République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahlab, Blida
USDB.

Faculté des sciences.
Département informatique.

Mémoire pour l'obtention
d'un diplôme d'ingénieur d'état en informatique.

Sujet :

Administration de réseau étendu par les agents mobiles

Présenté par : HADJI Naoufel.
TOUBAL Samir.

Promoteur : D.E Menacer

Soutenue le: 25/12/2006, devant le jury composé de :

M^me Aoussat, Maitre assistant, Université de Blida	**Président**
M^r. Cherif Zahar, Maitre assistant, Université de Blida	**Examinateur**

- 2005/2006-

Remerciements

Nous tenons à adresser nos sincères remerciements à notre promoteur MR.: Djemal Eddine MENACER qui a su nous conseiller par ses critiques constructives et nous guider dans notre modeste travail.

En deuxième lieu, nous remercions notre copromoteur KERDOUUSSE Mohamed pour les conseils qu'il nous a donné. Ainsi Mr Zine Eddine ZINO

Nous tenons à remercier le chef de centre de switching ainsi que toute l'équipe de l'IN pour avoir assurer notre projet de fin d'étude.

Nous remercions aussi les professeurs de département Informatique pour avoir assurer notre formation.

Nous remercions aussi les membres du l'ensemble département informatique, qui nous a assuré l'environnement adéquat afin de mener à bien notre travail.

Nos vifs remerciements s'accordent au Mme Aoussat d'avoir accepter de présider le jury de notre soutenance. Ainsi que Mr Cheirif Zahar d'être l'examinateur de notre Mémoire.

Nous exprimons notre plus grande reconnaissance et nos vifs remerciements à toute l'équipe de LOGITEL ALGER pour leur aide

Nos vifs remerciements vont également et tout particulièrement à nos Enseignants d'encadrement.

Nos remerciements vont aussi à tous ceux et celles qui ont participé de près ou de loin à l'élaboration du présent travail.

Enfin, nous tenons à remercier tous nos amis et collègues pour leur soutien moral tout au long de la préparation de ce mémoire.

Sommaire

Liste des Figures

CHAPITRE I : INTRODUCTION AU ADMINISTRATION DES RESEAUX

CHAPITRE II : LES PROTOCOLES DE GESTION DES RESEAUX

CHAPITRE III : LES SYSTEMES D'ADMINISTRATION

CHAPITRE V : ADMINISTRATION PAR LES AGENTS MOBILES

CHAPITRE V : CONCEPTION DE LA CONSOLE D'ADMINISTRATION

CHAPITRE VI: IMPLEMENTATION ET TESTS

Liste des tableaux

Introduction générale

INTRODUCTION GENERALE

De nos jours, toutes les entreprises sont équipées au minimum d'un réseau local, et pour les plus importantes d'entre elles de réseaux Wide Area Network. L'administration des réseaux est devenue un point critique dans toutes ces entreprises de par l'hétérogénéité des éléments composant le réseau. Du fait de cette hétérogénéité, il devient donc indispensable de surveiller en permanence les éléments clefs du réseau, afin que les utilisateurs ne soient pas affectés par des incidents de fonctionnement et que la perte d'exploitation soit la plus faible possible en cas d'incidents. Pour cela, les entreprises investissent dans des outils d'administration de réseau très coûteux et qui ne sont pas forcément adaptés à chacune d'entre elles. Le système d'administration de réseau fournit des mécanismes de surveillance, de gestion et de contrôle du réseau par la collecte des informations sur le fonctionnement du réseau, il joue un rôle prépondérant permettant aux administrateurs de visualiser en temps réel le comportement des éléments du réseau, les changements de configuration et les pannes mais permet aussi de superviser les problèmes liés à la sécurité. Un système d'administration de réseau est composé d'une ou plusieurs stations de supervision qui communiquent entre elles et avec les éléments du réseau par un protocole d'administration de réseau, les plus connus étant **SNMP** (Simple Network Management Protocol) ou **CMIP** (Common Management Information Protocol).

Les systèmes d'administrations actuels sont orientés propriétaires, c'est-à-dire que les stations d'administration sont liées à une machine et un environnement système particulier. De tels systèmes sont basés sur une station d'administration centralisée qui contrôle le fonctionnement de l'ensemble du réseau et archive les alertes reçues sous forme de tickets. De ce fait, la station d'administration devient rapidement un goulot d'étranglement au vu du trafic réseau d'administration qui lui arrive et des informations pertinentes qu'elle se doit d'offrir à l'administrateur du réseau. Elle devient donc un élément critique du réseau ce qui a motivé de nombreux travaux pour élaborer des systèmes d'administration réseaux à base d'agents mobiles.

Notre étude sur cette thématique d'agents intelligents dotés d'une capacité de mobilité, a pris la forme d'un développement d'une console d'administration de réseau basée sur la plateforme ProActive. Cette plate-forme offre les capacités nécessaires au support d'objets actifs et migrants. L'objectif essentiel de cette étude est de démontrer la faisabilité et les intérêts pratiques d'un système d'administration à base d'agents mobiles au-dessus de la plate-forme ProActive. En effet, avec cette technologie on veut minimiser la bande passante

utilisée par le trafic de gestion et réduire la durée de la connexion. Autrement dit, on ne veut pas encombrer le réseau par le trafic lié à l'administration.

Le présent mémoire est subdivisé en trois parties :

D'abord, une première partie présente l'état de l'art où nous évoquons les différents aspects de la gestion des réseaux en général, puis on spécifie la gestion classique des réseaux via les protocoles SNMP et CMIP.

En suite, une deuxième partie consacrée aux différents systèmes d'administration des réseaux à savoir le système centralisé, hiérarchique, le système basé sur la technologie du web et nous développerons par la suite l'avènement des agents mobiles et leur opportunité dans les systèmes d'administration de réseau, ainsi que l'évaluation des plates-formes objet offrant des outils de migration.

La troisième partie de ce mémoire comprend deux chapitres. Le premier concerne la conception d'un itinéraire d'administration suivi par celle de la console d'administration en utilisant le langage UML (**U**nified **M**odeling **L**anguage). Nous terminerons notre travail par un chapitre dont lequel nous évoquons les contributions apportées et les prolongements futurs de notre solution.

Partie I

Etat de l'art

Chapitre I

Introduction à l'administration des réseaux

Dans ce chapitre :

- L'administration des réseaux
- Les modèles d'administration
- La matière première utilisée dans les outils d'administration

CHAPITRE I : INTRODUCTION AU ADMINISTRATION DES RESEAUX

INTRODUCTION

Les télécommunications ont connu ces dernières années un essor formidable. Cela s'est traduit par une augmentation de la complexité de l'opération des réseaux. Cette complexité peut être analysée selon deux directions: l'évolution de la taille des réseaux et la complexité des équipements et protocoles mis en oeuvre.

L'utilisation des réseaux s'est généralisée. Dans les entreprises, on peut observer un abandon de l'informatique centralisée au profit d'une informatique distribuée au travers d'un réseau. Bien que des serveurs existent toujours (jouant ainsi le rôle des mainframes), les terminaux sont devenus plus intelligents et plus puissants.

Le passage de l'informatique centralisée au paradigme distribué a non seulement amené une croissance de la taille de réseaux, mais également une augmentation de la complexité des équipements utilisés. En effet, motivés par des besoins de performance, sécurité et versatilité, des équipements souvent complexes tels que routeur, switch et firewall ont fait leur apparition sur la scène de l'informatique d'entreprise.

La multiplication des réseaux et outils de communication que nous connaissons depuis quelques années entraîne un nouveau besoin : celui d'une gestion globale et distante d'un système et de chacun de ses sous-ensembles. C'est ce qu'on appelle l'administration d'un réseau. Cette terminologie recouvre l'ensemble des fonctions nécessaires pour l'exploitation, la sécurité, le suivi et l'entretien du réseau. L'administrateur doit pouvoir suivre en temps réel l'état de fonctionnement de son réseau (surveillance et diagnostic des incidents, mesure de la charge réelle, maintenance, contrôle, information des utilisateurs, ...) afin de pouvoir intervenir (réparation, ajouts ou retraits d'abonnés, contrôle des accès ...). Ses outils doivent lui permettre de suivre les performances de chaque élément afin d'identifier les points faibles. L'objectif étant de planifier, d'optimiser et de quantifier pour gérer les évolutions du réseau.

I. L'ADMINISTRATION DES RESEAUX

I.1. Définition

Administrer, c'est savoir gérer, cela devient indispensable pour les systèmes informatiques d'aujourd'hui. L'administration de ces systèmes est stratégique pour un bon fonctionnement global de l'entreprise, et cela reste ardu car il se matérialise par un ensemble d'éléments humains et matériels (équipement). Gérer un réseau revient à contrôler, coordonner et surveiller les différentes ressources mises en œuvre.

La mise en œuvre de la gestion se réalise au travers :

- D'activités: maintenance, exploitation, planification, surveillance et sécurité.
- De modèles: modèle informationnel, modèle architectural, modèle de communication, modèle fonctionnel et modèle organisationnel.
- De normes et de standards : CMIS, CMIP, MIB, SNMP, etc.
- D'outils : plates-formes et méthodes.

Dans les paragraphes qui vont suivre, on évoque les activités et les modèles d'administration, les normes et les standards seront traités implicitement plus loin.

I.2. Les activités d'administration

I.2.1. La maintenance

C'est l'ensemble des actions permettant de maintenir ou de rétablir un équipement ou une installation dans un état spécifié en mesure d'assurer un service bien déterminé. On distingue plus particulièrement:

a) **La maintenance préventive :** A pour objectifs:

- Diminuer les pannes et les imprévus de façon à réduire les coûts de la maintenance curative.
- Améliorer la qualité des produits.
- Prolonger la durée de vie du matériel.
- Améliorer la sécurité (pour le matériel comme pour le personnel).
- Améliorer la fiabilité et la maintenabilité du matériel.

b) **La maintenance curative :** Dans ce genre de maintenance, il faut traduire la faute constatée par l'utilisateur en terme approprié à la technologie réseau, ainsi il faut associer la panne à une catégorie aussi précise que possible: le composant ou la couche logicielle concernée.

I.2.2. L'exploitation

Elle recouvre les actions prises en vue de fournir, modifier et supprimer la mise à disposition d'équipements de services. Elle représente la phase opérationnelle pendant laquelle les décisions doivent être prises en temps réel. Elle comprend également la fourniture d'information des ressources consommées à des fins de facturation

I.2.3. La supervision

Elle se décline à travers les fonctions suivantes:

- Collecter les informations traduisant le comportement des différentes ressources
- Analyser les informations pour envisager et définir des actions.
- Stocker et archiver les informations.
- Réagir soit en émettant des commandes (pilotage) soit en informant l'administrateur.

I.2.4. La planification

Elle consiste à définir la topologie du système et à dimensionner chaque ressource du système de sorte que chaque usager puisse obtenir une communication dans des conditions optimales de qualité et de prix; il faut accomplir un certain nombre de tâches:

- Mesurer et prévoir la demande du trafic global entrant et sortant.
- Choisir pour chaque ressource une architecture adaptée.
- Optimiser les investissements en tenant compte des évolutions.
- Optimiser l'utilisation du système.

I.2.5. La sécurité

Elle constitue un domaine à part entière, cinq groupes de propriétés peuvent être distingués, à savoir: la disponibilité, l'intégrité, la confidentialité, l'auditabilité, et l'existence de preuve (non répudiation). Il existe plusieurs mesures pour éviter les sinistres: le contrôle d'accès (authentification et gestion des autorisations), le chiffrement, le scellement et la signature, etc.

I.3. Les modèles d'administration

Les buts de la gestion des réseaux sont:

- La gestion de tous les équipements du réseau quelque soit sa taille.
- La permission des opérations de gestion à distance.
- L'automatisation des tâches complexes de la gestion des réseaux.

Le cube d'administration de la figure (Fig I.1) montre les 3 domaines qu'il faut prendre en compte pour une bonne administration des réseaux. Le domaine technique touche tous les objets à administrer tels que les éléments physiques et logiciels. Le domaine fonctionnel inclut les fonctions à gérer. Le domaine décisionnel montre les objectifs de cette administration.

Dans ce présent travail, nous allons intéresser par le domaine fonctionnel et particulièrement par la gestion de la configuration, performances, erreurs et coût.

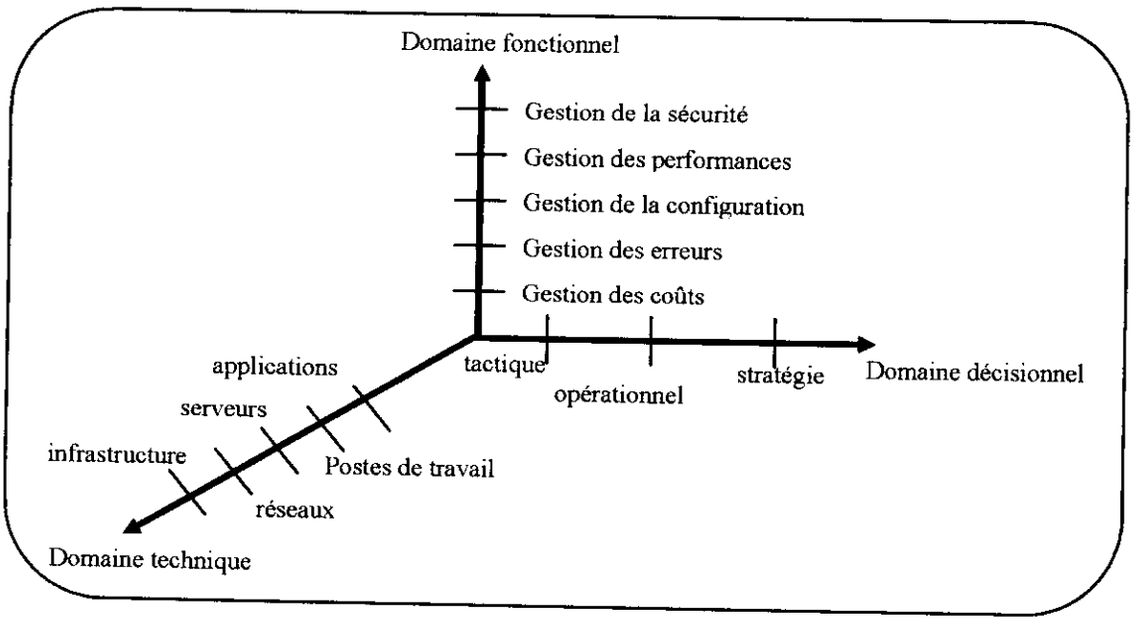


Fig.I.1 : Les dimensions de gestion de réseaux (cube d'administration)

I.3.1. Le modèle informationnel

Ce modèle fournit une vue à la fois unifiée et précise des ressources à gérer. Il structure l'information de gestion et présente un certain degré d'abstraction. Pour ce modèle l'ISO (ISO 10165) définit un modèle orienté objet par contre l'IETF (qui se limite à la gestion de réseaux TCP/IP) perçoit les objets à travers des variables stockées dans une base de données virtuelle (la base d'information de gestion (MIB)).

I.3.2. Le modèle organisationnel

Il décrit la nature distribuée de la gestion (norme ISO 10040) et comment le gestionnaire et l'agent échangent des informations de gestion à travers un protocole:

- Soit par une demande du gestionnaire et réponse de l'agent.
- Soit par un compte-rendu spontané de l'agent lors d'un événement.

I.3.3. Le modèle de communication

Il définit les protocoles qui permettent de recueillir les informations élémentaires et les statistiques auprès des agents représentant les ressources:

-A l'ISO, norme CMIP [Common Management Information Protocol].

-A l'IETF, norme SNMP [Simple Network Management Protocol].

I.3.4. Le modèle architectural

Il décrit la structure générale des entités réalisant les activités de gestion ainsi que leurs interfaces. La figure (Fig I.2) montre les relations entre les différents modèles.

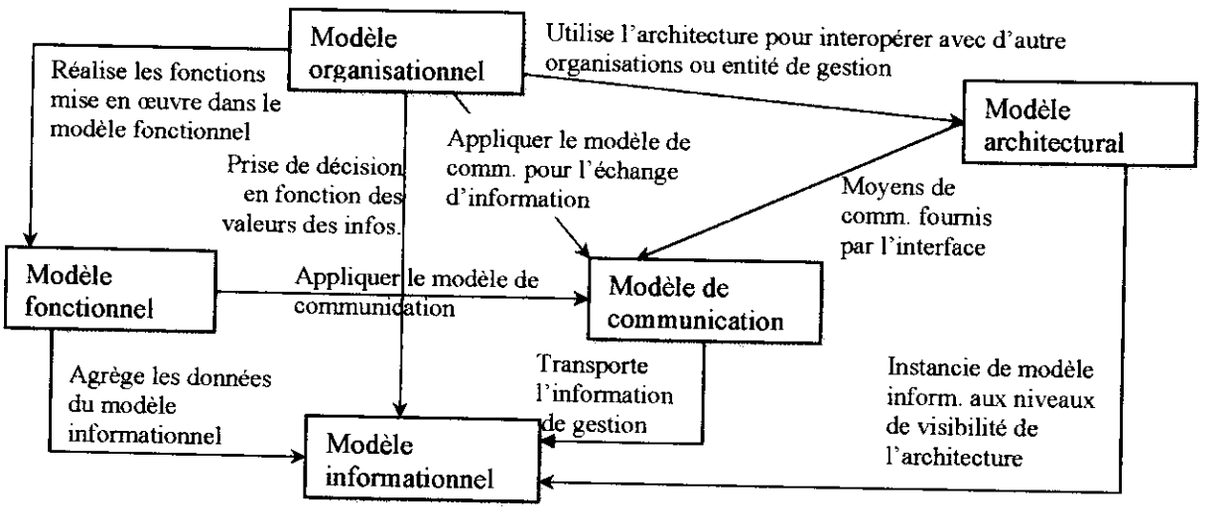


Fig.I.2 : Les relations entre les différents modèles

I.3.5. Le modèle fonctionnel

Il définit les opérations qui sont appliquées aux données du modèle informationnel. Ces opérations concernent les aires fonctionnelles de gestion (SMFA [Specific Management Functionnal Areas]):

a) La gestion des erreurs

Il s'agit de fournir des fonctions permettant de détecter puis d'isoler et enfin de résoudre des erreurs qui surviennent sur le réseau. Les anomalies empêchent le fonctionnement normal du système où elles se produisent de façon temporaire mais aussi de façon persistante. Les pannes sont détectées par un mécanisme de détection d'erreurs utilisant souvent les recherches dichotomiques ou des outils de tests. L'administrateur réseau est prévenu très rapidement par des alarmes.

Trois grandes caractéristiques de la gestion des erreurs :

- Fonction de surveillance : qui consiste à « monitorer » les pannes, c'est à dire prendre en compte les événements inhabituels, à créer et à examiner des journaux pour valider et confirmer les fautes.
- Fonction de localisation des pannes : qui va déterminer le ou les éléments causant ces troubles, ceci en tenant compte des effets de bords pouvant survenir.

- Fonction de détermination des pannes : qui s'effectue par l'analyse des journaux et par un choix d'actions curatives.

b) La gestion de la configuration

Elle manipule des données qui vont permettre de préparer l'initialisation et le lancement du fonctionnement des services d'interconnexion. Elle établit les paramètres qui contrôlent le comportement normal d'un système ouvert, elle diffuse les informations concernant l'état des systèmes, avertit d'éventuel changement d'état, etc. Elle permet de contrôler la configuration sur le plan fonctionnel et opérationnel. Elle permet à tout moment de dresser une topologie du réseau en établissant un inventaire des différents éléments qui le constitue.

Trois facettes peuvent être constatée:

- Une fonction d'installation : qui consiste à paramétrer un élément, à l'initialiser et à le mettre en service, ou à le supprimer. Cette fonction permet le contrôle et la mise à jour des différents paramètres, et tient à jour les différentes versions implémentées.
- Une fonction de contrôle et de surveillance : qui consiste à rester à l'affût de tout changement d'état.
- Une fonction de gestion des noms : qui est nécessaire pour permettre une localisation sans ambiguïté de tout élément du réseau.

c) La gestion de la sécurité

L'administration réseau consiste à assurer la confidentialité de l'accès aux données critiques de l'entreprise qui peut traiter des informations hautement sensibles comme des données bancaires ou militaires. Un encryptage des données est parfois nécessaire pour les véhiculer de façon plus sécurisée sur le réseau, l'administrateur devra contrôler ces codages.

Trois grandes fonctionnalités sont énoncées par l'OSI :

- Une gestion de la confidentialité, par l'utilisation de clé d'encryptage et de mécanismes d'authentification.
- L'audit pour surveiller les tentatives de connexion intempestives, ceci par la mise en place d'examens réguliers de journaux. Concernant les utilisateurs, il faut enregistrer et gérer aisément les droits de connexion de chacun.
- La gestion de sécurité offre aussi des fonctions qui permettent de créer, contrôler et supprimer des mécanismes et des services de sécurité, de rendre compte d'événements et de diffuser des informations pertinentes pour la sécurité.

d) La gestion des coûts

La part du budget dédiée au matériel informatique est importante dans une entreprise, or les dirigeants attendent un retour sur investissement. Comme certains éléments constituant un

réseau sont encore assez onéreux, le responsable informatique doit pouvoir justifier du montant des budgets qui lui sont alloués, pour cela il a besoin d'une gestion d'informations comptables.

D'autre part, il faut pouvoir évaluer le prix à facturer pour la prestation de services lors de la connexion au réseau. Une gestion des comptes est nécessaire pour connaître le montant fiduciaire à associer à chacun des utilisateurs. Une bonne gestion permet de déterminer qui utilise les ressources du réseau et dans quelle mesure afin d'imputer les coûts en conséquence.

Chaque machine d'un réseau étant clairement identifiée par une adresse, il est du rôle de l'administrateur de fournir à la direction des statistiques sur l'exploitation du réseau. Lors de l'utilisation d'une prestation, il faut déterminer les charges comptables et les limites de son utilisation pour envisager les mesures adéquates.

La gestion de la comptabilité passe donc par quatre fonctions qui sont :

- La charge des ressources, afin de fixer et surveiller la charge admissible des ressources.
- Le coût des ressources, pour fixer le prix d'utilisation des ressources.
- La facturation, pour récupérer les informations sur le coût imputable à un utilisateur donné.
- La gestion des limites des utilisateurs, dont le rôle est de déterminer et surveiller que les quotas d'utilisation des ressources des utilisateurs sont respectés.

e) La gestion des performances

Il est vital pour un responsable de réseau de connaître l'état de fonctionnement de celui-ci, c'est à dire de pouvoir, à tout moment, analyser et identifier les éléments causant une dégradation des performances. Car il faut pouvoir maintenir des temps de réponse sur le réseau à un niveau constant et acceptable pour l'utilisateur. Un segment de câble surchargé, un routeur mal configuré, sont des problèmes qu'un administrateur doit détecter avant l'apparition d'anomalies lors de l'utilisation du réseau.

La gestion des performances permet d'évaluer le comportement des ressources et l'efficacité des communications des données sur le réseau. Ainsi des informations statistiques, des journaux de divers états sont régulièrement mises à jour pour informer l'administrateur, elle permet donc de prévoir l'évolution du réseau et par conséquent de planifier les investissements nécessaires.

Pour cela il faut :

- Des fonctions de surveillance pour collecter des informations diverses sur les fautes et pour élaborer des statistiques sur le trafic. La gestion du trafic permet de réguler les données transitant sur le réseau de façon uniforme et optimisée.

- Une fonction d'observation de la qualité de service évaluera les paramètres de configuration, et leur incidence sur le fonctionnement du système. On pourra constater de cette manière si le temps de réponse est satisfaisant, ou la qualité d'une connexion à un serveur etc.

II. LA MATIERE PREMIERE UTILISEE DANS LES OUTILS D'ADMINISTRATION

On peut distinguer trois familles d'outils d'administration de réseaux :
Les commandes, les programmes de service et les logiciels d'administration de réseaux et systèmes.

Nous allons étudier quelques commandes utilisées dans l'administration de réseaux de tout type, notamment *ping*, *netstat*, *traceroute*. Ces commandes sont présentes sur de nombreux systèmes informatiques, y compris les postes de travail (le nom peut varier selon la plate-forme).

II.1. Description du protocole ICMP

ICMP [Internet Control Message Protocole] fait partie de la couche IP. Ce n'est pas un protocole d'administration à proprement parler mais certains outils d'administration l'utilisent. Il communique les messages d'erreurs et les autres évènements qui réclament de l'attention. Le format des messages ICMP est le suivant :

En-tête IP	**Message ICMP**		
20 octets	1 oct. (type)	1 oct. (code)	2 oct. (CRC)

II.2. Applications du protocole ICMP

II.2.1. Le programme PING

Ping s'emploie dans le monde UNIX, monde Microsoft, MacOS, Netware, OS/400, OS/390, etc. C'est le premier outil pour déterminer si une machine est accessible par le réseau. Ce programme donne également le temps d'aller-retour entre le système source et le système destination. Il envoie un message de requête ICMP echo en direction de la machine spécifiée (echo-request : type=8, code=0) et si la machine destination est connectée sur le réseau, il retourne un message ICMP réponse echo (echo-reply : type=0, code=0). Le pourcentage de paquets perdus et le temps d'aller-retour permettent aux administrateurs réseaux une analyse rapide des performances du réseau à un instant donné. Il permet également de connaître la route utilisée par les paquets IP. L'inconvénient est que la liste maximum de routeurs traversés ne peut excéder 9 (aller-retour) car l'en-tête IP a une taille

maximum de 60 octets. Un routeur étant multi-interface, il enregistre l'adresse IP de son interface de sortie (rfc 791). Les adresses IP indiquées permettent aux administrateurs de vérifier que les paquets utilisent une route cohérente.

II.2.2. Le programme Traceroute

Traceroute s'emploie dans le monde UNIX, OS/400, OS/390, etc. Elle existe aussi dans le monde Microsoft sous le nom de *tracert*. Cette commande affiche le chemin par lequel passe un appel au *host* distant spécifié. Traceroute n'est pas limité en nombre de routeurs traversés. Il utilise le protocole ICMP et le champ TTL de l'en-tête IP. Elle aide à localiser un problème logiciel ou matériel sur le réseau, par exemple à déterminer si un système intermédiaire ne transmet pas les messages correctement.

II.2.3. Netstat

Netstat s'emploie dans le monde UNIX, monde Microsoft, OS/400, OS/390, etc. Cette commande affiche diverses informations liées au *routing*. La syntaxe de cette commande varie d'une plate-forme à l'autre. La commande *netstat* permet d'obtenir des informations sur la connexion TCP-IP du *localhost*, ses tables de routage (*gateway tables*), ainsi que les *sockets* et les liaisons qu'il utilise.

II.2.4. Le programme telnet

TELNET n'utilise pas le protocole ICMP (il utilise TCP) mais c'est l'outil le plus utilisé pour déterminer les paramètres sur une machine via le réseau. Il permet de se connecter à distance et de simuler un écran de terminal. Datant de 1969, il est "implémenté" avec TCP/IP sur la plupart des systèmes et utilise une négociation d'options entre le client et le serveur pour déterminer les fonctionnalités fournies à chaque extrémité. Une fois connecté, il est possible de taper des commandes et de visualiser le résultat comme sur un écran terminal. Une connexion par telnet permet aux administrateurs de configurer et vérifier les paramètres d'un équipement. Sur un routeur, un hub, un commutateur, il aide au diagnostique d'un problème.

Telnet sécurisé (ssh) : est un outil qui remplace telnet, sécurisé par une procédure d'authentification basée sur des algorithmes de cryptage à la connexion, puis un cryptage durant la session. L'intérêt de telle procédure est de lutter contre l'écoute pirate des mots de passe circulant en clair sur nos réseaux actuels. Par souci de sécurité, des sites interdisent de plus en plus les connexions par telnet et n'autorisent que les connexions par ssh.

Chapitre II

Les protocoles d'administration des réseaux

Dans ce chapitre :

- Le protocole SNMP
- Le protocole CMIP
- Comparaison entre SNMP et CMIP
- Le protocole CMOT

CHAPITRE II : LES PROTOCOLES DE GESTION DES RESEAUX

I. LE PROTOCOLE SNMP

I.1. Historique

Issu des milieux de la recherche et de la défense américaine (DOD). L'activité de recherche et de standardisation au niveau de SNMP est menée sous la égide de l'IETF (Internet Engineering Task Force). La première version de SNMP a vu le jour en 1989. La seconde version, appelée SNMP v2 a été standardisée en 1992. Cette version apporte essentiellement des changements au niveau de la sécurité et de la confidentialité des opérations de gestion. Suite à des problèmes apparus lors de la mise en opération de la seconde version du protocole, ces aspects ont été classés obsolètes en 1996 lors d'un des meeting tri-annuel de l'IETF. Ce dernier rebondissement a eu pour effet, d'une part de renvoyer SNMP à son état de 1989 et d'autre part, de déclencher les recherches sur SNMP v3. Les travaux se poursuivent actuellement et petit à petit les différents éléments de l'architecture SNMP v3 sont proposés à la communauté scientifique. SNMP v3 (dans les aspects qui sont actuellement définis) est en passe de devenir un standard car il est accepté par le working group SNMP v3 en tant que standard proposé.

I.2. Définition

Le protocole SNMP (Simple Network Management Protocol) est un protocole d'administration de réseau issu du monde TCP/IP. C'est un protocole de couche applicative, il utilise l'UDP (User Datagramm Protocol) pour transporter les données entre les agents et le manager en mode non connecté, ce qui peut poser des problèmes de sécurité des échanges (aucune garantie que les paquets arrivent à bon port). Le choix de UDP contre TCP est justifié par la simplicité et la faible taille du code. Il existe par ailleurs des solutions SNMP non routables, c'est à dire sans les couches intermédiaires 3 et 4. Ces solutions permettent de réduire l'encombrement des agents SNMP dans les équipements. Elles sont adoptées par certains constructeurs pour la gestion des cartes adaptateurs.

I.3. Services offerts

SNMP permet de superviser les performances d'un réseau. Il permet de localiser à distance toute anomalie de fonctionnement. Il offre la possibilité de contrôler et de commander n'importe quel équipement, c'est à dire de changer des variables, modifier des configurations, initialiser ou interrompre le fonctionnement d'un élément ... Tous ces services

permettent de collecter des informations pour établir des statistiques et de consigner chaque action dans l'historique du réseau.

I.4. L'architecture de SNMP

I.4.1. Les entités SNMP

L'architecture SNMP est composée de stations gérantes et gérées (architecture client/serveur) ainsi que d'une base d'information (MIB : Management Information Base). Le noeud central d'un réseau est la NMS (Network Management Station). Elle donne au manager une vue d'ensemble de tous les équipements du réseau. Elle permet de centraliser les alertes et d'interroger chaque agent. Son interface graphique lui permet de visualiser chaque équipement sous la forme d'une icône spécifique dont la couleur représente l'état. Elle permet une représentation plus ou moins en profondeur de chaque équipement. Le manager contient un noyau SNMP dont le rôle est d'interpréter la MIB et d'associer des icônes aux objets gérés.

La MIB est une base de données 'virtuelle' contenant les caractéristiques de tous les équipements reliés au réseau. SNMP ne définit pas ses caractéristiques physiques mais son organisation logique (RFC 1066). Celle-ci est conforme à la structure d'information de gestion (SMI : Structure Management Information). Elle défini, grâce à un ensemble de règles, les caractéristiques des objets du réseau, la façon pour le protocole de gestion d'obtenir des informations sur ces objets, ainsi que la manière d'ajouter de nouveaux objets à la MIB. (Voir Fig II.1)

I.4.2. Principes de base

Le client étant le manager et le serveur l'agent. Contrairement aux architectures client/serveur habituelles ou un serveur interagit avec plusieurs clients, SNMP est l'exemple d'un schéma inverse: un client, souvent unique, communique avec plusieurs serveurs. Il y a un agent par équipement à gérer. L'agent conserve une série de variables qui modélisent l'équipement. Le manager, au travers de l'agent, consulte et/ou modifie les valeurs des variables à l'aide d'opérations de management.

Des changements d'état internes à l'équipement modifient eux aussi les valeurs des variables. La manière dont l'agent doit être informé de ces changements n'est pas spécifiée dans SNMP. Le manager interagit avec l'agent via un protocole de communication qui spécifie la dynamique de la communication (mécanismes de question/réponse), ainsi que la structure des messages échangés. En standardisant le protocole, le comportement de l'agent pour les opérations de management et la manière de décrire la MIB, on aboutit à SNMP.

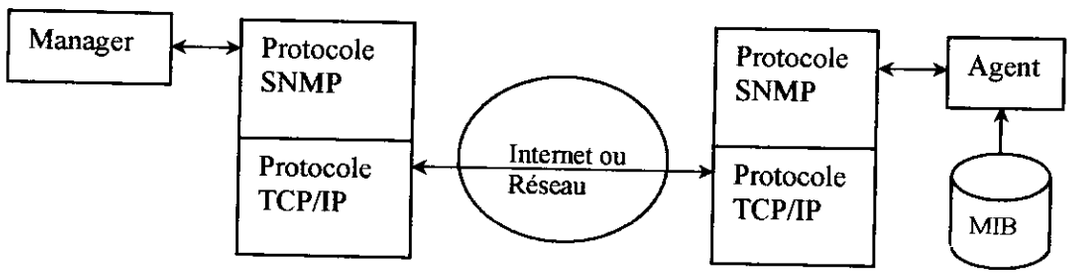


Fig.II.1 : Différentes entités de l'architecture SNMP

I.4.3. La base d'information de gestion (MIB)

La base d'information de gestion contient les variables qui forment le modèle informationnel de l'équipement. Une MIB (Management Information Base) est une collection de tous les objets que maintient un agent donné. Le langage de description ASN1 (Abstract Syntax Notation One), normalisé au niveau 6 du modèle OSI, est utilisé pour décrire les formats de données des objets. Chaque objet est décrit par des attributs pouvant prendre certaines valeurs.

La MIB est une structure arborescente très ramifiée, dont chaque sous arbre représente une organisation, depuis l'ISO et le CCITT jusqu'au plus petit constructeur souhaitant disposer d'informations propres aux équipements qu'il fabrique (MIB privées). La branche MANAGEMENT (sous arbre de INTERNET) contient les objets 'standards' dans deux sous-branches : MIB1 pour les objets nécessaires à la gestion d'un réseau TCP/IP et MIB2 pour des objets de description plus fine des fonctionnalités d'un équipement. La notation utilisée est préfixée. Par exemple pour accéder au groupe IP de la MIB1 on écrira :

ISO.ORG.DOD.INTERNET.MANAGEMENT.MIB1.IP qui est identifié par : 1.3.6.1.2.1.4 (voir Fig II.2).

Le compilateur de MIB permet à partir de la description textuelle d'un équipement, d'obtenir une table interne interprétable par le manager. Le browser de MIB fournit une aide par menu pour interroger la MIB sur des requêtes ponctuelles.

Actuellement, la base la plus utilisée est la MIB-II car elle est plus riche que la première version et contient des éléments propres à la gestion SNMP.

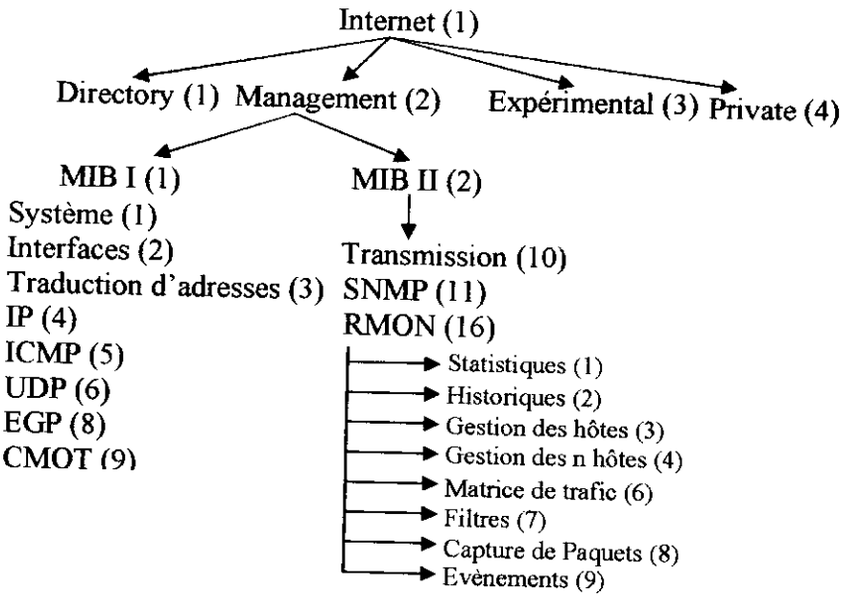


Fig.II.2 : La structure arborescente de MIB (Management Information Tree)

Nous allons décrire la MIB-II, dont les spécifications sont dans la RFC 1213. Elle décompose les informations d'administration des réseaux en 11 (onze) catégories standard illustrées dans le tableau II.1. Elle ajoute de nouveaux groupes par rapport à la MIB-I (Le groupe SNMP et CMOT). [Biz_97]

Catégorie MIB	**Permet d'avoir les informations sur :**	**Nb objets**
System	Le système d'exploitation de la machine ou du routeur	7
Interfaces	Chaque interface réseau	23
AT (translation)	La traduction d'adresse (ex. correspondance ARP)	3
Ip	Statistique sur le protocole IP	38
Icmp	Statistique sur le protocole ICMP	26
Tcp	Statistique sur le protocole TCP	19
Udp	Statistique sur le protocole UDP	7
Egp	Statistique sur le protocole EGP	18
Transmission	Ce groupe est prévu pour « raccrocher » d'autres modules de MIB qui concernent des médias de transmission plus spécifiques qui viennent compléter les informations contenues dans le groupe interface	0
Cmot	Compteurs pour CMOT (protocole OSI équivalent à SNMP)	0
SNMP	Statistique sur le protocole SNMP	30

Tab II.1 : Catégories d'information MIB-II

I.4.4. Structure des informations d'administration (SMI)

En plus du standard MIB, qui définit les informations spécifiques d'administration et leur signification, Il faut aussi un autre standard qui spécifie l'ensemble des règles utilisées pour définir et identifier les variables MIB. Ce sont les règles de gestion des informations d'administration, SMI (Structure of Management Information).

Une SMI est un ensemble de règles établies afin de définir et d'identifier des variables. On peut la considérer comme une couche au-dessus de ASN-1 dans la représentation des données. Cette structure correspond à une syntaxe utilisée pour la description d'un objet. Pour lequel elle détermine 3 attributs [Cha_99] : Identifiant, Syntaxe du type et Codage.

I.4.5. L'ASN-1 (Abstract Syntax Notation One)

ASN-1 est un langage formel qui présente deux caractéristiques principales : Une notation utilisée dans les documents manipulés par script et une représentation codée de la même information, utilisée dans les protocoles de communication. Dans les deux cas la notation formelle précise, élimine toutes les ambiguïtés possibles, tant du point de vue de la représentation que de la signification. Au lieu de dire, par exemple, qu'une variable contient une valeur entière, un concepteur qui utilise ASN-1 doit définir la forme exacte et le domaine des valeurs prises par cet entier. Une telle précision est nécessaire dans le cas de mises en oeuvre qui impliquent des machines hétérogènes n'utilisant pas toutes la même représentation des éléments de données. [Cha_99]

I.4.6. La MIB RMON

C'est une MIB spéciale qui contient les objets nécessaires à la télé administration via SNMP d'un équipement de mesure. Cette MIB est tout d'abord un standard qui émerge dans le monde SNMP. Elle doit fournir l'interopérabilité entre les équipements de mesure et les stations d'administration (les managers), permettant ainsi aux utilisateurs la coopération de différents constructeurs. Elle gère pour l'instant le niveau 2 (liaison) de l'OSI [Del_96]. Elle est largement adoptée pour la collecte de statistiques sur le comportement de réseaux distants. RMON est un sous-ensemble de SNMP qui est destiné à gérer les équipements d'extrémités et les connexions. Étant contenu dans SNMP, RMON a bénéficié d'un statut privilégié par rapport à tout autre mécanisme de supervision et a été immédiatement reconnu comme une norme internationale.

RMON comprend deux éléments ; un agent, qui est une sonde distante, attachée à chaque segment à contrôler et un client qui fournit l'interface de supervision.

I.4.7. Le Protocole SNMP

SNMP est un protocole bâti au-dessus de UDP/IP la figure suivante montre le positionnement de l'SNMP dans la pile du protocole IP :

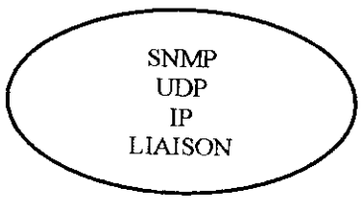


Fig.II.3 : SNMP et la pile de protocole IP

Les fonctionnalités SNMP

SNMP a choisi une approche originale de l'administration des réseaux car au lieu de définir un grand nombre de commande, SNMP ne définit que 5 types de messages entre manager et agent. Les trois premiers messages sont expédiés du manager vers l'agent et les deux derniers de l'agent vers le manager, comme le montre la figure ci-dessous.

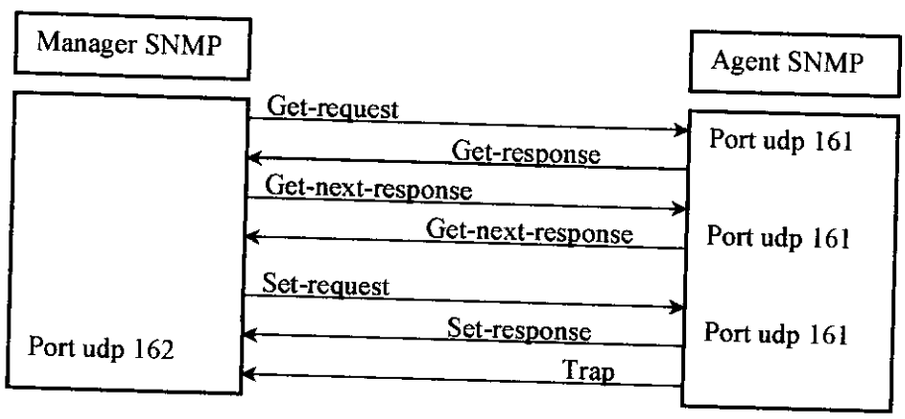


Fig.II.4 : Les messages SNMP

Get-request : permet à la station d'administration d'interroger un agent afin d'obtenir la valeur d'une ou plusieurs variables.

Get_next-request : permet la lecture d'une valeur suivant une ou plusieurs autres variables d'un agent sans en connaître le nom. Elle est utilisée pour récupérer des valeurs d'instances dont on ne connaît pas l'index à priori. C'est par exemple le cas des valeurs d'objets contenus dans les tableaux. Une commande GetNext sur un objet de la MIB permet d'obtenir la valeur de l'instance suivante qui le suit dans l'ordre lexicographique des OID : Cela permet de faire un parcours récursif des objets instances dans la MIB (opération appelée« walk »).

Set-request : permet de modifier la valeur d'une ou plusieurs variables d'un agent.

Get-response : Renvoie la valeur d'une ou plusieurs variables; il s'agit d'un message retourné par l'agent au manager en réponse aux opérations get-request, get-next-request et set-request.

Trap : il est aussi possible pour l'agent d'envoyer un trap au manager, pour lui indiquer que quelque chose vient de se produire sur l'agent, qui doit être porté à sa connaissance. Les traps sont envoyés au port UDP 162 sur le manager.

Format d'un message SNMP

Le message SNMP se compose de deux parties distinctes comme le montre la figure suivante : « Les messages SNMPv1 et v2 ont le même format »

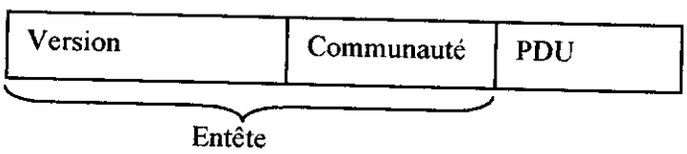


Fig.II.5 : Format du message SNMP v1 et v2

- Un identificateur de version : nom de version SNMP vaut « 0 » pour SNMPv1 et « 1 » pour SNMPv2. Le manager et les agents doivent utiliser la même version.
- Un nom de communauté : désigne le mot de passe (en clair sous forme de chaîne de caractères) entre le manager et l'agent. La valeur standard est la chaîne de 6 caractères « public ».
- Une PDU (Protocol Data Unit) : il existe 5 types de PDUs

0: Get-request, 1: Get-next-request, 2: Set-request, 3: Get-response, 4: Trap

Et voici un premier format qui est utilisé pour les PDU du genre GET, ou SET :

Type PDU	ID de requête	Statut d'erreur	Index d'erreur	Obj1.val1

Fig.II.6 : Format du PDU du genre GET, ou SET

L'identificateur de requête (ID) : Il est défini par le manager et retourné par l'agent dans le message get-response. Ceci permet au client (le manager) de faire le rapprochement entre la réponse du serveur (l'agent) et sa propre requête. Ceci permet en outre au manager d'émettre plusieurs requêtes vers un ou plusieurs agents, puis de trier les réponses.

Statut d'erreur : Le code de statut d'erreur est retourné par l'agent qui identifie l'erreur.

Index erreur : Si une erreur se produit, l'index d'erreur et l'offset entier spécifient quelle variable est en erreur. Il est défini par l'agent uniquement dans le cas des erreurs noSuchName, badvalue et readonly.

Obj/Val : Une liste des noms de variables et leur valeur suit les requêtes get et set.

L'encapsulation du message SNMP

Dans ce schéma ci-dessous, on décrit la forme d'un message SNMP avec une en-tête UDP et IP. On ne précise pas la taille en octets que pour les en-têtes IP et UDP, car le codage utilisé pour les messages SNMP (ASN-I) varie en fonction du type de la variable et de sa valeur.

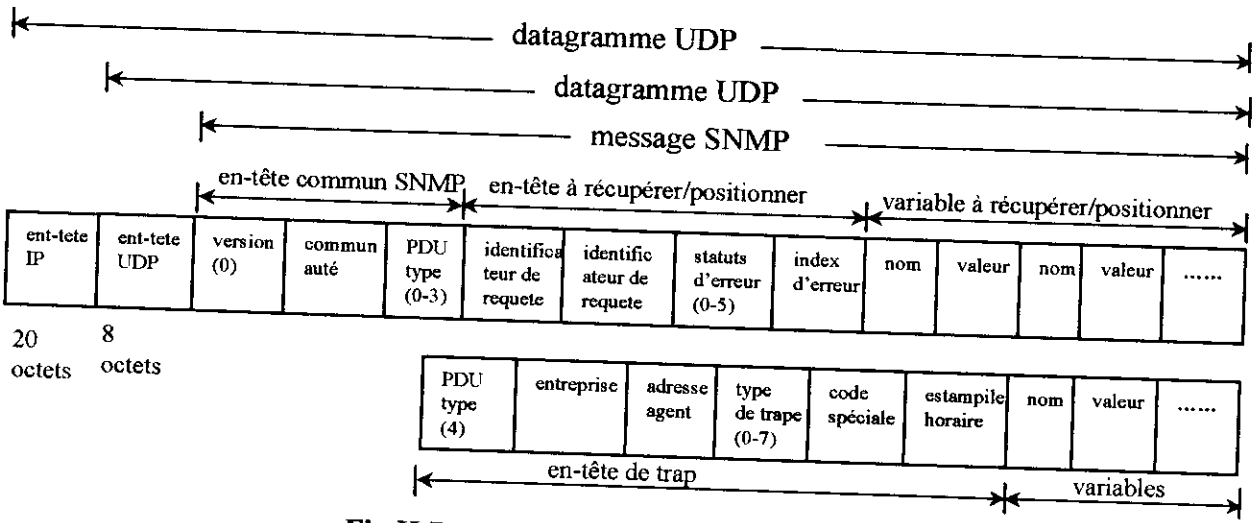


Fig.II.7 : Encapsulation des messages SNMP

Les primitives sont codées en ASN-I avec le code BER (Basic Encoding Rules). Le format des PDUs (Protocol Data Unit) SNMP suit globalement un même schéma, sauf pour les Traps SNMP.

I.4.8. Les agents SNMP

Les agents SNMP sont placés dans les équipements compatibles SNMP. Ils fournissent des informations à la partie de la MIB où se trouve l'objet qu'ils représentent. Deux cas particuliers existent :

L'agent Rmon : qui est situé dans une sonde et se comporte comme un analyseur de protocole local et peut déclencher des alertes de performances.

Le Proxy-agent : qui sert d'intermédiaire entre un ou plusieurs autres agents et le manager. Il est utilisé soit pour une conversion de protocole (SNMP <=>protocole propriétaire), soit pour regrouper les informations provenant d'un réseau local distant. Il ne fait remonter vers le manager que les conditions anormales et permet ainsi de limiter le trafic du réseau d'interconnexion, souvent à bas débit.

I.5. Les limitations de SNMP v1

Il y a quatre grandes faiblesses dans le protocole SNMP v1 :

- Tout d'abord, c'est un protocole trop gourmand.
- Il utilise uniquement le protocole UDP/IP.
- Il a une sécurité faible.
- Et finalement, il ne permet pas la communication de gestionnaire à gestionnaire.

Pour cela, deux autres versions ont vu le jour, et qui donnent des solutions à quelques problèmes de la première version.

I.6. Les évolutions

La sécurité dans le protocole SNMPv1 est basée sur deux mécanismes, l'authentification et les autorisations en utilisant le nom de communauté : [Biz_97]

a) Authentification : Se fait de façon très simple. Le nom de la communauté est placé en clair dans le message SNMP. Si le nom de la communauté correspond à une communauté définie au niveau de l'agent, l'entité SNMP émise est considérée comme étant authentifiée comme un membre de la communauté.

b) Autorisation : Une fois que l'entité SNMP émise est authentifiée comme un membre de la communauté, le client (noeud sur lequel se trouve l'agent) doit déterminer à quel niveau d'accès peut se placer la requête SNMP. A une communauté est associé des droits d'accès sur un sous-ensemble des variables de l'agent : "*Read-Only*" ou "*Read-Write*"

Chaque variable de l'agent a l'un des caractéristiques : "*Read-Only*" , "*Read-Write*" , "*Write-Only*" , "*Not-accessible*" .

I.6.1. SMP ou SNMPv2

SNMP deuxième version a été proposée en juillet 1992 comme une mise à jour de SNMP. Il reste simple et conserve la petite taille et l'implémentation rapide de SNMP, mais corrige ses principaux défauts. Il est prévu pour fonctionner aussi bien avec l'architecture TCP/IP, qu'avec l'architecture OSI. Il incorpore toutes les procédures de sécurité qui avaient été développées pour une version intermédiaire nommée SNMP Secure. Il utilise notamment le chiffrement (algorithme DES), l'authentification par vérification du délai de transfert (procédé MD5) et la vérification du contrôle d'accès. Il possède de plus un 'pseudo' transfert de masse ('bulk-transfert') qui augmente l'efficacité de transfert de données de la MIB. SNMP v2 différait de SNMP v1 selon deux critères:

1. Le support pour sécuriser (authentification et confidentialité) les opérations a été renforcé. SNMP v2 offre la notion de groupe (*party*) défini comme étant un contexte d'exécution des opérations de gestion. Ce contexte définit, entre autres, l'ensemble des objets qui peuvent être gérés, le protocole de transport utilisé (ex: UDP) et la méthode d'authentification utilisée pour reconnaître la légitimité d'une requête.

2. L'ajout d'une opération GET-BULK qui permet de rapatrier vers le manager des grandes quantités de données en les répartissant sur plusieurs messages de réponse.

- **Autres nouveautés apportées par SNMPv2**

La 2eme version de SNMP apporte surtout une amélioration pour la sécurité des messages par rapport à SNMPv1. Les différences majeures entre les deux versions sont [Del_96] :

1. De nouveaux types de PDUs sont ajoutés pour rendre la communication entre le manager et les agents plus efficaces et permettre la communication entre managers.
2. Deux nouvelles MIB sont définies: la MIB SNMPv2 et la MIB SNMPv2-M2M (Manager To Manager).
3. Avec SNMPv2 le domaine de transport ne se limite pas au modèle TCP/IP mais sur d'autres modèles comme OSI et IPX/SPX de Novell (Internet work Paquet eXchange).
4. SNMPv2 fournit des améliorations pour la sécurité par rapport à SNMPv1.Avec SNMPv1, le nom de communauté passé du manager à l'agent est un mot de passe en clair. SNMPv2 gère l'authentification et la confidentialité.
5. SNMPv2 a définit un nouveau mode d'administration autre que la communauté.

- **Cohabitation SNMP et SNMPv2**

L'une des fonctions les plus importantes est la coexistence entre l'ancien SNMP et SNMPv2. Les différents éléments du réseau qui répondent aujourd'hui à SNMP, ne sont pas obligés de migrer d'un seul coup vers le nouveau protocole SNMPv2. Alors que dans les équipements qui ont SNMPv2 implémenté par le constructeur, le fonctionnement en parallèle des deux protocoles a été prévu et une technique de Proxy Agent a été élaborée pour cette cohabitation.

La fonction de PROXY au niveau du gestionnaire permet de passer d'un protocole (SNMP) à un autre (SNMPv2). Le Proxy fait la traduction d'un monde vers l'autre et le gestionnaire peut ainsi réagir en gestionnaire SNMP ou SNMPv2. De plus, la MIB-II continue à être utilisée par l'ancien protocole SNMP. Elle est aussi prise en compte par SNMPv2 qui l'a enrichie de toute une nouvelle partie de l'arbre de l'Internet : OID (1.3.6.1.6).

I.6. 2. SNMP Version 3 (SNMPv3)

La troisième version de SNMP (définie dans les RFC2570 et 2574), introduit de nouveaux mécanismes de sécurité (forte authentification et confidentialité). Cette version est actuellement en chantier et consiste en une refonte complète de l'architecture existante. Tous les aspects n'ont pas encore été définis. Par contre, on connaît déjà le schéma général de l'architecture. Par rapport à SNMP v1 et v2, on trouvera des communications entre managers.

Par ailleurs, des agents intermédiaires (des Proxy) permettront de faire apparaître un ensemble d'agents comme un seul agent. Le Proxy pourra faire du filtrage sur les informations qui vont vers le manager afin d'alléger la tâche de ce dernier. Finalement, il sera possible de créer pour chaque manager un profil comprenant l'ensemble des objets gérés et les opérations possibles (on retrouve le principe de *party* présent dans SNMP v2).

La troisième version de SNMP se révèle complète et sécurisée, elle recourt à un système de chiffrement à clé privée USM (User-based Security Model) pour sécuriser la transmission des messages sur le réseau. Le sou système de sécurité met plusieurs modèles à disposition. Les requêtes sont reparties vers les générateurs et receveurs de notification et d'applications et le générateur de commandes. En plus elle interagit avec les anciennes versions du protocole.

Le standard SNMPv3 donne un cadre précis d'administration et suffisamment d'ouvertures pour permettre d'intégrer les extensions nécessaires, comme par exemple de nouveaux modèles de sécurité.

II. LE PROTOCOLE CMIP

De la même manière que SNMP, CMIP permet de réaliser l'administration d'un réseau en se basant sur une communication entre agents et manager. Ils utilisent tous les deux le concept de la MIB programmée en ASN1. Cependant, CMIP ne se base pas sur le modèle TCP/IP mais sur le modèle OSI. Il fonctionne en mode connecté, ce qui permet de sécuriser les échanges : chaque transmission demande un accusé de réception, correspondant à une garantie de délivrance.

Cependant, en cas de problèmes dans le réseau, CMIP peut être pénalisant, car le maintien d'une connexion peut représenter une surcharge supplémentaire de la bande passante et alourdir le trafic, déjà sujet à des difficultés. CMIP présente donc une grande fiabilité mais celle-ci entraîne une diminution de la rapidité du protocole. Il est peu utilisé, seuls certains opérateurs de grands réseaux (ex: les opérateurs telecom) l'ont adopté.

III. COMPARAISON ENTRE SNMP ET CMIP

On résume cette comparaison dans les points suivants :

- Alors que SNMP travaillait principalement par polling, c'est à dire par interrogations régulières de chaque agent, CMIP utilise le 'Reporting' : les stations gérées remontent vers la station gérante uniquement les alarmes et événements graves. Il n'y a pas d'interrogation par la station gérante, d'où un allégement du trafic. Le Reporting est particulièrement adapté aux

réseaux distants. Il nécessite cependant plus d'intelligence dans les stations gérées (la taille du noyau CMIP dans les stations est de 240ko contre 7 ou 8ko pour un noyau SNMP).

- Contrairement à SNMP qui est plutôt orienté vers une extraction individuelle de champs d'information, CMIP est orienté vers l'extraction collective d'attributs. Une seule requête permet alors la remontée d'une table entière d'attributs.
- CMIP offre une très grande sécurisation (droits d'accès ...), alors que le mécanisme de sécurité était une des faiblesses de la première version de SNMP. CMIP est donc un outil plus complet que SNMP mais aussi beaucoup plus lourd à installer et à utiliser.
- La grande faiblesse de SNMP est le manque de service, car il n'y en a que cinq disponible, alors que pour CMIP/CMIS il y en a dix. Mais cet avantage implique aussi que CMIP/CMIS est nettement plus complexe que SNMP.

IV. LE PROTOCOLE CMOT

Ce protocole de gestion est à l'origine une migration de SNMP vers la standardisation de l'OSI. CMOT vient de CMIS over TCP/IP. Il existe un groupe de travail l'OSI Internet Management (OIM) qui essaye de produire des spécifications pour que les services de CMIS et le protocole CMIP soient implémentables sur TCP/IP. Comme SNMP était un protocole de secoure pour permettre d'attendre la version de l'OSI, la solution initialement envisagée, pour permettre une transition plus facile, était d'utiliser la même base d'objets par SNMP et CMOT. Il est rapidement apparu que cette contrainte était non réaliste car en SNMP on manipule essentiellement des variables et en CMIP on manipule des objets dans le sens de la technologie orientée objet.

L'architecture CMOT consiste à ajouter au-dessus des protocoles TCP ou UDP, une couche présentation simplifiée (LPP Lightweight Presentation Protocol), les services d'application ACSE (Association Control Service Entity), ROSE (Remote Operation Service Entity) et bien sur CMIP/CMIS. On voit bien l'incompatibilité avec SNMP, car ACSE utilise un mode connecté.

Les constructeurs souhaitent que tous les protocoles (CMIP ou SNMP) puissent émerger d'une station à une autre, c'est à dire d'une architecture à une autre (de OSI à TCP/IP et de TCP/IP à OSI). Ce qui fait qu'il y a eu une tentative de regroupement des deux protocoles avec le protocole CMOT (CMIP Over TCP/IP), de manière à regrouper les deux approches. Mais aujourd'hui, le protocole CMOT n'a pas un grand succès.

CONCLUSION

Le succès de SNMP de part sa simplicité d'intégration, de mise en place et d'utilisation, le consacre comme un standard dans le domaine de l'administration des réseaux d'entreprise, d'autant plus que la solution de l'OSI (CMIP) est lourde à mettre en place.

L'utilisation de SNMP s'est étendue au-delà du réseau, dans le monde du système et des applications. La version 2 de SNMP apporte de très nettes améliorations par rapport à la première version tout en essayant de rester « simple ». Cette version permet une plus grande sécurisation des données et augmente la facilité et la rapidité d'accès à la MIB. La version 2 étant compatible avec SNMP v1, l'essor de SNMP risque de s'accroître de façon plus importante. Les limitations des versions 1 et 2 sont comblées avec la version 3. Ce dernier standard demande toutefois plus de travail d'implémentation et de mise en oeuvre.

Après cette présentation des protocoles d'administration, Nous allons aborder, dans le chapitre suivant, les différents systèmes d'administration.

Partie II

La gestion par les agents mobiles

Chapitre III

Les systèmes d'administration des réseaux

Dans ce chapitre :

- Systèmes d'administration de réseau centralisés
- Systèmes d'administration hiérarchiques
- Systèmes d'administration fondés sur les technologies du Web

CHAPITRE III : LES SYSTEMES D'ADMINISTRATION

INTRODUCTION

Dans ce chapitre nous allons présenter les systèmes d'administration des réseaux en spécifiant la définition, les avantages et les inconvénients de chaque système. Un système d'administration de réseaux est constitué d'un ensemble d'outils qui permettent aux administrateurs de gérer le réseau. L'administrateur accomplit son travail en utilisant les applications d'administration qui sont dédiées aux tâches d'administration citées précédemment et qui offrent une interface graphique conviviale. S'exécutant sur les éléments du réseau, les agents d'administration qui sont pilotées par les applications d'administration à travers des protocoles d'administration de réseaux, ont pour rôle d'une part de collecter et transmettre les informations d'administration et d'autre part de contrôler (initialiser, modifier) les paramètres de configuration de l'élément de réseau considéré.

Dans les paragraphes qui vont suivre nous présentons les systèmes d'administration de réseaux constituant l'architecture traditionnelle d'un système d'administration de réseaux ainsi les systèmes d'administration hiérarchiques, puis nous décrivons les systèmes d'administration reposant sur les technologies du Web. Dans le quatrième chapitre nous évoquons les approches fondées sur l'utilisation du code mobile.

I. SYSTEMES D'ADMINISTRATION DES RESEAUX CENTRALISES

I.1. Définition

La figure (Fig III.1) présente l'architecture d'un système d'administration de réseaux organisé autour d'une plateforme d'administration centralisée. Dans un tel système, toutes les applications d'administration s'exécutent sur une machine unique, appelée station d'administration de réseaux (SAR). Outre les applications d'administration, un noyau d'administration s'exécute sur la SAR et offre à ces dernières des services de base communs pour le stockage des informations, l'administration dans une base de données et pour la communication avec les agents d'administration situés sur les éléments du réseau. La station d'administration de réseaux communique avec les agents d'administration par l'intermédiaire d'un protocole d'administration (SNMP, CMIP etc..) pour échanger les informations d'administration définies par les MIBs.

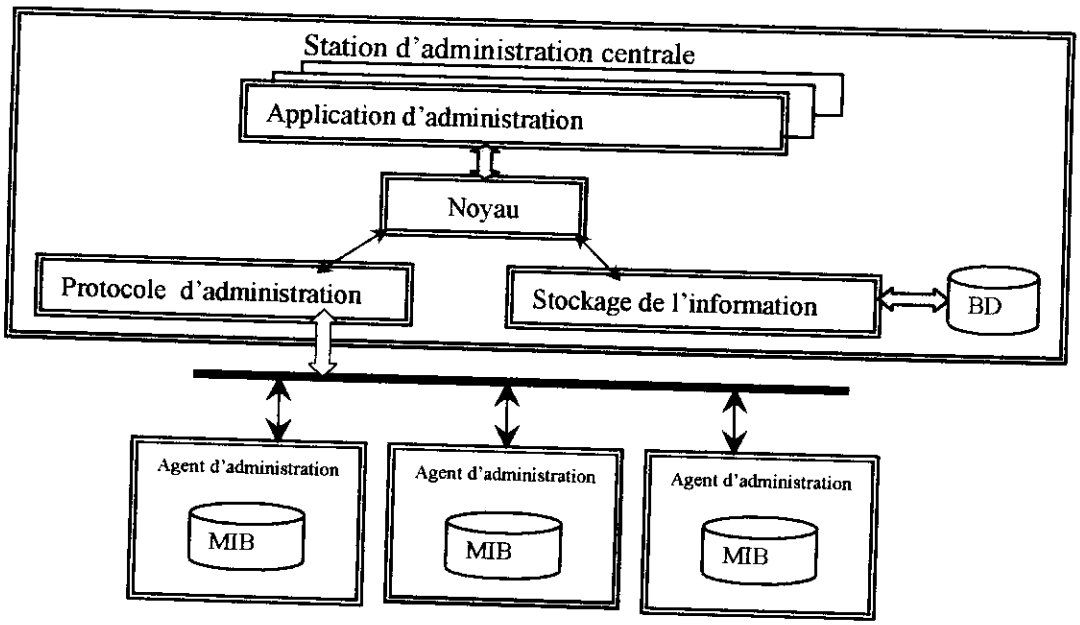


Fig III.1 : Un système d'administration centralisé de réseaux

I.2. Avantages et inconvénients des systèmes d'administration des réseaux centralisés

Un des principaux avantages des systèmes d'administration de réseaux centralisés est que l'administrateur peut visualiser et agir sur les éléments du réseau depuis un site unique. En contrepartie, il doit toujours se trouver à proximité du système pour effectuer ses tâches d'administration.

La centralisation des informations d'administration rend aisés les traitements nécessitant la corrélation de plusieurs données. Cependant, le système d'administration constitue un point unique de défaillance

Des mécanismes doivent être offerts afin de ne pas perdre les informations d'administration d'une part et d'assurer la continuité de service de la station d'administration d'autre part. La sauvegarde des informations d'administration est du ressort du système de gestion de base de données utilisé dans le système d'administration.

La continuité de service ne peut être assurée qu'au prix de redondance matérielle ou logicielle. [Akh_99].

Un autre inconvénient d'une architecture d'administration centralisée est qu'elle constitue un goulot d'étranglement pour les réseaux de grande taille. Le système concentre en effet toutes les informations et tous les traitements sur ces informations. Sa charge peut alors être très importante ainsi que celle de ses liens de communication, ce qui peut avoir des

répercussions sur la fréquence des requêtes envoyées aux agents d'administration. Par ailleurs, les systèmes d'administration de réseaux sont le plus souvent liés à une architecture matérielle et un système d'exploitation.

Enfin, les mises à jour d'un système d'administration de réseaux nécessitent de passer par le cycle de compilation et réinstallation. Pour faire face à ces inconvénients, des systèmes d'administration de réseaux distribués ont été conçus. Nous les décrivons ci-après.

II. SYSTEMES D'ADMINISTRATION HIERARCHIQUES

Les systèmes d'administration hiérarchiques sont une extension des systèmes fondés sur une plateforme d'administration (centralisée). Dans le cas de réseaux de grande dimension, ces systèmes permettent de répartir la collecte des informations d'administration sur plusieurs machines afin de décharger la plateforme d'administration. Un système d'administration hiérarchique est composé de plusieurs stations d'administration (Fig III.2) une station d'administration centrale et des stations d'administration subordonnées pilotées par la station d'administration centrale.

- La station d'administration centrale dispose de la base de données des informations d'administration et exécute les applications d'administration.
- Les stations d'administration subordonnées placées sous le contrôle de la station centrale ont pour rôle de communiquer avec les agents d'administration et de répondre aux requêtes de la station centrale [Akh_99].

Les systèmes d'administration de réseaux hiérarchiques ne constituent pas une solution au problème de tolérance aux fautes des plateformes d'administration puisque la station d'administration centrale reste un point unique de défaillance. De même la défaillance d'une station subordonnée peut entraîner la perte d'alarmes concernant les éléments des sous réseaux qu'elle gère.

La mise en place d'un système d'administration hiérarchique est complexe. Les stations d'administration subordonnées doivent être configurées ou reconfigurées manuellement dans le cas d'une évolution du réseau. Cette configuration doit être effectuée soigneusement de manière à éviter qu'un élément du réseau soit surveillé par plusieurs stations subordonnées, ce qui introduirait une consommation de bande passante inutile.

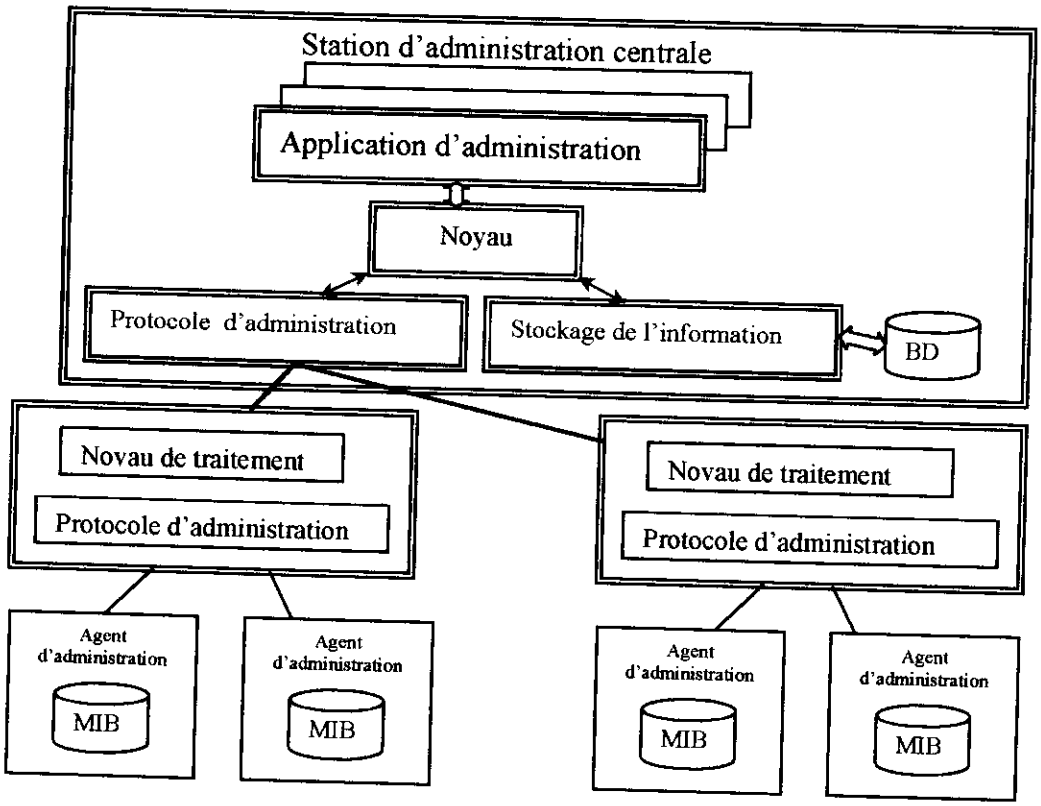


Fig.III.2 : Architecture d'un système d'administration hiérarchique

III. SYSTEMES D'ADMINISTRATION FONDES SUR LES TECHNOLOGIES DU WEB

Le World Wide Web se présente comme un nouveau paradigme d'accès et de présentation de l'information. Récemment les technologies du Web comme le langage HTML [Htm_98], le protocole HTTP [Htt_98] ou le langage Java ont été utilisées pour fournir aux applications d'administration une interface uniforme et indépendante du système. Les applications d'administration ont une interface utilisateur sur le Web et interagissent soit directement avec les agents d'administration (agents SNMP) sur les éléments du réseau soit à travers un serveur qui lui même interagit avec les agents d'administration. La première approche d'accès direct et la seconde d'accès indirect.

Accès direct : Des piles SNMP permettant un accès direct aux agents SNMP ont été développées dans le langage Java. Elles offrent une interface Java aux requêtes SNMP get, get_next et set. Le langage Java permet de disposer d'une interface uniforme indépendante du système. Cependant, ces piles communiquent directement avec les agents SNMP situés à distance sur des éléments du réseau. Les contraintes imposées aux navigateurs Web pour des

raisons de sécurité doivent alors être relâchées pour permettre ces communications. Une autre possibilité est d'utiliser ces piles SNMP au sein d'une application Java (sans faire appel au Web). Un inconvénient de ce type d'approche, lié à l'absence de serveur d'administration, est la duplication de requêtes qui peut mener à un important trafic sur le réseau.

L'accès indirect : L'architecture WebNMS [AdN_98] est représentative des approches à accès indirect. Elle est présentée dans la figure (Fig IV.2) Le système WebNMS se compose d'un serveur mis en ouvre dans le langage Java et de clients qui s'exécutent dans des navigateurs Web intégrant une machine virtuelle Java [Bou_01]. Le serveur est constitué d'un serveur Web qui permet la communication avec les clients Web. Les clients chargent des pages HTML depuis le serveur. Le serveur comprend également des servlets qui sont des petits programmes Java qui sont lancés par des requêtes des clients. Le serveur comprend en outre une base de données et un serveur d'applets SNMP. Ce dernier agit en tant que relais pour la transmission des données SNMP, ce qui permet aux navigateurs Web de communiquer à l'aide du protocole SNMP avec les éléments du réseau.

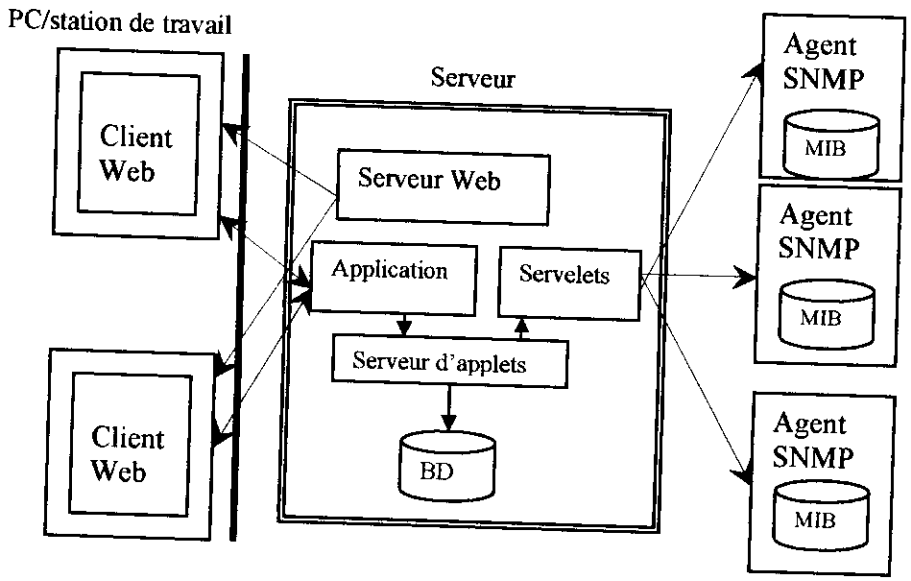


Fig.III.3 : L'architecture du système WebNMS

Les approches fondées sur le Web fournissent une interface uniforme. Elles permettent également d'accéder aux applications d'administration depuis plusieurs endroits du réseau (un site équipé d'un navigateur Web et d'une machine virtuelle Java suffit). Cependant les architectures proposées s'appuient sur un serveur centralisé et souffrent donc des inconvénients inhérents aux approches centralisées : faible extensibilité, goulot d'étranglement du serveur, sensibilité aux défaillances du serveur ainsi que toutes les requêtes

émises à destination des agents d'administration sur les éléments du réseau transitent via un serveur centralisé. Il en résulte un ralentissement de la surveillance en temps réel des éléments du réseau. Il n'y a aucune décentralisation des fonctions d'administration sur les éléments du réseau.

IV. CONCLUSION

Pour obtenir de plus amples informations sur le réseau, un administrateur peut utiliser le protocole SNMP (ou HTTP) pour communiquer avec des équipements actifs qui composent le réseau qu'il gère, afin de prendre des décisions efficaces quant à la mise en œuvre de services du réseau. Des outils plus ou moins intégrés dans des plates-formes existent pour l'aider dans son métier, que ceux-ci soient libres de droits ou commerciaux.

Cependant la difficulté voire l'impossibilité d'étendre les fonctionnalités offertes par les plates-formes d'administration et notamment celles qui sont commerciales, justifie d'envisager d'autres solutions : Les plates-formes à agents mobiles présentent en général la propriété de permettre à l'administrateur de programmer de nouvelles fonctions. Par ailleurs ces plates-formes offrant le concept d'agents mobiles permettent d'introduire plus d'autonomie dans la réalisation des fonctions d'administration et donc de décharger d'autant le travail de l'administrateur. Donc la solution envisagée est le passage à une autre technologie basée sur le paradigme d'agent mobile intelligent.

Les motivations qui nous ont conduit à recourir à cette approche (Agent Mobile) sont multiples :

- Les connexions réseau peuvent être gérées au mieux, on transfère du code exécutable sur le site de stockage de données et les actions sur ces données donnent des résultats moins volumineux que les données elles-mêmes.
- De plus, on peut traiter les résultats avant la transmission selon leur type.
- On peut aussi grouper les transmissions pour augmenter l'utilisation d'une liaison pendant toute la durée d'une connexion.
- Un autre aspect concerne le travail en mode asynchrone qui améliore les performances perçues par l'utilisateur. En effet, l'utilisateur peut activer des actions dans le réseau et accéder aux résultats plus tard quand ces derniers sont déjà stockés localement sur la station nomade. L'accès local peut être beaucoup plus rapide qu'à travers le réseau et les temps de réponse perçus par l'utilisateur peuvent être plus satisfaisants.
- Les systèmes basés sur des agents et en particulier, sur des agents mobiles, offrent le fort potentiel pour effectuer des tâches d'administration: autonomie et possibilité de gain en bande passante, etc.

- Le poste de travail sur lequel s'exécute le gestionnaire mobile de réseaux peut être déconnecté pendant l'utilisation des agents mobiles. Les résultats de leur exécution sont obtenus lors de la reconnexion. L'utilisation du réseau peut être ainsi minimisée.
- En outre le réseau n'a pas à être disponible pendant toute la durée d'exécution des agents mobiles puisqu'il n'est utilisé qu'au moment de l'envoi et de la réception d'agent.

De manière récurrente l'utilisation des agents mobiles pour l'administration réseau donne les performances comme critère de préférence des agents mobiles face au modèle Client/Serveur SNMP.

Dans le chapitre suivant nous allons détailler la notion d'agent mobile et son utilisation dans le domaine de l'administration des réseaux.

Chapitre IV

L'administration par les agents mobiles

Dans ce chapitre :

- Introduction aux agents
- Les plates formes d'administration réseaux fondées sur les agents mobiles

CHAPITRE V : ADMINISTRATION PAR LES AGENTS MOBILES

I. INTRODUCTION AUX AGENTS

Depuis 1993 et le démarrage du World Wide Web, à la fois, les techniciens, les hommes d'affaires, les banquiers et les hommes politiques s'y précipitent. Et depuis cette période, le nombre de sites, le nombre de pages, le nombre d'utilisateurs augmente sans arrêt. Mais quelques problèmes sont apparus, qu'il faudra régler afin de ne pas effrayer les investisseurs : les temps de réponse catastrophiques, les recherches difficiles dans cette montagne d'information, la sécurité des sites, la confidentialité des informations, la criminalité naissante, la protection de l'ensemble du réseau et son administration. La technologie des Agents arrive à point nommé pour aider à gérer cette « gigantesque pagaille ». [Irs_05].

I.1. Définition

Le domaine des agents est très récent. Les premières tentatives remontent tout au plus à quelques années et le terme d'agent logiciel est encore mal défini. Si on se réfère à la définition du dictionnaire du latin : « agens » : celui qui agit. *« Un Agent est une personne chargée des affaires et des intérêts d'un individu, d'un groupe ou d'un pays, pour le compte desquels elle agit »*. On peut retirer deux aspects fondamentaux :

- Un Agent accomplit quelque chose.
- Un Agent agit à la demande de quelqu'un (Agent ou utilisateur).

Comme dans le cas de toute technologie nouvelle il n'y a pas de définition universelle, mais de multiples. Citons, par exemple, la définition donnée par Caglayan et Harrison: *« Agent logiciel : entité informatique qui réalise de manière autonome des tâches pour un utilisateur »* [Emm_04].

Une deuxième définition : « Un agent logiciel est une entité autonome dotée de connaissances (données) et d'un comportement (code) privés ainsi que d'une capacité d'exécution propre. »[Mos_01]

I.2. Les caractéristiques des agents

I.2.1. Caractéristiques fondamentales

Un agent se caractérise fondamentalement par :

L'autonomie : L'agent travaille sans intervention directe, jusqu'à un point défini par l'utilisateur.

L'interactivité : L'agent doit pouvoir exercer des actions sur son environnement et réciproquement.

La réactivité : Un agent perçoit l'environnement dans lequel il se trouve et peut répondre aux changements qui s'y produisent.

I.2.2. L'intelligence

L'intelligence est basée sur un agrégat de caractéristiques (la capacité d'apprendre, la capacité sociale et une haute autonomie) chacun pouvant avoir des degrés divers.

La capacité d'apprendre : Un agent aura la capacité d'apprendre s'il sait acquérir de la connaissance, de l'information ou des habitudes.

La capacité sociale : Les agents interagissent avec les autres agents (et éventuellement des êtres humains) grâce à des langages de communication entre agents. Cette capacité sera la base pour la coopération entre les agents.

Haut degré d'autonomie : L'agent fonctionne sans intervention directe humaine ou autre (autonomie) et en plus il a une forme de contrôle sur ses actions et sur leur état interne (haut degré).

I.2.3. Mobilité

Afin d'éclaircir la définition de mobilité et d'éviter la confusion, nous devons prendre en compte deux types de mobilités :

Mobilité relative, ou par requêtes : Dans ce cas il n'y a pas un réel déplacement de l'agent. Celui-ci lance une succession de requêtes à destination de différents serveurs. C'est le cas des agents de recherche qui interrogent différents moteurs de recherche afin de fournir à son utilisateur une synthèse des résultats.

Mobilité réelle de l'agent : Le processus agent se déplace d'un serveur à un autre, sur le réseau. Le code de l'objet est transporté et ses données aussi. Ensuite, il continue son exécution sur la nouvelle machine. Quand nous parlerons d'Agent mobile, c'est ce dernier cas qui sera pris en compte.

I.2.4. D'autres caractéristiques

Les caractéristiques suivantes peuvent être ajoutées aux caractéristiques de base pour permettre à l'agent d'exécuter sa tâche. Il s'agit ici d'une liste non exhaustive, d'autres caractéristiques peuvent s'ajouter et se combiner.

Coordinativité : L'agent est capable de coordonner ses actions par rapport à un utilisateur ou un autre agent.

Compétitivité : L'agent est capable d'agir dans un environnement où d'autres agents interviennent. Le but est le même pour tous les agents présents, mais un seul l'atteindra. Les autres échoueront et, forcément, tous les coups sont permis.

Exemple : Un agent commercial peut chercher les meilleurs prix et les meilleurs services pour un produit donné. En négociant avec les fournisseurs, d'une manière plus rapide et plus optimale qu'un autre agent ou utilisateur. Un agent très compétitif pourra induire ses « adversaires » sur de fausses pistes.

I.3. Les types d'agents

La technologie agent est bouillonnante. De très nombreux acteurs interviennent sur le sujet : universités, centres de recherche et sociétés privées. Leur classification des agents dépend de leur utilisation. L'importance de cette classification est qu'elle permet de donner des attributs aux agents. La classification de l'OMG (Object Management Group) qui date de 1999 est :

- Les agents d'information.
- Les agents intelligents.
- Les agents d'interface utilisateur.
- Les agents commissionnaires
- Les agents mobiles

Et, pour compliquer un peu plus, chaque type d'agent peut se diviser en plusieurs sous-types [Omg_05].

Tout le long de notre travail nous allons étudier les agents mobiles et particulièrement dans le domaine d'administration réseau.

I.4. Les agents mobiles

I.4.1. Définition

Les agents mobiles se déplacent d'une machine à l'autre. Pour migrer, un agent mobile transfère son code et ses données sur le nouveau site puis continue son exécution sur ce site là. A l'issue de la migration, le processus de la machine initiale est détruit par une commande du nouveau processus. L'intérêt de ces agents concerne les recherches dans beaucoup de données sur des sites éloignés. Ces agents peuvent servir à nous assister dans nos tâches d'administration quotidiennes. Ils filtrent par exemple les Emails, partent à la recherche de virus, détectent les intrusions dans les réseaux, en bref effectuent les opérations d'administration de base sans plus d'intervention humaine. Ils peuvent communiquer entre eux, analyser l'information qu'ils recueillent et avertir le cas échéant l'administrateur du réseau. Par ailleurs ils permettent d'éviter d'avoir une station d'administration centralisée et

statique, car au contraire ils peuvent être interrogés et lancés à distance de n'importe où. Toutes ces facettes des agents mobiles permettent aux administrateurs de systèmes de rentrer pleinement dans l'aire de la mobilité.

I.4.2. Types d'agents mobiles

Plusieurs types d'agents mobiles peuvent exister. Ils sont à regrouper en fonction des tâches génériques qu'ils sont en mesure d'exécuter. Nous détaillons ci-dessous les principaux types d'agents mobiles que nous avons rencontrés dans la littérature.

Statique : Un agent statique est un agent mobile qui n'exécute en général qu'une seule migration. Cette migration s'effectue depuis la station de départ vers un nœud bien défini au lancement. A l'arrivée sur le nœud, l'agent mobile ne migre plus mais exécute une tâche prédéfinie.

Visiteur : Un agent visiteur est un agent mobile qui visite successivement les différents nœuds de la plate-forme afin d'y appliquer la même fonction d'administration du réseau, par exemple, en récupérant les versions des systèmes d'exploitation installées sur les éléments du réseau.

Collecteur de données : Un agent collectant les données est un agent mobile qui nécessite un point de rendez-vous avec un ou plusieurs autres agents mobiles pour les décharger des données que ces derniers ont collectées. Cet agent de collecte peut toutefois récupérer des données qui ont été laissées à sa disposition sur les différents systèmes par d'autres agents mobiles. Après la collecte, cet agent transporte les données jusqu'à un système défini à l'avance.

Transporteur : Un agent transporteur est un agent qui transporte des agents mobiles d'un réseau à un autre, ou qui transporte les agents mobiles dans un sous-réseau. Il permet d'éviter la multiplication des procédures d'identification des agents mobiles lors du passage d'un réseau à un autre (par exemple si les contraintes de sécurité imposées aux agents mobiles diffèrent).

Agent de suivi d'incident : Un agent de suivi d'incident est un agent qui est programmé pour réagir aux données qu'il analyse.

I.5. Qualités de service dans les environnements d'exécution d'agents mobiles

Comme dans tout environnement distribué, des problèmes de sécurité ou des défaillances peuvent survenir et avoir un impact sur l'exécution des agents mobiles. Il est par conséquent important que les environnements d'exécution d'agents mobiles garantissent des propriétés de qualité de service comme la sécurité ou la tolérance aux fautes.

I.5.1. Sécurité

Garantir la propriété de sécurité dans les environnements d'agents mobiles est important du fait de la nature même du modèle d'exécution des agents mobiles et du fait des interactions des agents mobiles avec plusieurs systèmes et ressources. Quatre types de problèmes de sécurité ont été identifiés pour les environnements d'exécution d'agents mobiles (voir Fig.IV.1)

- ❖ sécurité site agent : un agent doit être protégé d'un site malveillant et vice versa.
- ❖ sécurité inter-agent : un agent doit être protégé d'autres agents malveillants.
- ❖ sécurité intersite : un site malveillant peut essayer de modifier un autre site.
- ❖ sécurité entre les sites et les parties tierces non autorisées : les parties tierces peuvent essayer de modifier les sites.

Les techniques de cryptologie traditionnelles peuvent être appliquées pour apporter une solution aux trois derniers types de problème de sécurité. Seul le premier problème est particulier aux systèmes d'agent mobile. Le problème de sécurité site agent est double. D'une part, les sites doivent être protégés des agents malveillants qu'ils sont susceptibles d'accueillir. Les agents mobiles, lorsqu'ils ont accès aux ressources locales d'un site sont en effet susceptibles de les manipuler de façon malveillante.

D'autre partie les agents eux mêmes doivent être protégés des actions malveillantes pouvant être effectuées à leur rencontre par un site sur lequel ils s'exécutent. Du fait de l'exécution d'un agent par une place sur un site, le site peut avoir accès au code source de l'agent et le modifier de façon malveillante.

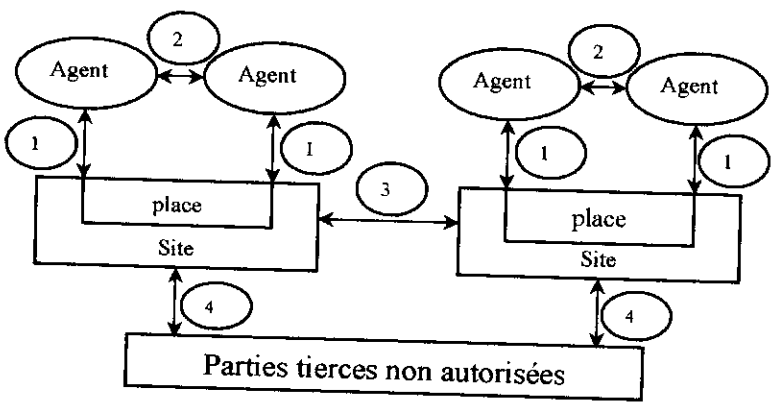


Fig IV.1. Illustration des problèmes de sécurité dans les environnements d'agent mobile

❖ **Protection des sites contre les agents malveillants au cours de leur exécution**

Un agent mobile peut avoir accès aux ressources du site sur lequel il est situé. Il peut donc profiter des accès aux ressources locales pour propager des virus, des vers ou des chevaux de troie. Il peut masquer sa véritable identité en usurpant l'identité d'un autre et mettre en place des attaques de dénis de service. Une approche classique pour protéger les sites des agents malveillants est de contrôler les accès des agents aux ressources locales du site.

❖ **Protection des agents contre les sites malveillants**

La protection des agents mobiles contre des sites hostiles est spécifique au domaine. Plusieurs approches ont été proposées :

L'approche organisationnelle : permet seulement aux sites dignes de confiance de gérer des systèmes d'agents mobiles.

L'approche de confiance réputation : permet aux agents de migrer uniquement vers des sites dignes de confiance ou qui ont une bonne réputation.

l'approche de protection par boite noire : L'objet de cette approche est de faire en sorte que l'agent apparaisse comme une boite noire de façon à ce que son code et ses données ne puissent pas être lus ou modifiés. La cryptographie mobile [Akh_99] est une étape vers l'approche protection par boite noire. La spécification de l'agent est convertie en code exécutable et en un ensemble de données encryptées. Le cryptage de données empêche un éventuel attaquant de lire ou modifier les données.

Quasiment tous les environnements d'agents mobiles existants utilisent l'approche organisationnelle.

I.5.2. Tolérance aux fautes

L'agent de par son modèle d'exécution qui implique une interaction avec plusieurs sites est enclin à disparaître à cause de la défaillance d'un site ou de la déconnexion soudaine et imprévue d'un site sur lequel il s'exécute. Pour certains types d'applications il est essentiel que les environnements d'exécution d'agents mobiles offrent des mécanismes de tolérance aux fautes. Les problèmes impliqués sont les suivants :

- ❖ Un agent peut disparaître après avoir visité plusieurs sites. Si aucune précaution n'est prise, les résultats de son exécution sur ces sites peuvent être perdus.
- ❖ Il est important de détecter la disparition d'un agent pour en informer la place qui l'a lancé.
- ❖ Il se pose un problème d'atomicité pour l'exécution globale d'un agent qui se déroule successivement sur plusieurs sites.

Peu d'environnements d'exécution d'agents mobiles existants fournissent des mécanismes de tolérance aux fautes. Certains environnements offrent des mécanismes visant à faire face à la défaillance d'un site, et donc de la place qui s'y exécuté. Plus précisément, il s'agit d'assurer la reprise des agents mobiles qui s'exécutaient sur la place défaillante, pour ce faire un mécanisme de point de reprise permettant la sauvegarde est utilisé par certains environnements.

Les points de reprise sont conservés sur disque, ce support étant supposé fiable. Ils peuvent ainsi être utilisés ultérieurement pour restaurer l'agent en cas de défaillance. Cependant, les points de reprise et la restauration doivent être utilisés avec soin, par exemple, restaurer un agent qui est toujours actif sur un système distant pourrait produire plusieurs copies du même agent, ce qui requiert un traitement explicite.

II. LES PLATES FORMES D'ADMINISTRATION RESEAUX FONDEES SUR LES AGENTS MOBILES

Une plate-forme d'agents mobiles c'est l'environnement qui fournit une interface de programmation et d'exécution en offrant des primitives pour créer, lancer et obtenir les résultats des calculs effectués par les agents. Cet environnement est constitué d'un ensemble de programmes statiques appelés places (ou noeuds) s'exécutant sur les sites du système susceptibles d'accueillir des agents. Les places sont des programmes qui fournissent aux agents l'infrastructure de base pour leur exécution et leur permet de se déplacer.

Les plates formes d'administration réseau basées sur les agents mobiles existent depuis le début des années 1990, c'est-à-dire depuis les premiers travaux du MbD (Management by Delegation) [Emm_04] qui ouvraient la voie des plates-formes dans le cadre de l'administration réseau. Nous allons décrire les principes de l'administration par délégation, puis nous présenterons les plates-formes à base d'agents mobiles tournées vers l'administration réseau.

II.1. Principe de l'administration par délégation (MbD)

L'approche introduite par MbD [Emm_04] présente les concepts de la délégation des tâches d'administration de la station centralisée vers des éléments intermédiaires, qui à leur tour deviennent des stations d'administration agissant comme des proxies.

L'architecture de MbD, repose sur des possibilités de déporter les fonctions d'administration de la station centralisée vers ces proxy, appelés MbD agent. Le composant central de l'architecture proposée par MbD est le MbD agent (voir figure IV.2). C'est une combinaison entre le modèle hiérarchique de supervision et d'un agent proxy (le MbD Agent)

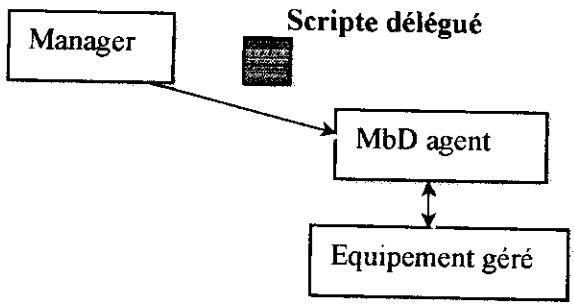


Fig IV.2: Architecture de MbD

Le MbD Agent fournit un ensemble de services qui peuvent être invoqués depuis la station d'administration, ou depuis autre MbD Agent. Le processus de délégation est engendré par la station d'administration sur laquelle ont été décrites les fonctions d'administration qui seront transmises à un MbD Agent. Ce processus permet de charger à la volée de nouvelles fonctions d'administration dans le proxy intermédiaire, ou de supprimer une fonction devenue inutile, tout cela pendant la durée de vie du MbD Agent. Dans l'architecture distribuée de MbD, un MbD Agent peut devenir à son tour le gestionnaire d'autres MbD Agents, et la station maître d'administration devient un agent proxy. Tout cela grâce aux possibilités de l'architecture qui permet de rajouter ou de supprimer des fonctions dans ses nœuds.

Dans cette architecture, il existe trois types de fonctions possibles : la fonction autonome qui a son processus (thread) d'exécution et ses données privées, la fonction exécutée à la volée et qui retourne son résultat et la fonction qui est un objet à l'écoute des demandes de services, accessible par tous les composants de l'architecture.

II.2. Quelques Plates-formes à agents mobiles tournées vers l'administration

Suite aux travaux menés sur l'MbD, un certain nombre de plates-formes à agents mobiles pour l'administration des réseaux, qui se révèlent être incontournables a vu le jour et nous allons lister quelques unes :

Grasshopper : est la première plate-forme à agents mobiles qui est compatible avec les standards industriels supportant des agents mobiles et leur gestion.

MIAMI : projet européen basé au dessus de Grasshopper, regroupe plusieurs entités différentes, comme des entreprises, afin de pouvoir administrer des noeuds situés à différents endroits en appliquant, par exemple, la même politique de sécurité à tous ces noeuds [Mia_04].

MAP: (Université de Catania, Italie) MAP sécurise en deux niveaux l'accès des agents mobiles sur les nœuds. Dans un premier temps lorsque les agents mobiles entrent dans un nouveau domaine (collection de nœuds regroupés sous la même autorité d'administration) ils obtiennent des droits d'accès sur les nœuds qui composent le domaine. De plus, une restriction peut être appliquée sur chaque nœud du domaine en fonction des exigences de sécurité mises en place par l'administrateur [Map_02].

MAGENTA: (IRISA, Campus Universitaire de Beaulieu, France). La particularité de MAGENTA est de prendre en compte la déconnexion de la station d'administration [Akh_99].

SOMA: (Université de Bologna, Italie.) Le point fort de SOMA est la supervision des agents mobiles et des nœuds d'accueil, en plus de la supervision des systèmes d'exploitation sous-jacents [Som_01].

James (Université de Coimbra, Portugal.) : Ce travail introduit la possibilité de superviser la plate-forme d'administration elle-même [Akh_99].

MobileSpaces: Cette plate-forme fournit des agents mobiles transporteurs d'agents mobiles. Ces agents mobiles transporteurs assurent les opérations de migration des agents qu'ils transportent dans un sous-réseau donné [Mos_01].

Aglets d'IBM: un Aglet est un objet représentant un agent mobile, les Aglets utilisent la notion de contexte qui correspond au nœud de ProActive ainsi le protocole **ATP** (**A**glet **T**ransfer **P**rotocol) basé sur le protocole HTTP [Agl_05].

Voyager : Cette plate-forme fournit des agents mobiles qui communique entre eux en utilisant un mecanisme de « forwarders » [Voy_05].

ProActive de l'INRIA **:** est une plate-forme utilisée pour écrire des applications réparties, parallèles, distribuées et concurrentielles (objets actifs, communication asynchrone, distribution, migration) [Emm_04].

ProActive est la plate-forme retenue pour mettre en oeuvre les fonctions de gestion de réseaux qui est le but de notre travail. Nous allons étudier en plus de détails cette plate forme dans les paragraphes qui suivent.

Cette liste n'est pas exhaustive, et d'autres exemples peuvent être trouvés, mais elle contient les plates-formes les plus connues. Toutes ces plates-formes sont bâties au dessus du langage Java. Mais il existe d'autres types de plates-formes où les tâches à réaliser sont définies par d'autres langages.

II.3. Installation des plates-formes

Dans ce paragraphe nous aborderons le problème du déploiement des plates-formes à agents mobiles, qui se base sur la mise en route des nœuds et sur des techniques pour l'enregistrement de ces nœuds. Lorsqu'il s'agit de travailler avec une plate-forme à agents mobiles, on introduit le concept de nœud. « Un nœud dans une plate-forme à agents mobiles est une entité logicielle qui permet la réception, le départ et l'exécution d'un agent mobile».

Ce nœud est une partie intégrante de la plate-forme. Pour cela le nœud fonctionne comme un daemon ou un service. Le nœud peut être mis en route systématiquement pendant la phase de démarrage du système (au boot). Le nœud peut être aussi démarré mais aussi ultérieurement arrêté à la demande de l'utilisateur, soit par des scripts automatiques de connexion, en utilisant des commandes à distance, ou encore en utilisant le planificateur de tâches fourni dans le système d'exploitation. Plutôt que d'utiliser des commandes systèmes pour démarrer ou arrêter les JVMs (java virtual machines) hébergeant les nœuds de la plate-forme, une solution intégrée à la plateforme peut être plus adaptée. En effet, le problème de l'arrêt des nœuds à distance peut se révéler fastidieux s'il est fait manuellement et parfois problématique, car il faut en particulier identifier tous les processus système faisant tourner les JVMs.

II.4. Services offerts par les plateformes

II.4.1. Migration d'agent

Cette fonctionnalité est implémentée de diverses manières dans les plates-formes par :

- l'utilisation des Sockets Tcp/IP,
- les appels de méthode via RMI (comme dans ProActive),
- un protocole propriétaire basé au-dessus de Http comme le Aglet Transfer Protocol. (ATP : Aglets)

II.4.2. Modification du code existant

Contrairement aux autres et notamment à la plate-forme Voyager [Voy_05]. ProActive stipule dans son API qu'une modification très minime doit être faite dans le code Java afin qu'il puisse devenir un code mobile. Agent MonAgent = ProActive.newActive(« mon_agent.class »,null, null). Dans cet exemple, le code « mon_agent » a pu être écrit sans se soucier qu'il soit actif et éventuellement mobile.

II.4.3. Communication inter-agents

Tous les systèmes offrent un mécanisme de Proxy pour supporter les échanges de message entre agents. Tous les systèmes implémentent un mécanisme de localisation de

l'agent afin de retrouver les agents pendant leurs différentes phases de migration. Ces mécanismes consistent soit en un serveur de noms (Aglets, Odyssey) centralisé ou réparti sur chaque site, ou en la mise en place de « forwarders » (Voyager, ProActive). Un forwarder permet de faire suivre le message de proche en proche selon l'itinéraire, afin d'atteindre l'agent. Un système de raccourci de la chaîne de forwarders est même implanté dans ProActive, permettant de limiter le temps de transmission du message à l'agent mobile. ProActive offre de plus la possibilité à un agent d'enregistrer sa référence sur un site donné (URL) : ProActive.Register(«//URL/MonAgent »).

Ainsi si un autre agent désire communiquer avec lui alors qu'il n'a pas de référence vers lui, il lui suffira de la récupérer :

MonAgent2 = ProActive.lookupActive(« //URL/MonAgent »).

Une fois la référence récupérée, on peut demander à un agent de récupérer une méthode, c'est-à-dire de communiquer avec lui. MonAgent2.foo().

II.4.4. Accès aux données de la MIB SNMP

Afin de pouvoir accéder aux informations de la MIB SNMP, il est utile que les agents mobiles puissent converser avec les agents SNMP. Certaines plates-formes ont intégré dans leur distribution des classes permettant de converser en SNMP. Cela implique que lorsqu'un agent mobile arrive sur un nœud sur lequel il doit exécuter une fonction d'administration requérant le protocole SNMP, il n'a pas besoin de télécharger les classes. Ces classes mettent en œuvre une API qui, pour masquer la complexité de l'utilisation du protocole SNMP, prédéfinit un ensemble de fonctions d'administration (par exemple, l'obtention de statistiques sur des interfaces réseaux). L'inconvénient évident est que certaines requêtes SNMP ne pourront pas être réalisées. Cette solution est mise en œuvre, par exemple par SOMA [Som_01], qui intègre une interface (nommée *Monitoring Application Programming Interface*), permettant de fournir une agrégation des résultats des indicateurs de surveillance. Ainsi l'agent mobile arrive et collecte directement la ou les valeurs des indicateurs. Une autre méthode est celle qui consiste à associer le code de l'agent mobile avec une API indépendante de la plate-forme qui donne accès à toute la pile du protocole SNMP (méthodes, fonctions, encodages, décodages, etc..), comme par exemple AdventNet [Adv_05]. En utilisant cette technique, l'agent mobile est libre d'accéder à n'importe quelle information contenue dans la MIB de l'agent SNMP. L'agent mobile va devoir télécharger les classes utiles pour réaliser sa fonction d'administration SNMP, ce qui peut être pénalisant en terme de performance si le lien réseau entre l'hôte de départ et l'hôte d'arrivée est de faible débit.

II.5. Les itinéraires

Les plates-formes présentées ci-dessus offrent toutes un système quasi-similaire d'itinéraire. Le système des itinéraires utilisé par ProActive et les méthodes fournies permettent à chaque agent de modifier son itinéraire, c'est-à-dire le rendre capable de détecter l'échec d'une migration vers un nœud, de décider de partir en visiter un autre, et cela sans être obligé d'en avertir la communauté d'administration. Cette possibilité est particulièrement intéressante en environnement hétérogène (où cohabitent des nœuds capables d'héberger une JVM, donc des nœuds et des agents ProActive, et des équipements réseaux).

II.6. Agents Mobiles d'administration

II.6.1. Administration décentralisée

Le code mobile offre la technologie nécessaire pour permettre l'administration de réseau avec un degré de flexibilité plus grand que les moyens traditionnels. Le code d'un agent mobile et les fonctions d'administration peuvent être téléchargés dynamiquement sur les équipements du réseau équipés de nœuds ProActive ou de façon réactive par les équipements réseaux (nouvelles tâches à effectuer). Par ce biais, les primitives d'administration ne sont transférées que lors de la migration des agents, évitant ainsi de consommer des ressources matérielles inutilement. Ainsi la capacité d'encapsulation du code mobile dans un agent ouvre un large éventail sur les possibilités offertes aux administrateurs de réseaux.

II.6.2. Souplesse de déploiement

Bien que développé sur un serveur central, le déploiement des agents mobiles et de leur code est largement facilité par l'adéquation des nœuds ProActive et des serveurs Http. En effet, toute la base d'un système à déploiement rapide et à évolution simplifiée peut se baser sur le déploiement d'un serveur Http pour la transmission des classes Java et par les possibilités offertes par ProActive pour le téléchargement dynamique des méthodes. Cette fonctionnalité est utilisée pendant la phase de migration des agents mobiles.

Chaque nœud ProActive décrit dans l'itinéraire récupérera à la demande de l'agent les classes Java nécessaires à son fonctionnement. Avant d'effectuer la première migration, il convient toutefois d'installer sur chacun des postes devrant recevoir la « visite » d'un agent mobile, le Jdk fourni par Sun Microsystem (java.sun.com) et les classes ProActive.

On associe au lancement des nœuds ProActive un paramètre définissant l'accès au serveur Http (URL). Pour offrir une solution plus complète, il convient, de prendre le temps de configurer les agents SNMP pour qu'ils puissent être interrogés lors du passage de l'agent mobile dédié à cette tâche. En effet, un agent mobile particulier pourra utiliser les fonctions

SNMP pour interroger les hôtes visités afin de connaître la valeur de certaines variables SNMP. On pourra donc entrevoir des agents multi-fonctions.

II.7. Les agents mobiles dans ProActive

Les modèles à objets actifs proviennent de l'unification des notions d'objet et d'activité, de la même manière que le concept d'objet lui-même provient de l'unification des notions de structure de données et de procédure.

Un objet actif se conforme donc à la règle des trois unités : unité de données (objets), unité de code (classes) et unité d'activité (threads). Le concept d'agent mobile résulte de la rencontre entre les techniques de programmation distribuée à objets et les techniques de code mobile. Dans son acception la plus courante, la notion d'agent mobile désigne une entité logicielle autonome qui se déplace de machine en machine afin de remplir une tâche. Même si diverses plates-formes d'agents mobiles ont été proposées (dans le cadre du langage Java, Aglets [Agl_05] et Voyager [Voy_05] pour les plus célèbres), aucune n'intègre de façon réellement transparente les aspects objets actifs communicants d'une part, mobilité d'autre part, comme réussit à le faire ProActive.

II.7.1. ProActive : objets actifs asynchrones communicants

Dans un souci de portabilité, la bibliothèque ProActive s'interdit toute modification du langage Java et de sa machine virtuelle. Nous sommes donc en présence d'une simple bibliothèque, qui utilise les outils standard (javac et JVM), ProActive s'interdit également la modification du code source écrit par l'utilisateur (pas de pré-traitement).

De plus, la bibliothèque est elle-même écrite entièrement en Java, ce qui autorise l'utilisation de n'importe quelle plate-forme Java et des outils associés, en particulier le fonctionnement dans un environnement ouvert et dynamique avec téléchargement dynamique de code ; ProActive utilise le mini-serveur http de la plate-forme Java pour transférer dynamiquement le *bytecode* nécessaire.

II.7.2. Modèle de base

ProActive offre un modèle de programmation parallèle et répartie que l'on peut qualifier de "***objets actifs***", et de "hétérogène" dans le sens où tous les objets ne sont pas actifs. En bref, le modèle de ProActive présente les caractéristiques suivantes :

- des objets actifs et accessibles à distance,
- la séquentialité des activités (processus purement séquentiels),
- une communication par appel de méthode standard,
- des appels systématiquement asynchrones vers les objets actifs,
- les continuations automatiques (un mécanisme transparent de délégation),

- l'absence d'objets partagés,
- une programmation des activités qui est centralisée et explicite par défaut,
- du polymorphisme entre objets standard et objets actifs distants.

II.7.3. Création des objets actifs dans ProActive

La bibliothèque a pour objectif de permettre la création d'un objet actif distant le plus simplement possible, à partir de code (donc d'une classe) initialement local et séquentiel. Un objet Java standard, créé par une instruction : A a = new A ("pfe", 7); peut être transformé en un objet actif distant de deux manières différentes comme le montre l'exemple suivant :

a) Instanciation-based :

A a = (A) ProActive. newActive (''A'', params, Node);

b) Object-based :

a = (A) ProActive.turnActive (a, node);

Instanciation-based

Passe par la création d'une nouvelle classe ou d'une interface qui sera instanciée avec l'activité FIFO par défaut. Si cette classe ou interface implémente l'interface-marqueur RunActive [Pro_05], l'activité propre de l'objet actif sera définie par la méthode RunActivity, en place de l'activité FIFO réalisée par défaut. Le second paramètre de cet appel statique (params) correspond aux paramètres effectifs à passer au constructeur lors de la création distante. Dans notre exemple, il peut être défini par :

Object [] params = {"pfe", new Integer (7)};

Il est possible de redéfinir la politique de service sans que l'objet actif implémente l'interface-marqueur RunActive en appelant une des méthodes newActive de la bibliothèque, qui prend en paramètre une activité.

A a = (A) ProActive.newActive("A",null, "//emp.edu.dz/VM1", RunActive, null)

Le dernier paramètre (node) spécifie la machine virtuelle où doit être placé l'objet actif (c'est habituellement une URL, par exemple, de la forme //Pfe/VM1). Dans le cas d'un nœud null, la création se fait dans la JVM en cours. Un autre objet actif passé en paramètre à la place du paramètre nœud (newActive est surchargée) permet la co-allocation (même machine virtuelle) avec un objet actif déjà existant. Enfin, le dernier type de création.

Object-based

Est encore plus dynamique puisqu'il prend un objet déjà créé, et en fait un objet actif accessible à distance. La sémantique est la suivante. Tout d'abord, si le code nécessaire à un accès distant sur ce type d'objet n'existe pas encore, il est généré dynamiquement. Si node est

la JVM en cours, les éléments nécessaires à la création d'un objet actif sont ajoutés à l'objet en question (une activité propre, une file d'attente, etc.). Si nœud n'est pas la JVM en cours, une copie de l'objet en question est transmise à la JVM nœud, et les éléments mentionnés ci-dessus sont ajoutés dans la JVM de cette copie. Dans tous les cas, l'objet original passif reste en place. Cette technique permet entre autre la répartition de code pour lequel on ne dispose pas de la source.

II.8. Migration

II.8.1. Migration forte

Pour une migration forte, le contexte d'exécution d'un agent est déplacé en même temps que son code et ses données, vers la place de destination. Ainsi, les environnements d'agents mobiles fournissant la migration forte définissent un modèle global d'agent et des méthodes pour l'internalisation et l'externalisation de l'état d'agent. Seulement quelques langages permettent d'externaliser l'état d'un agent à un niveau d'abstraction aussi élevé. Ces difficultés ont poussée la plupart des environnements d'agents mobiles à fournir un mécanisme de migration faible.

II.6.2. Migration Faible

La migration faible d'un agent, consiste à faire migrer uniquement son code et ses données. Il est difficile d'implémenter de façon complètement portable de la *migration forte* en Java de par l'absence de trois fonctionnalités : interruption préemptive, lecture et sérialisation du thread, écriture (reconstruction) du thread sur la JVM cible. Par souci de portabilité ProActive s'est interdit toute modification de la machine virtuelle, toute utilisation d'outils tels qu'un pré processeur de code source (ProActive agit donc uniquement à l'exécution) et finalement, en raison des problèmes de sécurité, toute modification des classes (le bytecode) au chargement.

Il existe en fait une seule primitive implémentée dans la bibliothèque ProActive pour fournir la migration des objets actifs communicants : migrateTo(). Cette primitive est surchargée en deux versions décrites dans le tableau IV.1.

	Permet la migration vers
static void migrateTo(URL)	Un site référencé par une URL
static void migrateTo(Objet)	Un site supposé d'un autre objet actif

Tab IV.1: Primitives de migration (méthodes statiques)

Un objet voulant migrer, appelle lui-même la méthode statique migrateTo() qui se charge de toutes les opérations nécessaires à la migration, notamment la sérialisation de tout le sous-système (un objet actif et tous ses objets passifs) et sa reconstruction à l'arrivée sur le site distant. Il est possible soit de migrer vers un nœud distant que l'on désigne explicitement, soit de migrer pour rejoindre un autre agent, sans que l'on sache explicitement sur quel nœud il se trouve.

II.8.2. Localisation d'objets mobiles dans ProAcive

Afin de pouvoir assurer la communication en présence de migration, il est nécessaire de pouvoir *localiser* après un nombre quelconque de migrations un objet actif. Pour être indépendant du système d'exploitation sous-jacent, la bibliothèque ProActive implémente le système de localisation dans sa couche applicative [Pro_05].

L'approche définie dans ProActive est une approche par répéteur. Plus précisément, la source d'un message n'a pas besoin de connaître la localisation exacte de l'agent mobile, il lui suffit de savoir que tout message qu'elle envoie sera correctement acheminé à destination. Pour réaliser cela, l'agent mobile qui quitte un nœud laisse derrière lui un objet répéteur qui sera chargé de faire suivre les messages vers son nouveau site d'exécution présumé (Fig IV.3).

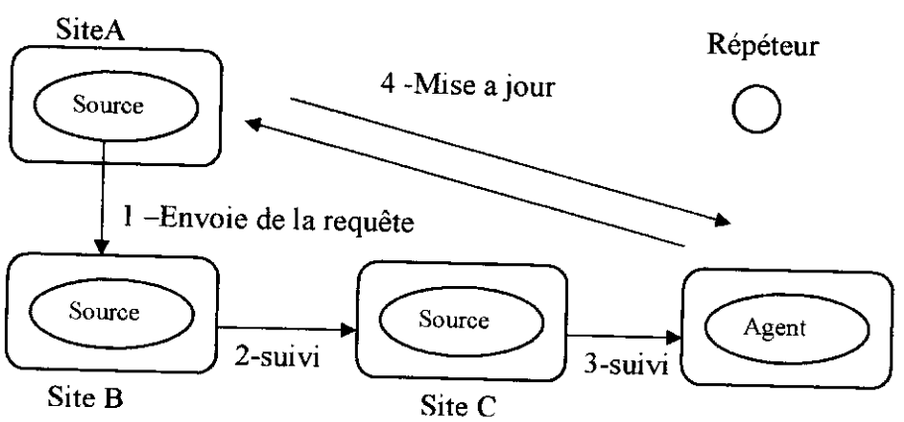


Fig IV.3 : Localisation avec répéteur

II.8.3. Exécution automatique de méthodes sur départ et arrivée

Une fonctionnalité intéressante à exploiter dans un système d'administration à agents mobiles est l'association d'un comportement à l'agent : il est possible de programmer l'agent pour qu'il effectue des méthodes, soit à son arrivée sur l'hôte distant, soit au moment de son départ. Donc la migration d'un objet actif mobile comporte deux phases principales correspondant au déroulement nominal de la migration, ce sont le *départ* d'un site et *l'arrivée* sur le site de destination. Une troisième phase existe en présence d'une levée d'*exception*.

ProActive permet de construire une abstraction qui va associer à chacune de ces phases une méthode à exécuter.

	Méthode à exécuter
static void onDeparture(String s)	au départ
static void onArrival(String s)	à l'arrivée
static void onException(String s)	lors d'une exception

Tab IV.2 : API d'exécution automatique des méthodes, notamment sur départ et arrivée

Des contraintes ont été imposées sur les méthodes susceptibles d'être exécutées automatiquement. Elles ne doivent prendre aucun paramètre et ne doivent retourner aucun résultat. Cependant il est possible d'avoir des fonctions complexes en utilisant les attributs de l'objet. Une utilisation typique de ces méthodes est la suppression et la reconstruction automatique d'éléments non sérialisables qui ne peuvent donc pas être déplacés avec l'objet, tels qu'une interface graphique.

Partie III

Conception et mise en œuvre

Chapitre V

Conception de la console d'administration

Dans ce chapitre :

- Notation UML
- Architecture
- Conception d'un itinéraire d'administration
- Console d'Administration

CHAPITRE V : CONCEPTION DE LA CONSOLE D'ADMINISTRATION

INTRODUCTION

Il y a de plus en plus d'éléments dans les réseaux des entreprises qu'un administrateur doit surveiller. Pour cela il dispose, normalement, d'outils pour analyser son réseau, surveiller son système, mais une console à agents mobiles peut apporter une aide substantielle à un administrateur. En effet, certaines tâches routinières peuvent être directement effectuées par des agents mobiles et relativement autonomes. Le mécanisme de migration fourni dans les plates-formes à agents mobiles permet de faire déplacer sans souci les agents mobiles et de faire exécuter automatiquement une tâche préprogrammée adaptée à une opération d'administration localement sur chaque élément. Il est à présent admis que les agents mobiles dans un cadre d'administration réseau permettent la réduction du temps de traitement des messages échangés entre des éléments du réseau, permettent de limiter la bande passante nécessaire aux opérations d'administration distantes, et enfin le traitement local des informations permet de réagir plus rapidement aux changements dans le réseau.

En restant dans ce cadre, notre objectif dans ce travail est de réaliser une console d'administration, fondée sur la technologie des agents mobiles, cette console permet à l'administrateur du réseau de suivre l'état de fonctionnement de celui-ci à tout moment en assurant les fonctions suivantes :

- L'inventaire des équipements du réseau avec la topologie,
- La configuration des équipements du réseau.
- L'analyse des performances. (statistiques et comptes rendus graphiques).
- La console d'administration permet la réception des traps qui peuvent aider l'administrateur dans les cas des défaillances.
- La console d'administration permet la gestion manuelle des Agents mobiles d'administration.
- Maintenir une base de données permettant l'extraction d'informations pour simplifier une analyse de l'historique des événements.

I. NOTATION UML

I.1. Définition

UML (*Unified Modeling Language*) est un langage standard conçu pour l'écriture de plans d'élaboration de logiciel. Il peut être utilisé pour visualiser, spécifier, construire et documenter les artefacts d'un système à forte composante logicielle [Gra_98]. UML est devenu la référence en terme de modélisation objet. Elle se veut être utilisée avec toutes les méthodes de modélisation objet. Ainsi, l'approche objet est un réflexe quasi-automatique dès lorsqu'on cherche à concevoir des logiciels complexes qui doivent "résister" à des évolutions incessantes. Cette approche objet consiste à penser à un problème, à un système en termes des concepts orientés objets, à savoir : objet et encapsulation, héritage et polymorphisme, message et interaction entre les objets. Il faut penser à l'objet non seulement lors de la programmation mais également lors de l'analyse et de la conception des systèmes informatiques. Ceci nécessite une notation unifiée afin de :

- représenter des concepts abstraits (graphiquement par exemple),
- limiter les ambiguïtés (parler un langage commun, ou vocabulaire précis),
- faciliter l'analyse (simplifier la comparaison et l'évaluation de solutions).

I.2. Diagrammes d'UML

Un diagramme est la représentation graphique d'un ensemble d'éléments qui constituent un système [Gra_98]. Il donne à l'utilisateur un moyen de visualiser et de manipuler des éléments de modélisation. Les différents types de diagrammes d'UML sont représentés dans l'extrait du méta modèle suivant :

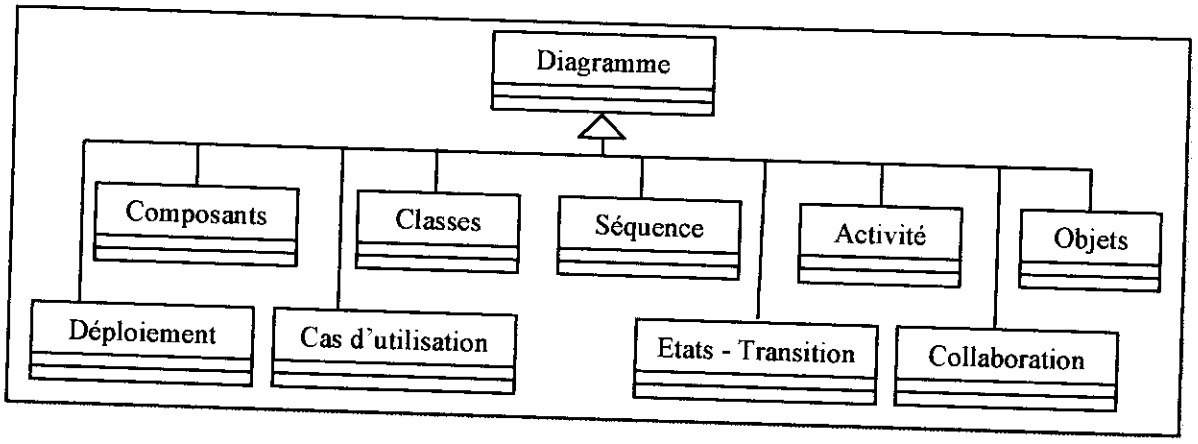


Fig VI.1 : Différents types de diagrammes définis par UML

- Les digrammes de cas d'utilisation qui représentent les fonctions du système du point de vue de l'utilisateur.

- Les diagrammes de composants qui représentent les composants physiques d'une application.
- Les diagrammes de classes qui représentent la structure statique en termes de classes et de relations.
- Les diagrammes de séquences qui sont une représentation temporelle des objets et de leurs interactions.
- Les diagrammes d'états-transitions qui représentent le comportement d'une application en terme d'états.
- Les diagrammes d'activités qui représentent le comportement d'une opération en termes d'actions.
- Les diagrammes objets qui représentent les objets et leurs relations.
- Les diagrammes de collaboration qui sont une représentation spatiale des objets, des liens et des interactions.
- Les diagrammes de déploiements qui représentent le déploiement des composants sur les dispositifs matériels.

Dans notre travail nous allons décrire quelques diagrammes que nous estimons nécessaires.

II. ARCHITECTURE

II.1. Architecture logicielle

Dans l'objectif de concevoir une console d'administration fondée sur la technologie des agents mobiles, nous avons cherché à distinguer les différents modules (ou services) qui composent cette console. Nous avons localisé quatre modules à savoir : un module de collecte et de description du réseau, un module pour la gestion des agents, un module pour la supervision et en fin celui de la persistance (Fig VI.2.).

a) Service de collecte et de description du réseau : Ce module permet de connaître les éléments constituant le réseau pour fournir des informations utiles à un administrateur pour qu'il puisse effectuer son travail. Pour connaître ce réseau, il faut collecter et analyser les informations disponibles sur les éléments qui y sont connectés. Ces éléments sont de type PCs, serveurs, routeurs, commutateurs, concentrateurs intelligents, imprimante réseau, etc, tout ce qui est en activité sur le réseau. Ce service est contrôlé par deux objets actifs de ProActive : l'*AgentDecouvert* qui est l'agent de découverte des éléments du réseau et de la topologie ; et l'agent *ChercheurNouds* qui sert à localiser les nœuds ProActive qui sont en cours d'exécution. La mise en œuvre de ces deux agents distincts est faite afin d'obtenir un délai d'exécution plus rapide.

b) Service gestionnaire des agents : permet de piloter les agents mobiles d'administration, c'est-à-dire de les créer, de les interroger, de les déplacer, et de les arrêter.

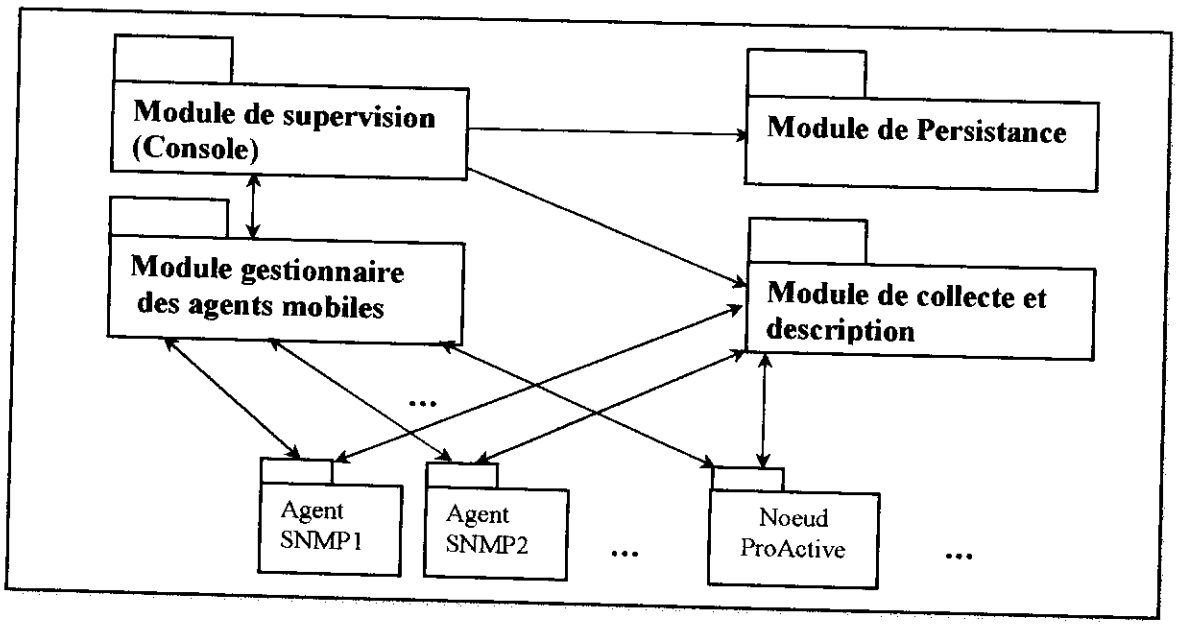


Fig VI.2 : Paquetages de domaine de la console d'administration

c) Service de supervision du cycle de vie des objets : permet à l'administrateur la visualisation des données collectées du réseau, ces informations permettront aux agents mobiles de se mouvoir sur le réseau à administrer. Autrement dit ce service permet la visualisation, l'utilisation et la supervision des objets composant l'application. Nous trouvons le service de pilotage, pour l'ensemble du réseau, des objets de l'application, objets mettant en œuvre les deux grandes familles de services que nous venons de décrire ci-dessus. Ce service de supervision pourra collecter les informations décrivant le réseau, d'une part à des fins de sauvegarde sur la station d'administration, et d'autre part pour avoir une vue en temps réel des éléments découverts automatiquement et de leur interconnexion. La supervision des agents mobiles est effectuée par un service de pilotage permettant de localiser automatiquement les agents mobiles pendant leur tâche d'administration par le biais d'un outil tel IC2D [Pro_05] (livré en standard avec ProActive).

d) Service de persistance : représente le récipient des données où on sauvegarde les données pertinentes et statistiques si la mémoire est saturée, pour une exploitation ultérieure. On détaillera les fonctionnalités de ce service plus loin.

II.2. Déploiement

Les diagrammes de déploiement représentent un ensemble de nœuds ainsi que leurs relations (lien physique), où chaque nœud englobe généralement un ou plusieurs composants. [Gra_98].

Composant : est un élément qui participe à l'exécution d'un système, et qui est exécuté par les noeuds.

Nœud : est un élément physique qui existe au moment d'exécution. Généralement, il représente une ressource de calcul ou un processeur.

Équipements actifs : Le terme équipements actifs, sous entend dans ce document, les éléments qui sont connectés sur un réseau, qui participent à son fonctionnement et qui sont administrables, c'est-à-dire équipés d'un agent SNMP.

La console d'administration doit être déployé comme suit :

6. **La SAR** (Station d'administration du Réseau) contient le manager, elle est connectée au réseau à gérer.
7. **Les Agents SNMP (équipements actifs)** : L'agent SNMP est installé sur chaque équipement du réseau par les constructeurs sinon c'est à l'administrateur de l'activer et le configurer. Pour plus de détail sur la configuration de SNMP dans les équipements du réseau (PCs, switchs et routeurs) voir l'annexe B ci-joint.
8. **Les nœuds ProActive** : Ce sont des nœuds du réseau dont lesquels l'agent SNMP est installé, ainsi ils sont capables d'héberger les agents mobiles, autrement dit, avant d'effectuer la première migration, il convient toutefois d'installer sur chacun des postes devant recevoir la « visite » d'un agent mobile, le **Jdk** (Java developer kit) fourni par Sun Microsystem (java.sun.com) et les classes ProActive. En effet, ces classes permettent de gérer la migration d'agents et sont donc nécessaires pour le développement des agents mobiles.

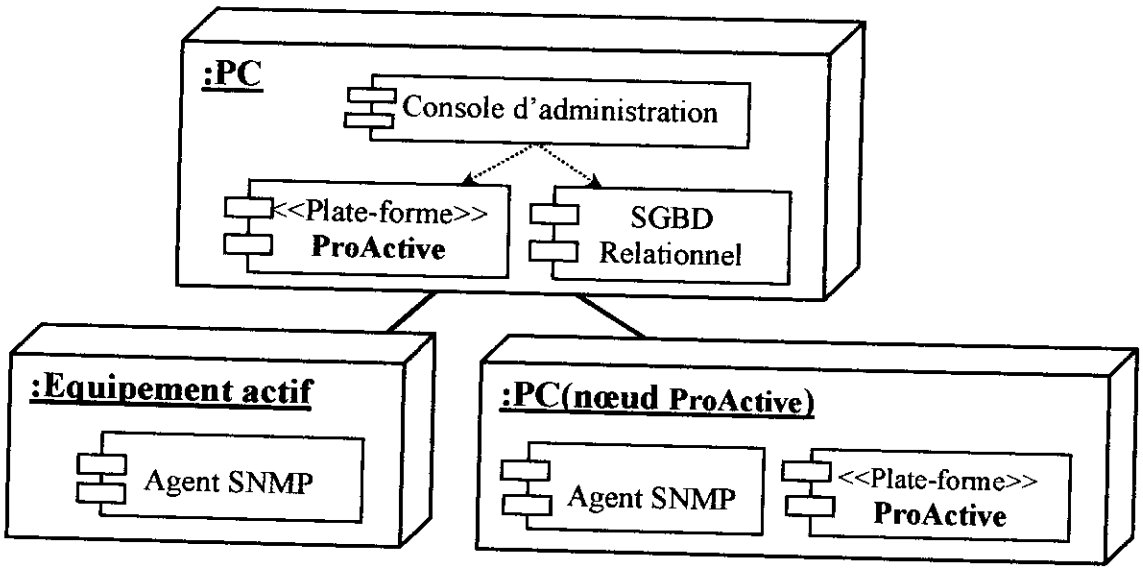


Fig VI.3 : Diagramme de déploiement de la console d'administration

Notre application sera déployée sur le réseau, une station d'administration qui gère les différents équipements du réseau (PCs et équipements d'interconnexion), elle requit la plate-forme **ProActive** pour la migration des agents, et un SGBD relationnel pour le stockage. La station d'administration (où la console est installée) communique avec les équipements actifs (switchs, routeurs, imprimantes, hubs...) via les requêtes SNMP (si ces éléments sont liés directement à la SAR), et avec les nœuds **ProActive** (PCs) via les agents mobiles. (Fig VI.3).

Avant d'entamer la conception de la console d'administration, nous avons jugé qu'il est nécessaire de concevoir un itinéraire pour l'administration, ce dernier sera utilisé à chaque fois par les agents mobiles d'administration afin de mouvoir sur le réseau.

III. CONCEPTION D'UN ITINERAIRE POUR L'ADMINISTRATION

Un itinéraire d'administration est un itinéraire qui permet à un agent mobile d'effectuer des tâches à la place d'un administrateur. Nous regroupons dans le concept d'itinéraire d'administration les opérations d'administration réseau qui sont effectuées selon la technologie Client/Serveur.

Tout en restant fidèle au modèle de **ProActive**, nous avons étendu la définition de ce qu'est une *Destination* pour que cela inclut les équipements actifs du réseau pour lesquels une opération d'administration doit être effectuée. Cette opération est une opération qui doit être réalisée en utilisant le protocole SNMP afin d'accéder aux informations de la MIB de chaque agent SNMP.

III.1. Destination

III.1.1. Les types d'intervention d'un agent mobile d'administration

Nous découpons les types d'intervention d'un agent mobile en quatre catégories, identifiables par leur mode de fonctionnement :

1. L'agent mobile effectue une migration sur un nœud afin d'y effectuer une tâche (1er cas sur la figure VI.4).

2. L'agent mobile doit se comporter comme un client d'un agent SNMP pour collecter des données de la MIB SNMP (2ème cas sur la figure VI.4).

3. L'agent mobile se rapproche au plus près de l'agent SNMP en effectuant au préalable une migration sur le site qui héberge l'agent SNMP. La communication Client/Serveur utilise l'interface de bouclage (*loopback*) de la couche IP (3ème cas sur la figure VI.4).

4. L'agent mobile se rapproche des agents SNMP en effectuant une migration sur un nœud appartenant au même réseau que ces agents. Toutes les requêtes Client/Serveur sont faites en utilisant le réseau local, plutôt que les liens inter-réseaux. Ainsi les requêtes peuvent se faire à la vitesse des liens locaux (4ème cas sur la figure VI.4).

Ces différents types d'intervention seront entièrement définis dans l'itinéraire d'administration qui sera fourni à l'agent mobile. On trouve donc dans les itinéraires d'administration des mécanismes permettant la migration pour les agents.

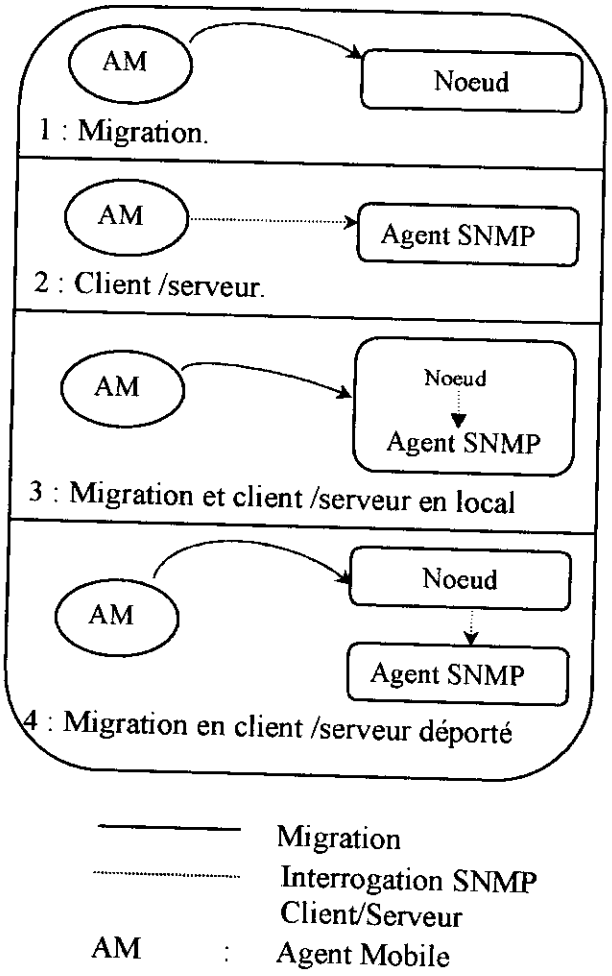


Fig VI.4 : Types d'intervention d'un agent mobile d'administration

III.1.2. Extension de la notion de Destination

Pour prendre en compte dans nos itinéraires les éléments compatibles SNMP, nous avons défini une destination particulière dénommée *DestinationSNMP*. Une *DestinationSNMP* contient l'ensemble des informations nécessaires pour effectuer une administration à distance (Client/Serveur) entre le nœud sur lequel est situé l'agent mobile et l'agent SNMP cible. On y mémorise le nom logique de l'agent SNMP cible (par exemple pfe_Blida ou à défaut son adresse IP), le nom de la communauté en lecture, le nom de la communauté en lecture/écriture et le port UDP de l'agent SNMP. Une *DestinationSNMP* contient donc toutes les informations nécessaires pour dialoguer avec un équipement actif. Toutefois, une *DestinationSNMP* ne permet pas à un agent mobile de migrer vers l'élément du réseau concerné puisque, les équipements actifs n'intègrent pas de nœuds Java pour accueillir

des agents mobiles. L'hiérarchie de la classe *DestinationSNMP* sera évoquée plus loin. (Voir diagramme de classe finale de la figure (VI.28).

Par héritage de la classe Destination, on pourrait facilement décrire un nouveau type de Destination, comme par exemple la destination *DestinationSNMPV3* pour supporter le protocole SNMP V3 ou une destination pour le protocole CMIP etc.

Pour effectuer des tâches en parallèle sur des éléments constituant le réseau, la création d'agents mobiles effectuant la même fonction peut s'avérer nécessaire. Un agent mobile père créerait des agents mobiles fils, et attendrait la synchronisation de ses fils pour collecter le compte rendu de l'exécution de la fonction. Ce type d'itinéraire peut être mis en œuvre par un nouveau type de destination (par exemple DestinationClone) qui clonerait un agent mobile et qui permettrait la synchronisation de tous les agents mobiles fils après la collecte de l'information (par exemple Destination Rendez-Vous pour la synchronisation à la fin des tâches).

La figure (FigVI.5) montre un itinéraire d'administration générique réalisé avec notre extension de **ProActive**. Il fonctionne comme suit :

- Contacter un serveur d'information : le module de supervision (On peut imaginer un serveur LDAP pour fournir aux agents les informations nécessaires concernant l'itinéraire.), afin de récupérer un itinéraire contenant la liste des éléments à visiter.
- Pour chaque élément de l'itinéraire faire
  - Cas de :
    - *DestinationProActive* alors migrer et exécuter l'action *onArrival* en tant que fonction d'administration.
    - *DestinationSNMP* alors contacter l'agent SNMP en rapprochant vers lui et effectuer l'opération d'administration réseau (en mode Client/Serveur distant)
  - Fin de Cas
- Fin Pour

En utilisant ces nouvelles définitions, les itinéraires d'administration pourront être constitués d'éléments de type *DestinationProActive*, de type *DestinationSNMP*, selon le type d'administration que l'on veut faire exécuter à un agent mobile d'administration.

III.2. Technique proposée pour l'obtention des éléments d'un itinéraire

Nous rappelons que nous sommes supposés effectuer de l'administration réseau sur plusieurs éléments, tous ces éléments ne sont pas forcément dans le même sous-réseau, ce qui va nécessiter des techniques appropriées pour mettre l'information concernant les éléments du

réseau à administrer à disposition. Nous présentons ci-après la technique que l'on a utilisée pour la mise à disposition des listes d'éléments permettant de construire des itinéraires.

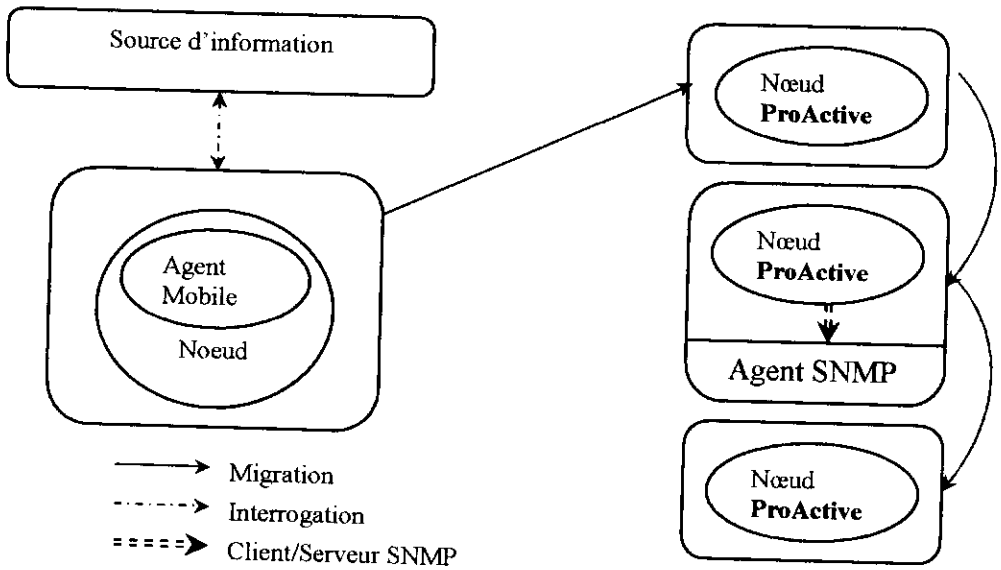


Fig VI.5 : Un itinéraire construit selon le modèle des destinations

Nous faisons transporter à l'agent mobile toutes les données qui lui sont nécessaires. L'objet actif, qui fournit la liste des éléments, est consulté une fois avant que l'agent mobile démarre son itinéraire. L'objet actif a la charge de fournir des informations les plus à jour possible aux agents mobiles, en collectant les données obtenues par un algorithme de découverte de la topologie (consulte le service de supervision).

En donnant des listes d'éléments complètes aux agents mobiles un gain de temps appréciable est réalisé pour le déroulement du suivi de l'itinéraire. Cependant, si un des nœuds devient défaillant, l'agent mobile progressera sur le suivi de son itinéraire en ne tenant pas compte du nœud défaillant. Une possibilité peut être offerte aux agents mobiles afin de prévenir le service d'itinéraire d'un ou plusieurs nœuds défaillants (utilisation de la méthode *onException*, de la plateforme ProActive). Il faut noter que, lorsque la destination cible de l'agent mobile est une *DestinationSNMP* (switchs, routeurs...), et qu'il y a plusieurs nœuds ProActive de même segment, on choisi le nœud le moins chargé pour recevoir l'agent, et à partir de ce nœud l'agent interroge la Mib de l'équipement cible.

IV. LA CONSOLE D'ADMINISTRATION

IV.1. Détermination des cas d'utilisation

Un cas d'utilisation décrit un ensemble de séquences d'actions, y compris des variantes, qu'un système exécute pour produire un résultat tangible pour un acteur. [Gra_98]

Avant de déterminer les cas d'utilisation, il faut tout d'abord déterminer les acteurs qui interviennent dans le système.

Un acteur représente un rôle joué par une personne ou une chose qui interagit avec le système. [pie_97].

Dans notre système de gestion de réseau par des agents mobiles, cinq acteurs peuvent être constatés à savoir : l'administrateur, l'agent mobile, l'agent SNMP, l'agent RMON et le serveur d'agent mobile (Nœud ProActive).

- **L'administrateur :** la personne qui contrôle et surveille le fonctionnement de la console d'administration, en configurant les options de la console, en faisant des statistiques et en prenant des décisions, etc.

- **L'agent mobile d'administration :** entité logicielle (programme) encapsule des méthodes d'administration de façon statique ou dynamique en fonction des besoins et se déplace d'un hôte vers un autre.

- **L'agent SNMP :** entité logicielle placée dans les équipements compatibles SNMP. Il fournit des informations à partir de la MIB où se trouve l'objet qu'il représente.

- **L'agent RMON :** entité logicielle située dans une sonde et se comporte comme un analyseur de protocole local et peut déclencher des alertes de performances. Un agent Remote Monitoring a besoin d'être configuré pour un type de donné particulier qui est dicté par la MIB RMON.

- **Le serveur d'agent mobile (Nœud ProActive):** Un nœud est une entité logicielle qui permet la réception, le départ et l'exécution d'un agent mobile. Ce nœud est une partie intégrante de la plate-forme ProActive. Pour cela le nœud fonctionne comme un daemon ou un service.

Après avoir défini les acteurs qui interagissent avec la console d'administration, on va déterminer pour chacun ses cas d'utilisation. Le résultat de cette étape est le diagramme de cas d'utilisation représenté dans la figure (Fig VI.6).

Acteur	Cas d'utilisation
Administrateur	◻ Configurer la console d'administration ▪ Spécification de la plage d'adresses du réseau à gérer. ▪ Configuration du protocole SNMP. ▪ Configuration des traps SNMP. ▪ Configuration du temps de rafraîchissement des données. ◻ Récupérer l'inventaire et la configuration des équipements avec la topologie du réseau. ◻ Créer un itinéraire manuellement. ◻ Créer un agent mobile d'administration manuellement en spécifiant sa tâche d'administration. ◻ Lancer un agent d'administration suivant un itinéraire existant. ◻ Gérer les Agents sur les différents hôtes et nœuds manuellement. ◻ Faire des statistiques sur le réseau. ▪ Statistiques concernant le matériel. ▪ Statistiques concernant les performances du réseau. ◻ Explorer la MIB d'un équipement. ◻ Modifier les propriétés (la configuration) d'un équipement du réseau. ◻ Mettre à jour la topologie.
Agent Mobile	◻ Effectuer une migration sur un nœud afin d'y effectuer une tâche d'administration (la collecte des statistiques, la configuration, etc.) ▪ L'agent revient à la station d'administration. ▪ L'agent s'auto détruit après qu'il termine ses tâches d'administration. ◻ Se comporter comme un client d'un agent SNMP pour collecter des données de la MIB SNMP. ◻ Se rapprocher au plus près de l'agent SNMP/RMON en effectuant au préalable une migration sur le site qui héberge l'agent mobile. ◻ Localiser les nœuds ProActive qui sont en cour d'exécution.
Agent SNMP	◻ Répond aux requêtes envoyées par les Agents mobiles d'administration. ◻ Envoie des traps vers le manager.

Agent RMON	▫ Répond aux requêtes envoyées par les Agents mobiles d'administration concernant les performances du réseau. ▫ Remonte des alertes au SAR (Station d'Administration Réseau).
Serveur d'agent mobile (Nœud ProActive)	▫ Recevoir l'agent mobile et lui permettre d'exécuter ses tâches d'administration, ce serveur réside dans tous les nouds qui sont capable d'héberger l'agent. ▫ Lancer de nouveau l'Agent mobile pour qu'il termine son itinéraire.

Tab VI.1 : Les cas d'utilisations de la console d'administration

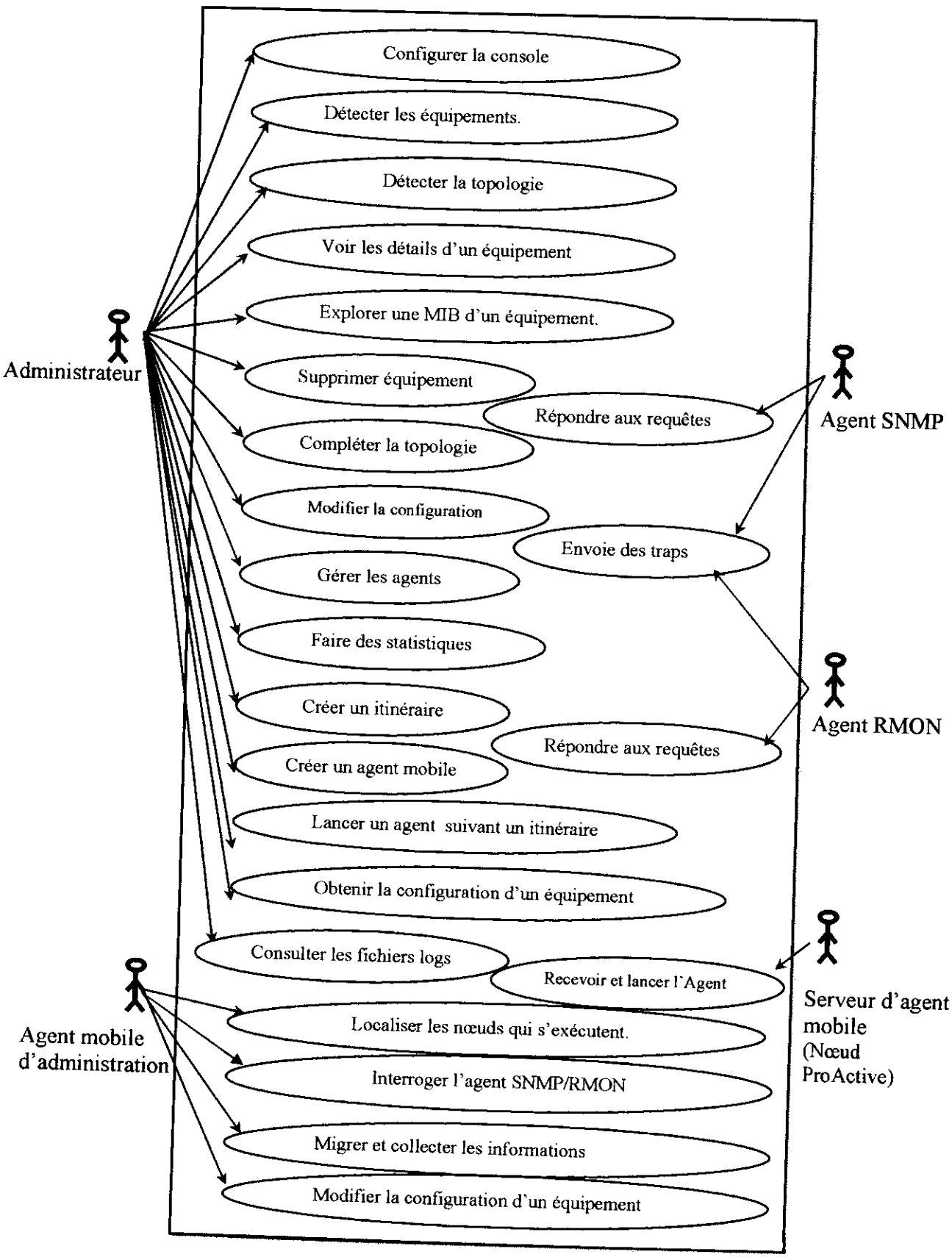


Fig VI.6 : Diagramme de cas d'utilisation de la console d'administration

IV.2. Descriptions des cas d'utilisation

Dans ce paragraphe, et pour chaque cas d'utilisation, nous allons décrire son déroulement et concevoir un diagramme de séquence pour les principaux cas. Un diagramme de séquence est un diagramme qui met l'accent sur l'ordre chronologique des actions d'une procédure et d'apporter une meilleure compréhension de l'enchaînement des événements et des messages dans le système.

IV.2.1. Configuration de la console d'administration

Spécification de la plage d'adresses du réseau géré

Ce cas d'utilisation comporte les actions suivantes :

- L'administrateur lance l'opération de modification de la plage d'adresses.
- Le système lit le fichier de configuration de la console.
- Le système lui répond par un formulaire de saisie dans lequel il y a la plage d'adresses utilisée.
- L'administrateur saisit la plage d'adresses.
- Le système contrôle les adresses saisies et enregistre la nouvelle plage d'adresses.

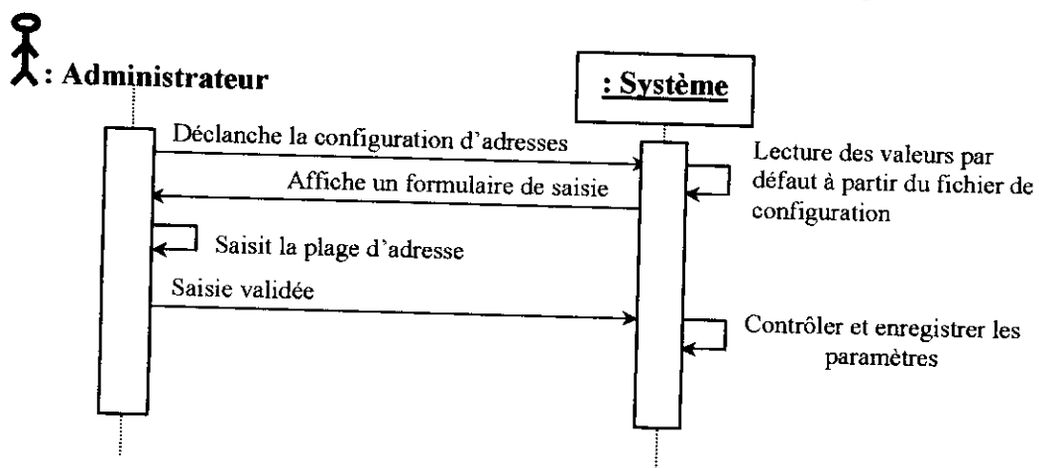


Fig VI.7: Diagramme de séquence «Spécification de la plage d'adresses du réseau»

Configuration du protocole SNMP

Dans ce cas d'utilisation s'exécutent les actions suivantes :

- L'administrateur lance l'opération de configuration du protocole SNMP.
- Le système lit le fichier de configuration de la console d'administration.
- Le système lui répond par une interface de saisie avec les paramètres utilisés par défaut de la configuration.
- L'administrateur valide les paramètres par défaut ou les modifie.
- Le système enregistre les paramètres validés par l'administrateur.

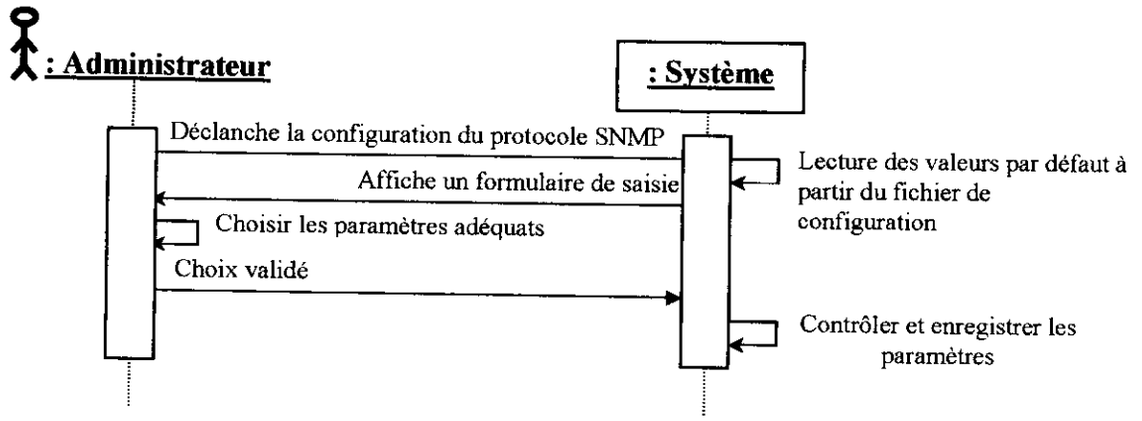


Fig VI.8: Diagramme de séquence «Configuration du protocole SNMP»

Configuration du temps de rafraîchissement des données

Ce cas d'utilisation est caractérisé par les actions suivantes :

- L'administrateur lance l'opération de modification de temps de rafraîchissement.
- Le système lit le fichier de configuration de la console.
- Le système lui répond par un formulaire de saisie dans lequel il y a le temps de rafraîchissement par défaut.
- L'administrateur saisit le temps.
- Le système contrôle les temps et enregistre les nouvelles valeurs.

Configuration des traps SNMP

Ce cas d'utilisation comporte les actions suivantes :

- L'administrateur lance l'opération de configuration des traps SNMP.
- Le système lit le fichier de configuration de la console.
- Le système lui répond par un formulaire de saisie dans lequel il y a le port à recevoir les traps.
- L'administrateur saisit le numéro de port et active la réception des traps.
- Le système contrôle le numéro de port et enregistre la nouvelle valeur, et crée un AgentTrap qui reste en écoute des messages SNMP.

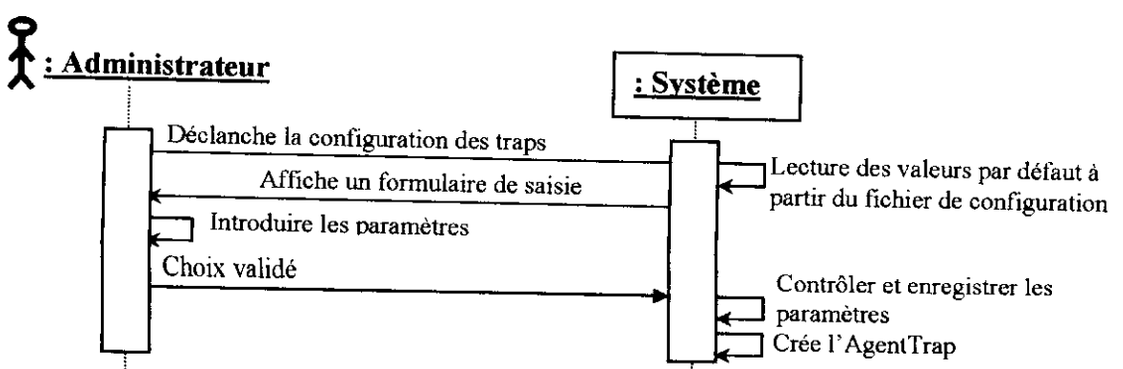


Fig VI.9 : Diagramme de séquence «Configuration des traps SNMP»

IV.2.2. Découverte du réseau

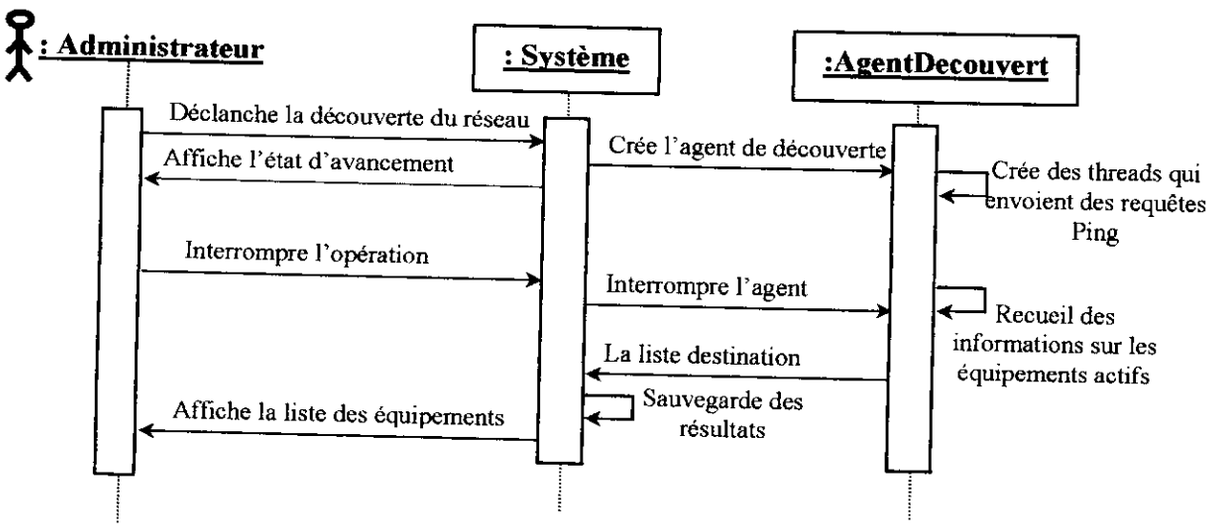


Fig VI.10 : Diagramme de séquence «Découverte du réseau »

Dans ce cas d'utilisation s'exécutent les actions suivantes :

- L'administrateur déclanche l'opération de découverte du réseau.
- Le système répond par une interface qui indique l'état d'avancement de l'opération.
- L'administrateur peut à n'importe quel moment interrompre cette opération.
- Le système crée L'AgentDecouvert qui s'occupe de la tâche de découverte du réseau, et ce dernier à son tour crée des threads qui envoient des requêtes Ping (ICMP) à chaque adresse de la plage, cette opération permet de construire une liste de destination
- Le système affiche dans la même interface la liste des équipements détectés, cette liste sera séparée en trois listes : la première concerne les nœuds qui peuvent hébergés l'agent mobile (*DestinationProActive*), la deuxième pour les équipements actifs (*DestinationSNMP*) et une troisième liste concerne les autres équipements détectés (NoDestination).

Ce cas d'utilisation est illustré dans la figure (Fig VI.10).

IV.2.3. Création d'un itinéraire manuellement

Ce cas d'utilisation contient les actions suivantes :

- L'administrateur déclanche l'opération de création d'un itinéraire pour l'agent mobile d'administration.
- Le système répond par une interface qui permet d'introduire les paramètres nécessaires.

- L'administrateur introduit les paramètres, puis valide les données.
- Le système contrôle les données introduites, crée et enregistre l'itinéraire d'administration.

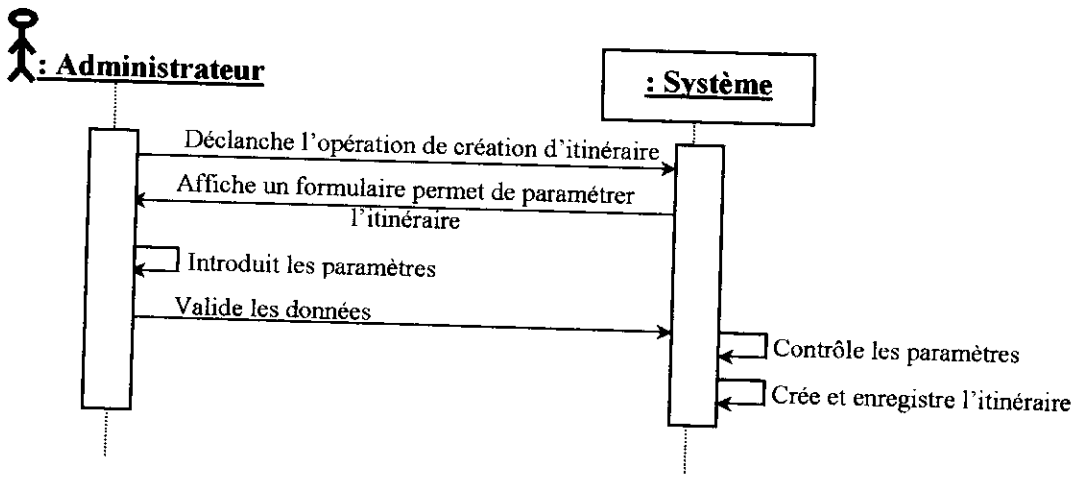


Fig VI.11 : Diagramme de séquence «Création d'un itinéraire manuellement »

IV.2.4. Création d'un agent mobile d'administration manuellement

Dans ce cas d'utilisation, on regroupe deux cas : la création et le lancement de l'agent. Ces deux cas peuvent être décrites par les actions suivantes :

- L'administrateur lance l'opération de création d'agent.
- Le système affiche une fenêtre qui permet à l'administrateur d'attribuer une tâche d'administration bien précise (spécifier le constructeur de l'agent) et de spécifier l'itinéraire de l'agent (les hôtes vers lesquels la migration aura lieu).
- Le système contrôle les paramètres introduits, s'ils sont valables il va créer l'agent mobile et l'envoie.
- Suivant l'itinéraire qu'il a, l'agent migre vers la première destination en effectuant la tâche d'administration puis vers la deuxième destination, etc.

La figure (Fig VI.12) illustre le déroulement de ce cas d'utilisation.

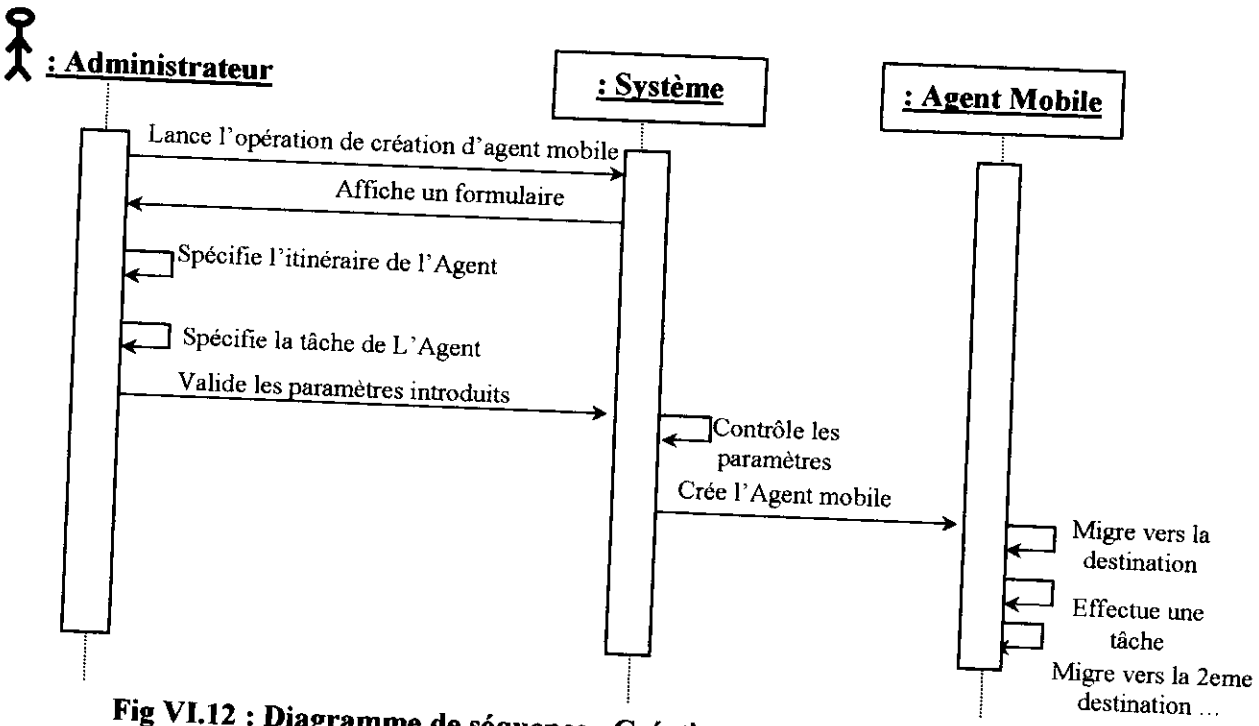


Fig VI.12 : Diagramme de séquence «Création et lancement d'un agent mobile»

IV.2.5. Détection de la topologie

Ce cas d'utilisation comporte les actions suivantes :

- L'administrateur déclanche l'opération de détection de la topologie du réseau.
- Le Système consulte la liste des équipements d'interconnexion.
- Le Système localise chacun de ces équipements.
- Le Système crée un agent (AgentDecouvert) en spécifiant son itinéraire.
- L'AgentDecouvert se rapproche vers chaque switch.
- L'Agent interroge l'Agent SNMP du switch (mode Client/Serveur).
- Le switch traite les requêtes reçues et envoie des réponses vers l'AgentDecouvert
- L'AgentDecouvert traite les réponses de chaque switch.
- L'AgentDecouvert envoie les résultats au Système.
- Le Système sauvegarde les résultats dans la base.
- Le Système repère le switch le plus sollicité (relié avec le maximum des switchs) qui va être la racine de l'arbre topologique.
- Le système affiche la topologie.

La figure suivante montre l'enchaînement de ces actions.

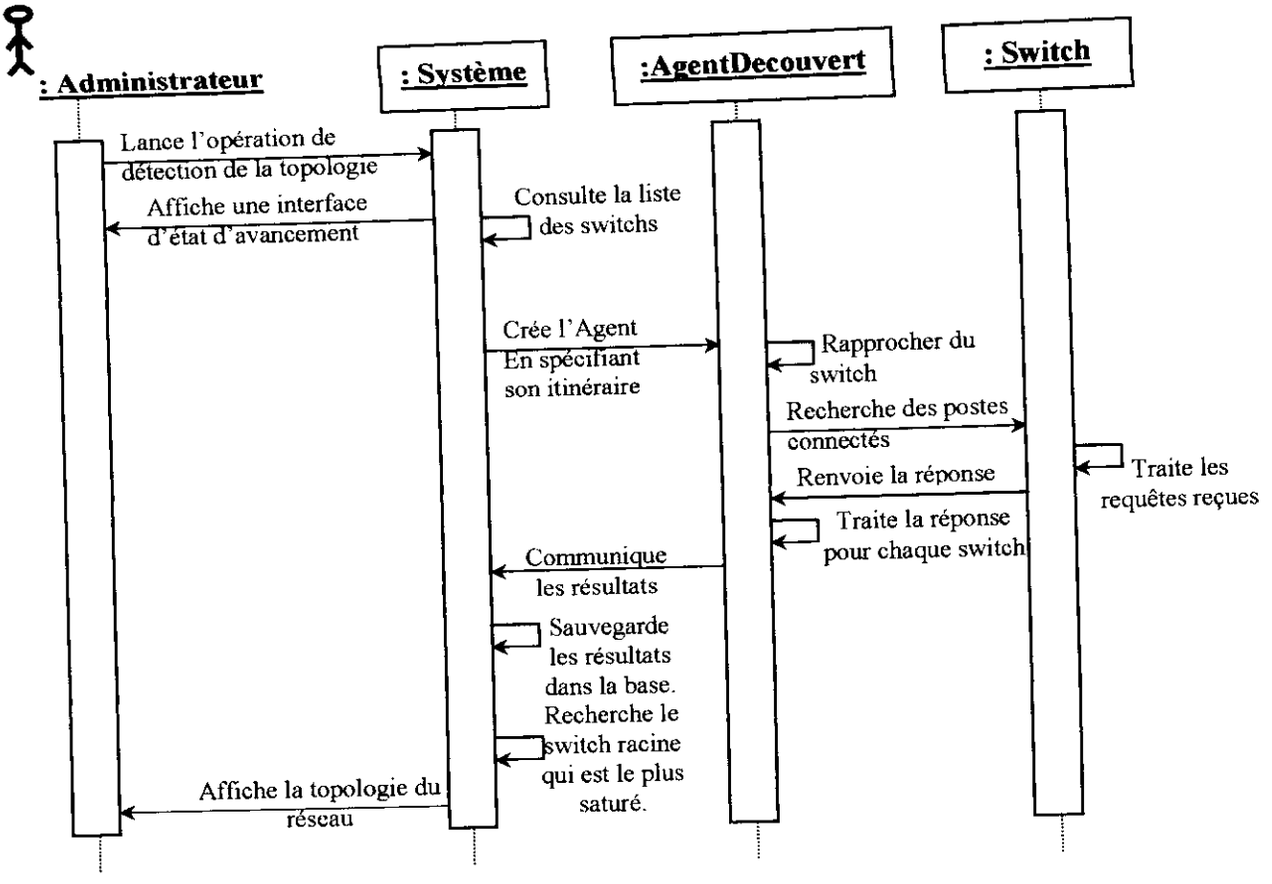


Fig VI.13 : Diagramme de séquence «Détection de la topologie »

IV.2.6. Elaboration des statistiques sur le réseau

Ce cas d'utilisation se déroule lorsque l'administrateur veut avoir des statistiques sur l'état et les performances du réseau. Dans ce cas d'utilisation, l'administrateur lance l'opération, le système lui affiche une fenêtre qui lui permettre de spécifier la nature des statistiques et sélectionne la machine sur laquelle il veut faire des statistiques. Dans le cas des statistiques sur l'état du réseau, le système consulte la base et récupère toutes les informations nécessaires. Mais dans le cas des statistiques de performances le système crée un Agent de collecte d'information pour faire ce travail.

IV.2.7. Exploration de la MIB d'un équipement

L'équipement est un Agent SNMP (ne peut pas héberger l'agent mobile)

Ce cas d'utilisation comporte les actions suivantes :

- L'administrateur lance l'opération d'exploration de la MIB.
- Le système lui affiche une interface dans laquelle l'administrateur choisit l'équipement actif en question.
- Le système crée un agent ExplorateurMib qui va migrer vers le noeud **ProActive** le plus proche de l'équipement en question et le moin chargé.

- A partir de ce noeud proche, l'agent mobile interroge l'Agent SNMP en mode client serveur (requêtes SNMP).
- L'Agent SNMP traite les requêtes et envoie des réponses.

L'équipement est un nœud ProActive (il peut héberger l'agent mobile)

Dans ce cas d'utilisation, le système crée l'Agent mobile qui va migrer vers le noeud ProActive, une fois l'agent est arrivé, il commence à interroger l'AgentSNMP en mode Client/serveur localement en utilisant l'interface de bouclage (*loopback*).

IV.2.8. Modification des propriétés (configuration) d'un équipement du réseau

Ce cas d'utilisation intervient lorsque l'administrateur veut modifier la configuration d'un équipement actif du réseau, elle comporte les actions suivantes :

- L'administrateur lance l'opération de modification de la configuration d'un équipement.
- Le système lui affiche une interface dans laquelle l'administrateur choisi l'équipement en question et introduit la communauté de lecture /écriture.
- Le système crée un agent de modification (AgentModification) qui va migrer vers un nœud le plus proche de l'équipement en question, ou bien vers l'équipement cible si ce dernier est un nœud ProActive.
- L'agent effectue la tâche de modification en envoyant des requêtes « Set-request » vers l'agent SNMP en mode Client/Serveur localement ou distant.
- Une fois l'Agent termine sa tâche de modification, il s'auto détruira après qu'il envoie un message à l'AgentGestionaire (ce dernier se trouve dans la SAR) indiquant la fin de la mission.

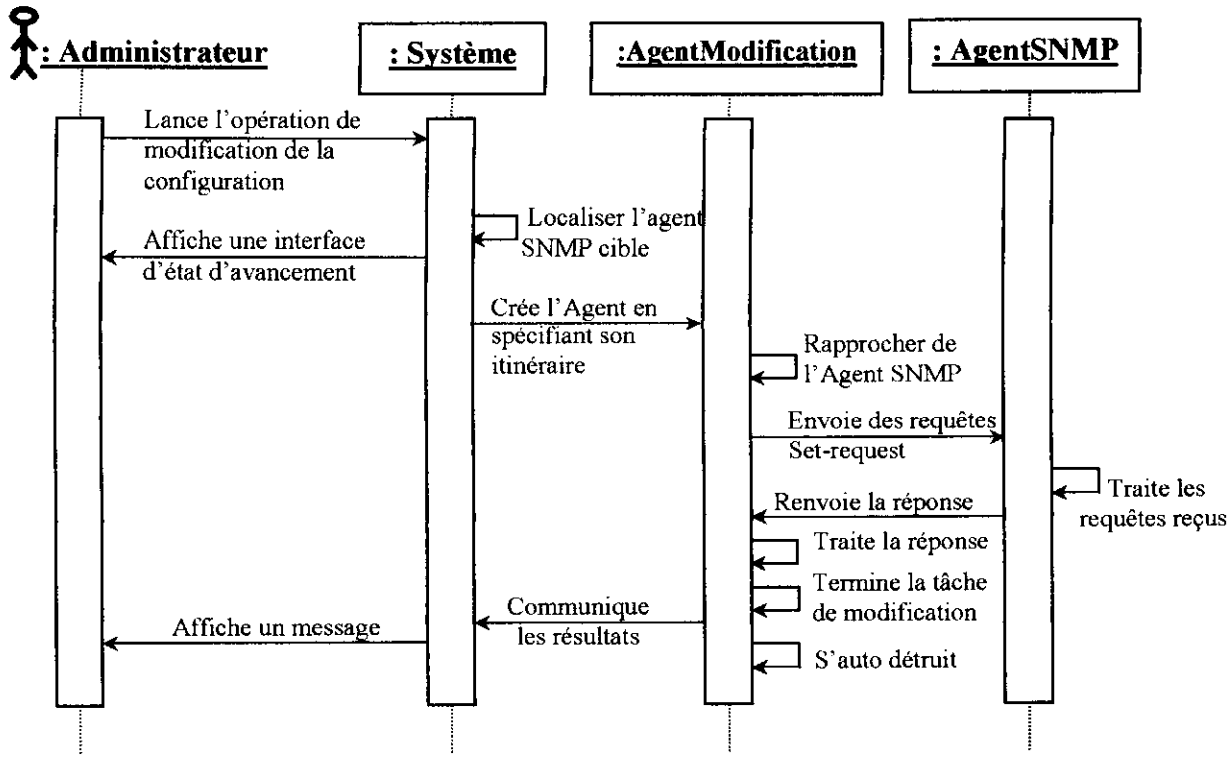


Fig VI.14 : Diagramme de séquence «Modification de la configuration d'un équipement»

IV.2.9. Enregistrement des nouveaux équipements dans la topologie

Ce cas d'utilisation apparaît lors de l'ajout d'un équipement qui ne sera pas détecté par l'agentDecouvert, tels que les concentrateurs (hubs...).Un tel cas comporte les actions suivantes :

- L'administrateur lance l'ajout manuel de l'élément dans la topologie.
- Le système recherche la liste des agents non connectés
- Le système répond par un formulaire de saisie.
- L'administrateur choisit l'agent à connecter au nœud sélectionné et le numéro de port.
- Le système vérifie les informations introduites.
- Le système met à jour la topologie du réseau.

La figure suivante illustre ce cas d'utilisation :

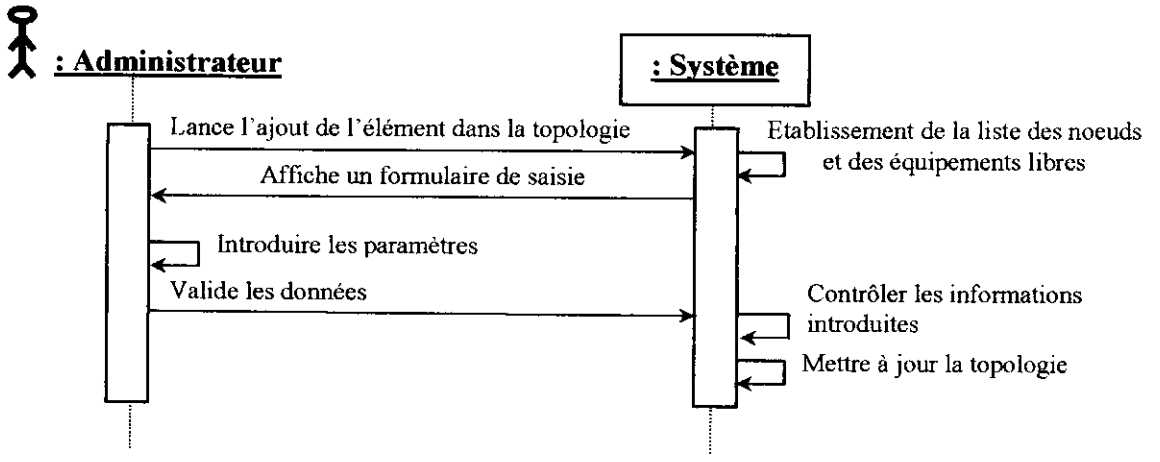


Fig VI.15 : Diagramme de séquence «Enregistrement des nouveaux équipements »

IV.2.10. Suppression d'un équipement de la topologie

Ce cas d'utilisation aura lieu lorsque l'administrateur veut modifier la topologie en supprimant un équipement. Elle commence par le lancement de la suppression par l'administrateur et se termine par l'enregistrement du nouvel état.

IV.2.11. La migration de l'agent vers un nœud ProActive

Deux cas peuvent être envisagés :

L'agent revient à la station d'administration après l'achèvement de la tâche

La figure VI.16 résume les différentes actions :

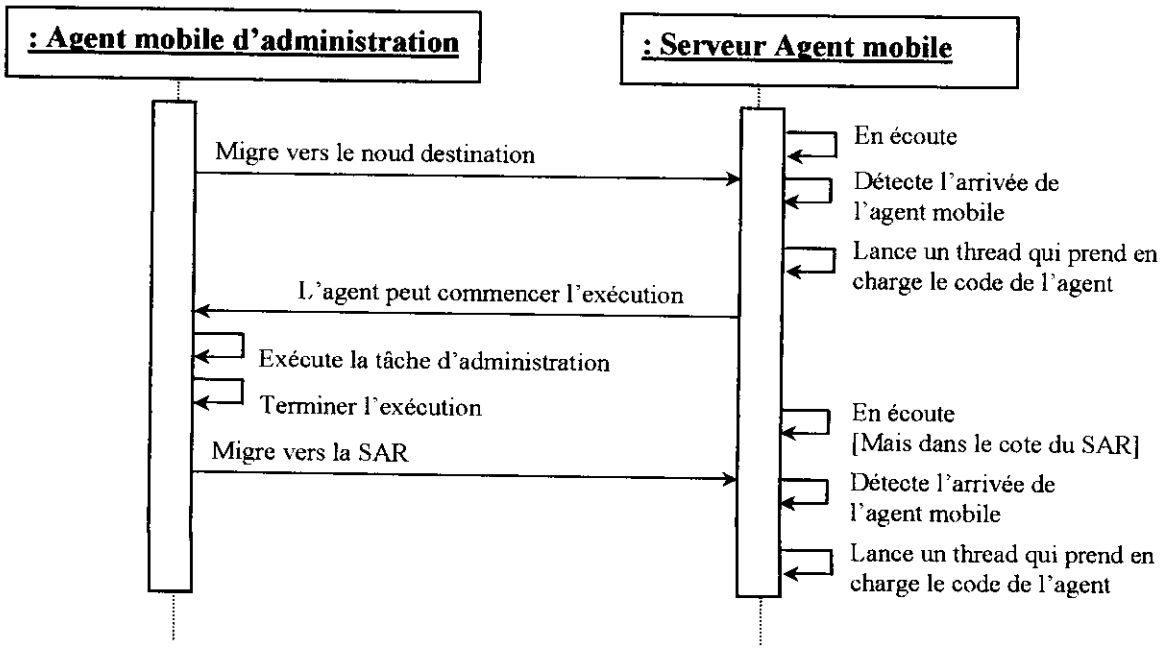


Fig VI.16 : Diagramme de séquence «Migration avec retour de l'agent»

L'agent s'auto détruit après qu'il termine sa tâche

La seule différence entre ce cas et le précédent est que lorsque l'agent termine sa tâche, il s'auto détruira.

IV.2.12. L'Agent se comporte comme un client d'un agent SNMP/RMON

Ce cas d'utilisation intervient pour collecter des données de la MIB SNMP/RMON d'un équipement qui ne peut pas héberger l'agen, il se compose des actions suivantes :

- L'Agent mobile d'administration se rapproche vers l'agent SNMP/RMON en migrant vers un noud (ProActive) qui se trouve le plus proche possible (simple liaison) de l'agent SNMP/RMON à interroger.
- L'Agent mobile ouvre une session SNMP.
- L'Agent mobile construit l'entête de la requête en mettant la version du protocole, le nom de communauté et l'adresse de l'agent à interroger (destination).
- L'Agent mobile d'administration crée le PDU (Protocol Data Unit) et l'envoie à l'agent SNMP/RMON.
- L'agent SNMP/RMON reçoit la requête et la vérifie (la version, nom de communauté, et l'adresse source)
- L'agent SNMP/RMON analyse la requête reçue puis il crée un PDU de réponse.
- L'agent SNMP/RMON envoie la réponse vers l'Agent mobile d'administration.
- L'Agent mobile d'administration transmet les informations récupérées à la SAR.

La figure ci-dessous illustre le déroulement de ce cas d'utilisation :

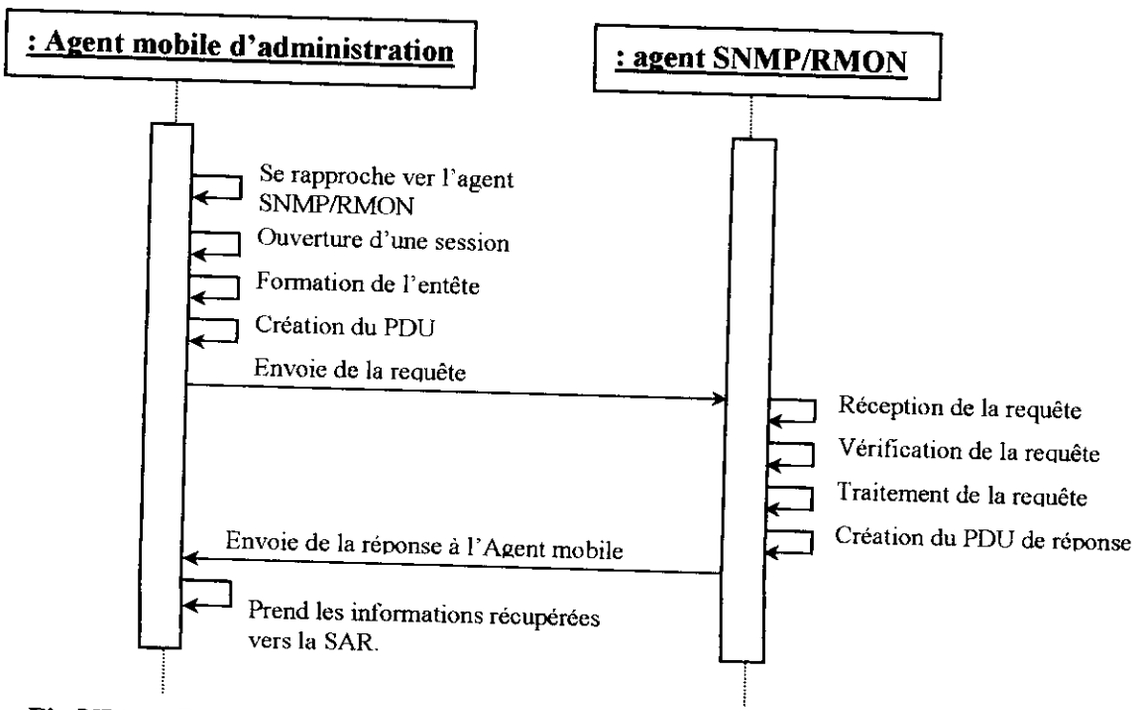


Fig VI.17 : Diagramme de séquence «L'interrogation d'un agent SNMP par l'agent mobile »

IV.2.13. Envoie des traps vers le manager

Dans ce cas, il se déroule les opérations suivantes :

- L'agent SNMP envoie des traps vers le manager.
- Le Système (plus exactement AgentTrap) interprète le trafic capturé (les traps).
- Le Système enregistre ces traps dans des fichiers.
- Le Système remonte des alarmes vers l'administrateur.
- L'administrateur consulte ces fichiers et prend des décisions.

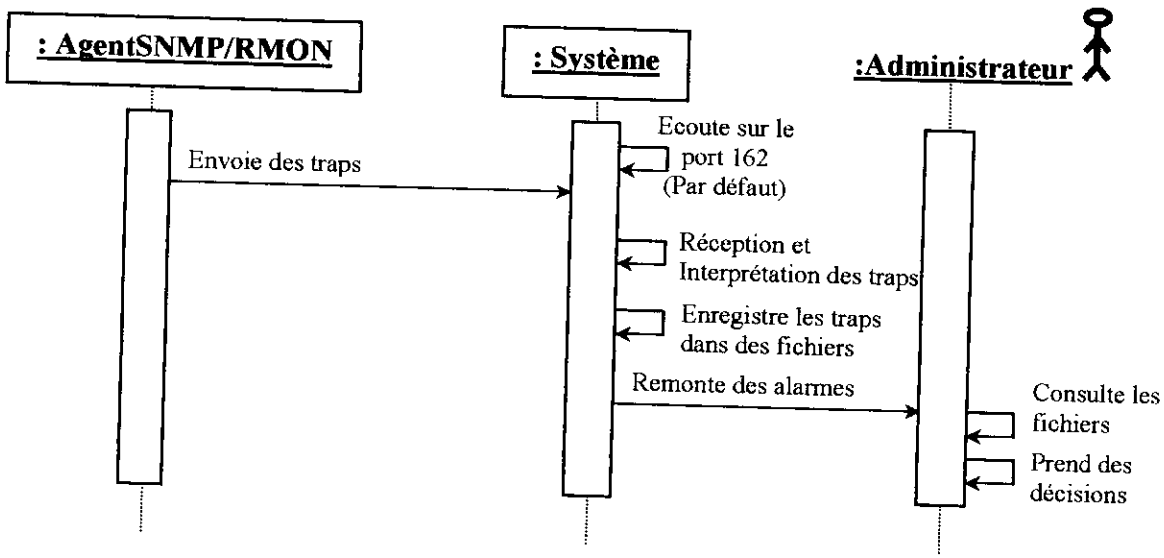


Fig VI.18 : Diagramme de séquence «Envoie des traps vers le manager»

IV.3. Diagrammes de collaboration

Un diagramme de collaboration entre objets vise à représenter du point de vue statique et dynamique les objets impliqués dans la mise en place d'une fonction de l'application. [Nat_97]

Nous avons choisi de représenter les diagrammes de collaboration des principales fonctions de notre système de gestion de réseau.

1. Configuration du temps de rafraîchissement des données
2. Détection des équipements du réseau.
3. Création d'un itinéraire manuellement.
4. Création d'un agent mobile d'administration manuellement en spécifiant sa tâche d'administration et son itinéraire.
5. Elaboration des statistiques sur le réseau.

IV.3.1. Configuration du temps de rafraîchissement des données

Ce cas d'utilisation est déclanché lorsque l'administrateur veut modifier la configuration du temps de rafraîchissement, il se réalise par la collaboration des objets instance des classes suivantes : Config-temps, Doc-config et JF-Config-temps. Ce cas d'utilisation commence par la lecture du fichier de configuration et se termine par la mise à jours de ce fichier. Il faut noter ici, que JF_Config_temps est un objet d'interface.

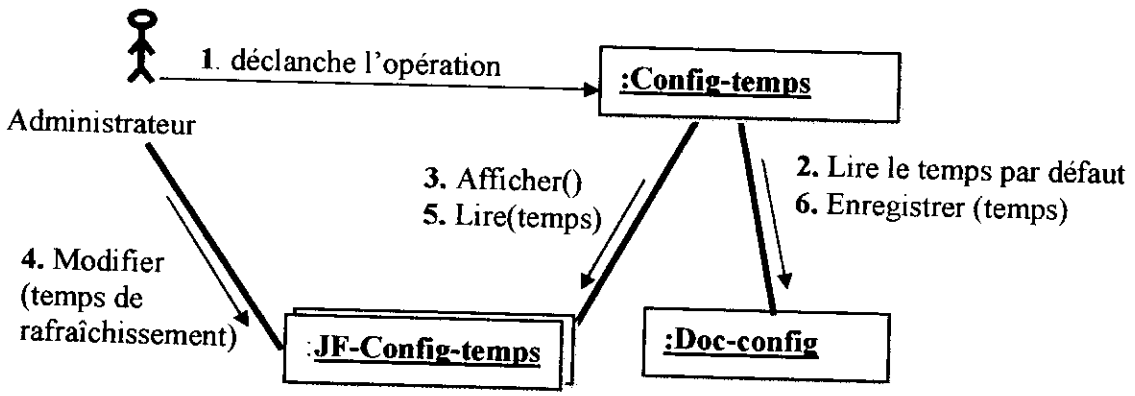
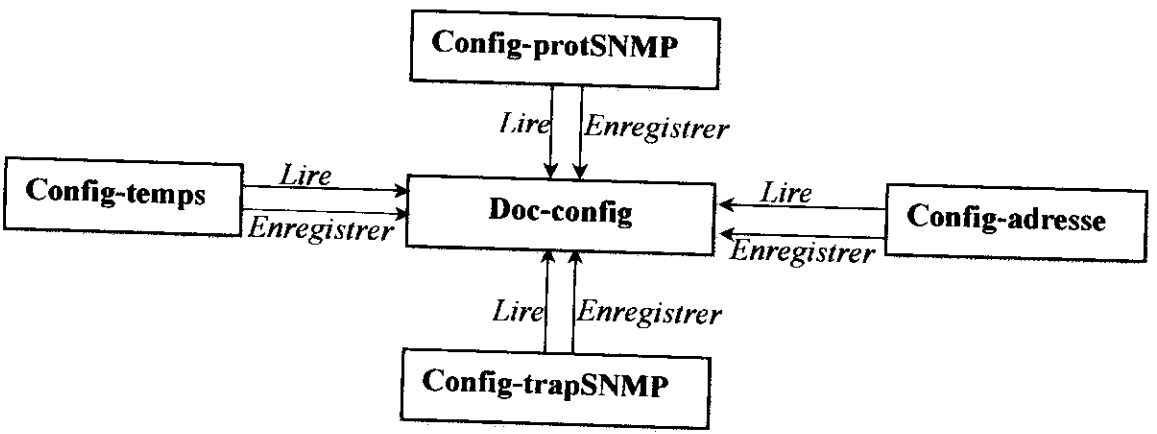


Fig VI.19 : Collaboration des objets « modification du temps de rafraîchissement »

Un diagramme de classes préliminaire, compatible avec la collaboration précédente et inclus les autres configurations tel que la configuration de la plage d'adresse, la configuration du protocole SNMP et celle des traps SNMP est représenté par la figure (Fig VI.20).



VI.20: Ebauche de diagramme de classes : Configuration de la console

IV.3.2. Découverte des équipements du réseau :

Cette fonction est contrôlée principalement par un Agent dit ***AgentDecouvert***, qui est un objet actif sur la plate forme ProActive. Dans la figure (Fig VI.21), nous allons identifier les objets qui participent à la réalisation de la fonction de découverte des équipements du réseau à savoir :

AgentDecouvert : objet actif, permet la création des threads de ping (Decouvthread) et affiche une interface à l'administrateur (JF-DecouvR) montrant l'état d'avancement de l'opération.

Decouvthread : chaque objet, permet l'envoie des requêtes ping à une adresse IP afin de vérifier l'existence de cette entité IP sur le réseau.

JF-DecouvR: Objet d'interface instancier de la classe javax.swing.JFrame.

AgentDestination : c'est un objet actif qui s'occupe de l'insertion des équipements détectés dans la base pour une utilisation ultérieur.

DestinationListe : un objet qui va récupérer l'équipement détecté et le classer dans l'une des trois listes :

- **DestinationSNMP :** Contient les équipements dotés par l'Agent SNMP mais pas par la plateforme ProActive (comme les switchs...).
- **DestinationProActive** : Contient les nœuds qui possèdent l'Agent SNMP et capable d'héberger l'Agent Mobile.
- **NoDestination :** regroupe la liste des équipements qui ne possède ni l'agent SNMP ni la plateforme ProActive.

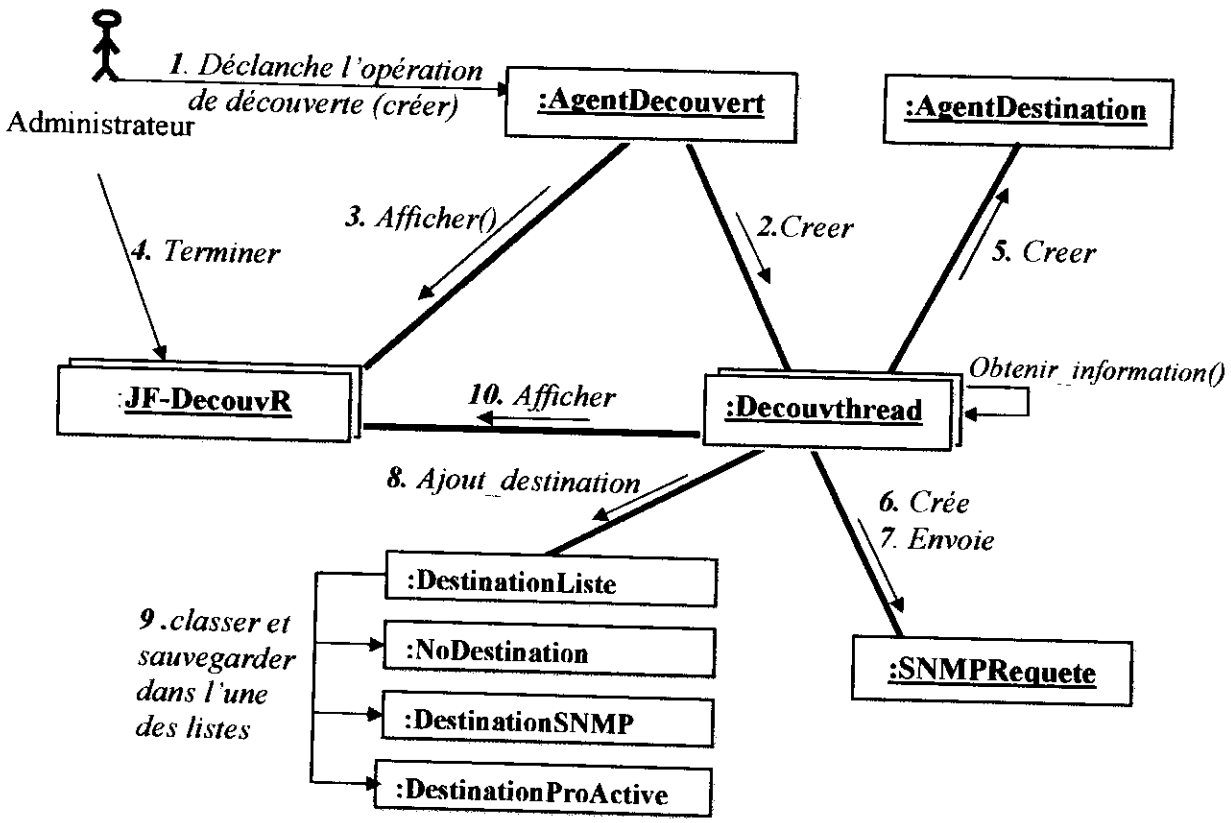


Fig VI.21 : Collaboration des objets « Détection des équipements du réseau»

La figure VI.21 montre la collaboration de ces objets pour la réalisation de la fonction de découverte du réseau. Le comportement de la procédure *Obtenir_informations()* est expliqué dans la figure VI.22.

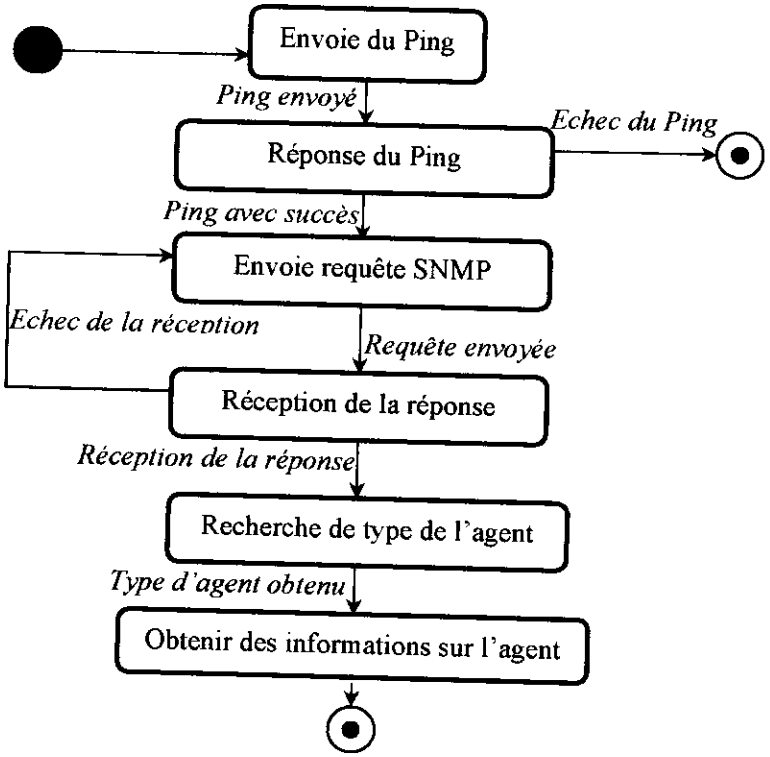


Fig VI.22: Comportement de la procédure *obtenir_information()*

IV.3.3. Création d'un itinéraire manuellement

Ce cas d'utilisation s'exécute lorsque l'administrateur veut créer un itinéraire d'administration qui se compose de plusieurs nœuds ProActive, sachant que chaque nœud appartient à une JVM, et plusieurs JVMs peuvent être trouvées dans le même poste. La réalisation de ce cas s'effectue par la collaboration des objets instances des classes : Interface, Itinéraire, JF-Itiniraire et DestinationProActive.

La figure VI.23 illustre cette collaboration des objets :

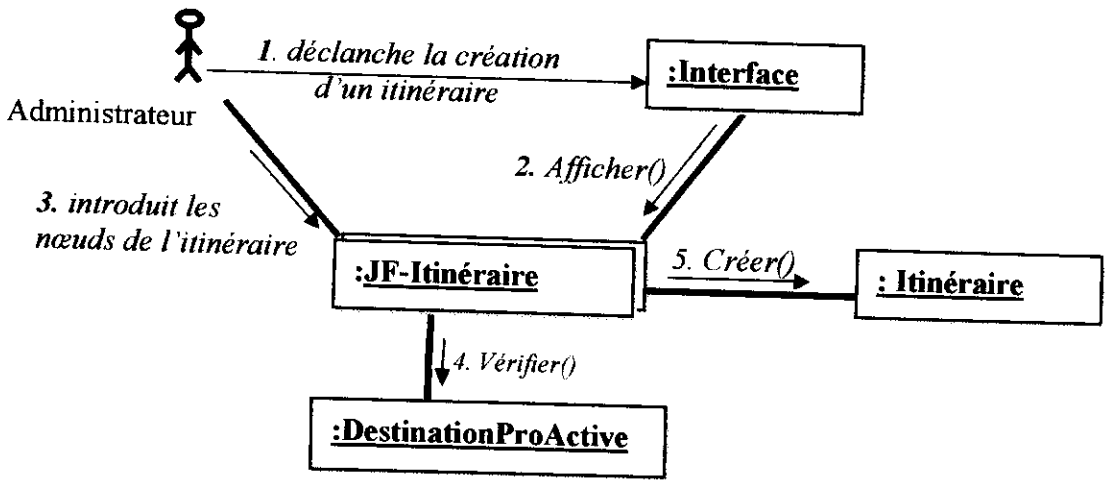


Fig VI.23 : Collaboration des objets « Création d'un itinéraire d'administration»

Le comportement de la procédure vérifier() est expliqué dans la figure VI.24 :

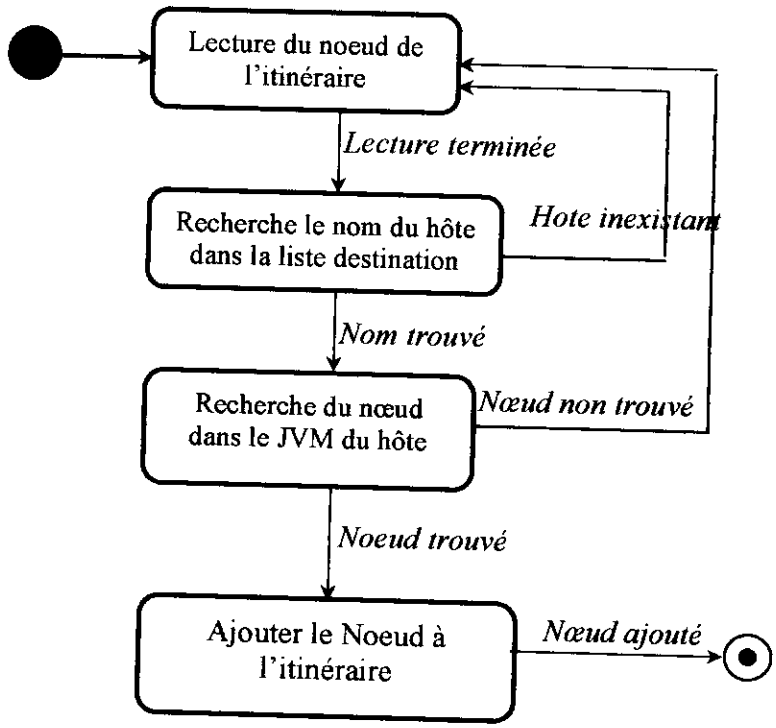


Fig VI.24: Comportement de l'opération Verifier()

IV.3.4. Création d'un Agent d'administration

Ce cas d'utilisation intervient lorsque l'administrateur veut créer un Agent mobile d'administration en spécifiant sa tâche d'administration et l'itinéraire qu'il doit l'emprunter.

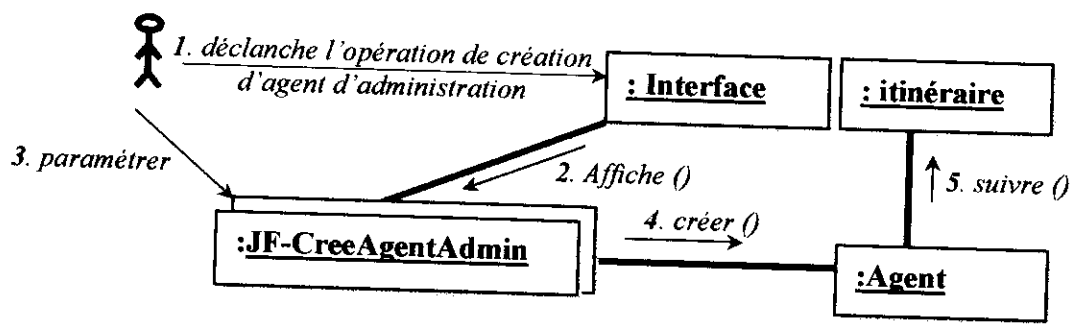


Fig VI.25 : Collaboration des objets « création d'un agent mobile manuellement »

IV.3.5. Elaboration des statistiques sur le réseau

La figure (VI.26) montre la collaboration des objets qui réalisent ce cas d'utilisation. L'administrateur déclanche l'objet « :Interface » qui à son tour crée un objet « :statistique ». Ce dernier envoie un objet actif ProActive « :AgentCollecteur » pour la récolte d'information, puis les afficher à l'aide de l'objet «:JF_Statistiques ».

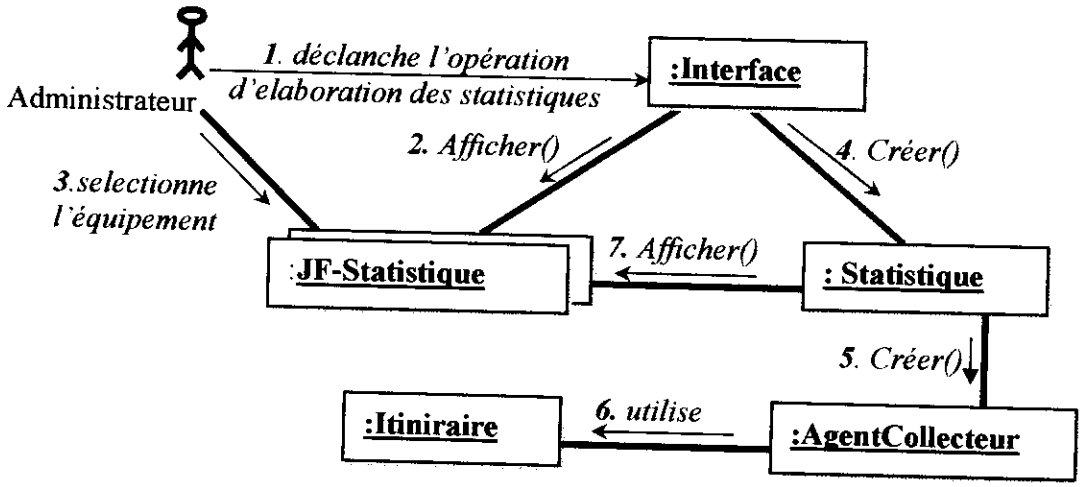


Fig VI.26 : Collaboration des objets «Elaboration des statistiques »

La figure suivante représente une ébauche de diagramme de classe de ce cas :

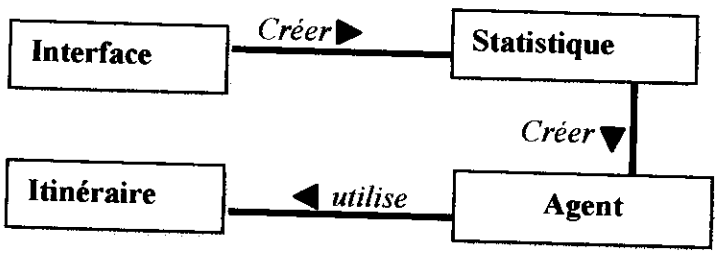


Fig VI.27 Ebauche de diagramme de classes «Elaboration des statistiques »

IV.4. Diagramme de classe

Après la description des grandes fonctions de notre système et l'identification des principales classes qui permettent leurs réalisations, nous allons maintenant élaborer un diagramme de classes global (voir figure VI.28). Ce diagramme n'est réellement que la synthèse des diagrammes de collaboration précédemment expliqués et d'autres. Il permet de définir les associations entre les différentes classes. Ces associations peuvent nous éclairer sur

le rôle d'un objet (instanciation de la classe) au sein du système ainsi que ses relations avec les autres objets.

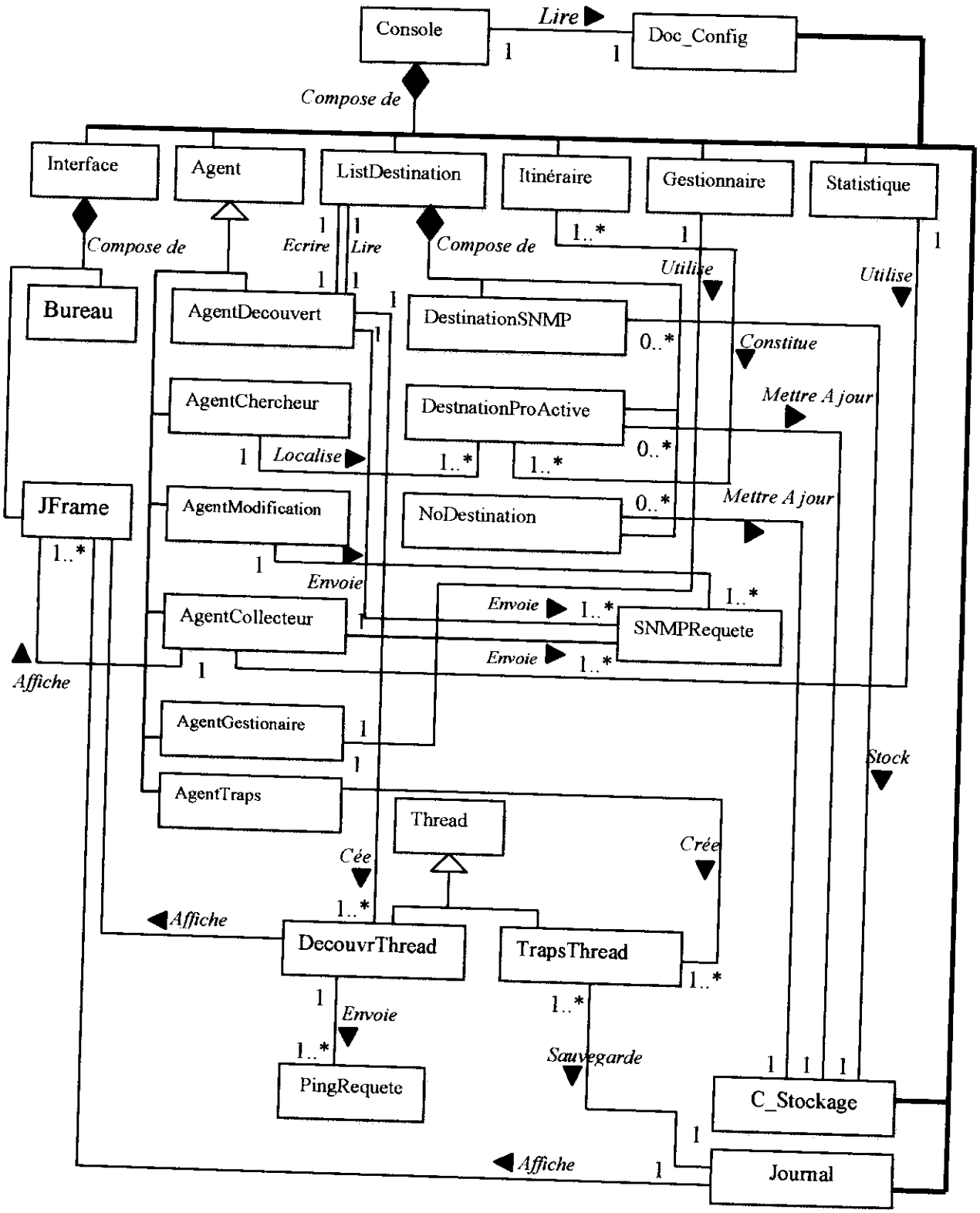


Fig VI.28 : Diagramme de classes final de la console d'administration

IV.5. La persistance

Les cinq principales raisons qui nous ont conduit à recourir au support de stockage de masse (disque dur) sont :

1. Restituer et afficher la topologie à chaque demande sans utiliser le réseau.
2. Connaître l'état d'un équipement existant c'est à dire actif ou éteint.
3. Détecter les nouveaux équipements connectés au réseau.
4. Détecter les changements des branchements dans le réseau lorsqu'une machine change de nœud de connexion (switch) ou même change le port dans le même switch.
5. Permettre une analyse de l'historique des événements.

Pour cela nous pouvons utiliser soit des simples fichiers, soit une base de données. Nous avons opté pour la deuxième solution afin de profiter des avantages qu'offrent les SGBDs (Systèmes de Gestion de Base de Données) tels que :

- ❖ Facilité de la recherche : c'est la cause principale, les simples requêtes SQL nous permettent d'extraire des données précises, dissimulées dans une énorme quantité de données.
- ❖ Gestion des transactions : les SGBDs offrent une meilleure gestion d'accès concurrentiel des utilisateurs pour la lecture et l'écriture (dans notre cas, chaque Agent est un utilisateur).
- ❖ Gestion d'intégrité des données : c'est le SGBD qui s'occupe de la gestion des liens de dépendance entre données, contrôle des valeurs autorisées des champs, etc.
- ❖ Facilité de la manipulation : les données sont bien structurées sous forme de champs et de tables, ce qui rend leurs utilisations plus faciles et efficaces.

Puisque le langage UML prend en compte la modélisation des données statiques, nous l'avons utilisé pour la modélisation de la base de données : (Fig VI.29)

Concernant l'implémentation, nous allons utiliser un SGBD relationnel. Cela nous ne pose pas un problème parce que le passage d'un diagramme de classes UML (classes comportant uniquement des attributs) vers un modèle logique de données (MLD) est possible.

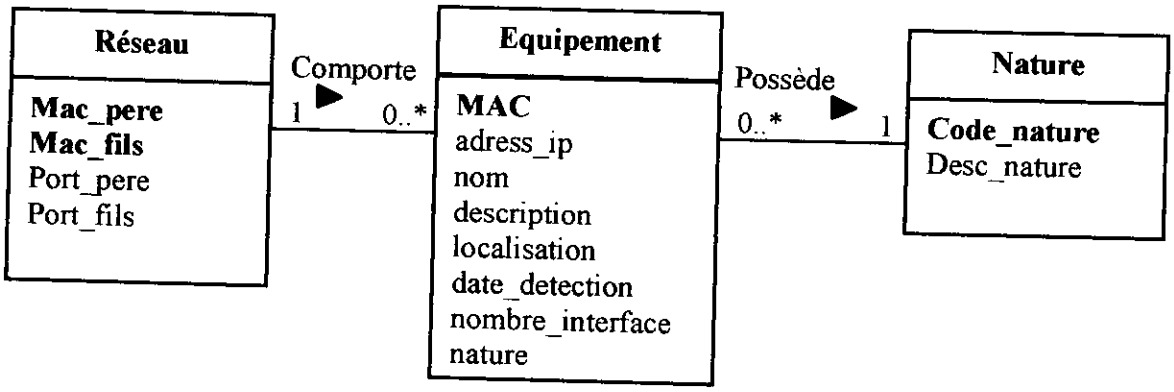


Fig VI.29 : Schéma conceptuel de la base de donnée

Pour la mise en ouvre de la base nous avons choisi le SGBD MySQL. Ce choix vient du fait que MySQL est un SGBDR multiutilisateurs, multitraitement, ce qui permet d'effectuer des connexions rapides et parallèles. MySQL est aussi portable ce qui lui permet de fonctionner sous différentes plates formes de systèmes d'exploitation. Avec des fonctions fortement optimisées et sa capacité d'utiliser des tables de différentes Bases de Données, MySQL prend en charge des Bases de Données de très grandes tailles. Avec la simplicité d'établir une connexion avec le langage Java en utilisant le JDBC (*Java DataBase Connectivity*), et en fin MySQL prend de la place dans les sites actuels parmi les grands SGBDR.

Remarques :

- Nous avons, lors de l'implémentation des tables, supprimé les clés étrangères car le SGBD MySQL ne prend pas en compte l'intégrité référentielle et nous avons inclus ces clés dans les clés primaires.
- La technologie JDBC (*Java DataBase Connectivity*) est un ensemble de classes permettant de développer des applications capables de se connecter à des serveurs de bases de données (SGBD).

Chapitre VI

Implémentation et tests

Dans ce chapitre :

- Environnement de développement
  - ProActive
  - Langage de programation
  - Environnement Java
  - Machine virtuelle Java
- Code d'administration
- Présentation et tests de la console d'administration

CHAPITRE VI: IMPLEMENTATION ET TESTS

INTRODUCTION

Dans ce chapitre nous allons mettre en œuvre l'objectif de notre conception, qui consiste à la mise en place d'une console d'administration réseau basée sur la technologie des agents mobiles. Pour cela nous allons commencer par une brève présentation de l'environnement de développement, en suite nous décrivons le code d'administration en spécifiant les squelettes de quelques agents et en fin nous terminons par une présentation de notre console.

I. ENVIRONNEMENT DE DEVELOPPEMENT

I.1. La plate forme ProActive

Comme nous avons déjà montré dans le chapitre IV, ProActive est une simple bibliothèque, qui utilise les outils standards (javac et JVM), la bibliothèque est elle-même écrite entièrement en Java, ce qui autorise l'utilisation de n'importe quelle plate-forme Java et des outils associés, en particulier le fonctionnement dans un environnement ouvert et dynamique avec télé-chargement dynamique de code ; ProActive utilise le mini-serveur http de la plate-forme Java pour transférer dynamiquement le *bytecode* nécessaire.

Installation de la plate forme ProActive

Proactive est disponible pour le téléchargement sous la licence de LGPL (Lesser General Public License), elle nécessite le JDK 1.5 ou plus (la version du JDK est importante) pour quelle puisse être installée sur votre poste.

Pour développer avec ProActive voici les étapes a suivre :

- Télécharger et décompresser le fichier ProActive [Pro_05]. Le tableau (Tab VII.1) résume des descriptions sur les répertoires et les fichiers contenant dans le répertoire principal.
- Inclure dans votre CLASSPATH les fichiers jar suivants : ProActive\ProActive.jar, ProActive\lib\asm.jar, ProActive\lib\log4j.jar, ProActive\lib\xercesImpl.jar, ProActive\lib\fractal.jar, ProActive\lib\bouncycastle.jar.
- N'oublier pas de lancer le JVM avec un fichier de sécurité (.policy). vous pouvez aussi spécifier un fichier de configuration log4j avec la propriété Dlog4j.configuration=file :pathToFile, si vous n'avez pas le spécifié un logger sera crée par défaut. Dans chaque poste, et pour le rendre un noued ProActive, il faut créer une

variable d'environnement JAVA_HOME pointe vers le jdk contenant les classes ProActive.

Exemple valable sous Windows: SET JAVA_HOME =C:\java\JDK1.5.

Repertoire ou Fichier	Description
ProActive.jar	Le byte code de ProActive pour l'inclure dans le CLASSPATH afin d'utiliser ProActive
ProActive_examples.jar	le byte code et les ressources de tous les exemples inclus avec ProActive
ic2d.jar	Le byte code et les ressources de IC2D la classe qui sera utilisée pour réaliser le gestionnaire des agents
Lib	La bibliothèque externe (external) utilisée par ProActive
Docs	La documentation de ProActive
scripts/unix	Unix *sh* les scripts pour exécuter les exemples sous Unix
scripts/windows	Windows *.bat* fichiers batch pour exécuter les exemples sous windows
Src	Le code source de ProActive pour la version source seulement
Compile	Le script pour compiler ProActive en utilisant Ant pour la version source seulement

Tab VII.1 : Le contenu du répertoire principal ProActive

I.2. Langage de programmation

Après une présentation de l'architecture d'administration des réseaux basée sur le paradigme d'agent mobile, et le choix de la plate forme ProActive qui est écrit entièrement en Java, on a choisi le langage de programmation Java, comme on a jugé qu'il est nécessaire de décrire Java en tant que machine virtuelle et support pour la mobilité.

I.3. Environnement Java

Java est un langage de programmation, l'abstraction d'un environnement homogène portée sur des plates-formes variées et des API riches et utiles pour la programmation :

- Java est un langage qui intègre plusieurs concepts : l'approche objet, le typage fort, la sécurité, les threads et les exceptions.
- Java est un environnement d'exécution qui s'appuie sur les progrès faits en matière de machine virtuelle depuis vingt-cinq ans pour concilier la portabilité du code avec le souci de performance.

➢ Java est un langage qui intègre des API utiles pour programmer les systèmes d'information (bases de données, texte, son, image, réseau, architectures distribuées).

I.3.1. Abstraction d'un environnement homogène

L'environnement Java fournit l'abstraction d'un environnement homogène grâce à :

➢ la spécification d'une machine unique, la *machine virtuelle Java (JVM)*

➢ et un format unique de fichier de code exécutable Java, le format de fichier *.class*.

La spécification d'une machine virtuelle Java unique et son implantation sur diverses plates-formes permettent de voir les diverses plates-formes comme une seule et même plate-forme. Ainsi, quels que soient les systèmes d'exploitation et les architectures de machines sous-jacentes, toutes les plates-formes auront l'apparence d'une même plateforme. D'autre part, le format de fichier de code exécutable Java, le fichier *.class*, est un format unique, portable et exécutable sur toute machine virtuelle Java. Ainsi, un code Java s'exécutant sur une plate-forme n'a pas besoin d'être porté ou recompilé pour être exécuté sur une autre plate-forme.

I.3.2. API de l'environnement Java

Un des aspects importants de l'environnement Java est la richesse de ses librairies de classes Java, accessibles via l'interface de programmation d'applications (API: Application Programming Interface). Cette API, illustrée par la figure (Fig V.1), propose divers outils pour faciliter la programmation de systèmes d'information variés et manipuler des bases de données, du texte, du son, des images, le réseau ou les architectures distribuées. Les librairies de classes Java sont organisées en plusieurs packages, dont le package *java.net* qui contient des outils concernant le réseau, le package *java.io* qui propose des outils de manipulation d'entrées/sorties, le package *java.applet* qui fournit une interface d'accès aux navigateurs web et le package *java.lang* qui contient les classes et interfaces les plus usuelles

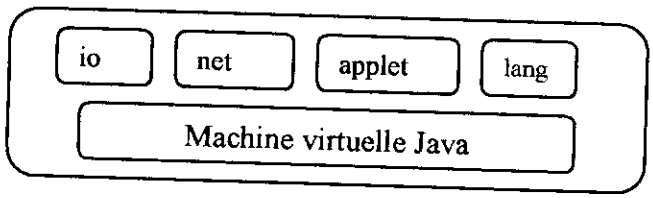


Fig V.1 : API de l'environnement Java

I.3.3. Java comme support pour la mobilité

Java a été conçu initialement pour les environnements et applications mobiles, qui se déplacent d'un site à un autre, à travers le réseau. C'est pour cette raison que l'API de Java

intègre divers outils pour la programmation d'applications mobiles : des mécanismes pour la mobilité du code Java et des mécanismes pour la mobilité des données Java. Les expressions « code Java » et « classe Java » seront utilisées indifféremment, de même que les expressions « donnée Java » et « objet Java ».

Mécanismes pour la mobilité du code Java

L'environnement Java fournit un mécanisme de *chargement dynamique de classes* Java, [Bou_01]. Le chargement d'une classe est l'opération qui consiste à construire, à partir du code exécutable d'une classe Java, les structures d'exécution qui décrivent cette classe dans la JVM. Le chargement de classes est dit dynamique car il peut être fait au cours de l'exécution des applications, lors de la première utilisation d'une classe. Ce mécanisme permet au programmeur d'une application de définir ses propres politiques de chargement de classes Java, en spécialisant :

- le format de la classe à charger (fichier *.class*, fichier ZIP, fichier compressé),
- la localisation source de la classe (système de fichiers, réseau),
- le moyen de localiser la classe à charger dans la localisation source.

Le mécanisme de chargement de classe est représenté par la classe *java.lang.ClassLoader*. Il peut servir à transférer du code Java à travers le réseau, tel qu'illustré par la figure (Fig V.2). La classe *java.net.URLClassLoader* est une sous-classe de la classe *java.lang.ClassLoader*, elle permet de charger des classes Java à partir de localisations identifiées par des URL.

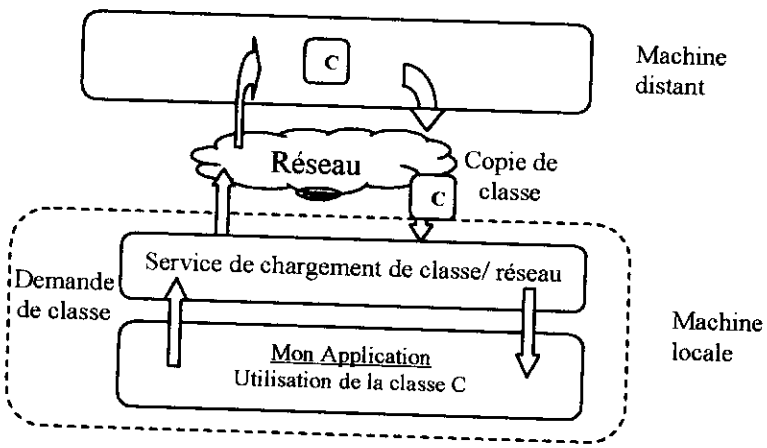


Fig V.2 : Chargement de classe à travers le réseau

Un autre mécanisme de mobilité des applications Java est l'*applet*, représentée par la classe *java.applet.Applet* .Une applet est un code Java auquel fait référence une page HTML d'un serveur web. La localisation du code de l'applet peut être le serveur web lui même ou un

autre site. Lorsqu'un client web accède à une page HTML contenant une applet, le code de l'applet est chargé à partir de la localisation de l'applet, pour être exécuté sur le site du client.

Mécanismes pour la mobilité des données Java

L'environnement Java fournit un mécanisme de *sérialisation/dé-sérialisation* des objets Java [Bou_01]. La *sérialisation* d'un objet Java consiste à convertir la structure de données décrivant un objet en un flot d'octets. La figure (Fig V.3) illustre une sérialisation d'objet Java. La *dé-sérialisation* est le processus de conversion de la forme sérialisée d'un objet (flot d'octets) en une copie de l'objet. Un objet Java est sérialisable si sa classe ou une de ses super-classes implante l'interface *java.io.Serializable*.

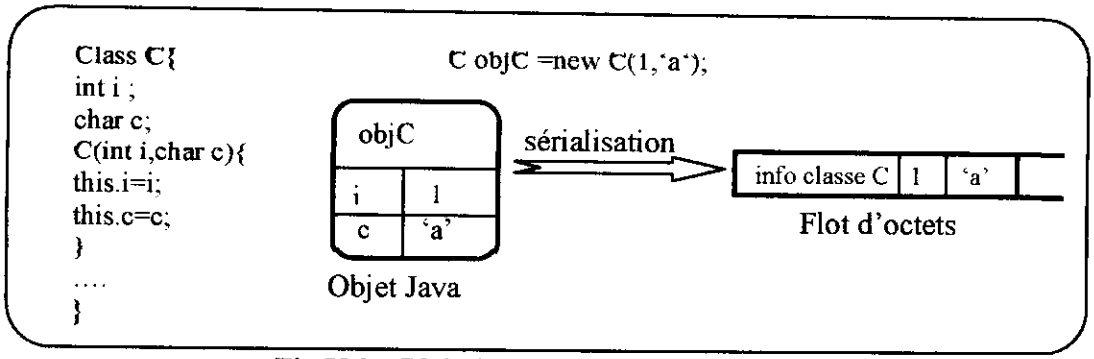


Fig V.3 : Sérialisation d'un objet Java

La figure (Fig.V.4) représente les différentes étapes de transfert d'un agent d'une machine à une autre et compris l'étape de sérialisation.

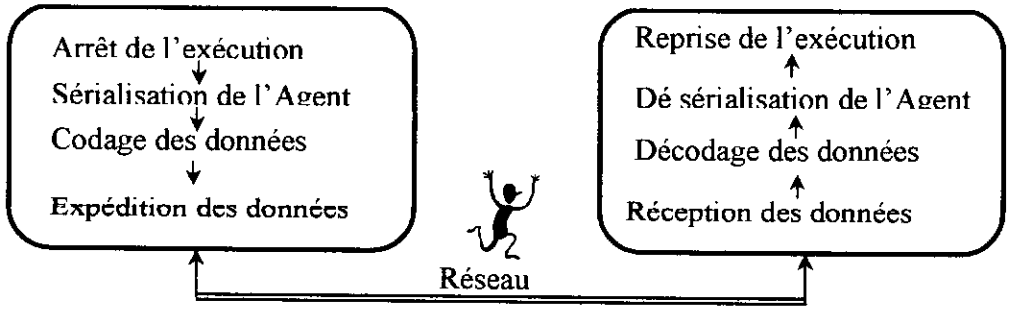


Fig V.4 : Transfert des agents

L'*externalisation* est une forme de sérialisation qui permet de contrôler plus finement les données écrites dans le flot d'octets. L'utilisation de l'externalisation au lieu de la sérialisation classique peut réduire le temps de sérialisation des objets Java jusqu'à 40% [bou_01]. Un objet Java est externalisable si sa classe ou une de ses super-classes implante l'interface *java.io.Externalizable*. La sérialisation/dé-sérialisation peut en particulier être

utilisée pour le transfert d'objets Java à travers le réseau. Ainsi, les données sérialisées sont envoyées d'une machine source vers une machine destination, à travers un flux réseau reliant ces deux machines.

I.3.4. Machine virtuelle Java

La machine virtuelle Java (JVM : *Java Virtual Machine*) est dite virtuelle car elle représente une machine abstraite définie par une spécification. Pour exécuter un programme Java, il est nécessaire d'avoir une mise en œuvre concrète de cette spécification abstraite. La machine virtuelle Java désigne ainsi une des trois entités suivantes :

- une spécification abstraite,
- une mise en œuvre concrète
- et une instance d'exécution.

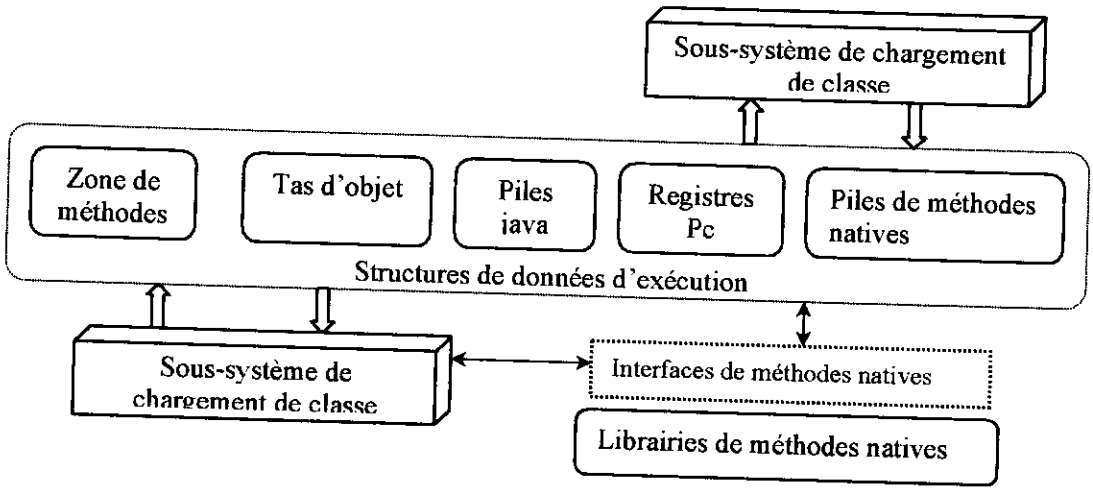


Fig V.5 : Architecture de la machine virtuelle Java

Sous-système de chargement de classe : Chaque machine virtuelle Java possède un *sous-système de chargement de classes* ou *chargeur de classes*. Ce sous-système a pour objectif de charger en mémoire, dans la zone de méthodes de la JVM, les classes et interfaces utilisées et décrites par des noms complets. Plusieurs chargeurs de classes peuvent cohabiter au sein d'une même JVM. Ainsi, le programmeur d'une application peut définir et utiliser ses propres chargeurs de classes.

Sous-système d'exécution : Chaque machine virtuelle Java possède un *sous-système d'exécution*, qui est le mécanisme responsable de l'exécution du bytecode contenu dans les méthodes des classes chargées. De même que la machine virtuelle Java, le sous-système d'exécution peut prendre une des trois formes suivantes : une spécification abstraite, une mise en œuvre concrète ou une instance d'exécution.

La spécification abstraite décrit le comportement d'un sous-système d'exécution en termes d'instructions de la JVM. La mise en œuvre concrète peut être une mise en œuvre logicielle, matérielle ou une combinaison des deux. Une mise en œuvre concrète d'un sous-système d'exécution peut, par exemple, être basée sur un interprète de bytecode, un compilateur à la volée, une exécution native ou une combinaison de ces différentes techniques. Enfin, une instance d'exécution d'un sous-système d'exécution est un thread Java.

II. CODE D'ADMINISTRATION

II.1. Les opérations de base et leur utilisation dans le code de l'agent

Nous utilisons pour programmer nos fonctions d'administration de réseau la bibliothèque SNMP fournie par AdventNet [Adv_05]. Il est possible d'utiliser l'API fournie par AdventNet afin de collecter les informations sur les agents SNMP.

Il suffit d'hériter du squelette de la classe Agent, en se focalisant sur les méthodes qui sont déclenchées automatiquement et qui implémentent le code fonctionnel : tout particulièrement, OnArrival et OnDeparture c'est-à-dire les méthodes qui sont déclenchées par le MigrationStrategyManager (classe ProActive permet la gestion des itinéraires). Dans le corps de la procédure OnArrival, on injecte le code d'administration ; c'est une instruction compréhensible par l'agent qui veut dire « lorsque vous arrivez sur le Nœud ProActive destination exécuter ce code ».

II.2. Linéarité du code

L'application simple d'administration de réseau que nous avons mis en place se compose essentiellement d'un « lanceur d'agent » et du code générique et linéaire des agents mobiles qui ont des tâches d'administrations différentes à exécuter.

II.2.1. Le code du lanceur

```
public static void main (String[] args)
/** Fonction principale */
{Agent Montest;
try {
Montest = (Agent)ProActive.newActive("Agents.Agent",null,null);
PrepareTravel(Montest);
try {
Thread.currentThread().sleep(10000); }
catch (Exception e) {e.printStackTrace();}
v=new Vector(Montest.getTrace());
for (int loop=0;loop<v.size();loop++) {
System.out.println(v.elementAt(loop)); }
System.exit(0);
}
catch (Exception e) {
System.err.println("Error : "+e.getMessage());
e.printStackTrace();
}
}
```

Fig VII.1 : Le code du lanceur d'agent

Ce code du lanceur se décompose en quatre parties, et est facilement programmable.

1. Déclaration d'une variable locale faisant référence à la classe de l'agent mobile.
2. Ici on rend le code de l'agent actif (ProActive.newActive). Les paramètres de création sont dans l'ordre : le nom de la classe à instancier, les paramètres à passer à la classe et l'URL du nœud ProActive sur lequel doit être créé l'agent.
3. La fonction PrepareTravel (Montest) n'est pas donnée ici, mais consiste uniquement à préparer l'itinéraire que doit suivre l'agent mobile. Un itinéraire est constitué en une séquence de paires : //Nom_machine/Nom_du_node_proActive, le nom de la méthode à exécuter en arrivant sur le nœud.
4. On attend suffisamment longtemps pour que l'agent mobile ait fini son itinéraire. Une autre solution consiste à interroger l'agent pour savoir s'il est revenu à son point de départ. La façon la plus générique est de lancer l'application en donnant comme paramètre le temps que doit attendre le « lanceur » avant de récupérer les résultats. Il serait aussi possible de bloquer « le lanceur » tant que l'agent mobile n'est pas revenu à son point de départ. Si, dans notre cas, le temps donné à la fonction « *sleep* » n'est pas assez grand, l'agent est tout de même interrogé quelque soit son emplacement courant, ce qui permet de récupérer que les informations recueillies sur les nœuds précédemment visités. Ainsi en quelques lignes de programmation il est facile de transformer le code d'une classe Java en

code mobile, quelque soit le travail que réalisera la classe. (Voir paragraphe II.7.3 du cinquième chapitre).

II.2.2. La classe générique de l'agent

Par soucis de clarté, les méthodes liées à ProActive sont encapsulées dans des méthodes propres à l'agent mobile. Cela ne détériore pas significativement les performances des méthodes de la plate-forme ProActive. Le code présenté ci-dessus, s'apparente à un code que l'on peut facilement qualifier de *générique* puisqu'il permet de décrire la structure principale du code d'un agent, c'est-à-dire ce qui est lié à la mobilité.

Il est important de respecter en fait quelques procédures simples avant de faire migrer le code de l'agent mobile comme par exemple définir son itinéraire. Cela est fait par la méthode *AgentInsertDestination*, qui informe l'agent sur l'itinéraire qu'il devra suivre. Une fois que l'itinéraire est complètement défini, les méthodes décrites ci-dessous sont exécutées pour diriger la migration de l'agent.

```
public class GenericAgent implements Active, Serializable
{
Itinerary myItinerary;
int TIME_SLICE = 1000;
Process myproc;
String Local=new String("Local -a");
Vector v=new Vector();
public void GenericAgent() {}
/**
Cette fonction pour qu'elle soit exécutée par l'agent lorsque il arrive*/
public void AgentOpenService()
{ //Maintenant,rien à faire , parceque c'est un Agent genirique }
/** Cette méthode pour faire une tache d' administration */
public void AgentExternal() { }
/** Cette méthode pour est faite pour obtenir le résultat de l' agent */
public Vector getTrace() { return new Vector(v); }
/** Cette méthode est utilisé pour obtenir la destination suivante de l'Agent */
public Destination getNextDestination()
{ try {
if (myItinerary == null) {myItinerary=ProActive.itineraryGetCurrent(); }
Destination r=this.myItinerary.getAndSetNextDestination();
return r;
} catch (Exception e) {e.printStackTrace();}
return null; }
```

```
/*Cette méthode pour convertir un noued de localisation vers un nom du hote */
private String NodeToString(String dest)
{ StringTokenizer st=new StringTokenizer(dest,"/");
String s=st.nextToken();
return s;
}
/** Cette méthode pour envoyer l'agent vers la destination choisie */
public void moveToNext()
{ Destination r = this.getNextDestination();
if ( r == null)
{ System.out.println("Erreur destination Null "); return; }
try {
ProActive.onArrival(r.getMethodName());
System.out.println("Agent.moveToNext : "+r.getNodeName());
System.out.println("And will execute on Arrival "+r.getMethodName());
ProActive.migrateTo(r.getNodeName());
}
catch (MigrationException e)
{ System.out.println("Migration Errror !"); }
catch (Exception e)
{ System.out.println("Error : No ProActive Node");
String s=this.NodeToString(r.getNodeName());
AnotherManagement(s);}
}
/** Le main de l' agent */
public void live (Body b)
{ while(b.isActive())
{ b.waitForRequest(TIME_SLICE);
if (b.requestLine.isEmpty()) { this.moveToNext(); }
else
{ b.service.serveOldest(); }
}
}
/** C'est une simple méthode pour construire un itinéraire */
public void AgentInsertDestination(String Destination, String Method)
{ try {
if (myItinerary == null) // On initialise le premier itineraire
{ myItinerary=ProActive.itineraryGetCurrent(); }
myItinerary.add(Destination,Method);
} catch (Exception e) { e.printStackTrace();}
}
}
```

Fig VII.2 : Classe générique de l'Agent

• La méthode *getNextDestination()* permet à l'agent de découvrir le prochain nœud ProActive qu'il devra visiter.

• La méthode *moveToNext()* permet tout d'abord de définir le comportement qu'aura l'agent lors de son arrivée sur le prochain nœud, et de déclencher la migration. Une erreur peut

survenir lors de la tentative de migration (nœud inexistant), et dans ce cas on force l'agent mobile à effectuer une administration réseau traditionnelle déportée (*AnotherManagement()*).

• La méthode *NodeToString(Node)* permet de récupérer le nom de l'hôte distant à administrer, ce qui est utile lorsqu'il a été impossible de trouver un nœud ProActive sur cet hôte.

• La méthode live *(Body)* est la méthode principale de l'agent. Elle permet de répondre à tous les appels de méthodes distants sur l'agent, et en plus dans notre cas d'agent générique, de continuer la progression conformément à l'itinéraire. Il est à noter que les appels de méthodes sont traités selon l'ordre FIFO par défaut (serveOldest(), mais que le modèle de ProActive permet de programmer n'importe quelle autre politique de service des requêtes, sachant que celles-ci sont stockées (requestLine) dès leur réception. Le paragraphe précédent présente le code du « lanceur » d'agent. Ce code simple rend le code de l'agent actif (*ProActive.newActive(..)*), dont un thread se charge de l'exécution de la méthode *live()* décrite ci-dessus. Cette méthode correspond à la vie de l'agent et tant que celui-ci est actif, deux types de comportement sont possibles :

• Soit la file des messages est vide, alors l'agent progresse dans son itinéraire,

• Soit l'agent a reçu un message et doit le traiter en fonction de la politique de gestion de file choisie.

L'agent sait alors détecter, dans ce cas, quelle est la méthode qu'il doit exécuter. Chaque méthode de l'agent peut être entièrement écrite indépendamment des autres méthodes (voir les méthodes de l'agent générique), car c'est la base du fonctionnement de ProActive. En effet, le programmeur n'a pas à se soucier de quelle façon va être appelée sa méthode pour l'écrire.

Exemple d'utilisation : Le recueil d'information sur les machines détectées

Le recueil d'information sur les machines du réseau détectées s'effectue par l'envoie d'un agent mobile qui va interroger localement l'agent SNMP (si l'équipement est un nœud ProActive) sinon il rapproche de l'équipement en question et l'interroge en mode client/serveur. Il faut noter ici que dans le cas des machines qui ne possèdent pas l'agent SNMP, on utilise le Netbios pour récupérer les informations nécessaires.

L'AgentCollecteurInfor de la figure VII.2 dialogue en SNMP V2 avec des équipements actifs et récupère quelques informations.

```
Class AgentCollecteurInfor extends GenericAgent implements java.io.Serializable {
private ItineraryManager itiManager;
public AgentCollecteurInfor () {}
// Quoi faire pour une destinationSNMP
public void secureV2OnArrival() {
//On récupère la destination SnmpV2 dans la variable snmpDest
super.OnArrival();
mgt.snmp.SystemDescr sysDescrV2= new mgt.snmp.SystemDescr();
sysDescrV2.setSystemDescr("v2",snmpDest.getUsername(),
snmpDest.getAuthPassword(),
snmpDest.getAuthProtocol(),snmpDest.getPrivPassword());
}
// Preparer l'itiniraire
public void prepareItinerary (ItineraryManager itiManager) {
// Preparer un ItineraryManager
this.itiManager=itiManager;
itiManager.prepareItinerary(itineraryServerHome);
}
}// fin de la classe AgentCollecteurInfor
```

Fig VII.3: Un Agent générique permettant de collecter quelques informations

II.3. La gestion des traps SNMP

Il est évident que lorsque l'on parle d'un système d'administration réseau, la console qui gère le réseau doit être en mesure de prendre en compte les événements qui surviennent sur le réseau. Plus précisément dans notre cas, la récupération des *traps* et le traitement de ces événements. La gestion et la compréhension de ces traps SNMP imposent un long travail afin de mettre à disposition dans la console d'administration réseau toutes les *OID SNMP* correspondant chacun à une trap d'un équipement. Toutefois, et à titre d'exemple, le code (figure VII.4) basé sur la gestion des traps SNMP de AdventNet, permet simplement de capturer la *trap* (fonction callback et d'instancier ensuite un agent mobile (figure VII.5) qui aurait pour travail sa gestion.

Dans notre exemple, l'agent mobile affiche la provenance de la *trap* SNMP, mais pourrait tout aussi bien déclencher un ou plusieurs autres agents mobiles pour effectuer une tâche d'administration liée à cet événement. Pendant la phase de traitement de cette *trap* par l'agent mobile, si un autre événement survient, alors les données relatives à cette *trap* peuvent être transmises à l'agent mobile. En effet, le système de communication de ProActive permet d'échanger des messages avec un agent mobile pendant son processus de migration, quelque soit l'emplacement de celui-ci. Par un tel système, l'agent mobile responsable de la gestion des *traps* pourra corréler les différents événements si nécessaire.

```
public boolean callback(SnmpSession session,SnmpPDU pdu, int requestID){
// Verifier la version
if(pdu == null) return false;
if (pdu.getCommand() == api.TRP_REQ_MSG) {
System.out.println("Trap recu de: "+pdu.getAddress() +", community: " +
pdu.getCommunity());
System.out.println("Enterprise: " + pdu.getEnterprise());
System.out.println("Agent: " + pdu.getAgentAddress());
System.out.println("TRAP_TYPE: " + pdu.getTrapType());
System.out.println("NUMIRO: " + pdu.getSpecificType());
System.out.println("Temps: " + pdu.getUpTime()+"\nVARBINDS:");
for (Enumeration e = pdu.getVariableBindings().elements();e.hasMoreElements();)
System.out.println(((SnmpVarBind) e.nextElement()).toTagString());
}
else
System.err.println("Pas de trap recu!!.");
try {
//creer un Agent mobile et l'envyer pour voir qu'est ce qui se passe
AgentTrap agtrap = (AgentTrap) ProActive.newActive("AgentTrap",null);
agtrap.setTrapFrom(pdu.getAddress().toString(),
pdu.getCommunity().toString(), pdu.getEnterprise().toString());
agtrap.check();
} catch (Exception e) {e.printStackTrace();}
return true; // API CallBack de AdventNet
}
```

Fig VII.4 : Instanciation d'un agent mobile de gestion de traps

```
public class AgentTrap extends Agent implements java.io.Serializable {
ArrayList nodeRessource = new ArrayList();
ArrayList arpTable = new ArrayList();
String ipAddress, community, trapOID;
public AgentTrap() {
super();
}
// une methode appelé par un " trap listener"
public void setTrapFrom(String ipAddress, String community, String trapOID) {
this.ipAddress=ipAddress;
this.community=community;
this.trapOID=trapOID;
}
// aller et verifier
public void check() {
System.out.println(" trap recu de "+ipAddress);
}
}
```

Fig VII.5: Un agent générique permettant de gérer une trap SNMP

III. PRESENTATION ET TESTS DE LA CONSOLE

Introduction

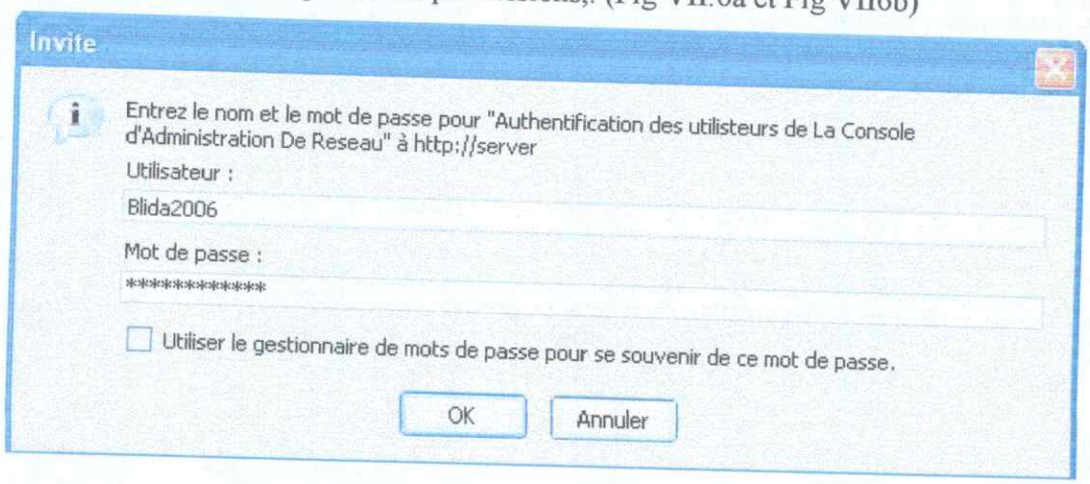
Les tests de l'agent mobile ont été faits sur différentes plates-formes afin de pouvoir prouver qu'il s'agit effectivement d'un système multi plates-formes et indépendant du système d'exploitation hôte (Linux, Windows NT, Win9x, WinXP).

Plusieurs combinaisons ont été choisies afin de vérifier la robustesse du déplacement de l'agent et notamment lorsqu'il doit rencontrer sur son itinéraire une plate-forme ne supportant pas de nœud ProActive.

III.1. Présentation générale

La console d'administration que nous avons conçu est un outil de gestion des équipements réseau local et étendue, elle utilise la technologie des agents mobiles ainsi les protocoles de gestion SNMP, WMI (Windows Management Instrumentation), http/TCP. Donc il faut installer sur chacun des postes devant recevoir la « visite » d'un agent mobile, le **Jdk** (Java developer kit) fourni par Sun Microsystem (java.sun.com) et les classes ProActive. En effet, ces classes permettent de gérer la migration d'agents et sont donc nécessaires pour le développement des agents mobiles. Elles assurent la création des nœuds, le départ, l'arrivée et le transfert des messages qui doivent suivre les agents mobiles. Il est nécessaire aussi que n'importe quelle machine appartenant au réseau doive avoir un agent SNMP intégré en elle.

L'accès à l'application sera via un serveur WEB, chaque utilisateur de la application doit introduire son Login et Mot de passe afin d'accéder au menu principal, pour cela on a intégré dans notre application les sessions car, l'administrateur peut ajouter supprimer des utilisateurs et définir à chaque un ses permissions,. (Fig VII.6a et Fig VII6b)



FigVII.6a Ouverture session par le serveur http Mozila.

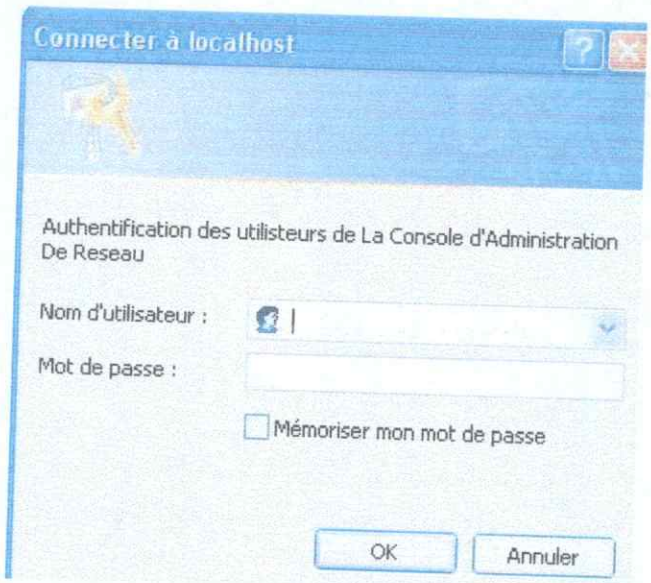
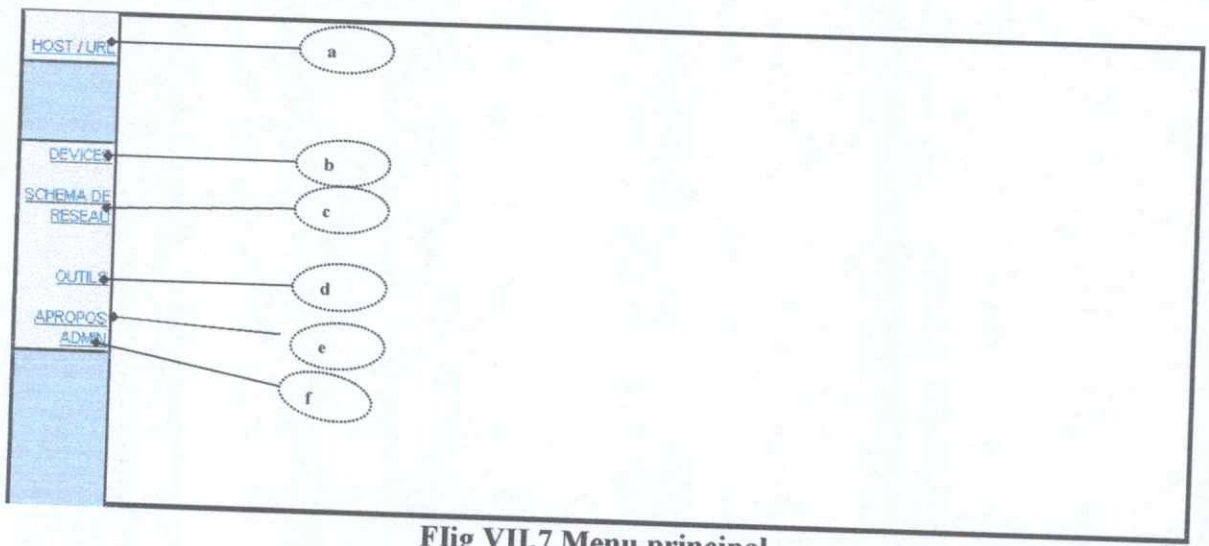


Fig VII.6b Ouverture de session serveur http Internet explorer

III .1.1. L'interface principale

L'interface principale de la application d'administration est sous forme d'une page HTML composé en deux modules : la barre d'outils et la zone d'affichage, comme la montre la figure (Fig VII.7).



FIig VII.7 Menu principal.

La barre d'outils : comporte un ensemble des boutons facilitant l'accès aux différentes commandes de l'application.

Interroger une station dans le réseau.

a. Découverte de réseau.

b. Afficher la topologie cartographique du réseau.
c. Les outils d'administration de réseau.
d. A propos.
e. Configuration de l'application.

1. **Zone d'affichage** : cette zone permet l'affichage des différentes fenêtres de l'application.

III.2. Présentation détaillée :

Les fonctions principales de notre application sont :

- Périphériques et leurs caractéristiques.
- Gestion et Supervision du réseau.
- Explorer la Mib Snmp et le Wmi.
- Gestion des utilisateurs.
- Pilotage et interogation des agents mobiles.

III.2.1. Périphérique : Il existe plusieurs façon de supervision les périphérique. On utilise DEVICE pour explorer tous les périphériques et les groups par ouverture de dossier ALL dans le group qu'on veut l'explorer. Clicker sur pour mettre à jour le périphérique ou son group de propriété. (Fig VII.8).

Explorer		Nom	Location	Alert
Tous				Info
localhost				Normal
sam		192.168.0.10		Normal
server		192.168.0.1		Normal
Routers				Info
Switches				Info
Hosts				Normal
localhost				Normal
sam		192.168.0.10		Normal
server		192.168.0.1		Normal
Internet				Info
Reseaux				Info
127.0.0.0				Info
192.168.112.0				Info
192.168.196.0				Info

Fig VII.8 Liste des équipements.

Pour afficher un nouveau matériel, clicker sur *HOST/URL*. Cette fonction affiche la fenêtre ci-dessous. Entrer le nom de la hôte ou l'adresse IP et le service snmp, La communauté par défaut est *Public*. Pour l'URL ou autre service réponse teste sélectionner *http* ou autre service.

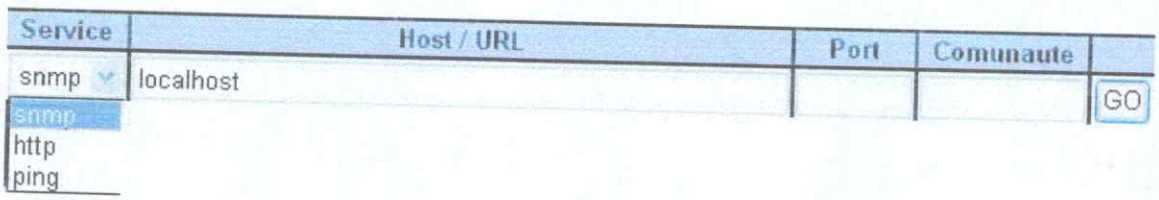


Fig VII.9 Host/Url.

On a utilisé le standard RFC1213-MIB-II afin de visualiser tous les interfaces. Pour plus d'information clicker sur un bouton ou une interface de la figure VII.10.

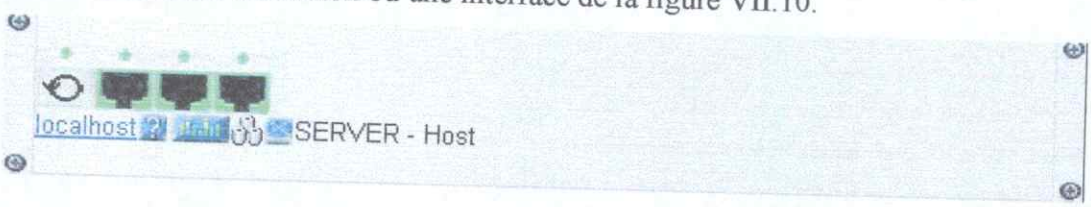


Fig VII.10 Interface de périphérique.

L'interface de périphérique comporte des indicateurs sur les cartes réseaux, par exemple le Symbole  indique qu'il existe une connexion réseau et celui-là  que le connecteur n'est pas actif, le click sur ces dernier affiche la figure (Fig VII.12) qui visualise la vitesse de connexion, adresse IP et le masque.

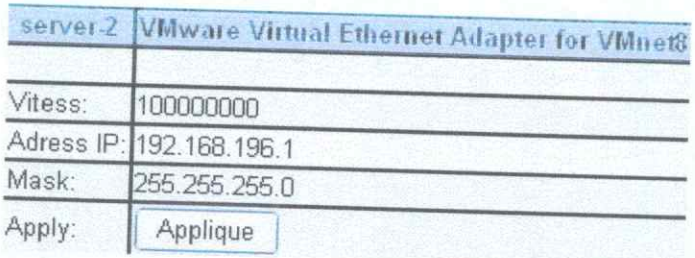


Fig VII.11 information sur interface réseau.

Pour savoir des informations concernant nom de système, description et temps de mis en marche sur une périphérique on click sur . (Fig VII.12).

server	
Description:	Hardware: x86 Family 15 Model 2 Stepping 9 AT/AT COMPATIBLE - Software: Windows 2000 Version 5.1 (Build 2600 Uniprocessor Free)
Temps De Mis En Marche:	0 days, 03:13:39
Nom:	SERVER
Apply:	Applique

Fig VII.12 description de périphérique

La fenêtre ci-dessous montre la table de routage.

Reseau	Mask	Hop Suivant	Metric	Age	Protocol	Interface
127.0.0.0	255.0.0.0	127.0.0.1	1	12248	Local	-
192.168.0.0	255.255.255.0	192.168.0.1	20	12248	Local	-
192.168.0.1	255.255.255.255	127.0.0.1	20	12248	Local	-
192.168.0.255	255.255.255.255	192.168.0.1	20	12248	Local	-
192.168.112.0	255.255.255.0	192.168.112.1	20	12248	Local	-
192.168.112.1	255.255.255.255	127.0.0.1	20	12248	Local	-
192.168.112.255	255.255.255.255	192.168.112.1	20	12248	Local	-
192.168.196.0	255.255.255.0	192.168.196.1	20	12248	Local	-
192.168.196.1	255.255.255.255	127.0.0.1	20	12248	Local	-
192.168.196.255	255.255.255.255	192.168.196.1	20	12248	Local	-
224.0.0.0	240.0.0.0	192.168.0.1	20	12248	Local	-
255.255.255.255	255.255.255.255	192.168.0.1	1	12248	Local	-

Fig VII.1.3 Table de routage.

L'administrateur peut ajouter, supprimer et modifier des propriétés de périphériques par l'appuie sur . Les changements effectués seront stocké uniquement dans la base de données de l'application non au matériel. (Fig VII.13)

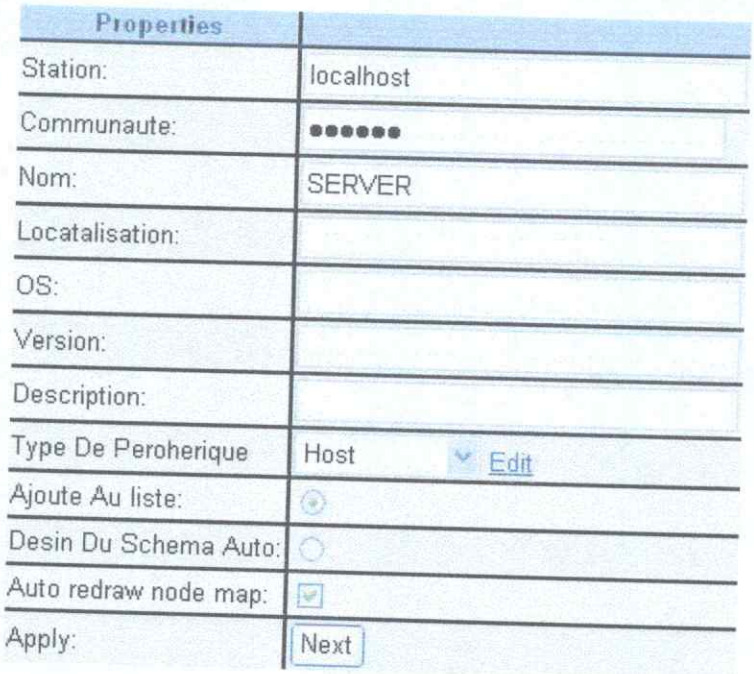


Fig VII.13 Mis-à-jour propriété des équipements.

On a introduit un service de réponse qui calcule le temps d'interrogation de périphérique, en spécifiant le type d'interrogation (Ping, http) et le nom du hôte ou l'adresse IP. Si on sélectionne le service PING on aura cette fenêtre présente le délai de réponse du requête PING.

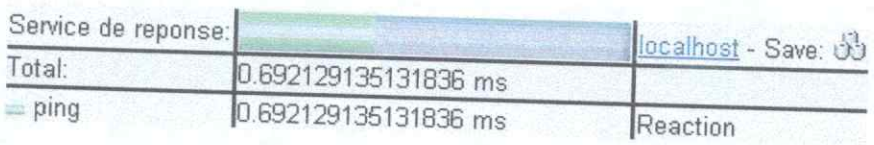


Fig VII.14 Ping.

Le service de réponse des stations permet de visualiser les résultats http d'une façon détaillé. Le temps de réponse de réseau et le DNS est mesuré Mseconds. En utilisant cet outil, Il est possible détecter certains problèmes lient au réseau ou au DNS ainsi que la diminution de performances. (Fig. VII.15).

Service de reponse:		localhost - Save:
Total:	198.58 ms	
DNS	169.135 ms	Temps pour recuperer le nom de la host apartir de l'adresse ip: 127.0.0.1
Reseaux	13.091 ms	Temps de teste Reseau
Application	14.298 ms	Reaction de l'application
Chercher	2.056 ms	Temps pour chercher une information

Content:

```
HTTP/1.1 200 OK
Date: Tue, 19 Dec 2006 17:23:18 GMT
Server: Apache
Last-Modified: Fri, 11 Mar 2005 10:04:16 GMT
ETag: "5bd0-7d-af5d8800"
```

Fig VII.15 HTTP.

III.2.2. Détection et schématisation du réseau

Détection du réseau : cette opération permet de détecter les périphériques existants dans le réseau par les agents mobiles, la recherche se fait de deux maniéres :

Détection de tout le réseau en spécifiant les nœuds maximum. (Fig VII.16)

Scanner Le Reseau	Adresse IP	
passrel par defaut:	localhost	
noeud maximum a scan:	512	
change la comunaute SNMP:		
comunaute SNMP par defaut:	public	SCANNER

Fig VII.16 Détection de réseau.

Lorsque l'agent mobile termine la recherche des équipements de réseau il mit à jours la base de données puis.

Détection des équipements de réseau par plage d'adresse (adresse début, adresse fin). (Fig VII.17)

Scan IP	IP-Address	
Debut de la plage IP:	192.168.0.1	
Fin de la plage IP:	192.168.0.10	
Change la comunaute SNMP:		
comunaute SNMP par defaut:	public	SCANNER

Fig VII.17 Détection des équipements par plage d'adresse.

1. **Schématisation de réseau :** Cette opération permet de d'afficher machines existantes dans le réseau en group selon leurs type, La fenêtre de la figures (Fig VII.18) indique tous les types des équipements, les types existants sont colores en vert les autres en bleu.

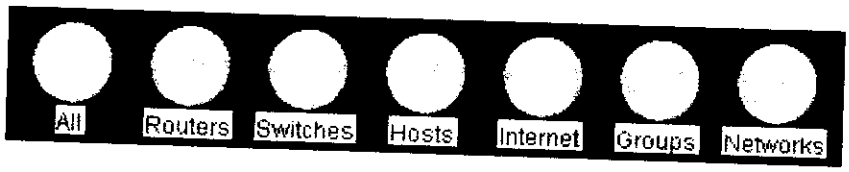


Fig VII.19 Vue globale des équipements.

Pour afficher le schéma de réseau on clicke sur le bouton *Networks*, les équipements de réseau paraient en 3D et on peut les déplacer dans la carte. (Fig VII.19)

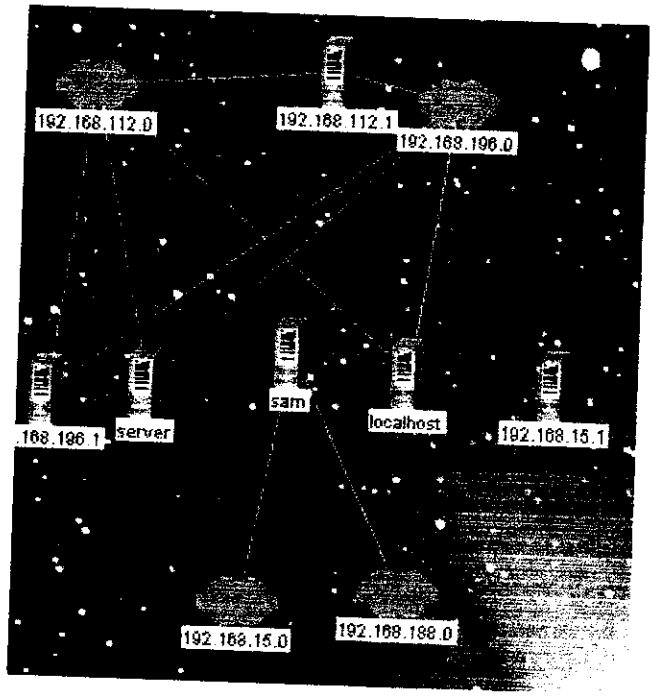


Fig VII.19 Topologie cartographique du réseau.

II. 2.3. Exploration de la Mib Snmp et WMI : notre application permet à l'administrateur réseau d'explorer et superviser tout périphérique contenant le SNMP ou WMI.

- Le WMI : le WMI (Windows management Instrumentation) est un protocole propre au système Windows, l'état de chaque périphérique, les interfaces physique et logique ainsi que Les importants systèmes statiques comme le CPU, mémoire, support de stockage et processus peuvent être visualisé. (FIG VII.20)

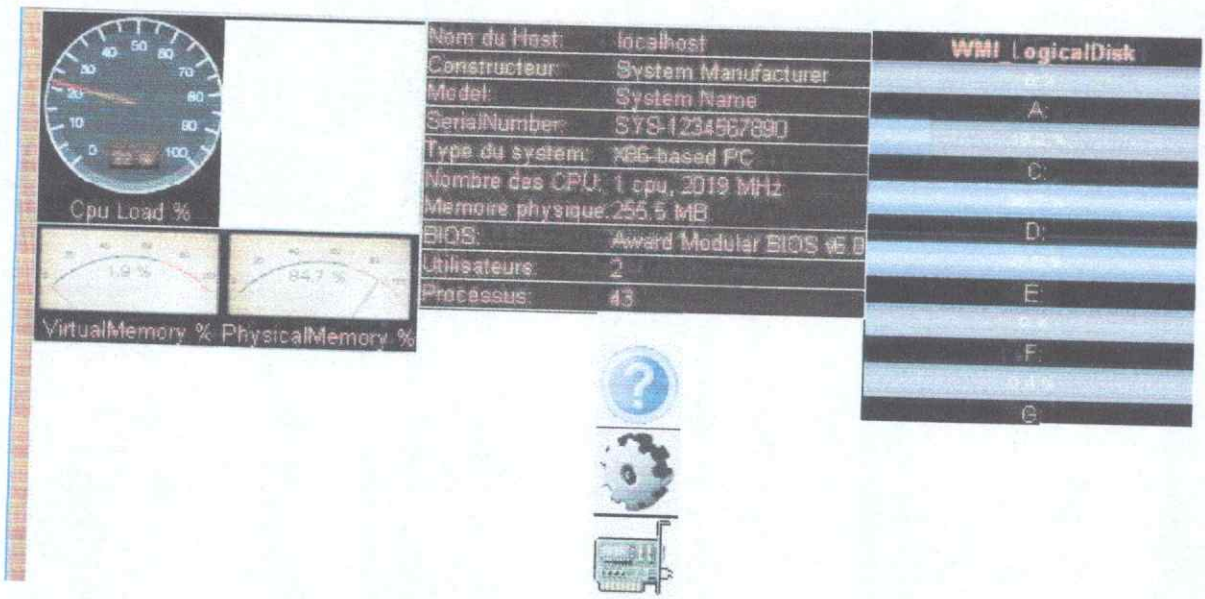


Fig VII.20 utilisation de protocole WMI.

De même la fenêtre de la figure (Fig VII.21) illustre la supervision des processus actifs avec leurs consommations des ressources.

Temps de repons/S	Memoire	PID	Nom	Long name	Type
		736	explorer.exe	C:\WINDOWS\	Applicatic
		892	csrss.exe	C:\WINDOWS\system32\ ObjectDirectory=\Windows SharedSection=1024,3072,512 Windows=On SubSystemType=Windows ServerDll=basesrv,1 ServerDll=winsrv:UserS	Applicatic
		3468	IEXPLORE.EXE	C:\Program Files\Internet Explorer\	Applicatic
		1996	svchost.exe	C:\WINDOWS\system32\ -k NetworkService	Applicatic
		4	System	System	Operating System
		576	IEXPLORE.EXE	C:\Program Files\Internet Explorer\	Applicatic
		2588	perl.exe	perl.exe C:\ASR\nino\mod_perl\status.pl	Applicatic
		1432	vmware-authd.exe	C:\Program Files\VMware\VMware Workstation\	Applicatic
		340	Notepade.exe	C:\notepade\ "C:\ASR\nino\mod_perl\nino.pl"	Applicatic
		1076	perl.exe	perl.exe C:\ASR\nino\mod_perl\eventaction.pl	Applicatic
		4028	perl.exe	perl.exe C:\ASR\nino\mod_perl\monitor.pl fast	Applicatic
		1596	avp.exe	avp.exe	Applicatic
		516	mspaint.exe	C:\WINDOWS\system32\	Applicatic
		3556	perl.exe	perl.exe C:\ASR\nino\mod_perl\monitor.pl default	Applicatic
		1936	svchost.exe	C:\WINDOWS\System32\ -k netsvcs	Applicatic
		1880	Apache.exe	C:\ASR\apache\bin\ -k runservice	Applicatic
		608	Apache.exe	C:\ASR\apache\bin\ -d C:/ASR/Apache	Applicatic
		2632	perl.exe	perl.exe C:\ASR\nino\mod_perl\monitor.pl slow	Applicatic
		1684	mysqld-nt.exe	C:\ASR\mysql\bin\	Applicatic
		3460	mmc.exe	C:\WINDOWS\system32\ /s C:\WINDOWS\system32\compmgmt.msc	Applicatic

Fig VII.21 Liste des processus actif sur la station.

L'explorateur de Mib : Nous avons implémenté un agent mobile *AgentTrap* qui a comme tâche la réception et l'analyse des traps envoyés par les déférents éléments de réseau, et les stocke dans la base de données. Donc, l'exploration de la Mib SNMP sera facile. (Fig VII.22)

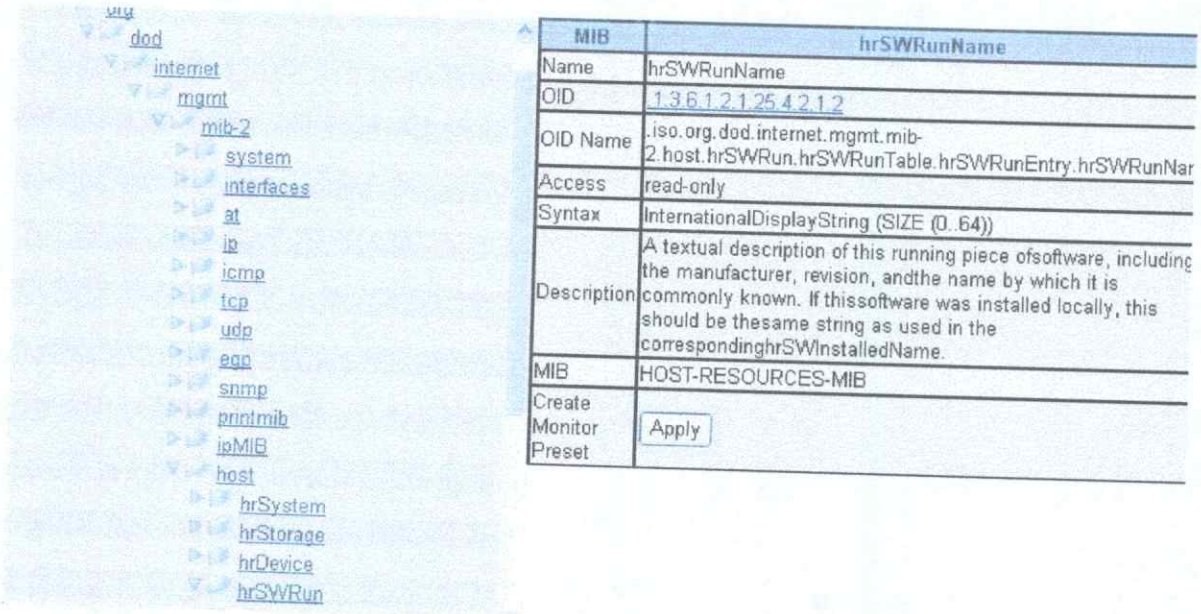


Fig VII.22 Explorateur de la MIB.

L'administrateur peut configurer la Mib afin d'afficher que les informations utiles. Il choisit uniquement les informations à afficher.

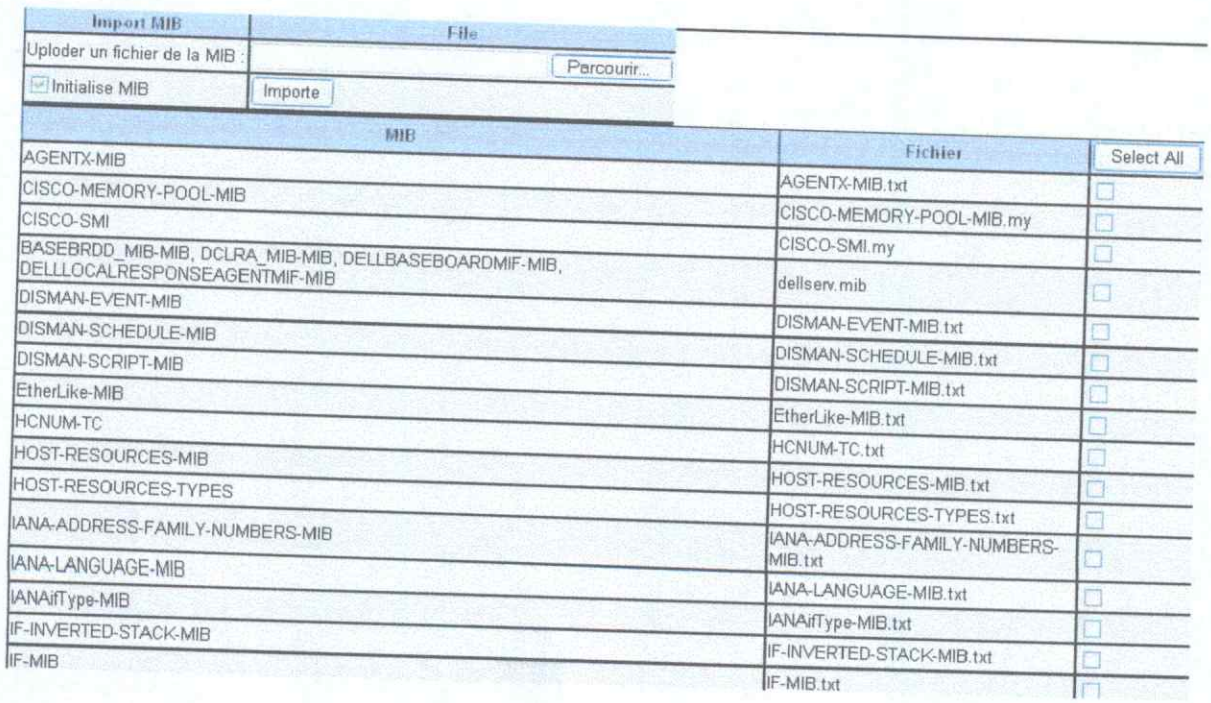


Fig VII.23 configuration de laMIB.

II.2.4. La gestion des utilisateurs et les sessions : l'accès à l'application sera via un serveur WEB, chaque utilisateur de la application doit introduire son Login et Mot de passe afin d'accéder au menu principal, pour cela on a intégré dans notre application les sessions car l'administrateur peut ajouter supprimer des utilisateurs et définir à chaque un ses permissions.

On click sur *admin* puis on sélectionne *Sécurité* la fenêtre de la figure (Fig VII.24) on trois choix *utilisateurs, groups, fonction et configuration.*

Utilisateurs	Ajout/Suprime des utilisateurs
Nouveau utilisateur	
Modifier	Cocher pour suprimer
naoufel	☐
samir	☐
nonux	☐
blida	☐
user1	☐
Apply:	Suprime

Fig VII.24 Ajout/Supprime des utilisateurs.

III.2.5. Pilotage et interrogation des agents mobiles

Nous avons utilisé les modules offerts par le service IC2D de la plateforme ProActive (livré en standard avec ProActive) pour la mise en œuvre du gestionnaire d'agent mobile.

Le gestionnaire des agents permet la visualisation de n'import quel agent de la plate forme ProActive sur n'import quel Nœud appartenant à une VM (Virtual Machine) dans un post quelconque du réseau, autrement dit ; il permet de contrôler et d'interroger tous les agents ProActive circulants sur le réseau.

Ce gestionnaire permet aussi de créer sur n'import quel nœud un agent ProActive, de le faire migrer manuellement vers un autre nœud d'une autre VM (déplacer le par la souris…).

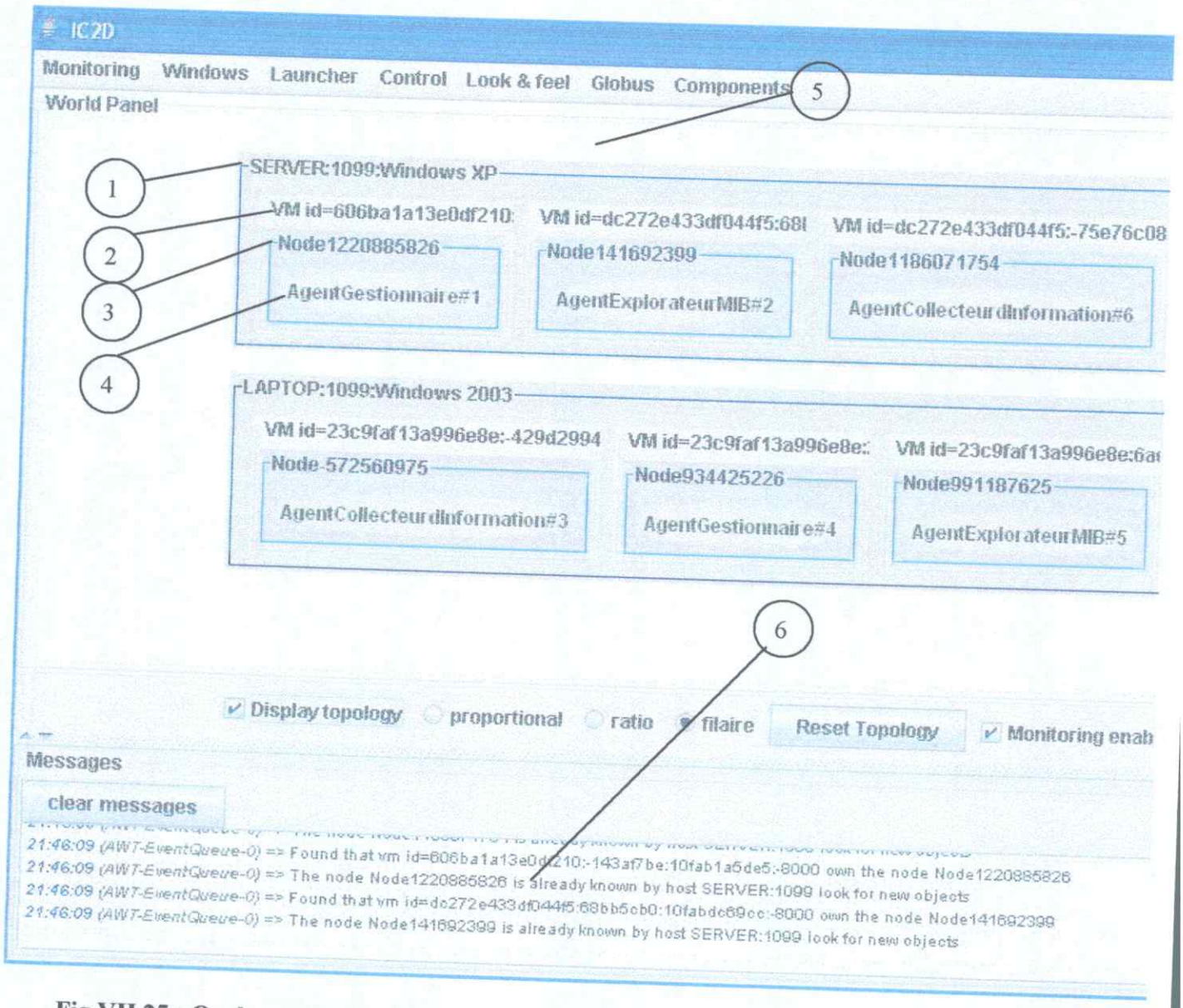


Fig VII.25 : Quelques services de la console et des agents mobiles d'administration vus par IC2D

1) Le nom de l'hôte avec le port et le système d'exploitation.
2) Le nom du VM et son identificateur unique.
3) Le nom du nœud ProActive.
4) Le nom de l'agent ProActive.
5) Zone pour visualiser les hôtes et les nœuds gérés.
6) Zone pour l'affichage des messages qui sont envoyés par d'autres noeuds et d'autres objets actifs.

Remarque : La recherche des nœuds ProActive se fonde sur RMI pour localiser un rmiregistry sur un hôte distant et s'il existe lui demander la liste des nœuds ProActive

enregistrés. Si l'hôte distant est protégé par un pare-feu, on subit le timeout RMI qui peut être long.

CONCLUSION

L'utilisation des agents mobiles dans un cadre d'administration réseau se justifie, mais sous certaines conditions. Une limitation de l'utilisation des agents mobiles est toutefois à faire, car il n'est pas forcément toujours rentable d'effectuer une administration par agent mobile plutôt qu'une administration traditionnelle. En effet dans le cas où la station d'administration ne désire obtenir que le trafic réseau instantané d'un ensemble d'équipements réseaux, la solution à agent mobile devient trop coûteuse en temps et en débit réseau. On retrouve ici l'avantage du protocole SNMP. Toutefois le gain par rapport à la solution d'administration SNMP devient intéressant lorsque le nombre de variables de la Mib SNMP et le nombre d'hôtes à visiter est grand. Autrement dit, la quantité d'information d'administration manipulée par une fonction d'administration est déterminante pour le choix de technologie de mise en oeuvre. Cette quantité dépend du nombre d'éléments de réseau concernés par la fonction d'administration, du nombre d'échantillons collectés, du nombre de variables concernées. La technologie client/serveur est bien adaptée à une collecte ponctuelle sur une courte durée tandis que la technologie d'agent mobile permet de gérer efficacement la collecte d'informations sur un élément de réseau pendant une longue période de temps ou pour une fréquence d'échantillonnage élevée.

Lorsque les conditions de l'utilisation de la technologie d'agent mobile sont vérifiées, elle devient bien adaptée pour l'exécution des fonctions d'administration du réseau. Cette technologie permet en effet de minimiser la bande passante utilisée et de réduire la durée de la connexion. Dans ce sens, elle est source d'économies.

Conclusion générale et perspectives

CONCLUSION GENERALE

Dans ce mémoire, nous nous sommes intéressés à la problématique de l'utilisation des agents mobiles dans un cadre d'administration réseau, en utilisant la plate-forme ProActive.

L'étude théorique des agents mobiles dans l'administration des réseaux nous a permis de découvrir un domaine de recherche très intéressant.

Dans ce mémoire nous avons montré qu'une bonne gestion des réseaux est devenue une nécessité au sein des entreprises surtout avec l'augmentation de la taille des réseaux d'aujourd'hui et la diversité des équipements utilisés. Récemment la technologie d'agent mobile a suscité beaucoup d'intérêt. Par conséquent, le développement de systèmes d'exécution d'agents mobiles est actuellement en plein essor. Contrairement aux agents d'administration standard qui réalisent des fonctions d'administration minimales, les agents mobiles sont envoyés sur des éléments de réseaux pour y effectuer certaines fonctions d'administration. L'intelligence déportée sur ces éléments de réseau permettrait de réduire encore plus les communications avec la station d'administration centrale.

En premier lieu nous avons donné un aperçu général sur l'administration des réseaux dont lequel nous avons évoqué les concepts de base pour comprendre le métier d'un administrateur réseau.

Nous avons ensuite étudié les systèmes d'administration des réseaux à savoir les systèmes centralisés, hiérarchiques et les systèmes d'administration fondés sur les technologies Web, et on a présenté les insuffisances de ces trois architectures, et qu'il est nécessaire d'envisager un autre système basé sur le paradigme d'agent mobile.

Dans la dernière partie nous avons présenté la conception de la console d'administration et les différents résultats de l'implémentation.

Finalement, il ne faut pas penser que l'intelligence et l'autonomie d'un agent mobile sont capables d'égaler celles d'un agent humain, dont la structure est encore trop peu connue pour qu'on puisse le répliquer.

PERSPECTIVES

Les perspectives de notre projet sont très ouvertes et motivantes en même temps, car les recherches dans le domaine des agents et de la mobilité ne cessent de se développer. Elle est toutefois la propriété des développeurs et centre de recherche qui travaillent souvent de façon autonome, Aussi, ne serait-il pas judicieux d'associer toutes les connaissances sur le sujet (quelque soit le pays et le domaine de recherche), afin de pouvoir obtenir des outils probants et fiables.

Pour les futurs travaux, on peut envisager un serveur LDAP (Lightweight Directory Access Protocol) à part entière pour fournir aux agents d'administration les informations nécessaires concernant l'itinéraire à suivre. Un système d'administration de réseaux accessible à l'administrateur quelque soit l'endroit ou il se trouve, c'est à dire quelque soit le poste de travail qu'il utilise et quelque soit le réseau employé pour la connexion de son poste de travail au réseau administré. Autrement dit un gestionnaire d'agents à part (n'est pas lié à la SAR), pour alléger la SAR, et pour que la console soit accessible de n'import quel nœud du réseau en utilisant la technologie Web.

L'intégration dans la console d'autres outils pour assurer les autres domaines de la gestion des réseaux tel que la gestion de la sécurité, et qui prend en considération la topologie virtuelle du réseau (VLAN).

BIBLIOGRAPHIE

Livres :

[Guy_98] : GUY PUJOLLE «Les réseaux», 2eme édition, Eyrolles, 1998.

[Bou_01] : Sara BOUCHENAK, these, Mobilité et Persistance des Applications dans l'Environnement Java, 2001.

[Biz_97] : Bizien Vincent et Fontaine Jean-Guilllaume « La gestion de réseaux SNMP et CMIS/CMIP » année 97 EPITA.

[Del_96] : Delobelle Alexis et Demaison Ludovic« La gestion des réseaux informatiques » année 96 EPITA.

[Ric_01] : W.Richard Stevens «TCP/IP illustre protocoles» volume 1 Vuibert édition 2001.

[Akh_99] : Akhil Sahai « Conception et réalisation d'un gestionnaire mobile de réseaux fondée sur la technologie d'agent mobile» année 1999.

[Cha_99] : Charlemagne Gomez, Florent Ledru et Franck Tegnet « gestion de réseaux et protocoles » année 99 EPITA.

[Irs_05] : Les agents intelligents sur internet, IRESTE DUTIL 10

[Gra_98] : G. BOOCH, J. RUMBAUGH, I. JACOBSON, Le guide de l'utilisateur UML, Eyrolles, 1998.

[Pie_97] : Pierre Alain Muller « Modélisation objet avec UML » édition 1997.

[Nat_97] : N. LOPEZ, J. MIGUEIS, E. PICHON, Intégrer UML dans vos projets, Eyrolles, 1997.

[Sta_03] : W. Stallings ; SNMP, SNMPv2 and CMIP: the practical guide to network management standards, Addison Wesley ; 1993.

[Emm_04] : THESE Emmanuel Reuter « Agents Mobiles : Itinéraires pour l'administration système et réseau » Mai 2004.

[Emm_00] : Les Agents Mobiles et Actifs pour l'Administration de Réseaux, Emmanuel Reuter, DEA Réseaux et Systèmes Distribués Juillet 2000.

[Ham_03] : Hamlati Anis, Zentor Hanane Kheira « Gestion des Réseaux » mémoire de fin d'étude d'ingénieur année 2003 université de BLIDA.

[Fpv_98] : A. Fuggetta, G.P.. Picco, G. Vigna, Understanding Code Mobility, IEEE Transactions on Software Engineering 24(5), 1998, pp. 342-361.

[And_99] : Andrew Tanenbaum. Réseaux : Architecture, protocole, applications, 3ème édition, Dunod. Prentice Hall, 1999.

[Sup_04] : CHELLAL Abd El Hamid, NEGGAZI Brahim « Conception et réalisation d'un

superviseur réseau Ethernet » mémoire de fin d'étude d'ingénieur année 2004 EMP.

Internet :

[Htt_98] : HTTP Version 1.1 http://www.w3.org/Protocols.

[Htm_98] : HTML Version 4.0 http://www.w3.org/MarkUp.

[Voy_05] : Voyager. ObjectSpace, Inc., 2005. www.objectspace.com.

[Agl_05] : Les Aglets java www.Aglets.com

[Adv_05] : AdventNet. AdventNet SNMP tools, 2005. http://www.adventnet.net.

[Gos_95] : Gosling (J) and McGilton (G.). The Java language Environment : A white paper.

[Pro-05] : INRIA, 1999. http://www-sop.inria.fr/oasis/ProActive.

[AdN_98] : Advent Net Monitor _ Advent Network Management Inc, http://www.adventnet.com.

[Omg_05] : Object Management Group : Agent Working Group «http://www.objs.com/isig/Agents.html » Technical Report Sun Microsystems, http:// java.sun.com-95.

[Mia_04] : Mobile Intelligent Agents for Managing the Information Infrastructure http://www.cordis.lu/infowin/acts/analysys/products/ thematic/agents/ch3/miami.htm.

[Map_02] : A. Puliafito and O. Tomarchio. Security Mechanisms for the MAP Agent System. In In 8th Euromicro Workshop on Parallel and Distributed Processing (PDP2000), pages 84–91, January 2000. http://sun195.iit.unict. it/~otomarch/publications.html.

[Som_01] : P. Bellavista, A. Corradi, and C. Stefanelli. How to Monitor and Control Resource Usage in Mobile Agent Systems. In 3 rd IEEE Int. Symp. on Distributed Objects and Applications (DOA'01), Italy. IEEE Computer Society Press, September 2001. http://citeseer.nj.nec.com/ bellavista01how.html.

[Mos_01] : Ichiro Satoh. Network processing of Mobiles Agents, by Mobile Agents, for Mobile Agents. In 3th International Workshop on Mobile Agents. for Telecommunications Applications, MATA'01, volume 2146, pages 81– 92. Lecture Notes in Computer Science (LNCS), August 2001. http: //citeseer.nj.nec.com/575168.html.

[Bel_99] : P. Bellavista, A. Corradi, and C. Stefanelli. An Open Secure Mobile Agent Framework for Systems Management. Journal of Network and Systems Management, 7(3), september 1999. http://citeseer.nj.nec.com/ bellavista99open.html.

Annexes

Annexe A

Description de la MIB II

ANNEXE A

DESCRIPTION DE LA MIB-II

1.Groupe system

Nom	Type de données	R/ W	Description
sysDescr	Displaystring		Description textuelle de l'entité.
sysobjectID	ObjectID		Identificateur du vendeur dans le sous-arbre 1.3.6.1.4.1
sysUpTime	TimeTicks		Temps en centièmes de seconde depuis le redémarrage du management de réseau.
sysContact	DisplayString		Nom de la personne et comment la contacter.
sysName	Displaystring		Nœud du nom de domaine (FQDN).
sysLocation	DisplayString		Localisation physique du nœud.
sysservices	[O..127]		Valeur indiquant les services fournis par le noeud. C'est la somme des couches du modèle OSI supporté par le noeud. Les valeurs suivantes sont juxtaposées, en fonction des services fournis : 0x01 (physique), 0x02 (liens de données), 0x04 (Internet), 0x08 (application).

TAB 1 : Variables simples dans le groupe system

2.Groupe interface

Nom	Type de données	R/W	Description
ifNumber	INTEGER		Nombre d'interfaces réseau sur le système

TAB 2 : Variable simple groupe *ifInterface*

Table d'interface, index = <*IfIndex*>			
Nom	**Type de données**	**R/W**	**Description**
iflrndex	INTEGER		Index de l'interface, entre 0 et if Number.
ifDescr	Displaystring		Description textuelle de l'interface.
ifType	INTEGER		Type, par exemple : 6=Ethernet, 7=802.3 Ethernet, 9=802.5 token ring, 23=PPP, 28=SLIP, et beaucoup d'autres valeurs.
ifmtu	INTEGER		MTU de l'interface.
ifspeed	Gauge		Vitesse en bits/sec.

ifoperstatus	[1-3]		Etat courant de l'interface : 1=up, 2=clown, 3=testing.
ifLastchange	TimeTicks		Valeur de sysUpTime quand l'interface est entrée dans l'état actuel.
ifInoctets	counter		Nombre total d'octets reçus, y compris les caractères de trame.
ifInUcastPkts	counter		Nombre de paquets unicasts délivrés aux couches supérieures.
ifInNUcastPkts	counter		Nombre de paquets non unicasts (ex : broadcast ou multicast) délivrés aux couches supérieures.
ifInDiscards	counter		Nombre de paquets reçus rejetés même s'il n'y avait pas d'erreurs dans le paquet (ex : out of buffers.
ifInErrors	counter		Nombre de paquets reçus rejetés à cause d'erreurs.
ifInUnknownProtos	counter		Nombre de paquets reçus rejetés à cause d'un protocole inconnu.
ifoutoctets	counter		Nombre total d'octets transmis, y compris les caractères de trame.
ifoutUcastPkts	counter		Nombre de paquets unicasts reçus des couches Supérieures.
ifoutNUcastPkts	counter		Nombre de paquets non unicasts (ex : broadcast ou multicast) reçus des couches supérieures.
ifoutDiscards	counter		Nombre de paquets sortants rejetés même s'il n'y avait pas d'erreurs dans le paquet (ex : out of buffers.
ifoutErrors	counter		Nombre de paquets sortants rejetés à cause d'erreurs.
ifoutQLen	Gauge		Nombre de paquets dans la file d'attente de sortie.
ifSpecific	GbjectiD		Une référence à la définition de la MIB spécifique à ce type de média.

TAB 3 : Table d'interface

3.Groupe AT

Table de translation d'adresse, index = *<atiflndex>.1.<atNetAddress>*			
Nom	**Type de données**	**R/W**	**Description**
atIfIndex	INTEGER		Numéro d'interface:ifIndex
atPhysAddress	PhysAddress	•	Adresse physique. (entrée est invalidée
atNetAddress	Network Address	•	si la longueur de la chaîne est 0.Adresse IR

TAB 4: Table de translation d'adresses : atTable

4.Groupe ip

Nom	Type de données	R/W	Description

ipForwarding	[1_2]	•	1 signifie que le système retransmet les datagrammes IP, et 2 qu'il ne le fait pas.
ipDefaultTTL	INTEGER		Valeur de TTL par défaut quand la couche de transport n'en fournit pas.
ipInReceives	counter		Nombre total de datagrammes IP reçus de toutes les interfaces.
ipInxdrErrors	counter		Nombre de datagrammes IP rejetés à cause d'erreurs d'en-tête (ex : erreur de somme de contrôle, mauvais numéro de version, TTL dépassé, etc..
IpInAddrErrors	counter		Nombre de datagrammes IP rejetés à cause d'une erreur d'adresse.
ipForwDatagrarns	counter		Nombre de datagrammes IP avec essai de transmission sattransmissionretransmission.
ipInUnknownProtos	counter		Nombre de datagrammes IP adressés localement avec un champ protocole invalide.
IpInDiscards	counter		Nombre de datagrammes IP reçus rejetés à cause d'un manque d'espace buffer.
IpInDelivers	counter		Nombre de datagrammes IP délivrés à un module de protocole approprié.
IpoutRequests	counter		Nombre total de datagrammes IP passés à IP pour transmission. N'inclue pas ceux qui sont comptés par ipForwDatagram.
IpoutDiscards	counter		Nombre de datagrammes IP sortants rejetés à cause d'un manque d'espace buffer.
IpoutNoRoutes	counter		Nombre de datagrammes IP rejetés parce qu'aucune route n'a été trouvée.
ipReasmTimeout	INTEGER		Nombre de secondes maxi pendant lesquelles les fragments reçus sont maintenus avant ré assemblage.
IpReasmRegdq	counter		Nombre de fragments IP reçus qui ont besoin d'être rassemblés.
IpReasmoxs	counter		Nombre de datagrammes IP rassemblés avec succès.
IpReasmFails	counter		Nombre d'échec de l'algorithme de ré assemblage IP
IpFragsORs	counter		Nombre de datagrammes IP qui ont été fragmentés Avec succès.
IpFragFails	counter		Nombre de datagrammes IP qui avaient besoin d'être fragmentés, mais qui n'ont pas pu l'être à cause du flag «dont fragment».
IpFragcreates	counter		Nombre de datagrammes IP générés par fragmentation.
ipRoutingDiscards	counter		Nombre d'entrées de routage rejetées même si elles étaient valides.

TAB 5 : Variables simples dans le groupe ip

Table d'adresse IP,			**index** = <*ipAdEntAddr*>
Nom	**Type de données**	**R/w**	**Description**
ipAdEntAddr	IpAddress		Adresse IP pour cette ligne.
ipAdEntIfIndex	INTEGER		Numéro d'interface correspondante : ifIndex.
ipAdEntNetMask	IpAddress		Masque de sous-réseau pour cette adresse IP

ipAdEntBcastAddr	[0..1]		Valeur du bit le moins significatif de l'adresse de broadcast.Normalement 1.
ipAdEntReasmMaxSize	[0..65535]		Taille du datagramme IP le plus grand qui peut être rassemblé sur cette interface

TAB 6 : Table d'adresses IP : ipAddrTable

Table de routage IP, index = *<ipRouteDest>*			
Nom	**Type de données**	**R/W**	**Description**
ipRouteDest	ipAddress	•	Adresse IP de destination. Une valeur de 0.0.0.0 indique une entrée par défaut.
ipRouteIfIndex	INTEGER	•	Numéro d'interface : ifIndex.
ipRouteMetricl	INTEGER	•	Métrique de routage primaire. La signification de cette métrique dépend du protocole de routage (ipRouteProto). Une valeur de -1 signifie qu'il n'est pas utilisé.
ipRouteMetric2	INTEGER	•	Métrique de routage alternatif.
ipRouteMetric3	INTEGER	•	Métrique de routage alternatif.
ipRouteMetric4	INTEGER	•	Métrique de routage alternatif.
ipRouteNextHop	ipAddress	•	Adresse IP du routeur de saut suivant.
ipRouteType	INTEGER	•	Type de route : 1 = autre, 2 = route invalide, 3 = directe, 4 = indirecte.
ipRouteProto	INTEGER		Protocole de routage : 1 = autre, 4 = redirection ICMP, 8 = RIP, 13 = OSPF, 14 = BGP, et autres.
ipRouteAge	INTEGER	•	Nombre de secondes depuis que la route a été mise à jour ou déterminée correcte.
ipRouteMask	ipAddress	•	Le ET logique de ce masque et de l'adresse IP de destination est comparée à ipRouteDest.
ipRouteMetric5	INTEGER	•	Métrique de routage alternatif.
ipRouteInfo	objectID	•	Référence aux définitions de la MIB spécifique à ce protocole de routage.

TAB 7 : Table de routage IP :ipRouteTable

Table de translation d'adresse IP, index = *<ipNetToMediaiftndex><ipNetToMediaNetAddress>*			
Nom	Type de données	R/W	Description
ipNetToMediaIfIndex	INTEGER	•	Interface correspondante : ifIndex.
ipNetToMediaPhysAddress	PhysAddress	•	Adresse physique.
ipNetToMediaNetAddress	IpAddress	•	Adresse IP.
ipNetToMediaType	[1_4]	•	Type de translation : 1 = autre, 2 = invalidée,3 = dynamique, 4 = statique.

TAB 8 : Table de translation d'adresses IP : ipNetToMediaTable.

5. Groupe icmp

Nom	Type de données	R/W	Description
icmpInMsgs	counter		Nombre total de messages ICMP reçus.
icmpInErrors	Counter		Nombre de messages ICMP reçus avec des erreurs (ex : somme de contrôle ICMP invalide).
icmpInDestUnreachs	counter		Nbre de messages reçus ICMP destination non accessible
icmpInTimeExcds	counter		Nombre de messages reçus ICMP temps dépassé.
icmpInParmProbs	counter		Nombre de messages reçus ICMP problème de paramètre.
icmpInsrcQuenchs	counter		Nombre de messages reçus ICMP extinction de source.
icmpInRedirects	Counter		Nombre de messages reçus ICMP redirection.
icmpInEchos	counter		Nombre de messages reçus ICMP requête echo.
icmpInEchosReps	counter		Nombre de messages reçus ICMP réponse echo.
icmpInTimestamps	Counter		Nombre de messages reçus ICMP requête d'estampille horaire.
icmpInTimestampReps	Counter		Nombre de messages reçus ICMP réponse d'estampille horaire.
icmpInAddrMasks	counter		Nombre de messages reçus ICMP requête.de masque d'adresse.
icmpInAddrMaskReps	counter		Nombre de messages reçus ICMP réponse de masque d'adresse.
icmpoutmsgs	Counter		Nombre total de messages ICMP envoyés.
icmpoutErrors	counter		Nombre de messages ICMP non émis à cause d'un problème d'ICMP (ex : manque de buffers).
icmpoutDestunreachs	counter		Nombre de messages émis ICMP destination non accessible
icmpoutTimeExcds	counter		Nombre de messages émis ICMP temps dépassé.
icmpoutParmProbs	counter		Nombre de messages émis ICMP problème de paramètre.
icmpoutsrcQuenchs	counter		Nombre de messages émis ICMP extinction de source.
icmpoutRedirects	counter		Nombre de messages émis ICMP redirection.
icmpoutEchos	counter		Nombre de messages émis ICMP requête echo.
icmpoutEchosReps	counter		Nombre de messages émis ICMP réponse echo.
icinpoutTimestamps	counter		Nombre de messages émis ICMP requête d'estampille horaire.
icmpoutTimestampReps	counter		Nombre de messages émis ICMP réponse d'estampille horaire.
icmpoutAddrMasks	counter		Nombre de messages émis ICMP requête de masque d'adresse.

TAB 9 : Variables simples du groupe icmp.

6.Groupe tcp

Nom	Type de données	R/W	Description
tcpConnstate	Integer [1..12]		Etat de la connexion : 1 = CLOSED, 2 = LISTEN, 3 = SYN SENT, 4 = SYN RCVD, 5 = ESTABLISHED, 6 = FIN WAIT 1, 7 = FIN _WAIT 2, 8 =CLOSE WAIT, 9 = LAST ACK, 10 = CLOSING, 11 = TIME_WAIT, 12 = efface TCB. La seule valeur à laquelle le manager peut positionner cette variable est 12 (c'est-à-dire la terminaison immédiate de la connexion).
tcpConnLocalAddress	IpAddress		Adresse IP locale. 0.0.0.0 indique que le scrutateur est d'accord pour accepter les connexions sur toutes les interfaces.
tcpConnLocalPOrt	[0..65535]		Numéro de port local.
tcpConnRemAddre	IpAddress		Adresse IP distante.
tcpConnRemPort	[0..65535]		Numéro de port distant.

TAB 10 : Table de connexion TCP : tcpConnTable.

Nom	Type de données	R/W	Description
tcpRtOA lgorithm	INTEGER		Algorithme utilisé pour calculer la valeur du time out de retransmission : 1 = aucun des suivants, 2 = un RTO constant, 3 = MIL-STD-1778 Appendice B, 4 = algorithme de Van Jacob son.
tcpRtoMin	INTEGER		Valeur mini du timeout de retransmission, en millisecondes.
tcpRtoMax	INTEGER		Valeur maxi du timeout de retransmission, en millisecondes.
TcpMaxConn	INTEGER		Nombre maxi de connexions TCP.
tcpActiveopens	Counter		Valeur -1 si dynamique Nombre de transitions des états CLOSED à SYN SENT.
tcpPassiveopens	Counter		Nombre de transitions des états LISTEN à SYN RCVD.
tcpAttemptFails	Counter		Nombre de transitions des états SYN SENT ou SYN RCVD à CLOSED, plus le nombre de transitions de SYN RCVD à LISTEN.
tcpEstabResets	Counter		Nombre de transitions des états ESTABLISHED ou CLOSE WATT à CLOSED.
tcpCurrEstab	Gauge		Nombre de connexions en cours dans les états ESTABLISHED ou CLOSE WAIT.
tcpInsegs	Counter		Nombre maxi de segments reçus.
tcpoutsegs	Counter		Nombre maxi de segments émis, en excluant ceux qui contiennent seulement des octets retransmis.
tcpRetranssegs	Counter		Nombre total de segments retransmis.
tcpInErrs	Counter		Nombre total de segments reçus avec une erreur (comme une somme de contrôle invalide).
tcpoutRsts	Counter		Nombre total de segments émis avec le flag RST à 1.

TAB 11 : Variables simples dans le groupe tcp

7. Groupe udp

Nom	Type de données	R/W	Description
UdpInDatagrams	compteur		Nombre de datagrammes UDP délivrés aux processus utilisateur.
udpNoPorts	compteur		Nombre de datagrammes UDP reçus pour lesquels aucun processus applicable ne correspond au port de destination.
udpInErrors	compteur		Nombre de datagrammes UDP non livrables pour une raison autre que pas d'application sur le port de destination (par exemple erreur checksum UDP).
udpoutDatagrams	compteur		Nombre de datagrammes UDP émis.

TAB 12 : les variables simples de groupe udp

Nom	Type de données	R/W	Description
udpLocalAddress	IpAddress		Adresse Ip locale de scrutateur. .0.0.0.0 indique que le scrutateur voudrait recevoir des datagrammes sur n'importe quelle interface.
udpLocalPort	[0..65535]		Numéro de port local de ce scrutateur.

TAB 13 : udpTable

8-Le groupe EGP 1.3.6.1.2.1.8

Il contient les informations propres aux protocoles EGP

NOM	Type de	R/W	Description
EgpIn	counter		Le nombre total des messages EGP reçus sans erreur.
EgpErrors	counter		Le nombre total des messages EGP reçus prouvés en erreur.
EgpOutMsgs	counter		Le total de messages EGP localement produits.
EgpErrors	counter		Le nombre de messages EGP générés localement non envoyés à cause des limitations de ressources à l'intérieur d'une entité EGP.

TAB 14 : groupe EGP

9-groupe SNMP : 1.3.6.1.2.1.11

NOM	Type de	R/W	Description
SnmpInPkts	counter		Nombre total de PDU reçues de la couche inférieure.
SnmpOutPkts	counter		Nombre total de PDU envoyées à la couche inférieure.
SnmpBadVersions	counter		Nombre total de PDU reçues ayant un numéro de version différent du numéro local.
SnmpBadCommunityNames	counter		Nombre total de PDU ayant un nom de communauté inconnu.
SnmpBadCommunityUses	counter		Nombre total de PDU ne pouvant être traités dans le cadre de cette communauté.
SnmpInASNParseErrs			Nombre total d'erreurs au moment de l'interprétation d'un objet ASN.1
SnmpInBadTypes	counter		Nombre total de PDU avec un type inconnu

TAB 15 : le groupe SNMP

Code de retour

NOM	Type de	R/W	Description
SnmpInBigs	counter		Nombre de PDU reçues avec comme ErrorStatus'too bigs'(la réponse ne tient pas dans un message UDP).
SnmpInNoSuchNames	counter		Nombre de PDU reçues avec comme ErrorStatus'noSuchNames'(nom inconnu).
SnmpInBadValues	counter		Nombre de PDU reçues avec comme ErrorStatus'badValue'.
SnmpInReadOnlys	counter		Nombre de PDU reçues avec comme ErrorStatus'readOnly'.
SnmpInGenErrs	counter		Nombre de PDU reçues avec comme ErrorStatus'genErr'.

TAB 16 : le variable de code de retour SNMP

Objets Statiques en Entrée :

NOM	Type de	R/W	Description
SnmpInTotalReqVars			Nombre d'objets de la MIB qui ont fait l'objet d'une requête Get-Request ou get-Next .
SnmpInTotalSetVars			Nombre d'objets de la MIB qui ont fait l'objet d'une requête set-Request .
SnmpInGetRequests	counter		Nombre total de PDU Get-request traitées par le protocole SNMP.
SnmpInGetNexts	counter		Nombre total de PDU Get-Next traitées par le protocole SNMP.
SnmpInSetRequests	counter		Nombre total de PDU Set-request traitées par le protocole SNMP.
SnmpInGetRsponses	counter		Nombre total de PDU Get-Rsponse traitées par le protocole SNMP.
SnmpInTraps	counter		Nombre total de PDU Trap traitées par le protocole SNMP.
SnmpOutTooBigs	counter		Nombre total de PDU SNMP qui ont été générés par l'entité de protocole et pour lesquelles la valeur du champ 'ErrorStatus' est 'tooBig'
SnmpOutNoSuchNames	counter		Nombre total de PDU SNMP qui ont été générés par l'entité de protocole SNMP et pour lesquelles la valeur du champ 'ErrorStatus' est 'noSuchName'
SnmpOutBadValues	counter		Nombre total de PDU SNMP qui ont été générés par l'entité de protocole SNMP et pour lesquelles la valeur du champ 'ErrorStatus' est 'BadValue'
SnmpOutGenErrs	counter		Nombre total de PDU SNMP qui ont été générés par l'entité de protocole SNMP et pour lesquelles la valeur du champ 'ErrorStatus' est 'genErr'

TAB 17 : les objets statiques en entrée du groupe SNMP

Objets Statique En Sortie :

NOM	Type de	R/W	Description
SnmpOutGetRequests.	counter		Nombre total de PDU de type Get-Request qui ont été générés par l'entité de protocole SNMP
SnmpOutGetNexts.	counter		Nombre total de PDU de type Get-Next qui ont été générés par l'entité de protocole SNMP
SnmpOutSetRequests.	counter		Nombre total de PDU de type Set-Request qui ont été générés par l'entité de protocole SNMP
SnmpOutGetResponses.	counter		Nombre total de PDU de type Get-Response qui ont été générés par l'entité de protocole SNMP

SnmpOutTraps.	counter		Nombre total de PDU de type Trap qui ont été générés par l'entité de protocole SNMP
SnmpEnableAuthTraps	Integer		Indication sur la configuration de l'agent SNMP concernant la génération du Trap d'Authentification

TAB18 : les objets statiques en sortiée du groupe SNMP

ASN-1:(Abstract Syntax Notation one)

I. Type des variables de la MIB

Nom	type	Octets	Signification
Integer	Numérique	4	Entier (32 bits en général)
Counter32	Numérique	4	Compteur 32 bits entiers non signé
Gauge32	Numérique	4	Valeur non signé
Integer32	Numérique	4	32 bits mémé sur une UC de 64 bits
Uinteger32	Numérique	4	Comme Integer32 mais non signé
Counter64	Numérique	8	Compteur de 64 bits
TimeTicks	Numérique	4	En centième de secondes depuis un instant donné
Bit String	Chaîne	4	Suite de 1 à 32 bits
OctetString	Chaîne	>= 0	Chaîne de caractères de longueur variable
DisplayString			Une chaîne de mots de 8 bits
Objet Identifier	Chaîne	> 0	Liste d'entier
IpAddress	Chaîne	4	C'est une chaîne de 4 octets reprenant l'adresse IP
PhysAddress	Chaîne	6	C'est une chaîne d'octets spécifiant une adresse physique (par exemple une adresse Ethernet sur 6 octets).
SÉQUENCE			comparable à une structure en langage C Liste ordonnée d'autres types ASN.1
SÉQUENCE OF			vecteur, dont tous les éléments ont le même type de donnée
udpLocalAddress	IpAddress		contient l'adresse IP locale
udpLocalPort	Integer		Comprise entre 0 et 65535, qui précise le numéro de port local.

TAB 20 : Type des variables de la MIB

II ASN-1

A.S.N.1 est un langage formel qui permet de définire le nom et le type des variables de la base de données d'informations d'administration MIB. La précision de cette notation élimine toute ambiguïté de forme ou de contenu au niveau des variables. Elle présente deux caractéristiques principales : Une notation utilisée dans les documents manipulés par les humains et une représentation codée de la même information, utilisée dans les protocoles de communication. Les notions suivantes concernant ASN.1 sont nécessaires à connaître pour comprendre la MIB.

a)Module

Un module ASN.1 décrit une collection d'objets. La syntaxe de déclaration d'un module A.S.N.1 est la suivante :

NomModule Definition ::=
BEGIN
Relation avec les autres modules (clauses IMPORT et EXPORTE)

Définition des objets
END

b)Objets

Les objets définis avec ASN.1 peuvent être :

- Des types, avec des types simples comme INTEGER ou BOOLEAN,et des types construits permettant de définir des listes (SEQUENCE) et des tableaux (SEQUENCE OF) ;
- Des valeurs, c'est-à-dire des objets ayant un type précédemment défini ;
- Des macros, qui permettent d'étendre les définitions et définir de nouveaux types.

Par convention, les types commencent par une majuscule, les valeurs par une minuscule et les macros sont tout entières en majuscules. Les commentaires sont précédés de deux tirets.

Les types de base des objets A.S.N.1 sont données dans le tableau suivant : (tananboum)

Type	Description	Code
INTEGER	Entier 32 bits	2
OCTET STRING	Chaîne d'octets	4
BIT STRING	Chaîne de bits	3
OBJECT IDENTIFIER	Séquence d'entiers identifiant un objet à l'aide de son positionnement à partir de la racine de la MIB	5
ENUMERATION OF INTEGER	Permet de représenter une énumération de valeurs par des entiers non nuls (ex. : on(1),off :2))	5

TAB 21 : Les types de base des données ASN.1 autorisé dans SNMP

b)Objets tabulaires :

Ils sont décrits au moyen des types construits ASN.1. Le tableau suivant donne les types des objets tabulaires supportés par SNMP est utilisée dans les MIB.

Type	Description
SEQUENCE	Liste ordonnée d'autres types ASN.1
SEQUENCE OF TYPE	Liste ordonnée d'éléments du même type

TAB 22 : Les types tabulaires de données ASN.1 autorise dans SNMP

Contrairement à la plupart des protocoles TCP/IP, les messages SNMP n'ont pas une structure fixe mais utilisent la notation ASN.1.Un message SNMP contient, comme nous l'avant vu, trois parties principales : une version de protocole, un identificateur de communauté et une zone de données.

RFC1157-SNMP DEFINITIONS ::= BEGIN

IMPORTS
ObjectName, ObjectSyntax, NetworkAddress, IpAddress, TimeTicks
FROM RFC1155-SMI;
Message ::= SEQUENCE {
version INTEGER {version-1 (0)},-- Version 1 for this RFC
community OCTET STRING, -- Community name
data ANY -- e.g. PDUs
}

La zone de données se décompose en unités de données de protocole (PDU).Chaque PDU est constituée d'une demande émise par le client (le manager) ou d'une réponse émise par le serveur (l'agent).

-- Protocol data units

PDUs ::= CHOICE {*get-request GetRequest-PDU,*
get-next-request GetNextRequest-PDU,
get-response GetResponse-PDU,

```
        set-request         SetRequest-PDU,
        trap                Trap-PDU
}
GetRequest-PDU ::= [0] IMPLICIT PDU
GetNextRequest-PDU::=[1] IMPLICIT PDU
GetResponse-PDU ::= [2] IMPLICIT PDU
SetRequest-PDU ::= [3] IMPLICIT PDU

PDU ::= SEQUENCE {
        request-id INTEGER,                          -- Request identifier
        error-status INTEGER {                       -- Sometimes ignored
                noError (0),toobig (1),noSuchName (2),badValue (3),readOnly (4),genError (5)},
error-index INTEGER,                                 -- Sometimes ignored
variable-binding VarBindList }                       -- Values are sometimes ignored

Trap-PDU ::= [4] IMPLICIT SEQUENCE {
        enterprise OBJECT IDENTIFIER,                ,--Type of object generating trap
        agent-addr NetworkAddress                    s-- Only one type of network adresses
                                                     -- IP adress of object generating trap
        generic-trap INTEGER {                       -- Generic trap type
                coldStart (0),warmStart ( 1),linkDown ( 2),linkUp (3),authenticationFailure (4)
                egpNeighborLoss ( 5),enterpriseSpecific (6)
        },
specific-trap INTEGER,                               -- Specific code
time-stamp TimeTicks,                -- Elapse time since the last reinitialization of the entity
variable-binding VarBindList                 -- "Interesting" information
}

-- Variable binding
VarBind ::= SEQUENCE
{name ObjectName,
value ObjectSyntax}

VarBindList ::= SEQUENCE OF VarBind
END
```

Pour compléter la définition d'un message SNMP, il est nécessaire de préciser la syntaxe de chacun des cinq types de PDU, nous avons indiqué à titre d'exemple la structure de PDU *GetRequest* et *trap*.

}

Annexe B

Configuration du protocole SNMP

ANNEXE B

CONFIGURATION DU PROTOCOLE SNMP

Les points essentiels pour la configuration de SNMP dans un agent (équipement) sont :

- L'installation et l'activation du protocole SNMP.
- Déclaration des nom et emplacement pour chaque agent.
- Spécification des noms de communauté (lecture et écriture).
- L'activation des traps.
- La déclaration des adresses des NMS dans les agents.

Nous allons traiter deux types d'agent :

1. Equipement d'interconnexion,
2. Station de travail.

III.1.1. La configuration de SNMP dans les équipements d'interconnexion

Déclaration des noms de communauté :

-snmp-server community *nom de communauté* **ro/rw** : permet de configuré les non de communauté **ro** (lecture) ou **rw** (lecture/écriture).

Exemple : configuré « public » pour la lecture et « private » pour lecture/écriture

-snmp-server community public **ro**

-snmp-server community private **rw**

-no snmp-server community *nom de communauté* : permet d'éfacer le non de communauté

Déclaration de nom pour un agent :

-snmp-server contact *nom de l'agent SNMP*: permet de donné un nom à l'agent.

exemple: configuré "agent cisco 2900 » comme un nom de contact à l'agent.

-snmp-server contact agent cisco 2900

Déclaration de l'emplacement pour un agent :

-snmp-server location *emplacement de l'a*gent :permet de donné un emplacement à l'agent

exemple :configuré « étage '3 routeur '12 »comme un emplacement de l'agent

-snmp-server location étage '3 routeur '12

-no snmp-server contact *nom de l'agent SNMP* : permet d'éfacer le nom d'agent

-no snmp-server location *emplacement de l'agent* :permet d'éfacer l'emplacement.

L'activation des traps :

-snmp-server enable traps : Permettre d'activer l'envoyer des traps SNMP.

Exemple d'activation du trape :

-snmp-server enable traps

-no snmp-server enable traps: permet de désactiver les trapes

Déclaration des adresses de destination :

-snmp-server host *NMS_host+ nom de communauté*: permet de spécifier l'adresse (adresse du NMS) ou l'agent envoi des traps.

Exemple : configuré l'envoi des alerte à l'adresse « 192.168.0.1 »avec un nom de communauté public

-snmp-server host 192.168.0.1 public

Remarque:

-no snmp-server host *NMS_host* : permet d'effacer l'adresse de destination des traps de l'agent.

-show snmp : permet d'avoir les déférent information concernant le protocole SNMP.

III.1.2. La configuration de SNMP dans les stations de travail

Pour installer le service **Microsoft SNMP** sur toutes les versions de Windows 2000. il faut suivre les étapes suivantes :

1-Aller au panneau de configuration et choisir « ajout/suppression des programmes ».

2-Selectionner « ajout/suppression des composant Windows ».

3-Choisir « outils de gestion et d'analyse » puis cliquer sur « détail ».

4-Choisir le service SNMP.

Après l'installation du service, il est cependant possible de configurer cet Agent SNMP. Il est même très fortement conseillé d'effectuer ce paramétrage pour des raisons de sécurité. A partir de l'outil gestion de l'ordinateur qui se trouve dans (panneau de configuration\outils d'administration\..).

1-Dans « service d'application » sélectionner « service ».

2-Dans la liste des services sélection « service SNMP » et ouvrir sa fenêtre des propriétés.

Glossaire

GLOSSAIRE

A :

Agent SNMP

Logiciel assurant le pilotage d'un équipement dialoguant avec le gestionnaire (Manager).

API

(Application Programming Interface). Interface pour langage de programmation matérialisée par des primitives. elle permettent à une application d'accéder à des fonctions système d'un programme, par exemple pour communiquer ou extraire des données.

ARP

(Address Resolution Protocol) protocole de résolution d'adresse : Processus des réseaux IP permettant à un système hôte de trouver l'adresse MAC (Media Access Control) d'un hôte cible situé sur le même réseau physique, mais pour lequel seule l'adresse IP est connue. Sous ARP, les cartes de réseau contiennent une table qui assigne les adresses IP aux adresses matérielles des objets appartenant au réseau. Pour créer des entrées, le protocole ARP diffuse une requête contenant l'adresse IP de l'hôte cible, qui répond en indiquant son adresse physique. La carte de réseau ajoute alors cette adresse à sa table ARP et peut ensuite envoyer des paquets à l'hôte cible.

ASN.1

(Abstract Syntaxe Notation number One) Notation formelle qui permet de spécifier très facilement et sans sacrifier à la généralité les informations manipulées par les protocoles de télécommunications, indépendamment des systèmes informatiques, des logiciels et des modes de transfert des données. Il a été normalisé dans le cadre de l'ISO et du CCITT.

B :

BER

(Basic Encoding Rules) décrit le codage de chaque type SMI sous forme d'une chaîne d'octets.

C :

CIM

(Common Information Model) Norme développée par le DMTF permettant de décrire des données, des applications ou des entités de telle manière que les administrateurs et les programmes de gestion puissent contrôler les différentes applications et dispositifs des différentes plates-formes de la même manière, assurant ainsi l'interopérabilité à travers le réseau.

CGI

(Common Gateway Interface) Interface standard installée sur les serveurs HTTP permettant entre autres l'envoi de variables d'entrée et d'environnement à tout programme serveur dont on demande l'exécution (utile par exemple pour la saisie de formulaires via le Web).

CMIP

(Common Management Information Protocol) Norme de supervision de réseaux définie par l'ISO (décrivant comment transmettre des informations de la gestion ISO), il permet de fournir les services liés au standard CMIS.

CMIS

(Common Management Information Services) Norme de supervision de réseaux définie par l'ISO qui présente les services utilisables par les entités de gestion. CMIS définie un ensemble de primitives et la structures des messages échangeables par CMIP.

CMOL

CMIP Over LLC.

CMOT

CMIP Over TCP/IP.

D :

Débit binaire

(bit rate) Nombre de bits transmis pendant un intervalle de temps rapporté à la durée de cet intervalle. Un débit binaire s'exprime en bit/s ou en multiple de cette unité : kbit/s, Mbit/s, Gbit/s.

DES

Data Encryption Standard, Algorithme utilisé par UNIX pour crypter les mots de passe.

DHCP

(Dynamic Host Configuration Protocol) Protocole qui permet à un ordinateur qui se connecte sur un réseau local d'obtenir dynamiquement et automatiquement sa configuration IP.

DMI

(Desktop Management Interface) Norme développée par le DMTF qui permet la collecte d'informations sur l'environnement d'un ordinateur.

DMTF

(Distributed Management Task Force) Consortium industriel très actif dans la recherche et le développement de normes de supervision ainsi qu'à leur mise en place.

DNS

(Domain Name Server) Serveur spécifique disposant d'une base de données géante mise à jour très régulièrement (fonctionnant avec un système de tables de correspondance) délivrant le nom (du type nom.domaine.organisation) correspondant à l'adresse IP (utilisée par les ordinateurs clients ou serveurs pour communiquer entre eux).

DoD

Department of Defense

F :

Firewall

(Pare Feu ou Coupe Feu) Système logiciel de protection placé entre un réseau local (comme celui d'une entreprise) et un autre réseau (par exemple Internet). Cette barrière sert à assurer la sécurité des informations internes au réseau local en filtrant les entrées et en contrôlant les sorties selon une procédure automatique bien établie. Le firewall est un "mur" contre le piratage.

FTP

(File Transfer Protocol) Protocole de transfert des fichiers informatiques sur des réseaux supportant TCP/IP.

G :

GID

Group **ID**, identifiant de groupe.

H :

HTML

(HyperText Markup Language) HTML est un langage de description de pages - littéralement de marqueurs hypertextes. C'est le langage utilisé dans les pages accessibles sur le Web.

HTTP

(HyperText Transfer Protocol) Méthode utilisée pour transporter des pages HTML du Web sur le réseau. L'accès aux services Web se fait en donnant une adresse de type http://nom de domaine/répertoire.

I :

IAB

(Internet Activities Board). Cette commission assure l'orientation et la coordination de la majeure partie de la recherche et des développements relatifs aux protocoles TCP/IP. Elle oriente l'évolution d'internet, décide quels protocoles font partie de la famille des protocoles TCP/IP et définit les stratégies officielles.

ICMP

(Internet Control Message Protocol) Extension du protocole Internet qui permet la génération de messages d'erreurs, de tests et d'informations relatifs aux conditions de transmission sur le réseau.

IEEE

Institute of Electrical and Electronics Engineers

IETF

(Internet Engineering Task Force). Groupe de travail qui développe les nouveaux standards pour l'Internet. Force d'intervention principale de l'IAB. Souvent nommé groupement Internet.

IGRP

(Interior Gateway Routing Protocol) Protocole qui permet aux passerelles de construire leur table de routage par échange d'informations avec d'autres passerelles. L'IGRP est un protocole à vecteur de distance. Il est identique au protocole RIP de Berkeley.

Internet

Ensemble de réseaux et de passerelles, MILNET et NFSNET inclus, qui utilise la famille des protocoles TCP/IP. Il fonctionne comme un réseau virtuel unique et coopérant.

IOS

(Internetworking Operating System) Permet aux routeurs et commutateurs de fonctionner avec (IP, IPX) en réseau local, (X.25, RNIS, PPP, Frame Relay) en réseau étendu avec les protocoles de routage : RIP, IGRP.

IP

(Internet Protocol) Protocole de télécommunications utilisé sur les réseaux qui servent de support à Internet et permettant de découper l'information à transmettre en paquets, d'adresser les différents paquets, de les transporter indépendamment les uns des autres et de recomposer le

message initial à l'arrivée. Ce protocole utilise ainsi une technique dite de commutation de paquets. Sur Internet, il est associé à un protocole de contrôle de la transmission des données appelé TCP (Transmission Control Protocol) ; on parle ainsi du protocole TCP / IP.

IPX

(Internetwork Packet Exchange) Pile de protocoles utilisés pour l'expédition et le routage des paquets dans les réseaux Novell.

ISO

(International Organisation for Standardization). C'est une organisation internationale non gouvernementale dont le rôle est d'unifier et de coordonner les domaines techniques du traitement de l'information. Plus de 90 pays à travers le monde en sont membres en ayant une filiale nationale (AFNOR en France, ANSI aux USA). Les normes internationales ont leur correspondance en norme nationale exemple la norme ISO 7498 est l'équivalent de la norme AFNOR NFZ70-001 qui décrit le modèle OSI (Open System Interconnection). Ce modèle décrit le fonctionnement d'un réseau du niveau matériel au niveau "application" en découpant les problèmes en couches.

L :

LAN

(Local Area Network) Réseau informatique d'interconnexion locale d'ordinateurs, situé dans un rayon géographique relativement restreint et constitué d'au moins 2 ordinateurs reliés entre eux. Dans un réseau local, il est courant de trouver du matériel Ethernet RJ45 ou BNC pour relier des machines au sein du LAN.

LDAP

(Lightweight Directory Access Protocol) Structure d'annuaire standard sur les réseaux TCP/IP.

M :

MAC

(Media Access Control) Part du modèle de données IEE d'un réseau. La couche MAC implémente le protocole qui contrôle l'accès au réseau.

MIB

(Management Information Base) Base de données d'objets pouvant être consultée grâce à un système de supervision de réseau.Elle represente les données à administrer.

Mib-1 et Mib-2

Mib standard (version 1 et 2) décrivant l'équipement, ses ports de communication et ses statistiques globales de fonctionnement.

Mib privée

Mib propriétaire relative à un équipement géré, en complément des informations présentes dans les Mib standard.

MRTG

(Multi Router Traffic Grapher) Logiciel Unix gratuit basé sur SNMP permettant la traduction graphique de données pertinentes récupérées sur des entités du réseau.

N :

NIC

Network Information Center

NIS

Network Information Service

NIST

National Institute of Standards and Technology

NMS

(Network Management System) Système de gestion de réseau

NTP

(Network Time Protocol) Protocole de synchronisation du réseau.

O :

OMG

(Object Management Group) Structure de travail définissant des normes sur les méthodes, objets et modèles à l'issue de leurs activités de recherches.

OSI

(Open Systems Interconnection) Architecture à 7 couches qui normalise les niveaux de service et les types d'interactions entre les ordinateurs qui échangent des informations à travers un réseau. Elle décrit le flux des données entre la connexion physique et le réseau d'une part et le programme de l'utilisateur final d'autre part.

P :

Passerelle

(Gateway) Interface de programmation entre deux ressources informatiques de nature distincte.

PROM

Programmable Read Only Memory

Proxy

(Mandataire) Dispositif informatique associé à un serveur et réalisant, pour des applications autorisées, des fonctions de médiation, telle que le stockage des documents les plus fréquemment demandés ou l'établissement de passerelles.

Proxy-agent

Passerelle entre un agent standard (SNMP par exemple) et un environnement propriétaire à gérer.

R :

RAM

Random Access Memory

RARP

(Reverse Address Resolution Protocol) Protocole permettant aux hôtes d'un réseau de trouver leur adresse IP au démarrage.

RFC

(Request For Comment) Une série de documents techniques émanant de la communauté de recherche et du développement de l'Internet. les propositions de définition, de modification de protocoles ainsi que les normes TCP/IP.

RFC 1155
Stucture and identification of Management Information for tcp/ip based networks.

RFC 1212
Concise Mib definitions.

RFC 1215
A convention for defining traps for use with SNMP

RFC 1157
Simple Network Management Protocol.

RIP
(Routing Information Protocol) Protocole de routage interne avec vecteur de distance destiné aux réseaux TCP/IP.

RMON (agent)

Remote **Mon**itoring. Agent particulier situé dans une sonde. Il surveille les paramètres de fonctionnement d'un réseau local.(taux d'erreurs, collisions ...)

RNIS
(Integrated Services Digital Network) Réseau numérique à intégration de services. Norme internationale des réseaux numériques.

ROM

Read **O**nly **M**emory

RPC

Remote **P**rocedure **C**all

S :
SGMP
(Simple Gateway Management Protocol) Protocole de supervision de passerelles (gestion des routeurs) TCP/IP à l'origine du protocole SNMP.

SHTTP
(Secure HTTP) Version sécurisée du protocole HTTP.

SMI
Structure of Management Information, décrit les règles d'écriture des objets dans une MIB. SMIV1 est défini dans les RFC suivants: RFC1155, RFC1212, RFC1215.

SMP
(Simple Management Protocol). Evolution de SNMP, très proche de SNMP version 2. Il apporte une amélioration notable de performances et de sécurité d'utilisation par rapport à SNMP version 1.

SNA
(System Network Architecture) Architecture de réseau en couche introduit par IBM et qui servit, plus tard, de base au modèle OSI.

SNMP
(Simple Network Management Protocol) Protocole de supervision de réseaux conçu par l'IETF pour le monde IP. Il existe actuellement 3 versions. Il supporte les opérations SET, GET, GET-NEXT et TRAP permettant d'agir sur les objets d'une MIB.

SNMPv2

Deuxième version de SNMP. Similaire à SMP.

Sonde
Equipement de supervision remontant des alarmes de performance et d'incidents. Il assure les fonctions de capture d'un analyseur de protocoles.

T :
TCP
(Transfer Control Protocol) Protocole de contrôle de transmission d'informations apparu dans les années 1970 sur les réseaux de machines UNIX. Il fonctionne par découpage de l'information en paquets de données afin de les acheminer à l'adresse voulue sur le réseau. C'est le protocole en vigueur régissant les transferts de fichiers sur le Réseau Internet.
TP4
Classe de transport 4 orienté connexion défini dans la norme X224 du modèleOSI.
TMN
(Telecommunications Management Network) Protocole de supervision des réseaux de télécommunications conçu par l'UIT-T. Celui-ci est très évolué et utilise CMIP.

U :
UDP
(User Datagram Protocol) Protocole en mode non connecté qui, à l'instar de TCP, tourne au dessus d'IP. Il permet l'envoi et la réception de datagrames sur un réseau IP mais n'offre que peu de services de protection d'erreurs.

UID
(User ID), identifiant d'utilisateur
UML
(Unified Modeling Language) L'UML est le concept de modélisation objet. Son développement a débuté en 1994 et il a été normalisé par l'OMG début 1997.

V :
VLAN
(Virtual LAN) Réseau local (LAN) "virtuel" permettant de cloisonner des réseaux par l'intermédiaire de filtres de sécurité. Dans un VLAN, des groupes de travail comprenant PC et serveurs peuvent être formés dans l'entreprise, sans tenir compte des situations géographiques. Les fonctions d'autorisation d'accès sont définies à partir d'un commutateur.
W :
WAN
(Wide Area Network) Réseau pouvant s'étendre dans une zone très grande. Nécessite en général des liaisons Internet comme la fibre optique avec un débit très important par exemple. Le WAN permet donc l'interconnexion de sites distants au sein d'une architecture réseau.
WBEM
(Web-based Entrprise Management) Initiative du DMTF afin de proposer une solution de supervision de réseaux intégrée via le Web.

X :
XML
(eXtensible Markup Language) Meta-langage extensible dérivé de SGML permettant de structurer des données.