

L'évaluation du modèle gestion/agent dans la gestion de réseaux et services

Hanane Ez-Zahra Oumina

▶ **To cite this version:**

Hanane Ez-Zahra Oumina. L'évaluation du modèle gestion/agent dans la gestion de réseaux et services. [Stage] A04-R-466 || ez-zahra_oumina04a, 2004, 54 p. inria-00099921

HAL Id: inria-00099921
https://inria.hal.science/inria-00099921

Submitted on 26 Sep 2006

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

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

L'évaluation du modèle Gestionnaire/Agent dans la gestion de réseaux et services

MÉMOIRE

15 Juin 2004

pour l'obtention du

DEA de l'Université Henri Poincaré – Nancy I

(Spécialité Informatique)

par

Hanane Ez-Zahra Oumina

Composition du jury

Pr. Dominique Méry
Pr. Didier Galmiche
Pr. Noëlle Carbonell

Responsable de filière : Olivier Festor

Encadrant : Laurent Andrey

Mis en page avec la classe thloria.

Table des matières

Remerciements **1**

3

Introduction et Problématique **5**

1 Les plate-formes de gestion de réseaux et services **7**

1.1 Les plate-formes de gestion des réseaux et services 7

1.1.1 Le modèle SNMP/SMI 7

1.1.2 Le modèle CIM/WBEM 7

1.1.3 Le modèle JMX 8

1.2 Les applications de gestion des réseaux et services 9

1.2.1 Les types de applications de gestion 9

1.2.2 L'approche distribuée de déploiement des applications de gestion 9

1.2.3 L'approche centralisée 9

1.2.4 L'approche hiérarchique 9

1.2.5 L'approche distribuée 10

1.2.6 L'évaluation et l'optimisation des performances des applications de gestion 10

1.2.7 Conclusions 13

2 Le protocole SNMPv3 **15**

2.1 L'architecture de SNMPv3 15

2.2 Les aspects protocolaires 16

2.2.1 Les opérations de SNMPv3 16

2.2.2 Le format du message SNMPv3 17

2.2.3 La taille du message SNMPv3 18

2.2.4 La sécurité dans SNMPv3 18

2.3 Les aspects fonctionnels de SNMP 21

2.3.1 L'accès aux ressources avec SNMP 21

2.3.2 L'agent SNMP 22

3 Les fonctions de gestion et l'influence de la sécurité **23**
3.1 Les fonctions de gestion 23
3.1.1 Les variables de gestion 23
3.1.2 Les scénarios de gestion 24
3.2 L'impact de la sécurité sur la taille du trafic de gestion 25
3.2.1 Les paramètres de sécurité 25
3.2.2 L'exemple de la recherche de la variable *ipInReceives* 26
3.3 L'impact de la sécurité sur le temps de traitement de la requête 27
3.3.1 Les caractéristiques des tests 27
3.3.2 L'influence du niveau de sécurité 29
3.3.3 L'influence de la procédure de contrôle d'accès 31

4 Simulation et résultats **33**
4.1 Le but de la simulation 33
4.2 Les paramètres de simulation 33
4.2.1 Les topologies de la simulation 33
4.2.2 La taille des messages SNMP et les temps de traitement 34
4.3 Les résultats de la simulation 35
4.3.1 Le premier scénario de gestion 35
4.3.2 Le deuxième scénario de gestion 37

Conclusion **39**

Annexe A : Le comportement de *getwalk* et *getbulk* **40**

Annexe B : La taille du message SNMPv3 **42**

Annexe C : L'implémentation Net-SNMP v5.1 **44**

Bibliographie **47**

Table des figures

1 Le modèle d'une plate-forme de gestion 5

1.1 Les modèles de gestion 8
1.2 L'approche distribuée 10

2.1 L'architecture d'une entité SNMPv3 16
2.2 Le format du message SNMPv3 17
2.3 Les paramètres de sécurité du modèle USM 19
2.4 Les éléments d'un système géré par SNMP 21
2.5 Les aires fonctionnelles de l'agent SNMP 22

3.1 Les tâches de gestion 24
3.2 La structure de la table ARP 24
3.3 Les champs du bloc de sécurité selon le niveau de sécurité 26
3.4 La taille du trafic supplémentaire de gestion pour la requête *get* 26
3.5 Les caractéristiques des machines de test 27
3.6 La vitesse des algorithmes d'authentification 27
3.7 La vitesse des algorithmes de cryptage 28
3.8 Le temps de traitement de la requête *get* selon les niveaux de sécurité 29
3.9 Le temps de traitement de la première requête *getnext* de *getwalk* 30
3.10 Le temps de traitement de la requête *getbulk* 30
3.11 Le temps de contrôle d'accès 31
3.12 Le temps de contrôle d'accès en fonction de la taille des tables VACM 31

4.1 La taille des messages SNMP 34
4.2 Le temps de traitement à l'intérieur de l'agent 35
4.3 Le temps de traitement dans une topologie LAN pour le premier scénario de gestion 36
4.4 Le temps de traitement dans une topologie Internet pour le premier scénario de gestion 37
4.5 Le temps de traitement dans une topologie LAN pour le deuxième scénario de gestion 37
4.6 Le temps de traitement dans une topologie Internet pour le deuxième scénario de gestion 38

1 Les échanges des requêtes de *getnext* et *getbulk* 40
2 Le champ de varbinds requêtes/réponses de *getnext* et *getbulk* 41

1 Les champs du message SNMPv3 42
2 La taille des valeurs des champs dans le message SNMPv3 43

1 Les modules Net-SNMP 45

Remerciements

Ce travail n'aurait pas pu être réalisé sans le soutien et l'encouragement du Responsable du projet MADYNE à l'INRIA Lorraine, Monsieur Olivier Festor que je veux vivement remercier de m'avoir accueilli dans son équipe et offert tous les moyens nécessaires pour mener à bien mon stage de DEA.

Mes sincères remerciements vont à mon encadrant Monsieur Laurent Andrey qui a compté sur moi. Cette confiance, sans laquelle je n'aurais pas atteint mes objectifs. Je remercie Monsieur Laurent Andrey également pour avoir partagé mes soucis à chercher les bonnes solutions et pour ses conseils avisés.

Je voudrais également citer tous ceux et celles qui ont eu la patience de m'entourer tout au long de mon stage pour me prodiguer aide et soutien tant au plan professionnel que moral.

Résumé

La croissance exponentielle de l'Internet et la mutiplication des services et leur extention aux plate-formes mobiles rend les tâches de gestion des réseaux et services indipensables. Mais l'application de gestion distribuée devient de plus en plus contrainte par le plan fonctionnel qu'elle gère. Les travaux d'optimisation menés sur les applications de gestion élaborent leur tests de performances en se basant sur l'approche cetralisée de SNMP. Dans ce travail, on propose d'étudier le modèle *Gestionnaire/Agent* dans SNMPv3 pour revalider les tests précédents et estimer le coût de la sécurité et du passage à l'échelle.

Abstract

Exponential growth of internet and the number of services and their extention to mobile platforms make network and service management vital. But distributed management function becomes more constrained by the platform it manages. Performance tests done in works on optimization management applications are based on the centralized approach of SNMP. In this work, we propose to evaluate *Manager/Agent* model in SNMPv3 in order to revalidate previous tests and estimate the cost of security and scalability.

Introduction et Problématique

La gestion est une tâche vitale dans les réseaux et services. Avec l'expansion de l'Internet et la multiplication des services, le plan de gestion s'est intégré dans le plan fonctionnel. Il est devenu indispensable de maîtriser les fonctions de gestion pour gérer le réseau et les services de manière efficace sans influencer le plan fonctionnel.

Une plate-forme de gestion comme l'illustre la figure 1 est constituée d'un gestionnaire qui envoie des requêtes à un ensemble d'agents qui récupèrent les données de gestion de la base de données de gestion et les envoient au gestionnaire. Ces données de gestion décrivent l'état des agents et de leurs ressources. Toute ressource matérielle ou logicielle ayant un identifiant qui permet de l'identifier de manière unique. Une fonction de gestion est une application qui se déploie sur cette plate-forme pour gérer des ressources spécifiques dans le système de gestion.

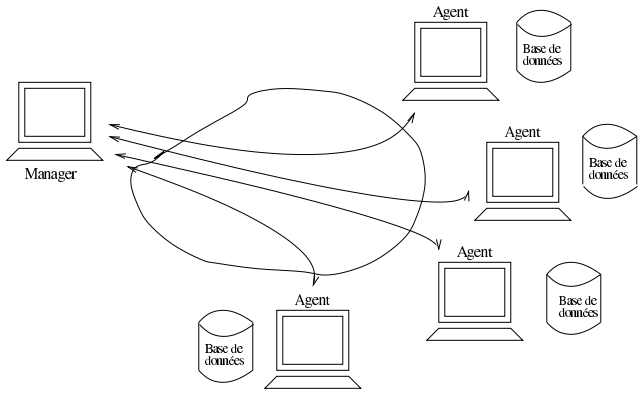


FIG. 1 – Le modèle d'une plate-forme de gestion

Plusieurs travaux visent à optimiser les applications de gestion. Ces travaux utilisent le modèle SNMP centralisé comme base de comparaison afin de prouver l'efficacité de leurs approches. Or, les performances du modèle SNMP sont peu discutées. En plus, les tests de performances effectués sur ces approches sont différents ce qui ne permet pas de comparer les approches proposées entre elles.

Dans ce projet, on propose d'évaluer le modèle *Gestionnaire/Agent*. Le but de cette évaluation est, d'abord, de revalider les scénarios de SNMP centralisé utilisés dans les études de performance des autres travaux. Ensuite, nous allons estimer le coût de la sécurité, devenue un critère d'efficacité de la fonction

de gestion, en terme de volume de trafic et de temps d'exécution.
On adopte le modèle SNMP/SMI dans notre travail vu qu'il est le plus répandu dans le domaine de la gestion des réseaux et services. En plus il est devenu incontournable de telle manière que toute nouvelle plate-forme de gestion prévoit une interface avec un agent SNMP. Nous avons considéré la troisième version du protocole SNMP puisqu'il est l'évolution naturelle de la gestion par SNMP et il offre des fonctionnalités de sécurité.

Dans le premier chapitre, nous allons présenter quelques plate-formes de gestion de réseaux existantes à savoir SNMP/SMI, CIM/WBEM et JMX, les caractéristiques des applications de gestion des réseaux et services et les travaux d'optimisation de ces applications. Dans le deuxième chapitre, nous allons présenter le protocole SNMPv3, ses aspects protocolaires et fonctionnels. Dans le troisième chapitre, nous allons spécifier les scénarios de gestion que nous allons utiliser pour les tests de performance, d'abord en terme de volume de trafic et ensuite en terme de charge CPU dans l'agent de gestion. Pour cette dernière étude, nous allons effectuer un benchmarking de SNMPv3 sur l'implémentation Net-SNMPv5.1 pour mettre en évidence le coût de l'authentification, du cryptage et du contrôle d'accès. Dans le quatrième chapitre nous allons procéder par une simulation sur ns-2 des scénarios de gestion.

Chapitre 1

Les plate-formes de gestion de réseaux et services

1.1 Les plate-formes de gestion des réseaux et services

1.1.1 Le modèle SNMP/SMI

Le modèle SNMP/SMI [9] a été développé par l'IETF (Internet Engineering Task Force) pour avoir une gestion simple des réseaux. Il est très répandu dans le domaine des réseaux est devenu quasi incontournable dans la gestion des réseaux et services. Le modèle se base sur les concepts : *structure d'information de gestion (SMI)* qui spécifie les règles de définition et de nommage des objets, *base d'information de gestion (MIB)* qui stocke de manière virtuelle ces objets, *agent / manager* et un protocole de communication entre eux : *Simple Network Manangement Protocole (SNMP).*

La première version de SNMP [9] implémente les fonctionnalités de base du modèle. La sécurité dans cette version se base sur une chaîne de caractère échangée en clair dans le réseau. La deuxième version de SNMP met à jour les opérations du protocole. La sécurité dans cette version se base sur SNMPsec défini dans [15]. Cette deuxième version est restée expérimentale. La troisième version du protocole [20, 30] vise essentiellement à inclure la sécurité des échanges entre le gestionnaire et l'agent, et la modularité dans modèle.

1.1.2 Le modèle CIM/WBEM

Web-Based Entreprise Management (WBEM) [13] est une approche unificatrice de gestion des réseaux et services basée sur les technologies et les standards de gestion existants *a priori* hétérogènes. L'architecture WBEM ne vient pas remplacer les approches existantes, mais elle cherche à intégrer l'information de gestion de toutes les approches existantes d'une manière uniforme et offre à l'utilisateur une vision unifiée de la gestion du réseau et des services.

Pour ce faire, le consortium DMTF (Distributed Management Task Force) a défini un support technologique de WBEM constitué de :

un *modèle d'information* CIMCommon Information Model (CIM), permet de décrire l'information de gestion des différentes plate-formes dans un format standard afin de pouvoir le partager entre toutes les applications de gestion,

un *codage spécifique* xmlCIM qui est une représentation des classes et des instances CIM en langage XML (eXtensible Markup Language),

un *mécanisme de communication* entre les entités de gestion HTTP(HyperText Transport Protocol).

Toutes les opérations définies dans la première version de WBEM sont de type *client/serveur* ce qui ne la rend pas plus avantageuse que les autres approches de gestion. Et les implémentations présentes du modèle CIM/WBEM sont peu utilisées.

1.1.3 Le modèle JMX

La gestion des réseaux s'élargit vers une gestion des services et des applications. Java Management eXtensions [12] est une initiative qui répond à deux objectifs : la supervision des applications Java et la supervision par Java. Elle propose en plus des solutions aux problèmes de dynamicité, de distribution des applications, de passage à l'échelle, et de l'évolutivité du modèle de gestion, à travers une architecture d'agent léger, flexible et extensible.

L'architecture JMX définit des interfaces de programmation qui permettent d'accéder depuis les objets JMX à des ressources distantes via d'autres protocoles de gestion comme SNMP ou WBEM. En plus l'architecture reste ouverte à l'intégration de nouveaux modèles de gestion encourageant ainsi le déploiement de cette architecture.

Le tableau de la figure 1.1 récapitule les principaux composants des trois modèles de gestion. Le modèle JMX ne propose pas de modèle d'information de gestion, les données de gestion sont spécifiés dans l'instrumentation Java de la ressource gérée.

Modèle	Modèle d'information	Codage de l'information	Protocole de transfert	Communication gestionnaire/agent	
				Synchrone	Asynchrone
SNMP/SMI	SMI	ASN.1	SNMP	+	+
CIM/WBEM	CIM	XML	HTTP	+	-
JMX	Java	Sérialisation Java	RMI &JSR160	+	+

FIG. 1.1 – Les modèles de gestion

1.2 Les applications de gestion des réseaux et services

1.2.1 Les types de applications de gestion

La gestion des réseaux et services consiste en un ensemble de fonctions qui contrôlent, déploient et gèrent les ressources du réseau et des services de bout en bout. Il y a quatre catégories de fonctions : les fonctions de gestion des événements, les fonctions de planification, les fonctions de configuration et les fonctions de gestion de pannes. Une fonction de gestion est efficace si elle remplit son rôle de supervision et de contrôle à moindre coût et si elle s'adapte aux les changements des environnements gérés.

Il y a deux types majeurs d'applications de gestion :

Les applications d'analyse : elles consistent à collecter les informations de gestion du réseau pour de longues périodes afin de construire un modèle d'évolution du réseau et des services. La fréquence de polling[1] dans ce type d'applications n'est pas très élevée.

Les applications réactives : permettent de gérer le réseau en temps réel, détecter les problèmes dans le système et chercher à les résoudre par des actions appropriées. Pour ce faire, il faut avoir une grande fréquence de polling pour pouvoir détecter les problèmes à temps.

1.2.2 L'approche distribuée de déploiement des applications de gestion

1.2.3 L'approche centralisée

Il existe trois approches de déploiement des modèles de gestion : centralisée, hiérarchique et distribuée. Dans l'approche centralisée, une seule station de gestion se charge d'envoyer des requêtes de gestion et de recevoir les réponses des agents. Si cette approche présente l'avantage de fournir un point unique de gestion des évènements du réseau et un contrôle plus fiable de la sécurité, elle atteint vite ses limites vis à vis de l'expansion rapide des réseaux gérés, des applications et des services.

1.2.4 L'approche hiérarchique

L'approche hiérarchique, basée sur un partage de la responsabilité de gestion, permet de distribuer les tâches de gestion sur différents niveaux, et de mieux supporter les pannes du réseau. Dans cette approche chaque station de gestion a une sous-tâche de gestion. On trouve des stations qui se chargent de la collecte des informations, d'autres qui font le traitement des données. La station de l'opérateur se trouve à la racine de l'arbre et peut avoir une sous-tâche de gestion comme elle peut servir seulement d'interface utilisateur. Cette approche a été appliquée dans le cas de la gestion par délégation de SNMP [25, 14, 3]. Mais, elle souffre des problèmes de synchronisation entre les différentes stations de gestion, des longues boucles de contrôle, de l'augmentation de la complexité et du coût de gestion avec la taille du réseau.

[1] Le polling est l'envoi des requêtes de manière périodique à un ensemble de nœuds pour avoir des informations sur leur état et celui de leurs ressources

1.2.5 L'approche distribuée

L'approche distribuée 1.2 combine les avantages des deux premières approches en offrant un système de gestion paire. Les gestionaires ont tous les mêmes responsabilités et ont tous accès aux informations de gestion de tout le réseau. Ils effectuent tous des analyses et des traitements facilitant ainsi la surveillence globale du réseau tout en optimisant le coût de gestion en terme de temps de traitement et de trafic généré. Dans cette approche, la station manager n'est plus qu'une interface pour l'opérateur qui sert à la visualisation. Néanmoins, cette approche présente des difficultés au niveau du partage des données de gestion et du degré de coopération entre les différentes stations de gestion.

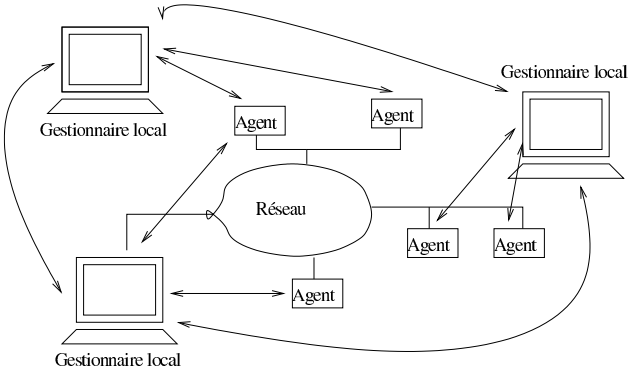


FIG. 1.2 – L'approche distribuée

Dans cette approche, chaque gestionnaire gère un sous ensemble d'agents du système de gestion. ensuite, il publie les données de gestion pour qu'elles soient connues par les autres gestionnaires. Ainsi, chaque gestionnaire a une vision globale du système de gestion mais il n'est responsable que d'une partie du système. On retrouve alors le modèle centralisé de la gestion dans le fait que le système géré est repartie en sous-systèmes où chacun est géré par un seul manager. L'aspect hiérarchique de l'approche distribué se manifeste dans le processus d'agrégation des données de gestion par les managers.

1.2.6 L'évaluation et l'optimisation des performances des applications de gestion

L'application de gestion des réseaux et services est une application distribuée qu'on met en oeuvre dans le plan fonctionnel pour le superviser. En effet, le plan de supervision est intègré dans le plan fonctionnel. Cette application doit remplir ses fonctions de manière efficace tout en optimisant son utilisation des ressources du réseau en terme de trafic de gestion généré et en terme de temps de traitement dans les nœuds du réseau. Ces paramètres dépendent étroitement de la configuration de la plate-forme, du type des fonctions de gestion et leur besoin en terme de temps CPU et bande passante.
Plusieurs travaux d'optimisation ont été effectués. Nous avons classé ces travaux dans quatre catégories que nous allons présenter dans la suite.

1) Les modèles et les algorithmes d'exécution des fonctions de gestion

A. Bauer propose dans [17] un modèle d'exécution des fonctions de gestion dans les systèmes distribués qui permet d'avoir la meilleure configuration de la plate-forme de gestion selon les opérations de gestion. Ce modèle permet d'attribuer à chaque politique de gestion : type d'opérations et type d'échange entre les agents, une configuration des agents de gestion : arrangement des agents et localisation des traitements. Le choix de la configuration est un consensus entre l'emplacement des agents et le type des ressources que l'on veut optimiser. La complexité du modèle augmente avec la complexité des opérations de gestion ce qui rend la convergence vers la configuration optimale difficile voire irréalisable dans certains cas.

Dans [8], une heuristique est proposée pour optimiser le trafic de gestion et le temps de traitement dans une plate-forme de supervision basée sur les sondes[2]. Ce papier vise à optimiser le délai d'exécution de bout en bout de la commande *traceroute*, utilisée pour la détection des congestions dans le réseau. Il propose un algorithme qui permet de choisir le nombre de sondes minimal que la station de gestion doit envoyer pour couvrir tout les nœuds du système sans participer à la congestion.

Un autre problème auquel les applications de gestion doivent faire face concerne le changement rapide des environnements gérés. Pour rester efficaces, les applications de gestion doivent s'adapter à ces changements et garder des informations de gestion les plus fraîches possibles. Dans [7], on propose de réduire le coût de gestion à travers un modèle d'anticipation des changements de l'environnement qui permet de déduire la fréquence de polling adéquate pour satisfaire les besoins de l'application en terme de granularité des données de gestion sans pénaliser la bande passante.

2) L'instrumentation du modèle gestionnaire/agent

R. Boutaba propose dans [28] un module de polling adaptatif dans le gestionnaire. Ce module se base sur la classification des agents selon des critères et des politiques de gestion, administratifs ou tout simplement géographiques. Chaque classe d'agents a une fréquence de polling unique qui s'adapte moyennant un sous-module d'adaptation aux variations du comportement du système de gestion. Ce module permet d'optimiser le trafic de gestion en adaptant la fréquence du polling aux composants du système distribué.

Une autre approche [21] vise à améliorer le coût de gestion en minimisant le trafic de gestion dans le modèle SNMP/SMI introduit le concept d'une couche de polling qui permet de gérer le polling redondant sur les mêmes données par des applications différentes et d'assurer une gestion efficace des applications nécessitant une bonne granularité des données de gestion.

Réduire le coût de gestion est aussi l'objectif dans [10] en optimisant l'accès à la MIB et donc le temps de traitement de la requête dans l'agent. [10] propose un environnement d'exécution des opérations de gestion. L'agent de gestion est muni d'une machine active qui gère les requêtes de gestion qui parviennent. Il permet de partager les informations de gestion entre les différentes applications de gestion en minimisant

[2] une sonde est un programme exécutable qui a des fonctionnalités spécifiques qui est envoé d'un nœud source vers un nœud destination pour collecter des informations

ainsi le temps de traitement de la requête dans l'agent. La machine active utilise un agent SNMP pour accéder aux variables de gestion dans l'équipement.

3) La technologie des agents mobiles

La technologie des agents mobiles est utilisée pour améliorer les performances de gestion [25, 3]. Elle permet de rapprocher le traitement des données ce qui augmente considérablement l'efficacité de l'application distribuée. Aussi, la technologie des agents mobiles permet de surmonter le problème de l'hétérogénéité des composants gérés. Un agent mobile est une entité autonome qui a la capacité de coopérer avec des applications et de naviguer dans le réseau d'un nœud à l'autre.

G. Pujolle présente dans [24] une comparaison entre la gestion basée sur SNMP centralisé et la gestion basée sur un agent mobile qui se déplace dans les nœuds du réseau et collecte les informations de gestion. L'application mise en oeuvre dans cette comparaison consiste à rechercher une variable de la MIB dans un système d'agents de gestion dont le nombre varie entre 1 et 250. [24] assume que, dans la gestion par SNMP centralisé, le manager envoie la requête à un agent et attend la réponse de celui-ci avant d'envoyer la requête à l'agent suivant. Ce modèle d'exécution (Polling en série) n'est pas obligatoirement adopté par les applications de gestion. Le manager SNMP peut envoyer la requête à un agent avant de recevoir la réponse de l'agent précédent (Polling parallèle). Ceci étant, l'étude comparative de [24] prouve que la gestion basée sur les agents mobiles permet d'optimiser le trafic de gestion. Quant au temps de traitement, les résultats présentés ne sont valables que dans le scénario d'exécution du polling en série.

La gestion basée sur les agents mobiles prouve son efficacité vis à vis du modèle SNMP centralisé en terme de trafic de gestion quant au passage à l'échelle du nombre d'entités gérées. Cependant le temps de traitement dans la technologie des agents mobiles reste élevé comparé à d'autres modèles de gestion.

4) Les patterns de navigation

Le concept des patterns de navigation [18] définit une abstraction du flux de contrôle d'exécution des opérations de gestion. Etant donnés une topologie, un état du réseau et une opération de gestion, le patron de navigation définit le graphe d'exécution de l'opération, le degré de parallélisme et la synchronisation de l'exécution distribuée et permet enfin d'agréger les résultats vers le gestionnaire. L'objectif des patterns de navigation est de séparer la sémantique de l'opération et le flux de contrôle. Les patterns de navigation permettent de définir pour chaque opération de gestion, le flux approprié pour minimiser la complexité du trafic et le temps d'exécution.

R. Stadler propose dans [19] un pattern de gestion et un environnement d'exécution de ce pattern. Il intègre la réaction de l'application de gestion vis-à-vis des erreurs du réseau, il contrôle la mobilité du code dans les nœuds du système et il offre une optimisation pour la distribution du code de l'application de gestion. La station de gestion envoie le code à un premier nœud du système. Ce dernier se charge de déterminer le pattern de navigation approprié et l'exécute. [19] a élaboré une comparaison entre le pattern de gestion proposé et SNMP centralisé avec un polling parallèle en considérant le temps d'exécution de l'application de gestion comme critère de comparaison. Il y a une optimisation très nette du

temps d'exécution avec le pattern de gestion quant au passage à l'échelle du nombre des équipements gérés.

Le temps d'exécution de SNMP centralisé pris en considération dans cette comparaison est celui calculé sur l'environnement d'exécution proposé pour supporter le pattern de gestion alors que le modèle SNMP n'a pas besoin de cet environnement. En plus dans le calcul du temps d'exécution de SNMP centralisé avec un polling parallèle, le temps de traitement dans l'agent n'est pas pris en considération.

1.2.7 Conclusions

La sécurité

Les travaux réalisés dans le cadre d'optimisation des applications de gestion se focalisent sur l'opération de gestion elle même et néglige les fonctions supplémentaires qui contribuent au bon déroulement de l'opération.

L'efficacité d'une application de gestion dépend aussi de la qualité des données qu'elle fournit. Si les données de gestion sont modifiées au cours de la route ou bien arrivent avec de grands délais, l'application de gestion n'est plus efficace. L'évaluation des fonctionnalités de sécurité de la gestion doit s'intégrer dans le cadre d'optimisation des applications de gestion.

Les tests de performance

Les modèles d'optimisation proposés sont soutenus par des études de performances qui montrent leur efficacité. Ces études de performances se basent sur des scénarios de gestion qui diffèrent d'un travail à l'autre ce qui ne permet pas d'établir une étude comparative entre les différentes approches.

Pour que les études de performances de deux modèles de gestion différents soient comparables, il faut utiliser la même plate forme de tests. [24] étudie les performances des agents mobiles dans la gestion avec SNMP en comparaison avec SNMP centralisé. Il utilise des variables de la MIB *System* pour les tests de performances. [1] fait du benchmarking de SNMP. Il utilise une MIB expérimentale construite pour ce but. Les tests da performance des deux travaux ne sont pas comparables puisqu'ils ne considèrent pas la même-plate forme de test.

Le passage à l'échelle

Dans la gestion des réseaux et services, le système de gestion peut se composer de milliers d'agents. [24] prouve le bon comportement des agents mobiles quant au passage à l'échelle en comparaison avec SNMP centralisé dans un système de gestion qui ne dépasse pas 250 agents. Mais avec la croissance rapide de l'Internet, un système géré peut dépasser les centaines d'agents.

Chapitre 2

Le protocole SNMPv3

Dans ce chapitre, nous allons présenter le protocole SNMPv3. En effet, une compréhension fine du protocole est nécessaire pour mener à bien nos analyses des données numériques.
Dans la première section, nous allons présenter l'architecture de SNMPv3. Dans la deuxième section, nous allons présenter les aspects protocolaires de SNMPv3 et ses fonctionnalités de sécurité. La troisième section portera sur les aspects fonctionnels du protocole. Dans la quatrième section nous allons présenter l'implémentation Net-SNMPv5.1.

2.1 L'architecture de SNMPv3

L'architecture de l'entité SNMPv3 2.1 est construite de façon modulaire pour permettre l'ajout ou la suppression de certains modules. Une entité SNMP est constituée de deux modules principaux :

- **La machine SNMP** identifiée par un $snmpEngineID$ (construit à partir de l'adresse MAC et l'adresse IP de la machine). Ce module est responsable du traitement du message. Il reçoit et expédie les paquets SNMP à la couche transport, contrôle la version du message, les paramètres de la sécurité et le contrôle d'accès des utilisateurs. La machine SNMP est constituée de quatre sous-systèmes :
  - *L'expéditeur de messages* qui est responsable de l'envoi et de la réception des paquets SNMP et de gestion du flux de données entre les différents modules de l'entité,
  - *Le processeur de messages* définit le format du message selon la version de SNMP et prépare l'extraction des données du message,
  - *Le sous-système de sécurité* fournit les services de sécurité,
  - *Le sous-système de contrôle d'accès* définit les droits d'accès des utilisateurs aux variables de la MIB.

- **Les applications** traitent les requêtes contenues dans le message SNMP. *Le générateur de commandes* manipule les données de gestion. *Le Répondeur de commandes* permet l'accès aux données de la MIB. *Le générateur de notifications* crée les messages asynchrones d'alarmes. *Le récepteur de notifications* traite les messages asynchrones. *Le proxy* achemine les messages entre deux entités SNMP.

L'agent et le manager n'implémentent pas les mêmes applications et ne traitent pas les messages de la même manière. Le manager implémente les applications suivantes :

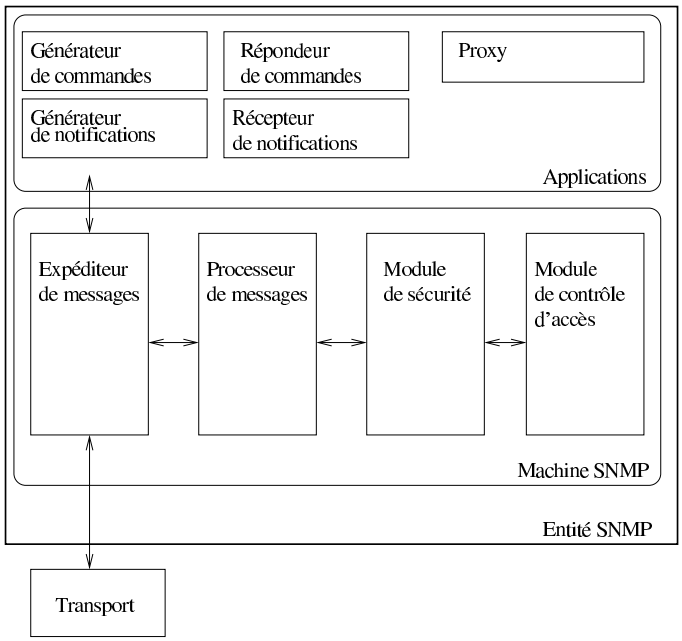


FIG. 2.1 – L'architecture d'une entité SNMPv3

- Générateur de commandes : gère et manipule les données distantes par des requêtes SNMP.
- Générateur de notifications : génère des messages asynchrones de notification pour les autres managers.
- Récepteur des notifications : reçoit les alarmes des agents et des managers.

L'agent implémente, comprend trois applications :

- Répondeur de commandes : répond aux requêtes du manager.
- Générateur de notifications : génère des messages asynchrones de notification.
- Proxy : qui achemine les messages entre les entités SNMP.

L'agent est muni d'un module de contrôle d'accès qui permet de contrôler l'accès des utilisateurs aux variables de la MIB.

2.2 Les aspects protocolaires

2.2.1 Les opérations de SNMPv3

Le fonctionnement de SNMP est basé sur un ensemble de requêtes, de réponses et sur un nombre limité d'alertes. Il y a trois type d'échanges entre l'agent et le manager :

- L'échange de Requêtes/Réponses : La station de gestion envoie des requêtes à l'agent qui retourne des réponses.
- L'échange d'Alertes : Lorsqu'un évènement anormal surgit dans l'élément géré, celui-ci envoie une alerte au manager.
- L'échange Inform/Réponses : C'est l'échange des informations de gestion entre deux managers dans un système de gestion. Le message SNMPv3 qui permet cet échange est $Inform - PDU$.

SNMP utilise le protocole UDP avec le port 161 pour recevoir les requêtes de la station de gestion et le port 162 pour recevoir les alertes des agents.
SNMP compte quatre types de requêtes :
. La requête *GetRequest* permet la recherche d'une instance d'objet[3] sur l'agent.
. La requête *GetNextRequest* permet la recherche de l'instance de l'objet suivant.
. La requête *GetBulk* permet la recherche d'un ensemble d'instances d'objets.
. La requête *SetRequest* permet de changer la valeur d'un objet sur un agent.
A ces requêtes, l'agent répond par un message *GetResponse*.

Les alertes sont envoyées quand un évènement se produit qui pourrait intéresser la station de gestion. Les alarmes possibles sont :*ColdStart, WarmStart, LinkDown, LinkUp, AuthenticationFailure*.

2.2.2 Le format du message SNMPv3

Les requêtes, les réponses et les alarmes SNMP sont transportées dans des PDUs SNMP. Une PDU SNMP est encapsulée dans un message SNMP. Le message SNMPv3, comme l'illustre la figure 2.2, est constitué des champs suivants :

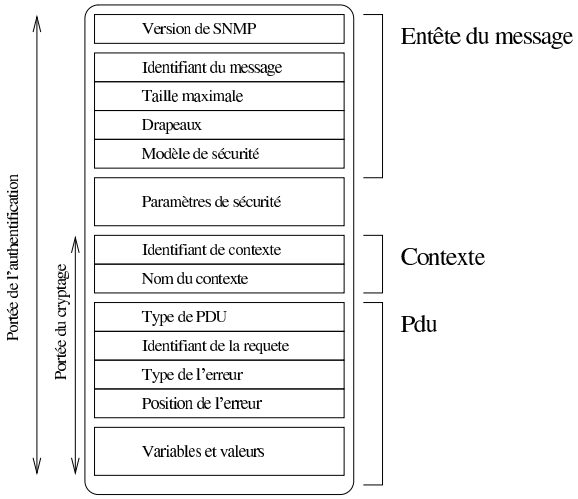


FIG. 2.2 – Le format du message SNMPv3

1. **La version** : indique la version du protocole SNMP
2. **Identifiant du message** : utilisé pour coordonner les requêtes et les réponses entre l'agent et le manager. Il est aussi utilisé par le module de traitement des messages pour coordonner le traitement des message par les différents modules de l'entité.
3. **Taille maximale** : indique la taille maximale du message que le manager peut accepter d'un agent.
4. **Drapeaux** : un champ de un octet. Seuls les trois bits du poids le plus faible de l'octet sont utilisés. Le premier bit est le drapeau de l'authentification, le deuxième est celui du cryptage. Alors, les deux

[3]un objet est une variable d'une MIB qui sont nommés

premiers bits indiquent le niveau de sécurité du message : $noAuthNoPriv$(message en clair sans authentification), $authNoPriv$ (message authentifié) ou $authPriv$(message authentifié et crypté). Le troisième bit signale si le message attend un rapport en retour.

5. **Modèle de sécurité** : l'architecture de SNMPv3 permet la co-existence de plusieurs modèles de sécurité. Ce champ désigne le modèle utilisé dans le message. Exemple :USM.
6. **Paramètres de sécurité** : contient les paramètres de sécurité du message.
7. **Identifiant et Nom de contexte** : ces deux chaînes d'octets identifient de manière unique dans l'entité SNMP le contexte auquel la PDU est destinée.
8. **Type de PDU** :désigne le type de la PDU.
9. **Identifiant de la requête** : identifiant du PDU.
10. **Type de l'erreur** : un entier qui indique le type de l'erreur dans la PDU si il y en a.
11. **Position de l'erreur** : un entier qui indique l'emplacement de l'erreur dans la PDU.
12. **Variables et valeurs** : la liste des varbinds[4] de la requête.

2.2.3 La taille du message SNMPv3

ASN.1 (Abstract Syntax Notation One) est un langage formel qui définit une **syntaxe de définition** de la donnée et une **syntaxe de transfert** qui définit la façon dont les données sont converties sans ambiguïté en une suite d'octets pour la transmission. ASN.1 utilise le codage BER (Basic encoding Rules). Il code chaque donnée en trois champs : $Type$, $Longueur$ et $Valeur$. ASN.1 définit des types simples comme les entiers et les chaînes d'octets, et des types structurés qui sont une composition de plusieurs types simples.

Les champs *Type et Longueur*

Dans le codage BER, le nombre d'octets allouées aux deux champs $type$ et $longueur$ est :
- le champ $Type = 17 + \sum_{k=1}^{n} 2 * NumberOfVarbinds$.
- le champ $Longueur \geq 17 + \sum_{k=1}^{n} 2 * NumberOfVarbinds$

Le champ *Valeur*

La taille des champs $Valeur$ dans le codage BER du message SNMPv3 dépendent de du nombre des variables dans la liste des varbinds de la construction des différents champs du message. L'annexe B detaille le codage BER du message SNMPv3 et la taille en octets attribuée a chaque champ selon les spécifications [5] et [6].

2.2.4 La sécurité dans SNMPv3

Le modèle de sécurité USM

Le module USM (User based Security Model) assure l'authentification et le cryptage de données. Il est capable de protéger les messages SNMPv3 contre le rejeu, contre les délais, contre les changements au

[4]un varbind est un couple identifiant d'objet-valeur par exemple : [sysUpTime , 15248]

cours de la transmission et contre la lecture des données par les entités tierces.

Le bloc de sécurité du message SNMP porte alors six valeurs comme l'illustre la figure 2.3.

msgAuthoritativeEngineID
msgAuthoritativeEngineBoots
msgAuthoritativeEngineTime
msgUserName
msgAuthenticationParameters
msgPrivacyParameters

FIG. 2.3 – Les paramètres de sécurité du modèle USM

- **msgAuthoritativeEngineID** : c'est un champ de type OCTET STRING dont la taille varie entre 5 et 32 octets, c'est lidentifiant de l'entité de référence dans la communication entre les deux entités SNMP.
- **msgAuthoritativeEngineBoots** : un entier entre 0 et $2^{31} - 1$, désigne le nombre de boots de l'entité de référence.
- **msgAuthoritativeEngineTime** : un entier entre 0 et $2^{31} - 1$, indique le temps en secondes dans l'entité de référence.
- **msgUserName** : un champ de type OCTET STRING qui indique le nom de l'utilisateur dont les clés secrètes sont utilisées pour authentifier et crypter le paquet.
- **msgAuthenticationParameters** : un champ de type OCTET STRING. Si le message est authentifié, ce champs comprend les douze premiers octets de l'empreinte du message calculée par un algorithme de hachage.
- **msgPrivacyParameters** : un champ de type OCTET STRING de 8 octets. Si le paquet est crypté, ce champ comporte le vecteur d'initialisation utilisé pour initialiser l'algorithme de cryptage.

a- L'authentification

Le modèle USM propose deux algorithmes d'authentification : HMAC-MD5 qui génère une empreinte de seize octets et HMAC-SHA qui génère une empreinte de vingt octets. Le modèle USM n'utilise que les 12 premières octets de cette empreinte qu'il met dans le champs *msgAuthenticationParameters* du message authentifié.

b- Le mécanisme d'horodatage

Chaque machine SNMP doit maintenir deux objets *snmpEngineTime* qui indique le temps du système en secondes et snmpEngineBoots qui indique le nombre de boots du système depuis l'installation de l'agent. Le manager maintient localement une notion de ces deux objets pour chaque machine de référence. Le manager met ces valeurs estimées dans tout message destiné à la machine de référence ce qui permet à celle-ci de déterminer si le message n'est pas retardé ou rejoué. Le manager garde trois objets pour chaque machine de référence :

. *snmpEngineBoots* : contient la valeur la plus récente reçue de l'agent.
. *snmpEngineTime* : la valeur du temps dans l'agent. cette valeur est incrémentée chaque secondes pour garder la synchronisation du manager avec la machine de référence.
. *latestreceivedEngineTime* : stocke la plus grande valeur reçue de *snmpEngineTime*. Elle permet au manager d'éviter le rejeu des messages.

c- Le cryptage

Le modèle USM utilise l'algorithme de sécurité CBC-DES. Le message est divisé en bloc de 8 octets. Si la taille du message n'est pas multiple de 8, on ajoute des octets nuls pour arrondir. L'algorithme effectue le cryptage des données bloc par bloc. Il y a trois entrées : la clé de cryptage de 56 bits, le bloc de données de 64 bits à crypter et le bloc crypté de l'itération précédente. Pour démarrer l'algorithme un vecteur d'initialisation constitue la troisième entrée. Ce vecteur est construit à partir du nombre de boots et d'un entier aléatoire. Ce vecteur est mis dans le champ *msgPrivacyParameters* du message.

La découverte des agents

Lors d'une communication entre deux entités SNMP, une des deux est désignée comme étant de référence, dont les paramètres de sécurité sont mis dans le bloc de sécurité du message. l'entité dite de référence est celle qui reçoit le message SNMP qui exige une réponse, en général l'agent.
Pour adresser une requête à un agent, le manager doit connaître des informations sur l'agent et l'agent doit connaître des informations sur le manager à savoir :
- Nom d'utilisateur.
- Le protocole d'authentification.
- Le mot de passe secret qui doit être utilisé pour calculer l'empreinte. Ce mot de passe doit avoir une longueur de 16 octets si on utilise MD5 et 20 octets si on utilise SHA1.
- Le protocole de cryptage.
- Le mot de passe secret pour le cryptage qui a une longueur de 16 octets. Ces informations sont stockées dans une base de données persistante, *usmUserTable*.

Avant d'envoyer une requête, le manager doit connaître l'identifiant de l'entité SNMP distante *snmpEngineID* , snmpEngineboots et snmpEngnetime de l'entité de référence. Le processus de découverte permet de fournir ces données au manager. Il y a deux phases dans le processus de découverte : La première permet d'avoir l'identifiant de l'entité *snmpEngineID* distante. La seconde découvre le *snmpEngineBoots* et le *snmpEngineTime*. Cette dernière étape n'est accomplie que quand on exige une authentification.

Le modèle VACM

Le modèle View Access Contol Model de l'entité SNMP a pour rôle de contrôler les droits d'accès à un objet de la MIB par un utilisateur. VACM utilise les données : niveau de sécurité, modèle de sécurité, nom du contexte et les variables de la requête pour déterminer les droits d'accès d'un message SNMP. une PDU SNMP contient une opération qui doit être appliquée à un objet identifié par un OID et un contexte. Si une PDU contient plusieurs OIDs, le contrôle d'accès est vérifié individuellement pour chaque variable.

2.3 Les aspects fonctionnels de SNMP

2.3.1 L'accès aux ressources avec SNMP

Dans la gestion basée sur SNMP, un agent doit être capable d'accéder aux informations de gestion. Pour ce faire, le système géré doit contenir les blocs suivants :

- Un agent SNMP,
- une pile da transport en mode datagramme,
- des sous-systèmes contenant des informations de gestion,
- un traducteur de définitions entre le sous-système et l'agent SNMP,
- un mécanisme d'accès aux informations de gestion.

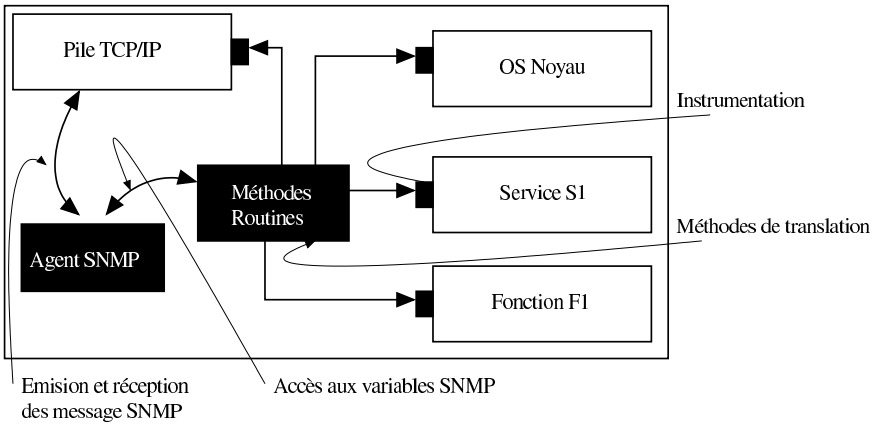


FIG. 2.4 – Les éléments d'un système géré par SNMP

La figure 2.4 montre les modules qui existent déjà dans un équipement, représentés par des blocs blancs, et les modules qu'il faut ajouter pour rendre l'équipement gérable par SNMP, représentés par des blocs noirs :

L'agent SNMP est l'entité de traitement. Il reçoit les messages de la couche transport, récupère les informations de gestion des sous-systèmes (OS noyau, Services, fonctions) et retourne les réponses et les alarmes.

Les routines et méthodes permettent de faire la traduction entre la structure des informations de gestion telles qu'elles sont définies dans la plate-forme SNMP/SMI, et leur structure dans les sous-systèmes. La plupart des systèmes ont été conçus sans tenir compte de la gestion par SNMP, devenue une nécessité.

L'instrumentation permet l'accès à l'information de gestion d'un sous-système. C'est un logiciel ou matériel additionnel qui permet de calculer et retenir les informations de gestion du sous-système pour qu'elles soient accessibles par l'agent.

2.3.2 L'agent SNMP

Les composants fonctionnels de l'agent SNMP

Un agent SNMP est constitué de trois aires fonctionnelles comme l'illustre la figure 2.5.

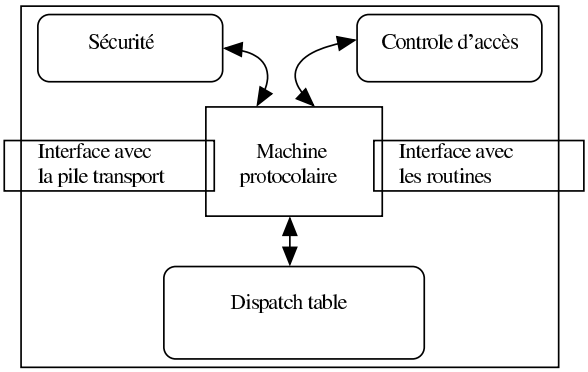


FIG. 2.5 – Les aires fonctionnelles de l'agent SNMP

- L'accès au transport pour échanger les messages SNMP avec les stations de gestion.
- La machine protocolaire qui remplit les fonctionnalités du protocole SNMP (les requêtes et réponses, sécurité ...).
- La dispatch table qui permet le mapping entre l'identité de la variable SNMP(OID) et la méthode qui permet sa lecture ou son écriture de l'instrumentation.

Chapitre 3

Les fonctions de gestion et l'influence de la sécurité

Dans ce chapitre, nous allons en premier lieu proposer des scénarios de gestion simples, et qui utilisent des variables qui se trouvent dans la MIB-II. Ces variables ont la particularité d'être significatifs dans le domaine de la gestion des réseaux et services et sont disponibles dans tout système qui utilise le modèle de gestion SNMP.
Ensuite, nous allons évaluer l'influence de la sécurité sur les scénarios de gestion en terme de volume de trafic de gestion.
Puis, nous allons procéder par un benchmarking de SNMPv3 sur l'implémentation Net-SNMPv5.1 (voir annexe B) pour estimer le surcoût du temps de traitement dans les agents.

3.1 Les fonctions de gestion

3.1.1 Les variables de gestion

On considère les fonctions de gestion de la figure 3.1. Le premier scénario permet de connaître le nombre des paquets IP reçus dans un agent. Cette information sert à estimer le taux d'utilisation de la bande passante par un client et permet au gestionnaire de localiser les clients responsables en cas de congestion dans le réseau et isoler les sources de problèmes. Aussi, cette information permet de déterminer le volume des données qui transitent dans les réseaux de cœur et les réseau d'accès. L'identifiant de l'objet de la MIB qui porte cette information est le scalaire *ipInReceives*.

Le deuxième scénario permet de récupérer la table *ARP* (Address Resolution Protocol) qui stocke les informations sur les adresses IP et MAC des machines dans le réseau. Cette donnée permet au gestionnaire de construire la topologie du réseau. Chaque nœud dans le réseau stocke des informations dans la table *ARP* sur les autres nœuds avec qui il est en communication. Chaque nœud est identifié par son adresse IP et son interface de communication et occupe une entrée dans cette table.

La structure de la table *ARP* est représentée dans la figure 3.2. Elle est constituée de quatre vecteurs :

Application de gestion	Les objets
Rechercher le nombre de paquets IP reçus	ipInReceives (scalaire)
Rechercher la table ARP (table)	ipNetToMediaIfIndex
	ipNetToMediaPhyAddress
	ipNetToMediaNetAddress
	ipNetToMediaType

FIG. 3.1 – Les tâches de gestion

- $ipNetToMediaIfIndex$: identifie l'interface de communication. Par exemple l'interface ethernet.
- $ipNetToMediaPhyAddress$: porte l'adresse physique.
- $ipNetToMediaNetAddress$: porte l'adresse IP.
- $ipNetToMediaType$: identifie le type de l'entrée si elle est statique, dynamique, ou invalide.

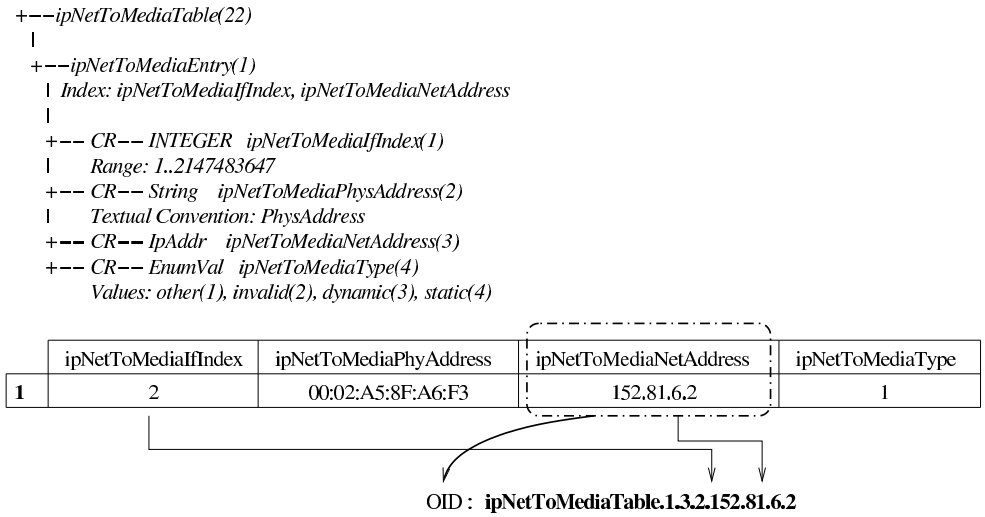


FIG. 3.2 – La structure de la table ARP

La table ARP est indexée par $ipNetToMediaIfIndex$ et $ipNetToMediaNetAddress$. C'est à dire, le OID de chaque instance d'objet de cette table est la concaténation, dans cet ordre, du nom du vecteur auquel appartient l'instance, la valeur du champ $ipNetToMediaIfIndex$ et la valeur du champ $ipNetToMediaNetAddress$ de l'entrée à laquelle appartient l'instance.

3.1.2 Les scénarios de gestion

La variable *ipInReceives* est un scalaire dont on connaît le OID. Nous allons utiliser la requête *get* pour rechercher cette variable.

Dans la deuxième fonction de gestion, il s'agit de ramener une table dont on ne connaît pas la taille a

priori. On peut utiliser l'une des requêtes *getwalk*[5] ou *getbulk* (voir l'annexe A). La requête *getwalk* permet de ramener un OID à chaque réponse. Le nombre de messages SNMP échangés entre un manager et un agent serait de $2 \times 4 \times (TailleARP + 1)$.

En revanche, si on utilise la requête *getbulk* on arrive à récupérer toute la table en une seule requête pourvu que l'on connaîsse sa taille. Toutefois, La taille de la table ARP est variable dans le temps dans un nœud et varie selon la nature du matériel géré (serveur DNS, serveur Mail, simple utilisateur...). Si on utilise *getbulk* avec des paramètres inadéquats, on risque de récupérer des informations non significatives ce qui compromet l'efficacité de la tâche de gestion.

Pour palier à ce problème, on propose d'utiliser la requête *getwalk* pour découvrir le premier vecteur *ipNetToMediaIfIndex* de la table ARP à des intervalles larges dont la taille dépend de la dynamicité du réseau pour avoir une idée sur la taille de la table. La requête *getbulk* sera utilisée dans des intervalles plus fins pour assurer la fonction de supervision.

3.2 L'impact de la sécurité sur la taille du trafic de gestion

3.2.1 Les paramètres de sécurité

Selon le niveau de sécurité du message, la taille du bloc de sécurité varie. Nous allons prendre le niveau de sécurité $noAuthNoPriv$ comme niveau de base pour estimer le nombre d'octets ajouté à la taille du trafic de gestion.
Le tableau de la figure 3.3 indique les champs utilisés dans le bloc de sécurité pour chaque niveau dans le modèle USM.

Les champs *snmpEngineID* et *UserName* sont communs à tous les niveaux de sécurité. Ces deux champs sont indispensables pour établir la communication entre l'agent et le manager.

Dans le niveau de sécurité *noAuthNoPriv*, les champs *snmpEngineBoots* et *snmpEngineTime* sont des entiers qui prennent la valeur 0 et ils sont non significatifs, c'est à dire qu'ils ne sont pas pris en compte pendant le traitement du message dans le module de sécurité. Les champs des paramètres de sécurité sont nuls (0 octets).

Dans le niveau *authNoPriv* le champs *authenticationParameters* porte le code MAC du message sur 12 octets et le champ qui porte les paramètres de cryptage reste nul[6].
Dans le niveau *authPriv*, le champs *privacyParameters* porte le vecteur d'initialisation de l'algorithme de cryptage sur 8 octets.

[5] *getwalk* est une commande qui consiste à faire une série de requêtes *getnext* de telle façon à ramener une partie de MIB(une table ou un vecteur)

[6] Un champ nul est codé en BER avec un type primitif $NULL$ qui a un champs *longueur* égal à zéro et un champ $Valeur$ qui n'existe pas

Niveau de sécurité	bloc de sécurité	Taille suplémentaire
noAuthNoPriv	msgAuthoritativeEngineID EngineBoots = 0 EngineTime = 0 UserName	
authNoPriv	msgAuthoritativeEngineID EngineBoots EngineTime UserName AuthenticationParameters	12 octets
authPriv	msgAuthoritativeEngineID EngineBoots EngineTime UserName AuthenticationParameters PrivacyParameters	20 octets

FIG. 3.3 – Les champs du bloc de sécurité selon le niveau de sécurité

3.2.2 L'exemple de la recherche de la variable *ipInReceives*

La figure 3.4 montre la taille supplémentaire induite par le niveau de sécurité *authNoPriv* et *authPriv* pour rechercher la variable *ipInReceives.0* en variant le nombre des agents dans le système de gestion. Le volume supplémentaire pour un échange entre un agent et le manager est la somme du nombre d'octets des paramètres de sécurité dans la requête et dans la réponse. Par exemple, si le manager utilise la niveau de sécurité $authPriv$, le volume supplémentaire dans un échange est de 40 Octets.

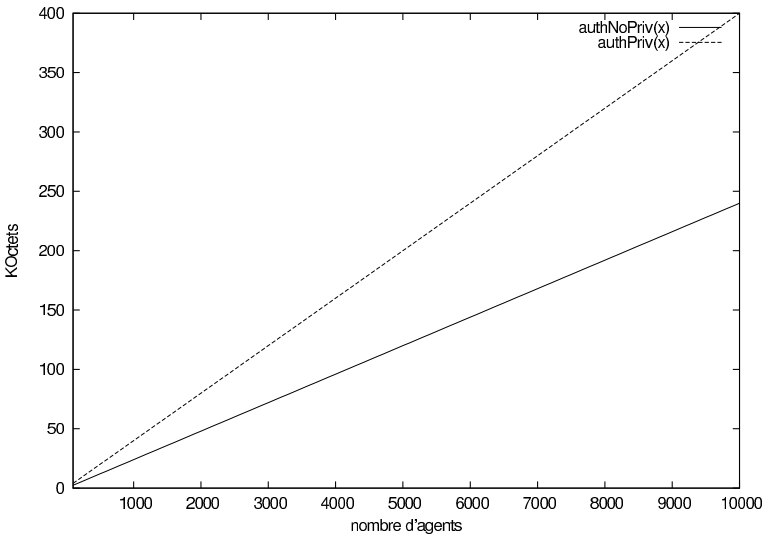


FIG. 3.4 – La taille du trafic supplémentaire de gestion pour la requête *get*

On appelle cycle un intervalle de polling. C'est à dire l'intervalle où le manager envoie une requête à chaque agent du système et reçoit toutes les réponses. Si le système est constitué de 100 agents, un cycle

contient 100 échanges requête/réponse.

Si l'on fixe la fréquence de polling à 5 secondes, pour gérer 10000 nœuds du système, le trafic de gestion supplémentaire pour le niveau $authNoPriv$ est : $2 \times 12 \times 10000/5 = 48ko/s$. Et pour le niveau $authPriv$ $80Ko/s$.

Pendant l'exécution, Une application de gestion doit obéir à un certain nombre de recommandations exigée par l'opérateur. Par exemple, le trafic de gestion ne doit pas dépasser 5% de la bande passante de bout en bout. Le gestionnaire de l'application de gestion peut jouer sur Le niveau de sécurité pour adapter l'exécution de l'application aux recommandations.

3.3 L'impact de la sécurité sur le temps de traitement de la requête

3.3.1 Les caractéristiques des tests

Nous avons installé Net-SNMPv5.1 sur deux machines de vitesses différentes ayant les caractéristiques matérielles et logicielles présentées dans le tableau de la figure 3.5.

	CPU	Mémoire	Système d'expolitation
Machine 1	Intel Xeon, 1700 Mhz	512 Mo	Linux Mandrake 9.2
Machine 2	Intel Pentuim III, 804 Mhz	256 Mo	Linux Mandrake 9.2

FIG. 3.5 – Les caractéristiques des machines de test

L'outil Net-SNMPv5.1 utilise la bibliothèque OpenSSL pour effectuer l'authentification et le cryptage des messages SNMP. Pour l'authentification, il propose les algorithmes MD5 et SHA-1. Pour le cryptage, il propose DES et AES. Ces algorithmes ont des temps d'exécution différents. La figure 3.6 montre la vitesse de ces algorithmes sur les deux machines de test.

		16 Octets	64 Octets	256 Octets	1024 Octets
Machine1	MD5	9948.72 Ko/s	33596.19 Ko/s	91091.61 Ko/s	155215.99 Ko/s
	SHA−1	6510.11 Ko/s	19955.88 Ko/s	50355.11 Ko/s	71529.17 Ko/s
Machine2	MD5	3216.73 Ko/s	11853.69 Ko/s	37187.62 Ko/s	122651.52 Ko/s
	SHA−1	3007.26 Ko/s	10136.10 Ko/s	27346.87 Ko/s	60523.61 Ko/s

FIG. 3.6 – La vitesse des algorithmes d'authentification

La figure 3.6 montre la vitesse d'exécution des algorithmes d'authentification sur des blocs de données

dont la taille varie entre 16 octets et 1024 octets. Plus la taille des données augmente, plus le nombre d'octets authentifiés par seconde augmente. On constate aussi que l'algorithme MD5 est plus rapide que l'algorithme SHA-1 sur les deux machines de test.

		16 Octets	64 Octets	256 Octets	1024 Octets
Machine1	DES	26204.60 Ko/s	27278.19 Ko/s	27155.73 Ko/s	26377.53 Ko/s
	AES	32384.62 Ko/s	33503.84 Ko/s	33467.83 Ko/s	33451.61 Ko/s
Machine2	DES	16272.46 Ko/s	16996.74 Ko/s	17155.36 Ko/s	17357.90 Ko/s
	AES	14835.88 Ko/s	15348.88 Ko/s	15495.01 Ko/s	15541.78 Ko/s

FIG. 3.7 – La vitesse des algorithmes de cryptage

La spécification du modèle de sécurité USM *RFC*3414 ne propose que l'algorithme de cryptage DES-CBC. La version 5.1 de l'implémentation Net-SNMP offre aussi l'algorithme AES. La figure 3.7 illustre la vitesse d'exécution des algorithmes AES et DES sur des blocs de données variant entre 16 et 1024 octets. On constate que la taille de données cryptées en une unité de temps ne dépend pas de la taille du bloc de données. On constate aussi que l'algorithme AES est plus rapide que DES dans la machine 1 tandis que dans la machine 2 on arrive à crypter plus de données par seconde avec DES qu'avec AES. Ceci est du à la nature du processeur des deux machines. Certes, AES est plus performant et plus robuste que DES.

L'objectif des tests

L'objectif des tests est d'estimer le temps de traitement d'une requête à l'intérieur de l'agent. Le temps calculé est la durée écoulée entre la réception de la requête de la couche transport et la sortie de la réponse du module SNMP vers le transport. Nous avons utilisé les algorithmes MD5 et DES pour les fonctions de la sécurité. Pour chaque valeur, nous avons effectué une série de *cent* mesures. Les valeurs présentées sont les moyennes de valeurs obtenues. Les tests que nous avons effectué sont :

1. La requête *get* sur la variable *ipInReceives* en variant le niveau de sécurité.
2. La requête *getwalk* sur le vecteur *ipNetToMediaIfIndex*. On varie le nombre des entrées *ARP* entre 5 et 80 et le niveau de sécurité.
3. La requête *getBulk* sur la table *ipNetToMediaTable*. On varie le nombre des entrées *ARP* entre 5 et 80 et le niveau de sécurité.

Nous avons compilé et exécuté Net-SNMPv5.1 avec le compilateur *gcc version* 3.3.1. Pour effectuer les mesures, nous avons introduit la fonction $gettimeofday()$ dans le programme de l'agent, dans les fonctions dont on désire estimer le temps de traitement. Nous avons utilisé l'outil *ethereal* pour visualiser le trafic et les messages échangés entre l'agent et le manager.

3.3.2 L'influence du niveau de sécurité

Le premier scénario de gestion

La vitesse de la machine influence le temps de traitement de la requête. La figure 3.8 montre aussi que ce temps varie selon le niveau de sécurité utilisé. L'agent SNMP utilise la bibliothèque OpenSSl pour assurer les fonctions d'authentification et de cryptage.

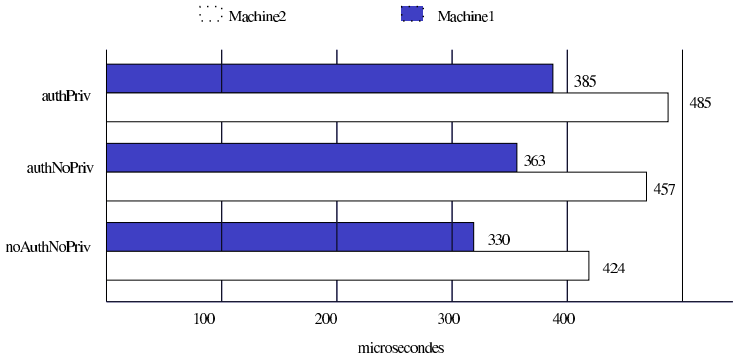


FIG. 3.8 – Le temps de traitement de la requête *get* selon les niveaux de sécurité

En prenant le niveau de sécurité *noAuthNoPriv* comme niveau de base, le temps de traitement d'une requête augmente de 10%, et pour le niveau de sécurité *authPriv*, il augmente de 16% par rapport au niveau de base.

Le deuxième scénario de gestion

a- **La requête *getwalk***

La requête *getwalk* consiste en un échange de requêtes *getnext* et de réponses pour rechercher une table ou un vecteur. La figure 3.9 représente le temps de traitement dans l'agent du premier *getnext* de la requête *getwalk* pour avoir le vecteur $ipNetToMediaIfIndex$ en variant le nombre des entrées dans la table *ARP* entre 3 et 80, et par conséquent le nombre d'instances d'objets à retourner.

On remarque que le temps de traitement de la première requête *getnext* augmente de manière quasi linéaire dans les deux machines. L'augmentation du temps de traitement de la requête est du à la procédure d'ordonnancement dans l'ordre lexicographique des instances. Plus le nombre d'instances augmente, plus cette procédure exige plus de temps CPU.

Les requêtes suivantes de la série des requêtes/réponses de *getwalk* ont un temps de traitement moyen qui varie selon le niveau de sécurité. Ces valeurs sont similaires au temps obtenu pour rechercher le scalaire *ipInReceives*.

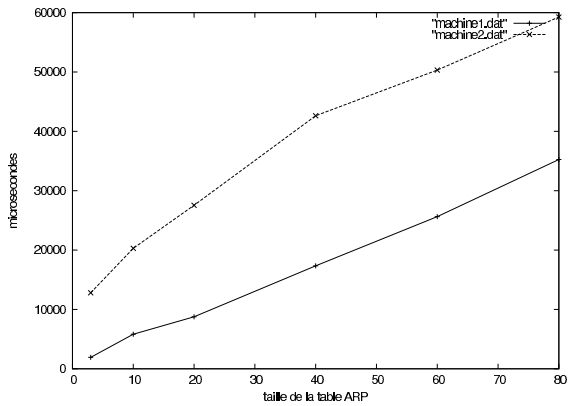


FIG. 3.9 – Le temps de traitement de la première requête *getnext* de *getwalk*

b- **La requête *getbulk***

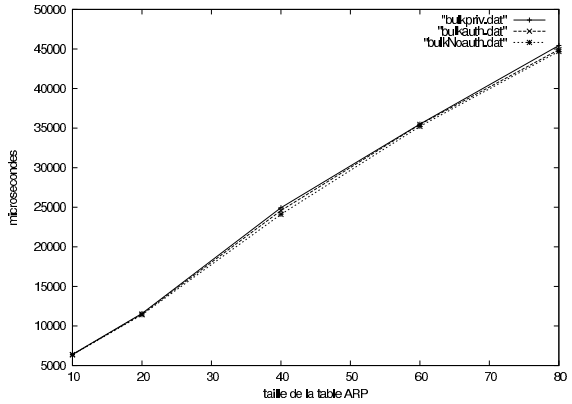


FIG. 3.10 – Le temps de traitement de la requête *getbulk*

La figure 3.10 illustre le temps de traitement de la requête *getbulk* pour rechercher la table *ARP* entière, en variant le nombre d'entrées entre 10 et 80, dans la machine 1.Nous avons réglé les paramètres de la requête de telle manière à avoir tous les variables dans une seule réponse. Le temps de traitement de la requête dans l'agent augmente de manière quasi linéaire. L'impact du niveau de sécurité est négligeable devant le temps CPU consommé par les autres étapes de traitement de la requête à savoir l'accès à la MIB et l'ordonnancement des instances dans l'ordre lexicographique.

Dans un agent, pour retourner la table *ARP* à 80 entrées dans la machine 1 avec le niveau de sécurité $authNoPriv$, la requête *getbulk* consomme $42ms$ alors que la requête *getwalk* atteint $158ms$ pour retourner le même nombre de variables avec le même niveau de sécurité. Pour *getnext* le temps estimé est la somme des temps de traitement de chaque requête pour retourner toute la table.

3.3.3 L'influence de la procédure de contrôle d'accès

La procédure de contrôle d'accès dans SNMPv3 nécessite l'accès à quatre tables desquelles on récupère des informations sur l'utilisateur, les vues qu'il a de la MIB et ses droits d'accès au variables. Ces tables sont stockées dans la MIB SNMP-VIEW-BASED-ACM-MIB dite vacmMIBObjects.

Au cours du traitement de la requête, le module VACM récupère le nom de l'utilisateur et le OID pour décider du contrôle d'accès. Si une requête contient plusieurs variables, alors la procédure de contrôle d'accès est répétée pour chaque OID avant de décider des droits d'accès pour l'utilisateur. Il suffit que l'utilisateur n'ait pas l'accès à un OID pour refuser le traitement de toute la requête.

	Requête get	Requête getnext	Requête getbulk
Machine 1	2µs	5µs	6µs
Machine 2	2µs	5µs	6µs

FIG. 3.11 – Le temps de contrôle d'accès

Nous avons testé le module de contrôle d'accès. Dans ce test, la tables des groupes de sécurité comporte 4 entrées et celle des vues comporte 3 entrées dans les deux machines. On constate que le temps de traitement dans le module VACM ne change pas avec les caractéristiques matérielles de la machine.

Nombre de groupes de sécurité	Nombre de vues	Get	Getnext	Getbulk
4	4	2µs	6µs	6µs
13	6	2µs	10µs	6µs
23	6	2µs	12µs	6µs

FIG. 3.12 – Le temps de contrôle d'accès en fonction de la taille des tables VACM

Nous avons changé la taille des tables des groupes de sécurité et de vues. La figure 3.12 montre que le temps de traitement de la requête *getnext* change avec la taille de la table. L'augmentation du temps de traitement reste acceptable par rapport au temps global de traitement de la requête.
La proportion des ressources consommées pour effectuer les fonctions de sécurité reste acceptable. Certes, le gestionnaire peut toujours jouer sur le niveau de sécurité pour changer le volume du trafic de gestion et la charge CPU dans ses agents.

Chapitre 4

Simulation et résultats

Ns-2 (Network Simulator Version 2) est un outil logiciel de simulation de réseaux. C'est un contrôleur d'événements orienté objet qui permet de simuler une variété de réseaux de type LAN et WAN. Ns-2 est un logiciel du domaine public disponible sur le web[7].

4.1 Le but de la simulation

L'objectif de la simulation est la détermination de l'influence de la sécurité sur le temps de traitement des scénarios de gestion dans les deux topologies LAN et Internet avec le passage à l'échelle du nombre des agents dans le système de gestion.

Dans un premier lieu, nous allons simuler le premier scénario de gestion dans les deux topologies. Ensuite, nous allons simuler le deuxième scénario de gestion.

4.2 Les paramètres de simulation

4.2.1 Les topologies de la simulation

La topologie LAN

La topologie LAN est constituée 1000 nœuds, une bande passante de 20Mbps et des délais de liens inférieurs à 1ms.

On considère la composition suivante du réseau LAN :
- 2% de serveurs,
- 13% de commutateurs,
- 5% de d'imprimantes,
- 80% de machines utilisateurs,
- un seul routeur.

[7]http ://www.isi.edu/nsnam/ns

on considère les tailles de la table *ARP* dans les différents équipements. Les pourcentages sont donnés par rapport au nombre total de nœuds dans le réseau :
- Dans les serveurs : 60%,
- dans les commutateurs : 40%,
- dans les imprimantes : 10%,
- dans les machines utilisateurs : 4 entrées,
- dans le routeur : 60%.

La topologie Internet

La topologie Internet de 1000 nœuds et 3000 liens une bande passante moyenne de 200Mbps et des délais entre 5ms et 40ms. Cette topologie a été généré avec la méthode *Map sampling* [26].

Dans une topologie Internet, l'activité de gestion de l'opérateur considère les ressources de communication du cœur du réseau. Il y a des routuers d'accès qui sont en communication directe avec l'utilisateur, et les routeurs de cœur qui constituent le cœur du réseau. Ces derniers n'ont pas une grande connectivité et n'hébergent pas des applications pouvant être adressées par des utilisateurs. Alors la table *ARP* dans ce type de routeurs n'est pas grande. En revanche, la table *ARP* dans un routeur d'accès dépend da la taille du réseau d'accès qui est connecté au routeur et au type d'activité dans ce réseau.

Nous allons considéré la répartition suivante pour les routeurs dans une topologie Internet. Les pourcentages sont données par rapport au nombre total des nœuds :
- 30% de routeurs de cœur.
- 70% de routeurs d'accès.

dans le cadre des simulations effectuées dans cette partie, on considère que la taille moyenne de la table *ARP* dans un réseau de cœur est de 5 entrées et dans un routeur d'accès, elle est de 80 entrées.

4.2.2 La taille des messages SNMP et les temps de traitement

Dans la section 2.1.3, nous avons présenté le codage ASN.1 du message SNMPv3. Nous allons nous servir de ces résultats pour établir la taille des messages SNMP pour les fonctions de gestion que nous allons simuler.

Fonctions de gestion	Type du Pdu	Taille du Pdu en octets		
		NoAuth	Auth	Priv
Recherche de ipInReceives	Get	125	137	145
	GetResponse	129	141	149
Recherche de la table ARP n le nombre d'entrées	Getbulk	124	136	144
	GetResponse	115 + 110*n	127 + 110*n	135 + 100*n

FIG. 4.1 – La taille des messages SNMP

On assume les données suivantes :

- Le champ $EngineID$ est sur 13 Octets. C'est la taille utilisée par l'implémentation Net-SNMP.
- Le champ $userName$ tient sur 8 Octets.
- Le champ $ContextName$ tient sur 8 octets.
- Les quatre entiers de l'adresse IP sont tous supérieurs à 127. C'est le cas échéant.

Le simulateur ns2 ne prend pas en considération le temps de traitement dans les nœuds. Des paramètres relatifs au temps de traitement dans l'agent SNMP seront ajoutés dans le modèle des événements de ns2 afin d'avoir des résultats proche de l'implémentation réelle. Nous allons considérer les temps de traitement dans la machine 1 résultant du benchmarking de SNMP sur l'implémentation Net-SNMPv5.1. La figure 4.2 résume les temps de traitement que nous allons utiliser dans la simulation.

En se basant sur les tests de performances élaborées dans le chapitre 3, on estime que le temps de traitement de la requête *getbulk* est linéairement proportionnel au nombre des entrées de la table ARP.

Les temps de traitement utilisés dans la simulation sont illustrés dans le figure 4.2. Ces temps de traitement sont issues des tests de performance effectués dans le chapitre précédent. Pour les temps de traitement de la requête *getbulk*. nous avons estimé que le temps de traitement est linéairement proportionnel au nombre des entrées dans la table ARP. Nous avons estimé que le temps de traitement dans le niveau $authNoPriv$ est augmenté de 100 microsecondes par rapport au niveau $noAuthNoPriv$ et il est augmenté de 200 microsecondes pour le niveau $authPriv$.

Le scénario de gestion	Temps [µs]		
	noAuth	Auth	Priv
Scalaire ipInReceives	330	363	385
Table ARP (getbulk)	158*n	158*n+100	158*n+200

(n nombre d'entrées ARP)

FIG. 4.2 – Le temps de traitement à l'intérieur de l'agent

Dans la simulation, on assume qu'il n y a pas de charge dans les liens et le taux de pertes de paquets est nul. On procède par un polling parallèle avec un intervalle de un milliseconde. L'intervalle de polling dépend de la capacité du manager à construire les requêtes. Par conséquent, il dépend des caractéristiques matérielles et logicielles de la machine du manager et du niveau de sécurité choisi pour cette requête.

4.3 Les résultats de la simulation

4.3.1 Le premier scénario de gestion

La figure 4.3 présente la durée du cycle dans la topologie Internet. On constate que le temps de traitement augmente avec le niveau de sécurité. Cette augmentation est induite par le temps de traitement de la requête à l'intérieur de l'agent. Elle reste raisonnable et dépend du nombre d'agents dans le système

de gestion. Plus le nombre d'agents augmente, plus la différence entre les temps de traitement entre les niveaux de sécurité augmente aussi.

Dans un système à 50 nœuds, la différence entre les temps de traitement du niveau *authNoPriv* et *noAuthNoPriv* est de 96 microsecondes soit 0.2% de plus si l'on considère le niveau $noAuthNoPriv$ comme niveau de sécurité de base. Dans un système à 500 agents, cette différence est de 2.289 millisecondes soit 1.03%. Entre les niveaux *authPriv* et *noAuthNoPriv*, dans un système de 50 agents, l'augmentation est de 0.33%, et dans un système de 500 agents, elle est de 4.87%.

L'allure des courbes est linéaire de la figure 4.3 est linéaire. Ceci est du au fait qu'il n'y a de trafic fonctionnel qui gène le trafic de la gestion. Il n'y a pas de gigue du aux files d'attentes. En plus les paquets de gestion sont petits. Alors ils ne sont pas fragmentés au cours de la transmission. Ces résultats rejoignent les résultats du papier [24].

Nombre d'agents	noauth nopriv[ms]	authnopriv[ms]	authpriv[ms]
50	110.755	110.972	111.117
100	143.338	143.484	143.581
150	189.648	189.824	189.942
200	261.054	261.229	261.346
250	300.600	302.030	302.983
300	417.784	423.828	427.856
350	460.649	467.798	472.564
400	527.713	528.930	535.959
450	602.519	606.320	615.072
500	630.073	652.362	673.402
600	811.151	831.102	850.621

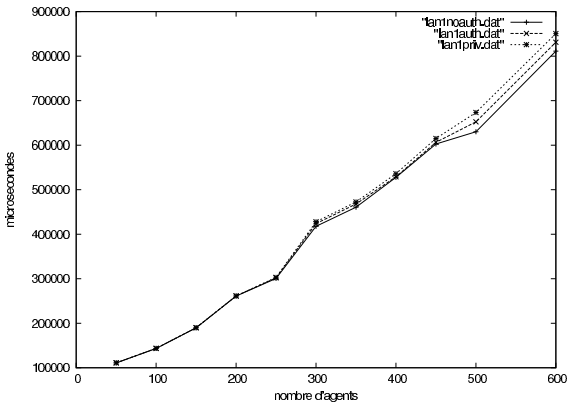


FIG. 4.3 – Le temps de traitement dans une topologie LAN pour le premier scénario de gestion

La figure 4.4 présente la durée de cycle en fonction de la taille du système de gestion et du niveau de sécurité de la requête. On constate que les temps de traitement sont plus importants que dans la topologie LAN. Les latences du réseau sont plus importantes dans la topologie Internet. L'augmentation du temps de traitement en fonction du niveau de sécurité est d'autant plus grande que la taille du système de gestion est grande.
On constate aussi qu'en changeant le niveau de sécurité, l'augmentation du temps de traitement représente un pourcentage négligeable par rapport au temps de traitement d'un cycle.

On conclut qu'en changeant le niveau de sécurité de $noAuthNoPriv$ à $authPriv$ la durée du cycle devient plus grande. L'augmentation est d'autant plus grand que la taille du système de gestion augmente.
On remarque aussi que le niveau de sécurité influence plus visiblement la durée de cycle dans une topologie LAN que dans une topologie Interent.

Nombre d'agents	noauth nopriv[ms]	authnopriv[ms]	authpriv[ms]
50	238.019	238.115	238.180
100	349.841	349.920	349.974
150	443.829	444.096	444.275
280	547.681	547.840	547.946
400	764.933	765.116	765.238
500	860.207	860.510	860.713
600	1049.401	1049.722	1049.937
700	1066.677	1066.887	1066.941
800	1191.024	1191.404	1191.500
900	1271.861	1272.095	1272.154

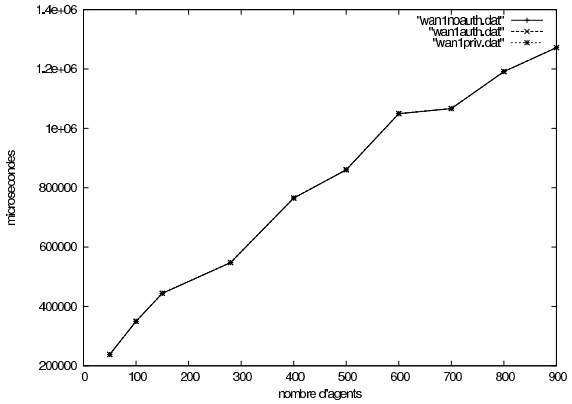


FIG. 4.4 – Le temps de traitement dans une topologie Internet pour le premier scénario de gestion

On constate que le temps de traitement est linéaire dans l'intervalle d'agents [50,250] dans la topologie LAN. Ceci rejoint les résultats retrouvés dans le papier [24]. En revanche, ce temps est beaucoup moins important dans nos résulats. Dans [24], l'implémenation AdventNet v2.2 a été utilisée pour estimer les temps d'accès au variables de la MIB.
La pente dans l'intervalle [50,250] est moins raide que dans l'intervalle [250,600]. Ceci nous amène à dire que l'étude du passage à l'échelle devient plus sérieuse quand on considère...

4.3.2 Le deuxième scénario de gestion

A partir du tableau de la figure 4.5, on constate que la durée du cycle augmente avec le niveau de sécurité. En terme d'unités de temps, le temps induit par l'augmentation du niveau de sécurité est plus important dans un système à 600 agents que dans un système à 100 agents.
En terme pourcentage de la durée de cycle, dans un système à 100 agents l'augmentation entre les niveaux $noAuthNoPriv$ et $authNoPriv$ représente 0.7% et entre les niveaux $noAuthNoPriv$ et $authPriv$, elle est de 2.86%. Dans un système à 600 agents, l'augmentation entre les niveaux $noAuthNoPriv$ et $authNoPriv$ représente 0.16% et entre les niveaux $noAuthNoPriv$ et $authPriv$, elle est de 0.25%.

Nombre d'agents	noauth nopriv[ms]	authnopriv[ms]	authpriv[ms]
50	115.246	115.560	115.803
100	161.066	163.807	165.669
150	265.565	268.540	268.832
200	360.047	363.137	365.230
250	740.500	743.448	745.446
300	1230.416	1235.652	1237.847
400	2208.408	2211.553	2213.682
500	3012.646	3019.273	3022.397
600	4021.698	4028.242	4032.061

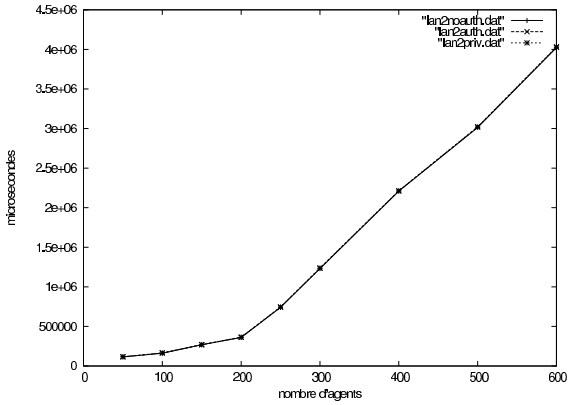


FIG. 4.5 – Le temps de traitement dans une topologie LAN pour le deuxième scénario de gestion

la courbe de la figure 4.5 peut être divisée en deux parties linéaires. L'intervalle du nombre d'agents [50,200] et l'intervalle [200,600]. Ce phénomène est du à la fragmentation des paquets. En effet, quand la taille du système de gestion augmente, la taille des tables *ARP* dans les différents équipements augmentent. Par conséquent, les réponses à la requête *getbulk* deviennent grandes et elles sont fragmentées. Ceci augmente le nombre des paquets envoyés dans le réseau ce qui induit l'augmentation du temps de traitement.

La pente du deuxième intervalle est trois fois plus grande que celle du premier intervalle. Ceci nous amène à prévoir un comportement critique de ce scénario de gestion quant au passage à l'échelle de la taille du système géré dans une topologie LAN.

Nombre d'agents	noauth nopriv[ms]	authnopriv[ms]	authpriv[ms]
100	609.882	610.923	611.950
200	1678.185	1679.225	1680.252
300	2264.386	2265.430	2266.458
400	2437.947	2439.227	2440.255
500	2633.618	2635.127	2636.184
600	2791.534	2793.047	2794.078
700	2840.352	2841.715	2842.750
800	3056.897	3058.901	3059.936
900	3217.190	3219.012	3220.060
1000	3374.627	3376.894	3377.922

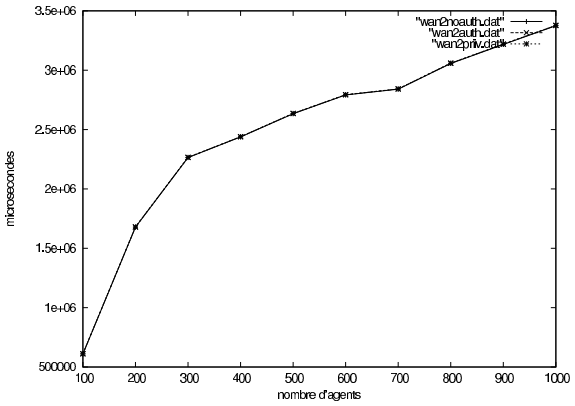


FIG. 4.6 – Le temps de traitement dans une topologie Internet pour le deuxième scénario de gestion

Dans une topologie Internet, l'influence de l'augmentation du niveau de sécurité sur la durée d'un cycle n'est pas considérable. En terme de pourcentage, dans un système à 100 agents, l'augmentation entre les niveaux *noAuthNoPriv* et *authNoPriv* représente 0.17% et entre les niveaux *noAuthNoPriv* et *authPriv*, elle est de 0.33%. Dans un système à 1000 agents, l'augmentation entre les niveaux *noAuthNoPriv* et *authNoPriv* représente 0.07% et entre les niveaux *noAuthNoPriv* et *authPriv*, elle est de 0.1%.

La courbe de la figure 4.6 a une allure logarithmique. Si on augmente la taille du système géré, la durée du cycle augmente lentement. Dans ce scénario la taille des tables *ARP* dans les équipements n'est pas proportionnelle à la taille du réseau.

Conclusion

Bilan

Nous avons effectué le benchmarking de SNMPv3 sur l'implémentation Net-SNMP v5.1. Selon les niveaux de sécurité, le temps de traitement dans l'agent augmente. Cette augmentation est significative, entre le niveau *noauthNoPriv* et le niveau *authPriv* le temps de traitement augmente de 16%.

Nous avons élaborés des scénarios de gestion que nous avons testé sur des topologies LAN et Internet avec les différents niveaux de sécurité offerts par le protocole SNMPv3.
L'influence de la sécurité sur le temps de traitement des fonctions de gestion reste raisonable. Quant au volume du trafic, il peut augmenter de manière considérable si la taille du système géré est grande ou bien si il y a des pertes de paquets de gestion.

Perspectives

Pour contrôler le trafic de gestion dans le réseau en tenant compte du niveau de la sécurité et de la capacité de traitement des agents et du manager, on propose, dans un travail futur, l'élaboration d'un pattern qui nous permet de définir le niveau de sécurité et la fréquence de polling à utiliser pour une fonction de gestion donnée sous un état du réseau bien déterminé, et des caractéristiques et des activités des équipements gérés.

Le modèle de trafic de gestion que nous avons utilisé dans le deuxième scénario de gestion, bien qu'il se base sur une répartition raisonnable des agents et leur caractéristiques, il ne considère pas la dynamicité des tables *ARP* dans le système géré. La simulation est effectuée dans certaines conditions de réseau qui ne sont pas génériques et les résultats obtenus, bien qu'ils sont intéressants pour l'évaluation du modèle Gestionnaire/Agent et l'impact de la sécurité, et ils nous ont permis de valider certains scénarios des travaux précédents, ils sont obtenus dans un cadre spécifique de paramètres et de facteurs.

Ceci nous amène à réflechir à la conception de patterns de performance pour l'ensemble des fonctions de gestion. Pour chaque système de gestion, selon les conditions du réseau et la nature de la fonction de gestion, on proposerait un pattern de gestion qui supporte l'exécution de cette fonction de manière efficace.

Annexe A : Le comportement de *getwalk* et *getbulk*

La requête *getnext* consiste à retourner la valeur du OID prochain. Pour retourner une table, le manager envoie une requête *getnext* avec l'identifiant de la table. L'agent recevant cette requête, recherche la table de l'instrumentation et ordonne les instances dans un ordre lexicographique avant de retourner la réponse qui contient la valeur du premier OID de la table. L'échange des requêtes/réponses entre l'agent et le manager continue jusqu'à ce que l'agent retourne une réponse avec $endOfMibView$ ou bien un OID dont le préfixe n'est pas l'OID de la table demandée initialment.

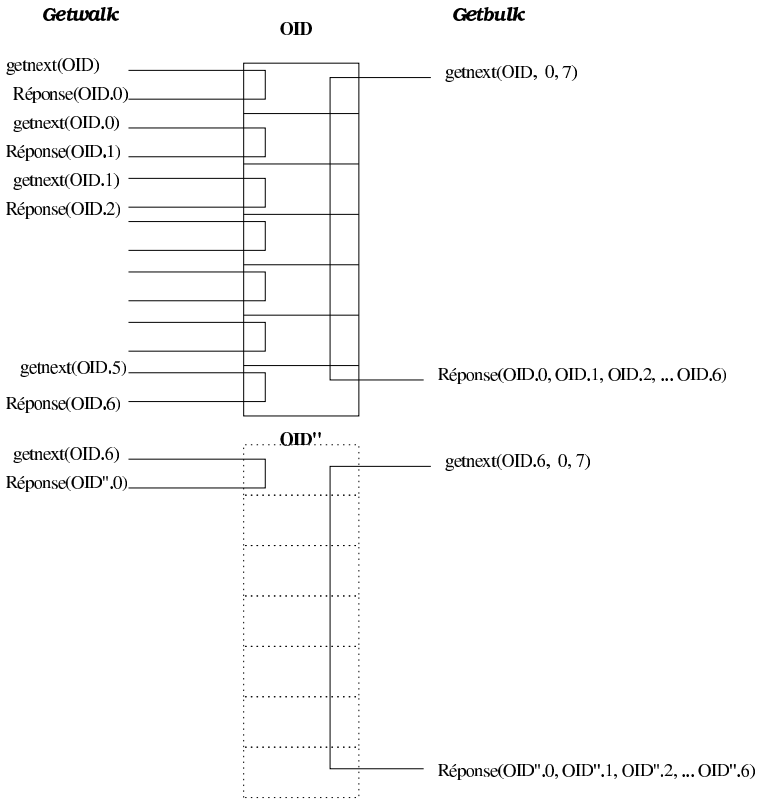


FIG. 1 – Les échanges des requêtes de *getnext* et *getbulk*

La requête *getbulk* est une optimisation de la requête *getnext* dans le sens où elle permet de retourner dans une même réponse plusieurs variables. dans la requête *getbulk*, il y a deux paramètres qui permettent de contrôler le nombre des variables dans une réponse :

- $non - repeaters$: indique le nombre des variables dans la liste des varBinds de la requête auquels il faut retourner une seule instance.
- $max - repetitions$: indique le nombre des répétions pour les variables qui restent dans la lisete des VarBinds.

Ces deux paramètres sont logés dans les champs : *type de l'erreur et position de l'erreur* de la PDU.

A chaque réponse reçue, le manager vérifie si le OID de l'objet retourné est préfixé par l'OID de la table demandée initialment. Si c'est le cas, il envoie une requête *getnext* à l'agent. Sinon il s'arrête estimant qu'il balayé toute la table. Quant à la requête *getbulk* la manager, la vérification se fait avec le dernier OID des objets reçus.

Le champ des varbinds dans les requêtes *getnext* et *getbulk*, on peut avoir plusieurs OID. Si une requête *getbulk* a les paramètres $n = 2$, $m = 3$ et une liste de 6 varbinds, la réponse d'une telle requête va porter $2 + 3 * 4$ soit 14 varbinds. Avec une requête *getnext*, nous avons 6 varbinds à la fois.

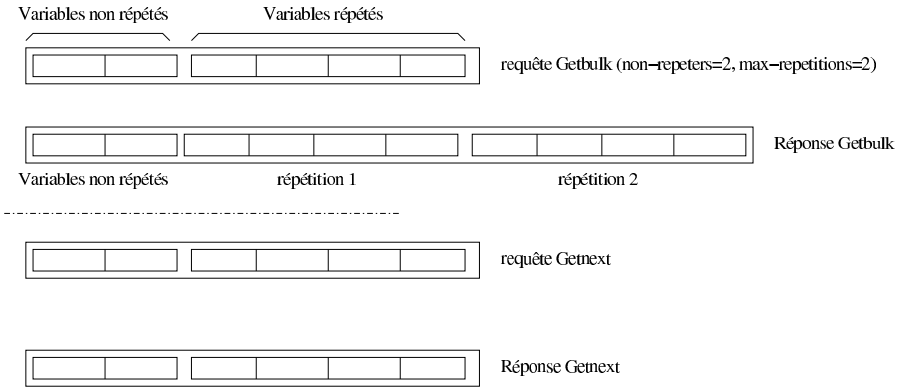


FIG. 2 – Le champ de varbinds requêtes/réponses de *getnext* et *getbulk*

La figure 2 illustre les requêtes et les réponses de *getnext* et *getbulk*...

Annexe B : La taille du message SNMPv3

La figure 1 montre les types et les containtes des champs du message SNMP tels qu'ils sont définis dans la spécification de SNMPv3. Chaque Pdu est un type structuré défini dans ASN.1 pour le contexte spécifique de SNMP.

Champ	Type	Contraintes	Taille en octets
Version	INTEGER	positif	4
Identifiant du message	INTEGER	0.. 2^31 – 1	4
Taille maximale	INTEGER	484.. 2^31 – 1	4
Drapeaux	OCTET STRING		1
Modèle de sécurité	INTEGER	0.. 2^31 – 1	4
Engine ID	OCTET STRING	5.. 32	5..32
EngineBoots	INTEGER	0.. 2^31 – 1	4
EngineTime	INTEGER	0.. 2^31 – 1	4
UserName	OCTET STRING		indéterminée
AuthenticationParameters	OCTET STRING		12
PrivacyParameters	OCTET STRING		8
Identifiant de contexte	OCTET STRING		5..32
Nom de contexte	OCTET STRING		indéterminée
Le Pdu	SEQUENCE (Specific Context)		
Identifiant de Pdu	INTEGER	0.. 2^31 – 1	4
Type d'erreur	INTEGER	positif	4
Position de l'erreur	INTEGER	positif	4
Variable	OBJECT IDENTIFIER		
Valeur			

FIG. 1 – Les champs du message SNMPv3

Le type OBJECT IDENTIFIER

Le type OBJECT IDENTIFIER de ASN.1 définit les objets dans la MIB avec une séquence de nombres entiers. La syntaxe de transfert BER prévoit le codage suivant pour ce type. Si l'objet est a.z.e.r.t, le codage sera $40 * a + z.e.r.t$. Chaque nombre est codé sur un octet. seuls les 7 bits du poids le plus faible de l'octet sont utilisés. Si un nombre est supérieur ou égal à 127, le huitième bit est mis à un indiquant qu'il y a un deuxième octet pour représenter le nombre.

Alors le nombre d'octets pour représenter un nombre i est :
- si $0 \leq i \leq 2^7 - 1$; Le nombre d'octets est 1.
- si $2^7 \leq i \leq 2^{14} - 1$; Le nombre d'octets est 2.
- si $2^{14} \leq i \leq 2^{21} - 1$; Le nombre d'octets est 3.
- si $2^{21} \leq i \leq 2^{28} - 1$; Le nombre d'octets est 4.

Le champ *Valeur*

La taille des champs $Valeur$ dans le codage BER du message SNMPv3 sont illustrés dans la figure 2 selon la partie du message. Pour chaque bloc on trouve une taille maximale et une taille minimale qui dépendent des containtes et du type des champs du message. Le bloc des valeurs dans une requête SNMP est nul et dans une réponse, il dépend du type de ces valeurs.

		Taille minimale	Taille maximale
Entête du message		17	17
Bloc de sécurité	NoAuthNoPriv	13 + UserNameSize	40 + userNameSize
	AuthNoPriv	25 + UserNameSize	52 + UserNameSize
	AuthPriv	33 + UserNameSize	60 + UserNameSize
Contexe		5 + ContexteName Size	32 + ContexteName Size
Entête du Pdu		12	12
Variables		obj de n nombres; soit i1,i2,...,im Les indices des nombres égaux à 127. Le nombre d'octets: $m+1+\sum_{\substack{i=3 \\ i\neq j \ j=i1,...,im}}^{i=n} [obj[i]/127]+1$	
Valeurs	Requête	Nul	
	Réponse	selon le type de la valeur	

FIG. 2 – La taille des valeurs des champs dans le message SNMPv3

Annexe C : L'implémentation Net-SNMP v5.1

Net-SNMP[8] est une implémentation du protocole SNMP qui offre un agent SNMP extensible, une librairie SNMP, des applications qui permettent de générer des requêtes et des alarmes SNMP, et de réccuperer des informations des agents et un explorer graphique de la MIB en Perl. Les fonctions de sécurité de Net-SNMP utilise la bibliothèque Open-SSL. Ce dernier doit être installé pour que l'agent puisse accomplir l'authentification et le cryptage.

L'agent SNMP supporte les MIBs suivantes :
- La MIB-II $RFC1213$,
- la MIB de l'agent SNMP,
- la MIB des ressources de la machine : système, mémoire, matériels et logiciels.

Cet agent accepte l'ajout d'autres MIBs arbitraires.

A la configuration d'un nouvel agent, un nom d'utilisateur, un mot de passe et un algorithme d'authentification, un mot de passe et un algorithme de cryptage sont requis. Ces données sont enregistrées dans la table *usmUserTable*.

Dans une entité SNMP, on peut créer plusieurs agents.On commence par créer un premier agent. les autres agents sont clonés sur le premier, mais on peut changer leurs configurations, leur droits d'accès et mots de passe après la création.

Le processus de découverte de l'agent par un manager est implementé en une seule phase. Le manager envoie une requête GET avec les champs *msgAuthoritativeEngineID, msgAutoritativeEngineBoots, msgAuthoritativeEngineTime* mis à zéro. L'agent répond avec un seul message contenant les trois valeurs dans le bloc de sécurité.

Les caches de Net-SNMP v5.1

Net-SNMP enregistre l'adresse IP du client et le statut qui désigne l'âge de l'adresse pour maintenir la

[8]http ://net-snmp.sourceforge.net

liste des managers avec qui il communique.

Il stocke aussi toutes les requêtes destinées à un agent : Le type de la requête, le OID de la requête, les droits d'accès à cet OID par le client et le statut d'exécution de la requête.

Les étapes de traitement des requêtes dans un agent

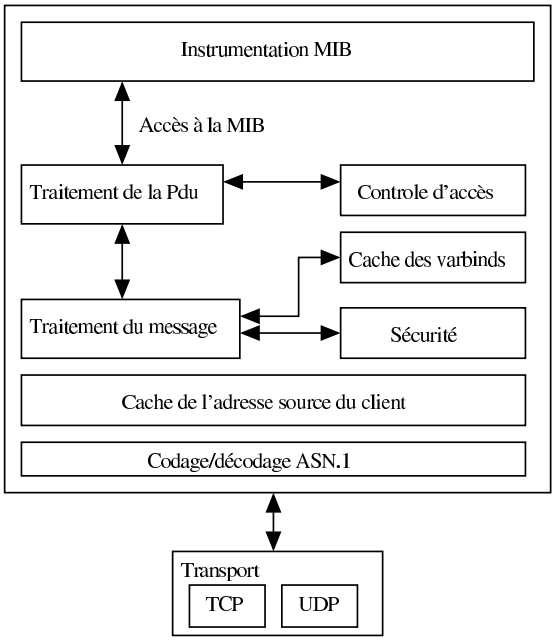


FIG. 1 – Les modules Net-SNMP

Les étapes de traitement de requêtes en réception sont :

1. La réception d'un paquet SNMP de la couche de transport et le décodage ASN.1 . Net-SNMP effectue le codage ASN.1 avec un analyseur qui lui est propre.
2. L'enregistrement de l'adresse du client dans un cache.
3. Le traitement du message SNMP : déterminer la version et la sécurité.
4. Contruire un sous arbre pour cacher les varbinds.
5. La vérification du contrôle d'accès à tous les variables dans le cache.
6. L'appel de la méthode qui permet de manipuler les ressourecs réelles.
7. La construction et l'envoi du message de réponse.

Bibliographie

[1] L. Tura A. Corente. Security performance analysis of snmpv3 with respect to snmpv2c. In *NOMS2004*, volume session 13, 2004.

[2] J. André. Mise en place d'une procédure de supervision des réseaux. In *,DESS, Université Paris 7*, September 2003.

[3] T. White B. Pagurek, Y Wang. Integration of mobile agents with snmp : why and how. In *Network Operations Manangement Systems(NOMS2000)*, pages 609–622, 2000.

[4] K. McCloghrie B. Wijnen, R. Prsuhn. View-based acces control model for simple network management protocol. In *RFC3415*, December 2002.

[5] R. Presuhn B. Wijnen, D. Harrington. Simple network management protocol framworks. In *RFC3411*, December 2002.

[6] U. Blumenthal B. Wijnen. User-based security model for simple network management protocol. In *RFC3414*, December 2002.

[7] H. Cruickshank C. Bohoris, G. Pavlou. Using mobile agents for network performance management. In *NOMS2000*, 2000.

[8] R. Barton C. Ozumutlu, N. Gautam. Optimizing mananging end-to-end network performance via optimized monitoring strategies. In *Journal of Network and Systems Manangement*, volume 10, March 2002.

[9] J. Case, M. Fedor, M. Schoffstall, and J. Davin. A simple network management protocol (snmp). In *RFC1157*, May 1990.

[10] Y. Shavitt D. Raz. Toward efficient distributed network management. In *Journal of Network and Systems Manangement*, volume 9, September 2001.

[11] B. Maglaris F. Stamatelopoulos, G. Koutepas. System security management via snmp. In *Swiss Federal Institute of Technology*, April 1997.

[12] O. Festor and L. Andrey. Chapitre 6 : Jmx : un standard pour la gestion java. In *Standards pour la gestion des réseaux et des services*, pages 213–251, 2003.

[13] O. Festor and N. Ben Youssef. Chapitre 5 : L'initiative wbem. In *Standards pour la gestion des réseaux et des services*, pages 159–211, 2003.

[14] Y. Yemini G. Goldszmidt. Distributed management by delegation. In *15th International conference on distributed computing systems*, June 1995.

[15] J. Galvin, K. McCloghrie, and J. Davin. Snmp security protocols. In *RFC135*, July 1992.

[16] E. Garcia. Syntaxe et systèmes d'information. In *IUT GTR, Montbéliard*, 2002.

[17] A. Bauer H. Abdu, H. Lutfiyya. Optimizing manangement functions in distributed systems. In *Journal of Network and Systems Manangement*, volume 10, december 2002.

[18] R. Stadler K.-S. Lim. A navigation pattern for scalable internet management. In *IM2001*, March 2001.

[19] R. Stadler K-S. Lim. Weaver : Realizing a scalable management paradigm on commodity routers. In *IM2003*, March 2003.

[20] D. Levi, P. Meyer, and B. Stewart. Snmpv3 applications. In *RFC2273*, Cisco ystems, January 1998.

[21] J. Labetoulle M. Cheikhrouhou. Efficient instrumentation of management information models with snmp. In *NOMS2000*, 2000.

[22] G. Samaras M. Dikaikos, M. Kyriakou. Benchmarking mobile-agents systems. In *technical report TR-01-2, University of Cyprus, Dept. of Computer Science*, November 2001.

[23] S. Ventura M. Jaton. chapitre 2 : Architecture de gestion de réseau. In *Gestion de réseaux IP*, pages 13–30, Juin 2002.

[24] G. Pujolle M. Rubinstein, O. Duarte. Scalability of a network management application based on mobile agents. In *Journal of Network and Systems Manangement*, volume 5, September 2003.

[25] K. Geihs M. Zapf, K. Herrmann. Decentralized snmp management with mobile agents. In *International Symposium on Integrated Network Manangement*, pages 623–635, May 1999.

[26] D. Magoni. Network manipulator version 0.9.6. In *Université Louis Pasteur*, 2002.

[27] K. McCloghrie and M. Rose. Management information base for network management of tcp/ip-based internets. In *RFC1156*, May 1990.

[28] R. Boutaba P. dini. Deriving variable polling frequency policies for pro-active management in networks and distributed systems. In *Integrated Management (IM1997)*, 1997.

[29] C. Pattinson. A study of the behavior of the simple network management protocol. In *DSOM2001*, October 2001.

[30] D. Perkins and E. McGinnis. Chapter 4 : Applying the snmp-based management model, chapter 6 : Snmp operations. In *Understanding SNMP MIBs*, November 1996.

[31] Y Saint-Hilare. Snmpv3-modulaire : une méthodologie de conception et de mise en oeuvre d'un protocole de gestion de réseau. In *Université du Québec*, pages 29–43, November 1998.

[32] W. Stallings. Snmpv3 : A security enhancement for snmp. In *IEEE communications surveys and tutorials*, November 2001.