB RFC1212
- Spécification SMI d'OSI

Numéro	Objet	Nombre de sous-objets	Description
1	System	7	Information générales concernant l'agent à travers le système
2	Interfaces	23	Informations concernant chaque interface IP de l'agent
3	Address Translation	3	La table de translation d'adresses qui réalise la correspondance entre l'adresse MAC et l'adresse IP
4	IP	38	Compteurs IP
5	ICMP	26	Compteurs ICMP
6	TCP	19	Compteurs TCP
7	UDP	7	Compteurs UDP
8	EGP	18	Compteurs EGP
9	CMOT	0	Compteurs pour CMOT (protocole OSI équivalent à SNMP)
10	Transmission	0	modes de transmission et protocoles d'accès de chaque interface. Remplacera at
11	SNMP	30	Statistiques du trafic SNMP
16	Rmon	20	Statistiques du trafic réseau

TAB. 6.1 – *La définition de la mib2*

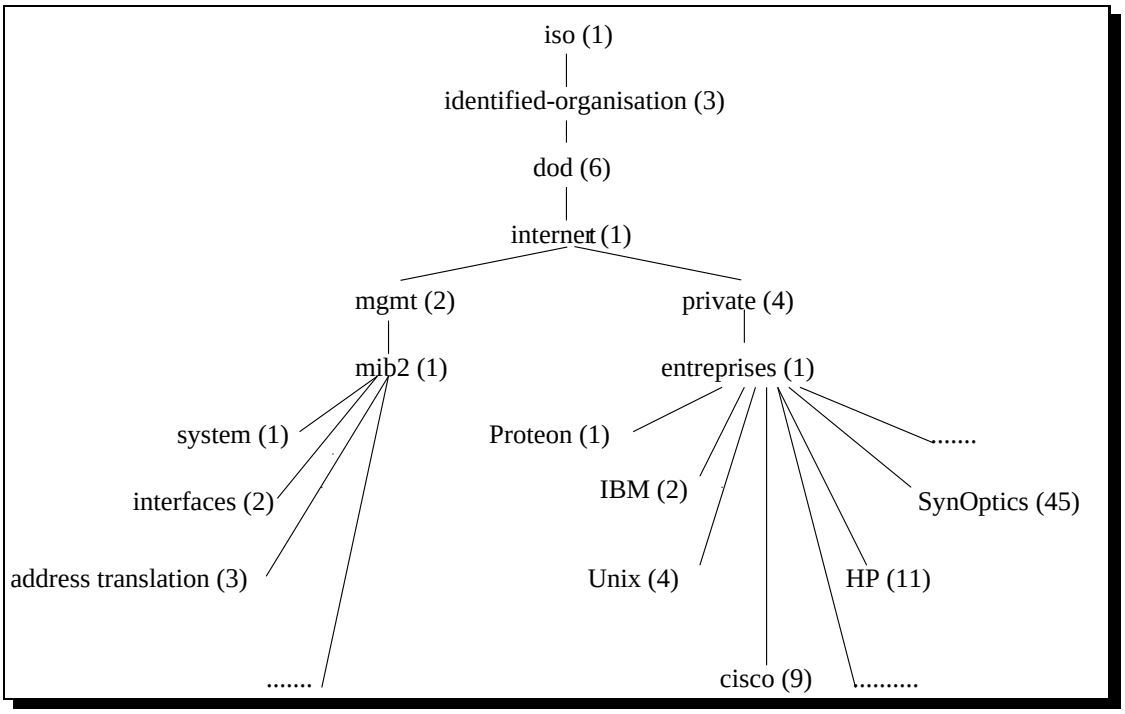


FIG. 6.7 – *La MIB privée*

6.5 Conclusion

Pour récapituler, l'installation d'entités SNMP est simple, ceci explique le fort déploiement de SNMP. Il permet de gérer les réseaux locaux, éventuellement hétérogênes. L'utilisation de SNMP dans l'approche du placement de tâches parallèles représente un bon choix. De plus SNMP profite de l'existence des MIB, aussi bien standards que privées. L'inconvénient de SNMP est le fait qu'il surcharge le réseau par ses requêtes. cependant une solution multi-gestionnaires peut l'atténuer.

Chapitre 7

L'apport des MIB

Dans ce chapitre nous essayons de montrer comment l'utilisation des MIB sert au placement de tâches dans les systèmes répartis, ceci en offrant les informations nécessaires pour la modélisation du réseau, la prévision des coûts du placement et la surveillance des nœuds du réseau.

7.1 Besoins

Le placement de tâches parallèles sur un NoW a besoin de localiser les machines et de connaître l'état de charge du système. Pour répondre à ces questions nous suggérons de décomposer le problème en trois niveaux :

la topologie , il s'agit de découvrir les machines disponibles, leurs performances et leurs services. Une fois les machines localisées, la découverte des composants qui les relient est importante. En fait ces éléments actifs sur le réseau influencent les coûts de communication. Ainsi le placement tient compte de l'impact de ces équipements sur le temps de transfert des messages entre les tâches. Ces informations sont plus ou moins suffisantes pour déduire la topologie du réseau, donc pour faire le placement statique, dans un système dédié au calcul parallèle et sans interférences avec le milieu extérieur. Notre contexte étant le placement dynamique, nous devons passer aux niveaux suivants ;

la surveillance , nous avons besoin de suivre l'état de fonctionnement des composants du réseau. La recherche est faite pour les machines, les équipements de raccordement (commutateur, HUB, routeur, passerelle) et les médias qui les relient ;

l'état de charge , elle concerne les différents composants du réseau. En fait nous ne cherchons pas les machines les plus performantes, mais

plutôt celles qui retournent rapidement des résultats. Par exemple l'utilisation d'un serveur trop chargé alors qu'il existe d'autres machines moins performantes avec plus de disponibilité de leurs processeurs, reste une mauvaise gestion. Un autre exemple, placer deux tâches communicantes sur des machines distinctes alors que le lien de communication est trop chargé ou non performant, n'est pas la meilleure alternative au placement des deux tâches sur une seule machine qui n'est pas trop chargée.

Cette décomposition donne une meilleure vision des informations recherchées. De plus elle reflète les périodes de recherche. En effet la topologie ne doit pas être recherchée à chaque placement, nous la considérons comme information à long terme. La surveillance des composants doit être suivie beaucoup plus que le niveau précédent, nous la considérons comme information à moyen terme. Le dernier niveau doit recherché au moment du placement, il est considéré comme à court terme.

7.2 Topologie du réseau

La topologie du réseau définit aussi bien la structure d'implantation physique que logique. En fait la configuration physique reste insuffisante pour déduire le coût de communication entre deux machines du réseau. On trouve plusieurs types de réseaux locaux, le plus répondu est *Ethernet*. Il est utilisé pour le réseau local **LORIA**, sa mise en place est faite grâce à un routeur le connectant au réseau Internet, deux commutateurs cisco, derrière les ports de ces derniers sont placées les machines sous forme arborescente en utilisant les **HUB**. La découverte de la topologie revient à chercher les informations décrites ci dessus.

identification : déterminer l'identité ou les services qu'offre un nœud donné du réseau. Cette information nous l'avons déduite à partir de sysServices. Par exemple un commutateur présente un service de type 2 (liaison de donnée) , un host présente un service de type 72 (supporte la pile TCP/IP et des applications au dessus) ... ;

implantation logique : revient à chercher le type des interfaces et estimer leurs impacts sur le coût des communications. Ces informations sont déduites des variables suivantes :

- ifType dans la sous-MIB interface, elle indique son type (csmacd, FDDI ...),
- ifSpeed se place sous la même sous-MIB, détermine la vitesse de l'interface du nœud, ceci permet de connaître son influence sur le coût de communication.

implantation physique : détermine les liens qui existent entre les nœuds du réseau. Ces informations sont repérées en général grâce aux tables de routage des composants de connexion, car un host n'a pas besoin de connaître dans ses communications à quel équipement il est connecté.
Pour les commutateurs nous utilisons les variables ci dessous.

- dot1dTpFdbTable qui est la table de routage se trouvant sous le groupe dot1dBridge. Elle retourne les adresses des machines qui sont derrière un port donné,
- dot1dBaseBridgeAddress et dot1dBaseBridgeAddress se trouvent sous le même sous groupe, elles permettent de connaître l'adresse du commutateur qui lui est connecté.

Pour les HUB, il existe une table standard rptrAddrTrackTable qui permet de récupérer les adresses des machines qui sont connectées aux ports du HUB. Cependant cette table est optionnelle et elle est souvent non implémentée. C'est pour cela nous avons fait recours aux MIB privées. Par exemple pour les HUB HP les variables ci dessous sont utilisées.

- hubportMacAddress détermine l'adresses MAC de la machine connectée à un port donné,
- hubAddressChunk détermine les adresses des machines qui sont derrière un port donné dans le cas où il est connecté à un autre équipement de raccordement.

D'autres informations peuvent être utiles pour d'autres types de réseaux, comme le réseau ATM où les machines mémorisent l'adresse IP de l'équipement connecté à une interface donnée : atmInterfaceMyNeighborIpAddress, aussi pour ceux de type FDDI dans la sous MIB privée de cisco la variable sysFddiRing-Next permet de déduire l'adresse MAC de la prochaine station dans le réseau FDDI.

L'utilisation de ces variables permet d'identifier les nœuds du réseau, déduire l'implantation physique et de mesurer l'impact de ses composants sur le temps de transfert des messages entre les tâches. Comme exemple, cette phase permet de déduire la topologie de la figure 7.1.

7.3 Surveillance

Le principe du placement dynamique est de tenir compte de l'état des nœuds et des liens de communication. En fait dans un système non dédié la probabilité de panne ou de dé-connexion de machines est élevée, en plus les liens de communication peuvent être surchargés ou bien les files d'attentes aux niveau des interfaces ont débordé. Pour cela nous cherchons des variables qui renseignent sur la disponibilité des

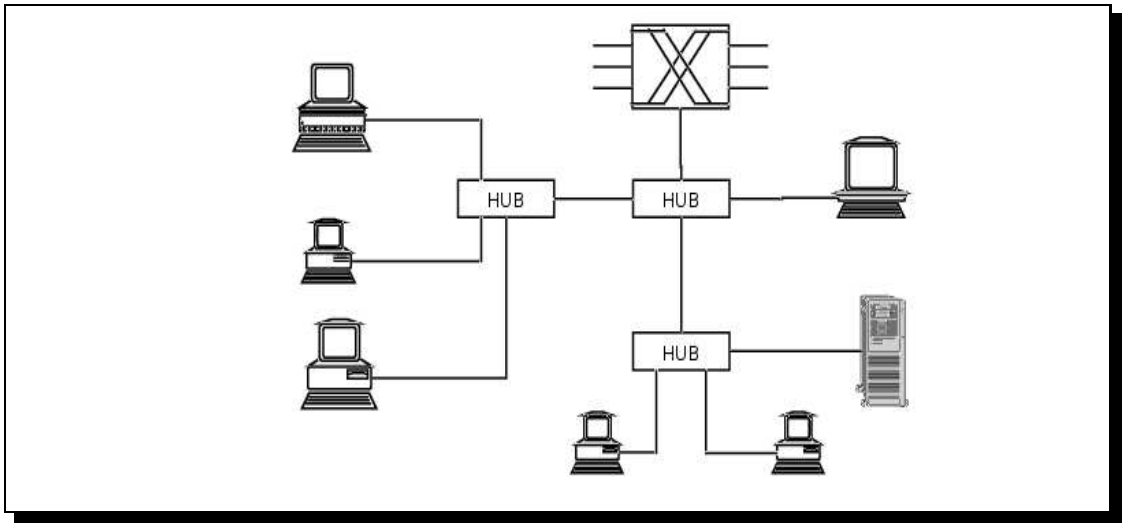


FIG. 7.1 – *Découverte de la topologie*

ressources.
Pour les hosts nous trouvons que les machines limitent le nombre de processus à héberger, donc une machine qui atteint ce maximum doit être éliminée au moment du placement. Cette information nous pouvons la déduire à partir de hrSystemProcesses et hrSystemMaxProcesses de la sous-MIB host. Cependant cette information n'est pas très utile vue que la performance de la machine dégrade avant d'atteindre ce maximum. Nous trouvons aussi pour les hosts la variable hrDeviceStatus qui décrit l'état de la ressource (réseau, disque ou processeur) .
Pour les interfaces, nous trouvons l'objet qui détermine son état ifOperStatus (actif, non actif ou en test) . Cette information nous la trouvons aussi sous la MIB privée cisco : locIfLineProt, aussi elle se présente sous forme de Trap : linkDown et linkUp. Cette même information existe pour les HUB à partir de hubPortsOperStatus .
Les variables trouvées sont suffisantes pour surveiller les composants du réseau. Elles permettent d'exclure du graphe du réseau déduit à la première phase, les nœuds qui sont en panne, hors service ou incapables de présenter de service. La figure 7.2 montre le résultat de cette phase en l'appliquant sur l'exemple ci dessus.

7.4 Placement

C'est le troisième niveau, il cherche l'état de charge des nœuds du réseau et des liens de communication (voir figure 7.3). La recherche

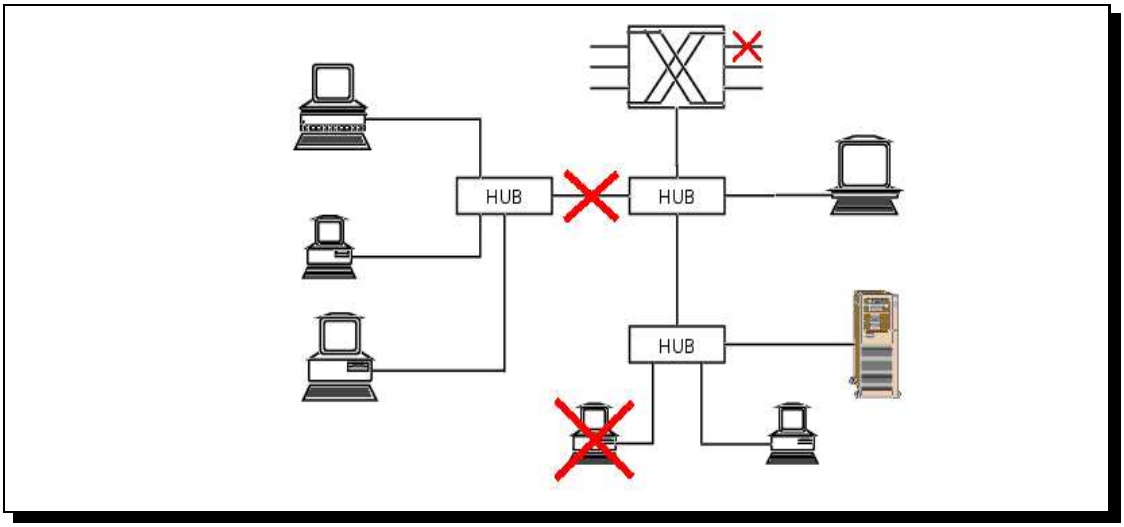


FIG. 7.2 – *Surveillance du réseau*

est faite pour les machines et le réseau.

machines : la première information que nous cherchons est la charge CPU des machines. Les variables trouvées sont :

- hrProcessorLoad indique la moyenne d'utilisation du processeur pendant la dernière minute. Cette information n'est pas très utile, car elle ne donne pas la vraie charge de la machine type UNIX,
- hrStorageAllocationFailures et hrStorageUsed donnent des informations sur l'espace mémoire et des statistiques sur les allocations mémoires, donc pour les tâches qui ont besoin d'un espace mémoire important, il est intéressant de connaître ces informations,
- busyPer, avgBusy1 et avgBusy5 sous la MIB privée cisco donnent le taux d'utilisation du processeur. Cependant sans connaître la construction des routeurs et des commutateurs, ce n'est pas possible d'utiliser ces informations.

Ces informations sont insuffisantes pour calculer la charge des machines. En fait le taux d'utilisation du processeur pendant une période donnée ne permet pas de prévoir sa future charge.

réseau : il y a des variables qui renseignent sur les équipement de connexion et sur les liens de communication.

1. les nœuds de connexion, il existe des informations statistiques sur les interfaces pour prévoir le coût des communication, ainsi nous utilisons la longueur de la file d'attente au niveau des interfaces : ifOutQlen.

2. les liens de communication, il existe des informations statistiques qui donnent une idée sur le lien de communication.
   - dot3StatsMultipleCollisionFrames, pour les interfaces likeEthernet, indique le nombre de paquets émis après un nombre très grand de collision,
   - dot3statsExcessiveCollisions et etherOutMaxCollisions donnent le nombre de paquets rejetés à cause d'excessive nombre de collision ou à cause d'un très long temps de transit à travers cet équipement.

   Ces informations peuvent être à la limite utilisées au deuxième niveau, si nous imposons des seuils pour les services, par exemple une machine connectée derrière un nœud de raccordement où il y a un très grand nombre de collision, peut être éliminée puisque un éventuel placement peut engendrer un coût élevé des communications.

Ces informations restent insuffisantes pour estimer l'état de charge du système, nous avons besoin d'autres variables qui reflètent la charge réelle des nœuds et des liens du système.

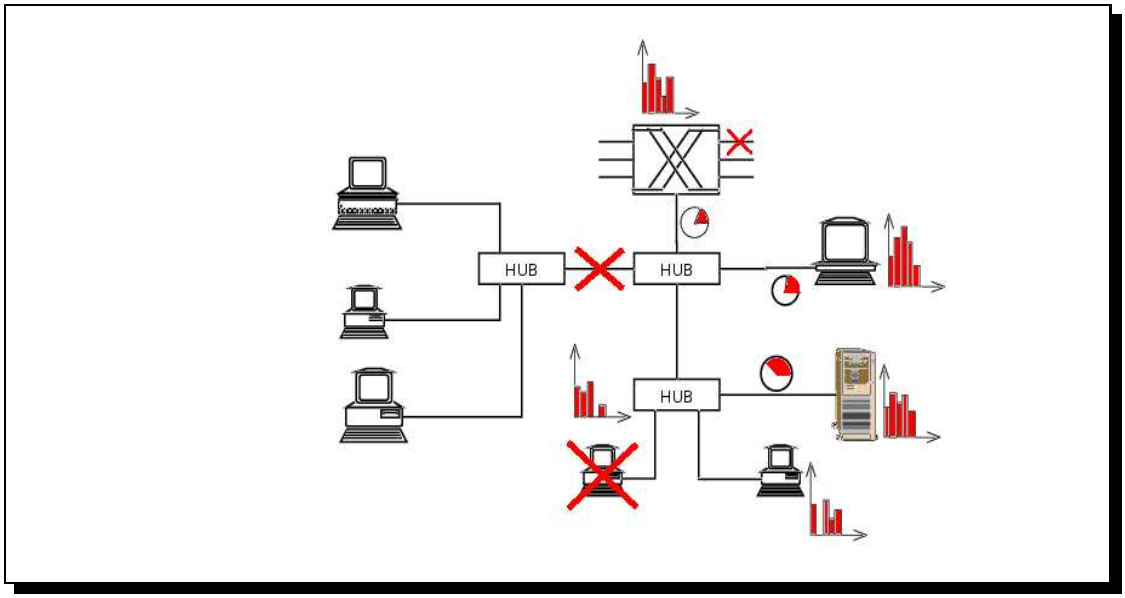


FIG. 7.3 – *Placement sur le réseau*

7.5 Démarche pour la découverte de la topologie

Les information qu'on trouve dans la MIB ont une durée de vie, par exemple une entrée dans la table de routage ne reste pas plus que

quelques minutes si la machine concernée n'a pas fait partie d'une communication. Donc nous avons deux solutions pour avoir les adresses des machines qui existent dans le réseau. La première se fait d'une façon manuelle, dès qu'on ajoute une nouvelle machine on met à jour la base de données. La deuxième méthode consiste à utiliser un processus qui écoute le réseau, ainsi nous pouvons déduire les adresses des machines qui communiquent sur le réseau. Cet processus est basé sur le protocole **ARP**[1], ainsi dés qu'il détecte un paquet **ARP**, il ajoute l'adresse de la machine communicante dans la base de donnée.
Cette base de donnée nous permet de connaître une grande partie des machines qui existent sur le réseau. Par la suite nous interrogeons toutes ces machines pour déduire le type de services qu'elles offrent. Bien-sûr il y a des machines qui ne répondent pas : certaines sont déconnectées du réseau, d'autres ne disposent pas d'agent. Nous ne pouvons pas affirmer que les machines qui n'ont pas répondu à la requête **SNMP**, sont déconnectées. Pour cela, nous faisons un Ping (basé sur **ICMP**[2]) pour connaître si elles sont connectées ou pas.
La première phase que nous venons de faire est très bénéfique, d'une part nous avons récupéré les adresses des machines et éventuellement leurs services, d'autre part elle permet de mettre à jour les tables de routage sur les nœuds de raccordement. Maintenant, il suffit d'analyser les différentes tables de routage disponibles pour déduire la topologie du réseau. Ces tables de routage peuvent aussi nous aider pour découvrir d'autres adresses des machines qui se trouvent sur le réseau et ne figurent pas dans la base de donnée.
Pour analyser les tables de routage, nous commençons par les tables des commutateurs, chacune nous retourne les adresses des machines qui sont derrière chaque port. Si pour un port donné les adresses des **HUB** ne figurent pas, alors la recherche derrière ce port ne peut pas aller très loin. Une solution à ce problème est d'utiliser le service *finger*, grâce auquel nous pouvons distinguer entre les machines et les autres élément du réseau. Par contre si les adresses des **HUB** sont repérées, nous pouvons détailler davantage la topologie derrière le commutateur. En fait le **HUB** nous précise pour chaque port l'adresses de la machine qui lui est connectée, ou les adresses des machines qui sont derrière si ce port est relié à un autre nœud de raccordement. Puisque l'architecture est sous forme d'arbre donc en partant des **HUB** qui se trouvent aux extrémités, nous arrivons de proche en proche à reconstruire le graphe avec une certaine qualité. Ce travail reste extensible dans le cas d'ajout de routeurs dans le réseau, la recherche commencerait à ce niveau puis passer aux commutateurs.

1. Addresses Resolution Protocol
2. Internet Communication Message Protocol

7.6 Conclusion

Les informations recherchées pour le placement, nous les avons structurées en troix niveaux. La topologie qui permet de décrire la répartition physique des nœuds du réseau, elle définit aussi les capacités et les performances des machines, composants de raccordement et des liens de communication. Le deuxième niveau est la surveillance permettant de suivre l'état des nœuds du réseau. le troisième niveau est l'état de charge, essaye de chercher la charge des machines pour y lancer les tâches, celle des équipements de connexion pour mesurer leurs impact sur le coût des communications inter-tâches. Les informations cherchées pour le premier niveau, sont présentes dans la MIB standard et les MIB privées. Pour le deuxième niveau, les informations qui existent sous la MIB standard sont suffisantes. Par contre, pour évaluer l'état de charge du système, les informations trouvées sont insuffisantes. La MIB RMON permet de surveiller les machines et le trafic, son inconvénient est que nous devons l'interroger périodiquement pour faire des statistiques. Il s'en suit que l'ajout d'autres variables à la MIB est une nécessité.

Chapitre 8

Conclusion

Dans ce document nous avons présenté le travail effectué durant le stage de fin d'études d'Ingénieur en informatique à la faculté des sciences de Tunis. Ce stage a été une bonne opportunité pour apprendre de nouvelles connaissances et s'initier au domaine de la recherche. Le sujet a permis de découvrir le domaine du parallélisme lié aux stations de travail (NoW) , ainsi que des mécanismes utiles pour obtenir de bonnes performances : le placement de tâches. Un autre aspect découvert, a été le domaine de la gestion de réseaux utilisé comme complément de l'environnement développé dans l'équipe **RESEDAS**, en particulier le protocole **SNMP**.

Au niveau de la synthèse des variables des MIB recherchées, nous avons proposé une décomposition en trois niveaux permettant de mieux structurer les informations. Les variables disponibles pour le niveau topologique sont suffisantes pour les besoins du placement, même si elles sont incomplètes. Pour le niveau de surveillance, toutes les informations sont présentes dans les MIB standard pour traiter efficacement cette opération. Par contre, au niveau des états de charge des équipements, les seules variables existantes, sont largement insuffisantes pour décrire les charges des machines et des liens. Il est nécessaire d'introduire de nouvelles variables, placées dans une MIB privée (par exemple resedasMIB) calculant les taux de charge des machines (la run-queue **UNIX**) et des réseaux.

Le second aspect du travail concerne la découverte de la topologie physique du réseau local. L'outil développé en Java permet de découvrir semi-automatiquement les équipements pourvues d'un agent **SNMP** ainsi que leurs caractéristiques. De plus, grâce aux tables de routage (niveau 2 **OSI**) nous avons pu établir la topologie des autres équipements possédant une adresse **IP**.

Malheureusement, des problèmes liés à la bibliothèque SNMP-Java utilisée, nous ont empêchés d'établir la topologie complète de l'ensemble du site.

De nombreuses perspectives s'offrent à nous dans la continuité de ce travail. Le premier aspect concerne la recherche automatique de la topologie en utilisant non seulement SNMP, mais également les outils de gestion tels que les pages jaunes (yp) et les protocoles de routage. Ainsi, une seconde perspective peut être envisagée qui est de définir la topologie pour des réseaux au de-là des routeurs. Une autre perspective est la génération graphique de la topologie du réseau.

Bibliographie

[1] E. Dillon. *Proposition pour la maîtrise de la programmation par échange de messages.* Thèse de doctorat, Université Henri Poincaré-Nancy, oct 1997.

[2] L. et Perkins. *Teach yourself JAVA in 21 days.* sans.net, 1996.

[3] L. Lamport. *Latex.* Addison Wesley, 1992.

[4] P. Rollin. *Réseaux haut débit.* Hermes, 1995.

[5] W. Stallings. *SNMP, SNMPv2 and RMON.* Addison Wesley, 1996.

[6] A. Tanenbaum. *Réseaux, Architectures, Protocoles, Applications.* InterEditions, 1990.

Annexe A

Glossaire

A.1 Ethernet

Ethernet est le réseau local le plus répondu, sa fiabilité et la modération de ses composants ont participé à son expansion. Le débit de 10M bit/s a été longuement satisfaisant pour les applications en réseaux locaux. Mais le besoin en bande passante a augmenté avec les applications client/serveur, multi-média. Pour répondre à ce besoin en débit trois technologies sont à adopter :

- les concentateurs commutés: offrent un débit de 10M bit/s pour les stations connectées, le concentrateur se charge de faire la liaison entre la station émettrice et la station de réception durant l'émission du paquet. Ainsi le nombre de collision diminue puisqu'il se réduit seulement au cas où plus qu'une station tente d'émettre à une même destination.
- Ethernet 100M ou la norme IEEE 802.14: on garde le protocole Ethernet, tout en offrant un débit de 100M bit/s, en changeant les coupleurs.
- la norme 100 BAS VG proposée par HP, elle est basée sur le principe du polling, on a besoin aussi de changer les coupleurs des stations, mais on peut conserver les systèmes de câblage.

La mise en oeuvre de ces trois technologies se fait grâce au HUB.

A.2 HUB

Boîtier multi-port pour Ethernet sur paire téléphonique et pour Starlan. Émule le comportement du bus et répète les signaux vers les stations et les HUB amont.

A.3 interface

C'est une frontière commune à deux équipements, en général définie par un ensemble de caractéristiques physiques, électroniques et fonctionnelles (protocole ou programme) autorisant l'échange d'informations entre ces équipements.

Annexe B

Introduction à CMIP

B.1 Principes de CMIP

C'est le *Common Management Information Protocol*, c'est un protocole de gestion des réseaux, il a été normalisé par l'ISO. On a pensé qu'il remplacerait le protocole SNMP, vue qu'il est plus large et plus complet. Ceci n'a pas lieu vue qu'il se limite aux environnements OSI. Le principe de CMIP ressemble à SNMP, en fait il utilise les requêtes PDU. En CMIP les variables sont vues comme une structure complexe de données ayant plusieurs attributs. Ainsi on trouve pour les variables des attributs qui décrivent ses caractéristiques (type des données, les droits d'accès) , des informations sur leurs comportements (les actions que chaque variable peut lancer) , enfin des informations sur les notifications qu'elles génèrent.

B.2 Comparaison entre CMIP et SNMP

Le fait que les variable dans CMIP ne reflètent pas seulement une information comme dans SNMP, mais ajoute des actions, constitue le majeur avantage de CMIP. Son deuxième avantage et contrairement à SNMP, est qu'il implémente des mécanismes de sécurité tels que l'autorisation, le contrôle d'accès et des procédures de login sécurisé.
Ces avantages nous poussent à se demander qu'est ce qu'il empêche

CMIP pour remplacer SNMP, ceci est dû à deux inconvénients de CMIP.

- CMIP a besoin de dix fois plus de ressources que SNMP et est donc supporté par beaucoup moins de réseaux,
- l'utilisation de ses variables n'est pas simple pour programmer, vue que le nombre est très grand.

Des recherches ont été faites afin de réduire la taille du protocole en changeant ses spécifications, mais les résultats n'ont pas pu concurrencer SNMP.

Annexe C

Les Types ASN.1

ASN.1 ou syntaxe abstraite des applications, est un langage formel qui a deux applications. Il définit une syntaxe abstraite des applications sur les données utilisées pour les protocoles PDU. Il est aussi utilisé comme notation dans les documents. Ainsi il permet de décrire les MIB SNMP et OSI. Il a été normalisé par ISO et UIT-T

C.1 Les types simples

- **INTEGER** : type entier pouvant être énuméré ou restreint,
- **OCTET STRING** : chaîne de caractères,
- **OBJECT IDENTIFIER** : identificateur d'objet,
- **NULL** : utilisé pour modéliser l'absence d'un champ dans une structure.

C.2 Les types prédéfinis

Les types prédéfinis sont tous composés à partir des types simples. Nous citons à titre d'exemple des types de SNMP.

- **Opaque** : type universel permettant la représentation de tout type d'informations sous forme de chaîne d'octets,
- **TimeTicks** : unité temporelle basée sur un entier non négatif,

- **Gauge : type entier utilisé notamment pour la définition de variables représentant des taux d'erreurs,**
- **Counter : compteurs.**

C.3 Les types structurés

- **SEQUENCE : permet la composition de structures,**
- **SEQUENCE OF : permet la composition de listes ordonnées,**
- **SET : ensemble,**
- **SET OF : permet la composition de listes.**

C.4 Les types ASN.1 utilisés dans SNMP

Type	Description	SNMP	SNMPv2
INTEGER	Entier dans l'intervalle $-2^{31} \ldots 2^{31} - 1$	x	x
OCTET STRING	Suite d'octets pour des données binaires ou textuelles	x	x
BIT STRING	Énumération de bits nommés		x
OBJECT IDENTIFIER	Identificateur de nom OSI ; séquence de maximum 128 entiers positifs	x	x
UInteger32	Entier dans l'intervalle $0 \ldots 2^{32} - 1$		x
Counter32	Entier positif incrémentable modulo 2^{32}	x	x
Counter64	Entier positif incrémentable modulo 2^{64}		x
Gauge32	Entier positif incrémentable ou décrémentable ne dépassant pas une valeur maximum comprise dans l'intervalle $0 \ldots 2^{32} - 1$	x	x
TimeTicks	Entier positif représentant une différence de temps modulo 2^{32} en centièmes de seconde	x	x
IpAddress	Adresse Internet de taille 32 bits	x	x
NsapAddress	Adresse OSI de taille variable indiquée dans le premier octet		x

TAB. C.1 – *Types ASN.1 autorisés dans SNMP*

Annexe D

Ordre lexicographique

Étant donné deux séquence d'entiers non négatifs $X(x1.x2.x3...x_m)$ et $Y(y1.y2.y3...y_n)$. X précède Y si et seulement si :

- **$(x_i = y_i)$ pour tout i tel que $1 <= i < k$ et $x_k < y_k$**
- **$(x_i = y_i)$ pour tout i tel que $1 <= i <= m$ et $n > m$**

Cet ordre est utilisé pour SNMP, il a un grand apport, en fait le gestionnaire ne connaît pas exactement toute la structure de la MIB que l'agent lui fournit. Donc il est intéressant d'accéder aux objets sans spécifier leurs noms et ceci grâce au GetNextRequest.