Évaluation de performance des architectures de gestion de réseaux : état de l'art et perspectives

Abdelkader Lahmadi, Laurent Andrey, Olivier Festor

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Évaluation de performance des architectures de gestion de réseaux: état de l'art et perspectives

Abdelkader Lahmadi — Laurent Andrey — Olivier Festor

N° 5598

Juin 2005

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--5598--FR

Évaluation de performance des architectures de gestion de réseaux: état de l'art et perspectives

Abdelkader Lahmadi , Laurent Andrey , Olivier Festor

Thème COM — Systèmes communicants
Projet Madynes

Rapport de recherche n° 5598 — Juin 2005 — 50 pages

Résumé : Les architectures de gestion sont de plus en plus complexes et des nouvelles approches et standards ne cessent d'apparaître. Ces approches de gestion sont souvent intégrées dans le plan fonctionnel de services supervisés dont elles impactent potentiellement les performances. Afin de maîtriser cet impact, nous travaillons sur l'évaluation de performances des systèmes de supervision. Ce rapport comprend l'état de l'art des travaux sur ce domaine. Il nous permet d'identifier les techniques, métriques, paramètres et facteurs de performance communs dans le but de définir une méthodologie générique d'évaluation des approches de supervision du point de vue performances.

Mots-clés : état de l'art, performance, architectures de gestion, supervision

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,

Performance evaluation of the network management architectures: state of the art and perspectives

Abstract: The complexity of network management architectures growth and new approaches and standards appears. Current approaches are usually integrated in the functional plan of users services plan, that impacts their overall performance. In order to quantify this impact, we evaluate the performance of these management architectures. This report presents a survey of a variety of performance evaluation studies of network management architectures, to identify the set of a common basic choices for performance evaluation: techniques, metrics, parameters and factors. Moreover, we propose a methodology to evaluate management architectures from a performance perspective.

Key-words: state of the art, performances, network management architectures, monitoring

Table des matières

1 Introduction **5**
1.1 Objectifs 6

2 État de l'art de l'évaluation de performance des systèmes de supervision **7**
2.1 Approches candidates pour l'évaluation de performance 7
2.1.1 La technique de mesure 8
2.1.2 L'évaluation par simulation 10
2.1.3 La technique analytique 11
2.1.4 Comparaison des techniques 13
2.1.5 Approches utilisées en supervision 14
2.2 Les métriques de performance 15
2.2.1 Granularité des opérations de gestion 17
2.2.2 Métriques de performance utilisées en supervision 17
2.3 Les paramètres de performance 19
2.4 Les facteurs de performance 19
2.5 Le facteur d'homogénéité 20
2.6 La charge 21
2.7 Examen et analyse des études d'évaluation de performance 22
2.8 Conclusion 24

3 Méthodologie d'évaluation de performance des systèmes de supervision **29**
3.1 Choix de techniques d'évaluation 29
3.2 Choix de métriques d'évaluation 31
3.3 Paramètres et facteurs de performance 32
3.4 Charges d'évaluation de performance 33
3.4.1 Paramètres de sélection de la charge 33
3.4.2 Charge de la supervision 35
3.4.3 Modèles de trafics réseau de la supervision 36
3.5 Impact de la supervision sur un système supervisé 37
3.5.1 Modèles de déploiement des agents de supervision 39
3.5.2 Scénarios des activités de supervision 40

3.5.3 Exemple d'utilisation de la fonction Impact 41

4 Conclusion **45**

Chapitre 1

Introduction

La complexité des infrastructures de gestion de réseau et de service ne cesse de croître et devient une préoccupation dans la communauté de gestion. Durant les 20 dernières années, des nouvelles architectures de supervision ne cessent d'émerger, elles varient du modèle OSI au modèle SNMP (v1,v2,v3) à la gestion basée sur les services web ou le standard JMX pour les applications Java. Ces architectures sont de plus en plus intégré au plan fonctionnel des réseaux et des services supervisés. La cohabitation du plan fonctionnel et du plan de gestion a remis en cause la solution de ressources allouées pour les fonctions de gestion. En effet, les activités de gestion engendrent un trafic réseaux important qui pourrait provoquer des congestions et dégrader les performances des systèmes supervisés.

Un autre facteur de complexification des architectures de gestion est l'explosion du nombre de composants à superviser (les équipements et les services). Les réseaux de télécommunications ont considérablement évolué et leur progression se poursuit à travers le monde. Ils relient de plus en plus d'objets désormais connectés : aujourd'hui les ordinateurs personnels dans les foyers et les terminaux mobiles, demain les objets du quotidien, les téléviseurs, les capteurs, l'électroménager. Ils sont le support de services de plus en plus nombreux : aujourd'hui l'accès à l'internet, la téléphonie sur IP et la diffusion de vidéo numérique, demain des services personnalisés, de la visioconférence et de la vidéo à la demande de très haut qualité. Enfin, ils sont devenus dynamiques, c'est-à-dire qu'ils changent automatiquement de topologie ou de comportement pour s'adapter au mieux au contexte. Les environnements et les contextes de ces composants à superviser sont aussi de plus en plus contraignants où les critère clés d'une gestion efficace sont l'amélioration de performances des services et la bonne utilisation de ressources des équipements.

Cette évolution des systèmes à supervisés a mis l'accent sur le problème d'allocation des ressources de supervision dans les réseaux. Les solutions triviales qui se basent sur l'allocation des ressources surdimmensionnées allant de l'ajout d'une station de travail dédiée à l'hébergement de ces fonctions dans un équipement, à l'allocation d'un réseau complet parallèle pour les seules fonctions de supervision, sont devenues invalide dans de tels environnements. En revanche, une allocation adéquate de ressources pour les fonctions de la

gestion nécessite l'évaluation de la maîtrise de leurs activités et définir leurs coûts associés. Cette maîtrise implique la connaissance de l'impact des fonctions de gestion sur le plan fonctionnel des réseaux et services supervisés ; et de définir une manière efficace avec laquelle ils doivent opérer.

Pour atteindre cette maîtrise, il est indispensable d'analyser la performance de ces architectures de gestion, qui est l'un des paramètres cruciaux dans tout système informatique. Cette analyse de performance doit permettre de quantifier le coût de la gestion, de définir l'impact du plan de gestion sur le plan fonctionnel des services et va jouer un rôle important dans l'optimisation des architectures de gestion ainsi que dans l'allocation de ressources associés.

1.1 Objectifs

Dans ce rapport[1], nous présentons un état de l'art des études publiées qui comportent un volet lié à l'évaluation de performance des architectures de gestion. Nous avons analysé une grande variété d'architectures de gestion et les études qui ont été faites pour évaluer leurs performances. Ces architectures varient de celles basées sur des approches classiques comme SNMP à des approches comme la gestion de réseaux basée sur XML et les services Web. Nous avons examiné deux types d'études d'évaluation de performance menées sur ces approches : des évaluations isolées portant sur une architecture de gestion, ou des évaluations portant sur la comparaison de deux ou plusieurs architectures de gestion.

Tout au long des analyses présentées dans cette étude, nous allons spécifier de façon incremantal une méthodologie adéquate pour l'évaluation de performance des architectures de gestion basée sur les études antérieures examinées. Cette méthodologie d'évaluation de performance, nous servira de guide pour nos travaux futurs portant sur l'évaluation de performance des architectures de gestion.

La première partie de ce rapport présente l'état de l'art de l'évaluation de performance des architectures de gestion, où nous avons identifié les différents techniques, métriques, paramètres et facteurs de performances. Dans la deuxième partie nous proposons une méthodologie du choix d'approches pour l'évaluation de performances des systèmes de supervision. La dernière partie est consacrée aux conclusions et une présentation de nos travaux futurs.

[1]Ce travail a été réalisé dans le cadre de projet RNRT AMARILLO

Chapitre 2

État de l'art de l'évaluation de performance des systèmes de supervision

La première décision à prendre avant de modéliser les performances d'un système est de décider *Quoi* évaluer, c'est à dire de définir le système, sa configuration, et les algorithmes ; sujets de l'évaluation de performance. Ces décisions englobent la sélection des métriques de performance, des paramètres de performance, des facteurs de performance, de la charge (*workload*) et le choix des méthodes à utiliser. Cet ensemble de choix de base représente les entrées du processus d'évaluation de performance d'un système. Après cette étape cruciale, toute étude de performances est basée sur un ensemble d'assomptions homogènes, explicites ou implicites.

2.1 Approches candidates pour l'évaluation de performance

Dans la littérature, il existe trois approches d'évaluation de performance de systèmes : les modèles analytiques, les simulations et les mesures. Le choix d'une approche ou d'une autre pour évaluer les performances d'un système dépend de plusieurs critères [22]. Nous pouvons classifier ces techniques en deux grandes catégories (voir figure 2.1) : les techniques basées sur la mesure et les techniques basées sur la modélisation. La première catégorie permet de quantifier les critères de performance en les mesurant directement sur un système réel. Dans cette catégorie, nous distinguons deux variantes d'approches de mesure : l'instrumentation du code source de l'application en insérant de codes de mesure (*in-side*), ou l'utilisation de moniteurs externes à l'application pour quantifier les mesures au cours de cycle d'exécution de l'application (*out-side*). La deuxième catégorie d'approches basées sur la modélisation,

permet de spécifier le système à évaluer afin de prédire ses performances. Les modèles élaborés par ces approches seront utilisés pour effectuer des simulations ou une étude analytique du système. La différence entre ce deux spécifités est que les simulations mettent en oeuvre des modèles conceptuels du système sous test (représentation conceptuel du système) qui nécessite leurs développement dans un outil de simulation. En revanche, la deuxième méthode (l'étude analytique) met en oeuvre des modèles calculables (files d'attentes, réseaux de Petri, ...) sous la forme de formules mathématiques.

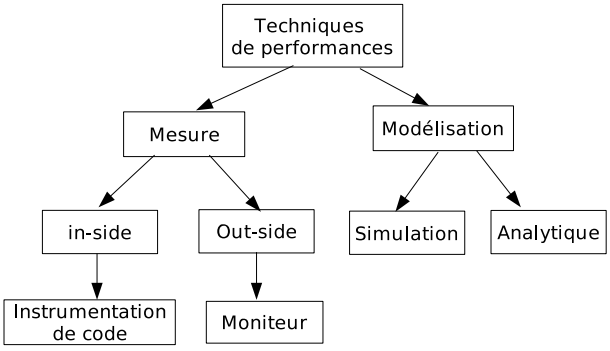

FIG. 2.1 – Classification des techniques d'évaluation de performance

2.1.1 La technique de mesure

La technique de mesure permet d'évaluer un système de gestion existant ou au moins un prototype de ce système, et elle se base sur des outils d'instrumentation. L'instrumentation d'un système est une méthode indispensable pour mesurer ses performances au cours de son exécution, en insérant des codes de mesure dans son code source, ou en utilisant des outils d'observations collectant des mesures. L'un de principaux avantages de cette technique est la réalité de ses résultats, puisque les données sont prises sur un système réel. Ce réalisme des mesures est l'un des arguments majeurs en faveur de cette technique pour évaluer les performances d'un système. Cependant, ces résultats, malgré leur réalisme, sont relatifs et variables. Ils dépendent de plusieurs paramètres, comme la configuration du système, la charge et le temps de mesure, qui sont généralement unique et relatifs à chaque test d'évaluation de performance. Ces résultats, ainsi, ne sont pas généraux mais spécifiques à un test particulier. Un autre inconvénient majeur de cette technique est le coût nécessaire pour sa réalisation. En effet, cette technique d'évaluation introduit un coût important dans un projet puisqu'elle nécessite des équipements réels, l'instrumentation des applications et un temps important de développement et de collecte de mesures. Un troisième inconvénient de cette technique est qu'elle n'est pas toujours réalisable. Dans certains environnements, il est impossible de procéder par cette technique pour évaluer les performances d'un système,

par exemple un système embarqué sur un satellite où la contrainte d'énergie rend impossible de procéder par cette technique pour évaluer les performances du système.

La technique de mesure se base sur deux méthodes : l'instrumentation de code du système sous test ou le monitorage du système au cours de son exécution. La première méthode consiste à placer des capteurs dans le code source du système pour mesurer ses performances sous une charge réelle. Cette facilité offerte par cette technique parvient à combler certaines lacunes d'autres techniques qui n'arrivent pas généralement à exécuter une telle charge réelle et complexe. Des bibliothèques d'instrumentation (par exemple ARM : Application Response Measurement [18]) existent et elles s'intègrent facilement dans le code source de l'application sous test en offrant un cadre efficace et transparent pour le représentation de données d'instrumentation. D'autres outils ne nécessitent pas une une modification ou une recompilation du code source de l'application mais instrumentent l'application à l'exécution. Parmi ces outils [1], nous citons le profiler fourni par Netbeans [38] ou Gprof (Gnu profiling Tool, [46]). La seconde méthode se base sur des moniteurs qui se positionnent comme des processus indépendants de l'application sous test. Ces moniteurs observent les activités de l'application au cours de son cycle d'exécution, collectent les mesures et les affichent ou les placent dans un fichier.

Il faut noter qu'il existe deux type de mesures : la mesure passive et la mesure active. La mesure passive ne perturbe pas le système sous test (idéalement). En revanche, la mesure active induit, généralement, des perturbations sur le système. Il existe deux approches de la mesure passive. Une approche de *vraie mesure passive*, où l'outil de mesure n'induit aucune perturbation sur le système sous test. À titre d'exemple un sniffer des paquets dans un réseau LAN sans fil (WLAN) ne perturbe pas le fonctionnement des autres équipements dans le réseau. Une approche de *faible mesure passive*, où le système sous test doit être modifié (par exemple instrumenté) mais sans changer son état de fonctionnement. À titre d'exemple, certains switchs Ethernet fournissent des ports spécifiques vers lesquels tous les paquets seront copiés sans perturber leurs transmissions. Un analyseur réseau peut s'attacher à ce type de port. L'utilisation de la mesure active ne nécessite pas seulement la modification du système sous test mais impacte sa charge et son état interne de fonctionnement. À titre d'exemple la technique packet-pair utilisée pour mesurer les bandes passantes dans l'Internet [30] perturbe la charge et l'état du réseau sous test. Dans des cas extrêmes, l'utilisation de cette approche de mesure active peut induire le phénomène de *Heisenbugs* [34] en faisant apparaître des bugs dans le système sous test.

Généralement, ces méthodes d'instrumentation et ces approches de mesure perturbent le système sous test et engendrent un coût (*overhead*) sur les performances réelles du système sous test. Ce coût est dû aux instructions supplémentaires à exécuter par l'application en utilisant la première méthode (instrumentation) ou aux ressources consommées par un moniteur externe à l'application dans le cas de la deuxième méthode.

[1] Ces outils sont généralement spécifique aux applications Java et utilisent les facilités offertes par l'instrumentation de la machine virtuelle Java.

2.1.2 L'évaluation par simulation

La simulation s'impose comme la technique la plus utilisée pour évaluer les performances de systèmes informatiques. Elle représente un moyen utile pour prédire les performances d'un système et les comparer sous plusieurs de ses configurations. Un atout majeur de cette technique est sa flexibilité puisque elle permet d'évaluer le système sous plusieurs conditions et configurations. Même, si le système est déjà implanté la technique de simulation reste favorable puisqu'elle offre une flexibilité, difficile à réaliser avec la technique de mesure. En revanche, la technique de simulation (aussi que l'approche par mesure) nécessite une excellente maîtrise des techniques statistiques pour une analyse pertinente des résultats, ainsi qu'une maîtrise de la programmation pour développer le modèle à simuler sous un langage approprié. La confiance ou une validation préalable des modèles utilisés par cette technique est indispensable (exemple, les limites du Random Way Point). L'inconvénient majeur de cette technique, est qu'elle nécessite, un temps potentiellement important et des ressources de calcul conséquentes. La simulation d'un système de supervision nécessite la

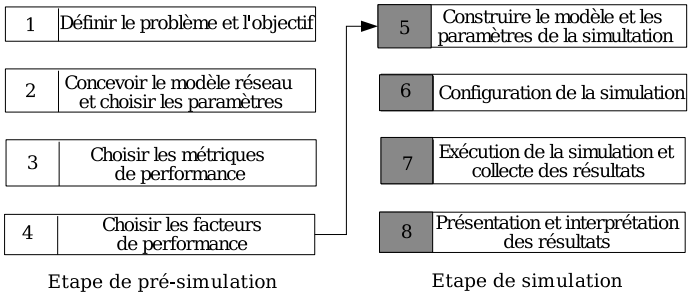

FIG. 2.2 – Les étapes de réalisation d'une simulation.

réalisation des étapes suivantes (voir figure 2.2) :

1. Définir le problème et l'objectif de la simulation : dans cette étape il faut essentiellement spécifier le système de supervision à évaluer et ses différents niveaux de détails. Dans notre contexte, l'objectif est d'évaluer les performances du système de supervision sous test.
2. Concevoir le modèle du système de supervision et définir l'ensemble de ses paramètres de performance. Notre objectif est d'évaluer les performances, ainsi, nous n'avons pas besoin de modéliser le fonctionnement exacte du système. Par exemple, la modélisation d'un agent de supervision ne nécessite pas la représentation de la récupération des informations de gestion du système d'exploitation, mais nous considérons juste les valeurs du temps nécessaire pour récupérer ces informations ou sa distribution mathématique (exponentielle, géométrique,...).
3. Choisir l'ensemble des métriques de performance du système de supervision.
4. Choisir l'ensemble des facteurs de performance du système de supervision.

5. Développer le modèle du système de supervision dans l'outil de simulation et fixer l'ensemble des paramètres de performance.
6. Configurer l'application développée dans l'outil de simulation afin de produire les résultats de performance.
7. Exécuter la simulation et collecter les données relatives aux performances du système de supervision.
8. Représenter les données collectées et procéder à leur interprétation.

Plusieurs modèles de systèmes existent dans la littérature [22] et sont utilisés pour la simulation. Essentiellement, les modèles sont temporels avec des valeurs des états du système continues ou discrètes en fonction du temps. D'autres modèles à base d'évènements existent qui spécifient les états du système qui peuvent être continus ou discrets. Ces derniers modèles peuvent utiliser les premiers pour spécifier la nature de valeurs des états (discrètes ou continues). Une variété d'approches de simulation existe, aussi, dans la littérature [22]. Ces approches sont essentiellement : l'émulation [2], la simulation dirigée par trace et la simulation discrète.

L'outil de simulation est un choix important lors de l'utilisation de cette technique. Deux principaux outils de simulation existent : Network Simulator (NS) [11] est libre source et OPNET [54] est un produit commercial [3]. Les différences majeurs entre ce deux outils consistent essentiellement en l'interface graphique conviviale de OPNET qui manque dans NS2 et l'étroite documentation de OPNET. Généralement, il est recommandé d'utiliser OPNET plutôt que NS2 pour effectuer des simulations, mais ces deux outils représentent les outils standards pour toute étude basée sur la simulation. Gilberto Flores et al [36] ont fait une étude comparative portant sur le degré de précision de ces deux outils dans le cadre de la simulation d'un réseau au niveau paquet. Un autre aspect à ne pas négliger est la crédibilité des résultats obtenus par la technique de simulation. L'étude de Pawlikowski et al [44] porte sur cet aspect.

2.1.3 La technique analytique

La technique analytique est l'un des moyen le plus rapide, comparé avec les deux autres, pour évaluer au moindre coût les performances d'un système. Elle se base sur la modélisation des systèmes sous forme de paramètres , de variables et un ensemble des formules mathématiques qui régissent leurs relations. Nous pouvons, ainsi, modéliser et évaluer les aspects variables d'un système. La technique analytique nécessite beaucoup de simplifications et d'hypothèses pour arriver à un modèle cohérent du système. Ces simplifications se présentent comme l'une de ses limites majeures et mettent en cause le degré de précision des résultats obtenus avec cette technique.

[2] L'émulation représente une des approches de la simulation. En revanche, elle se présente sous la forme d'un coeur de simulation avec des interfaces d'applications standard. Dans ce cas les applications ne sont pas des modèles.

[3] Une version gratuite de OPNET est disponible pour les chercheurs et les académiciens.

Cette technique se base sur plusieurs approches (Réseaux de files d'attentes, Réseaux de pétri, chaînes de Markov,...) pour modéliser un système et prédire ces performances. La théorie de réseaux de files d'attentes [26] est la plus utilisée dans l'évaluation de performances des systèmes informatiques. Elle représente et analyse, généralement, des systèmes à ressources partagées. Un modèle de réseau de files d'attente se présente sous la forme d'une collection de serveurs qui interagissent entre eux, représentant les ressources du système et d'un ensemble des clients qui représentent les utilisateurs partageant ces ressources. Ce modèle se représente formellement comme une graphe orienté direct avec des noeuds qui représentent ces serveurs et les liens entre eux représentent le comportement des sollicitations de clients à ces serveurs. Ces modèles sont analysés par plusieurs algorithmes de façon efficace pour obtenir des valeurs moyennes des indicateurs de performance du système modélisé. À titre d'exemple, nous citons l'algorithme de convolution ou l'algorithme d'analyse par valeurs moyennes (MVA : Mean Value Analysis) [5]. La construction de ces modèles nécessite la réalisation des étapes suivantes :

1. La définition des centres de service (serveurs). Cette définition inclue leur nombre, leur classe des clients et leur topologie.
2. La définition des paramètres du modèle : processus d'arrivée des clients, taux de service et le nombre des clients.
3. L'évaluation pour obtenir une description quantitative du système modélisé en calculant la valeur de ses indicateurs de performance (utilisation de ressource, débit de système et temps de réponse). Ces indicateurs peuvent être locales pour un serveur spécifique ou globales pour tout le système.

La figure 2.3 montre un réseau de files d'attentes simplifié d'une architecture de supervision. Nous identifions trois centres de service [4] : le superviseur, le réseau et l'agent. Ce réseau de files d'attentes représente les routes que les requêtes, provenant du superviseur, doivent suivre afin d'accomplir leurs tâches de supervision. Les paramètres du modèle sont les taux de services ($t_i, 1 \leq i \leq 3$) des différents centres de services.

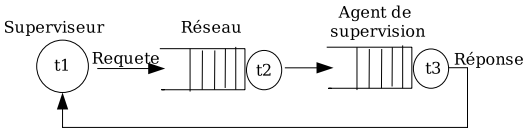

FIG. 2.3 – Un réseau de files d'attentes d'un système de supervision.

Des extensions des réseaux de files d'attentes sont apparues afin de représenter des nouveaux aspects des systèmes réels, comme par exemple la synchronisation, les contraintes liées à la concurrence, le multi-threading, et la possession simultanée des ressources. Le réseau de files d'attentes en couches (LQN : Layered Queueing Networks) [57] est l'une de ces extensions. Elle permet de modéliser des architectures client/serveur distribuées avec

[4] Il s'agit bien d'un modèle de supervision de type manager/agent.

des interactions concurrentes. Dans ce type de files d'attentes, un serveur devient un client pour d'autre serveurs tout en servant les requêtes de ses propres clients. Cela représente la différence essentielle entre les réseaux de files d'attentes classiques et les LQN. Il existe des outils permettant l'évaluation de performance de ces modèles LQN. Parmi eux, nous citons l'outil LQNS (Layered Queueing Networks Solver, [13]). La figure 2.4 montre un modèle LQN simplifié d'une architecture de gestion de type superviseur/agent. Un modèle LQN se

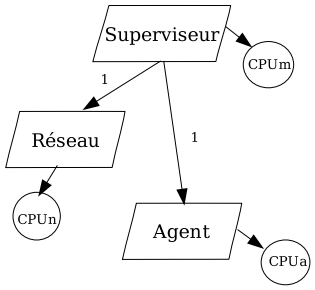

FIG. 2.4 – Un modèle LQN d'un système de supervision.

présente sous la forme d'un graphe acyclique dont les nœuds (les tâches) représentent les entités logicielles et matérielles du système et les arcs représentent les requêtes de service. Les tâches sont représentées par des parallélogrammes et les composants matériels (CPU) sous forme des cercles.

2.1.4 Comparaison des techniques

Critère	**Technique analytique**	**Technique de simulation**	**Technique de mesure**
Étape	N'importe	N'importe	Nécessite l'existence d'un système réel ou un prototype
Temps requis	Peu	Moyen	Variable
Outils	Des analystes	- Langages de programmation - Simulateurs	- Instrumentation - moniteurs
Précision	Faible	Moyen	Variable
Compromis de l'évaluation	Facile	Moyen	Difficile
Coût	Faible	Moyen	Haut

FIG. 2.5 – Comparaison des techniques d'évaluation de performance [22].

Le tableau 2.5 représente une comparaison qualitative de trois techniques d'évaluation de performance présentées précédemment.

2.1.5 Approches utilisées en supervision

Cette section vise à établir la liste des approches utilisées par les études publiées relatives à l'évaluation de performance des architectures de gestion. À première vue, ces études ont eu recours aux trois techniques (mesure, simulation et analytique) présentées précédemment.

Approches basées sur la mesure

Plusieurs études [58, 52, 37] ont utilisé cette technique pour évaluer des prototypes d'architectures de gestion de réseaux (agents mobiles, SNMP, services web). Bohoris et al [4] ont utilisé cette technique pour évaluer le prototype d'une plate-forme de gestion basée sur les agents mobiles. Ils ont instrumenté le code de l'agent pour mesurer la moyenne et l'écart-type de ses temps de réponse. Ils ont utilisé l'outil tcpdump pour mesurer le volume de trafic réseau au niveau TCP. Pavlou et al [43] ont appliqué cette technique sur un prototype d'une plate-forme de gestion basée sur les services web. Gavalas et al [16] ont appliqué cette technique sur une plate-forme de gestion basée sur des agents mobiles afin de valider les modèles analytiques qu'ils ont proposés dans leur étude.

Approches basées sur la simulation

Des études, comme [42, 48, 3], ont eu recours à la technique de simulation pour évaluer des infrastructures de gestion complexes sous différentes configurations en variant essentiellement le nombre d'agents et les topologies réseaux. Pattinson [42] a simulé une infrastructure de gestion basée sur SNMP avec un nombre d'agents allant jusqu'à 120 et avec des topologies réseaux de type LAN et WAN. Son modèle de simulation de SNMPv1 est discret, déterministe[5] avec des valeurs des états discrètes (nombre de requêtes, par exemple). Il a utilisé OPNET [54] comme outil de simulation. Rubinstein et al [48] ont appliqué la technique de simulation pour évaluer les performances d'une architecture de gestion basée sur les agents mobiles. Leur modèle de simulation est de type évènements discrets. Ils ont utilisé Network Simulator (NS) [11] comme outil de simulation.

Approches analytiques

Plusieurs études [2, 15, 35, 7, 16] proposent des modèles analytiques pour évaluer les performances des architectures de supervision des réseaux. Ces études mesurent d'une façon approximative (Un modèle analytique nécessite des simplifications du système sous test) les performances d'un système de gestion de réseaux suivant des modèles analytiques calculables. Par exemple, Chen et al [7] ont proposé des formules empiriques (intuitives) pour analyser

[5] Un modèle de simulation est dit déterministe, si la répétition de même entrées du modèle reproduisent les mêmes sorties.

les performances des différentes approches de gestion de réseaux (client/serveur centralisée, hiérarchique statique, mobilité faible et mobilité forte). Liotta et al [35] ont définis de façon analytique le délai de monitorage. Ce délai représente le temps nécessaire pour réaliser une opération de monitorage (une requête et une réponse). Vilà et Sheikh [55, 50] ont proposé des modèles analytique pour des applications de gestion de réseaux. La première [55], propose une étude du passage à l'échelle d'une application de gestion multi-agents dans des réseaux MPLS. Ce modèle repose sur les travaux de Woodside [25] portant sur l'évaluation de performance des systèmes distribués. Ils ont utilisé une formulation mathématique du taux de passage à l'échelle. Ce taux est défini comme étant le rapport entre la productivité du système sous une configuration de base et sa productivité sous une autre en variant un facteur d'échelle. La deuxième [50], propose une modélisation avec les LQM [57] (Layered Queueing Models) d'une application de gestion basée sur CORBA et l'analyse de ce modèle avec la méthode MOL [47] (Method of Layers).

2.2 Les métriques de performance

Une métrique de performance est un critère de mesure choisi pour quantifier les performances d'un système. Dans cette section, nous allons d'abord donner la définition de certaines métriques de performance qui existent dans la littérature [22], indépendamment des applications de gestion de réseaux. Nous allons ensuite, adapter chacune de ces définition dans le contexte de l'évaluation de performance des applications de gestion.

Deux catégories de métriques de performances existent [22]. La première catégorie met en relation les trois paramètres relatifs à un service d'un système *temps-débit-resource* : le temps mis par le système pour réaliser un service, le débit avec lequel le service est réalisé et les ressources consommées lors de la réalisation du service. Cette catégorie contient les trois familles de métriques suivantes :

1. Les métriques de réponse : ces métriques caractérisent la rapidité d'un système. Elles mesurent le temps écoulé entre l'invocation et la fin d'une opération. Pour une application de supervision deux métriques de réponse sont utilisées :
   - Le temps de réponse d'une simple opération de gestion. Par exemple, le temps nécessaire à un manager SNMP pour récupérer la valeur d'un ou d'un ensemble d'attributs depuis un ou plusieurs agents.
   - Le temps de début à l'achèvement d'un ensemble d'opérations de gestion. Par exemple, le temps nécessaire pour résoudre un problème de congestion dans un réseau ou celui nécessaire au rénumérottage d'un réseau de grande taille.

   Pour les applications de supervision, ces métriques de réponse doivent être minimisées et leurs points de mesure se situent au niveau du manager.
2. Les métriques de production : ces métriques mesurent la productivité (débit) d'un système. Elles représentent la quantité du travail accomplie par le système par unité de temps. Pour une application de supervision deux métriques de production sont utilisées :

- Le nombre d'opérations de gestion réalisées par unité de temps. Par exemple, le nombre de problèmes résolus par jour.
- La quantité de données de gestion générée pendant une opération de gestion. Par exemple, le nombre d'attributs scrutés par seconde par un manager.

Pour qualifier une application de supervision de performante, il faut qu'elle réalise le maximum d'opérations de gestion tout en produisant un minimum de données de gestion. Ainsi, La première métrique doit être maximisée et la deuxième minimisée. Les points de mesure de ces métriques se situent au niveau du manager.

3. Les métriques d'utilisation : ces métriques mesures les ressources consommées par un système. Nous identifions les métriques d'utilisation suivantes : l'utilisation CPU, l'utilisation mémoire et l'utilisation réseau. Les points de mesure de ces métriques se situent au niveau de l'agent et au niveau du manager. En effet, pour des architectures de supervision des services (comme JMX par exemple), ces métriques sont importantes, puisque l'agent de supervision et l'application supervisée peuvent coexister sur la même machine et vont partager les mêmes ressources. Ainsi, l'agent de supervision doit minimiser ces métriques d'utilisation.

Cette première catégorie propose un ensemble de métriques considérées comme des critères de performances *individuelles* puisqu'elles sont relatives, généralement, à des opérations ou à des algorithmes de gestion de réseaux.

La deuxième catégorie des métriques quantifie le comportement global du système sous test. Dans cette catégorie, nous identifions les métriques suivantes : la fiabilité, la disponibilité et le passage à l'échelle.

1. La métrique de fiabilité est définie par la probabilité qu'une erreur se produise lors de la réalisation d'un service, ou comme étant le temps moyen entre l'apparition des erreurs dans le système. Dans le cas d'une application de gestion nous pourrons mesurer la probabilité d'échec ou d'erreur d'une opération de supervision.
2. La métrique de disponibilité est définie comme étant la fraction de temps durant laquelle le système est disponible pour répondre aux requêtes des utilisateurs. Cette métrique est importante pour une infrastructure de supervision. Le but ultime de cette infrastructure est de superviser un système (des réseaux ou des services). Pour atteindre ce but, elle doit être opérationnelle d'une façon continue.
3. La métrique de passage à l'échelle : dans la littérature, nous trouvons plusieurs définitions [25, 6, 23, 31] du passage à l'échelle d'un système. Woodside et al [25] propose qu'un *système passe à l'échelle s'il peut être déployé d'une façon efficace et économique sur des plages convenablement définies et de tailles différentes.* Dans le cas des systèmes de gestion de réseaux, Vilà et al [55] propose la définition suivante : *un système de gestion de réseaux passe à l'échelle si ses performances ne se dégradent pas avec l'augmentation de la taille du système géré.* Un manager, par exemple, passe à l'échelle lorsque son temps de réponse pour une opération de monitorage d'un million d'objets supervisés ne se dégrade pas de plus de 50% par rapport à son temps de réponse de reférence.

Cette seconde catégorie propose des critères de performance considérés comme *globales*. Elles sont relatives au fonctionnement global de l'infrastructure de supervision. Cependant, la première catégorie des métriques peut être mesurée individuellement ou globalement pour le système sous test. Il faut noter que ces métriques sont fortement liées à la granularité des opérations de gestion. Par exemple, lors de l'évaluation des performances d'un manager en choisissant comme métrique le temps de réponse, nous devons spécifier la granularité de l'opération de gestion sous test. S'agit-t-il d'une simple opération requête/réponse ? ou d'une opération complète de scrutation ?

2.2.1 Granularité des opérations de gestion

Une opération de supervision signifie la récupération de la valeur d'un ou de plusieurs attributs de gestion par le manager en interrogeant un ou plusieurs agents. Nous identifions deux type d'opérations de gestion : unitaires et globales. Une opération unitaire est la composition d'une requête et d'une réponse envoyée par un manager vers un agent. Liotta et al [35] ont pris ce type d'opération pour évaluer des architectures de gestion basées sur les agents mobiles. Une opération globale est la composition d'un ensemble de requêtes et de réponses entre le manager et plusieurs agents. Stadler et al [33] ont pris ce type d'opération globale dans l'évaluation de leur approche de gestion basée sur les patterns de navigation.

2.2.2 Métriques de performance utilisées en supervision

Dans les études d'évaluation de performance des architectures de gestion de réseaux les métriques les plus utilisées sont le **volume de trafic** généré par une opération de gestion et son **temps de réponse**.

Le temps de réponse a été mesuré dans les différentes études d'évaluation de performance des systèmes de gestion. Luderer et al [37] ont mesuré ce temps au niveau du manager. Bivens et al [3] ont mesuré le temps de traitement d'une requête au niveau d'un agent. Chen et al [7] considèrent le temps de réponse d'une application de gestion comme étant le temps moyen pour détecter le changement de valeur d'un attribut de gestion au niveau d'un agent. Il s'agit du temps nécessaire pour qu'une application de supervision s'aperçoit que la valeur d'un attribut a changé au niveau d'un agent. Dans [7], l'analyse de cette métrique a été effectué pour différentes approches de gestion.

Le volume de trafic généré par une infrastructure de gestion a été analysé par la majorité des études d'évaluation de performance. Zapf et al, [58] ont montré qu'une infrastructure SNMP composée d'un superviseur et de 100 agents, effectuant une opération de monitorage toutes les 5 secondes génère un trafic de gestion de l'ordre de 477 Mo par jour. Ce volume de trafic conséquent, nécessite d'être maîtrisé pour que les applications de gestion ne deviennent pas des sources de congestion dans des réseaux avec des bandes passantes limitées.

La résistance au facteur d'échelle des applications de gestion de réseaux est analysée par [55, 14] en se basant sur la notion du taux de passage à l'échelle (voir section 2.1.5). Frolund et al [14] ont analysé ce critère pour une application distribuée de gestion en prenant comme facteur d'échelle le nombre des agents de gestion. Pour une architecture de gestion basée

sur JMX [12], nous pouvons appliquer cette métrique sur un agent en mesurant son taux de passage à l'échelle en variant le nombre des MBeans qui y sont enregistrés. Cette métrique s'applique ainsi sur deux niveau : au niveau de manager en prenant comme facteur d'échelle le nombre des agents et au niveau d'un agent en prenant comme facteur d'échelle le nombre d'objets de gestion.

Cette analyse des métriques de performances d'une application de supervision, nous a permis de les classifier selon leurs points de mesure et selon leurs points de perception. Nous identifions deux points de mesure : les métriques mesurées sur le manager et les métriques mesurées sur l'agent. Les points de perceptions sont de deux types : individuels ou globaux. Les métriques individuelles sont relatives aux opérations ou aux algorithmes de gestion. Dans cette famille nous identifions la première catégorie de métriques présentées précédemment (réponse, production et utilisation). Cependant, les métriques globales quantifient le comportement global de l'application de supervision sous test. La figure 2.6 représente le modèle de classification que nous proposons pour les métriques de performance des architectures de gestion.

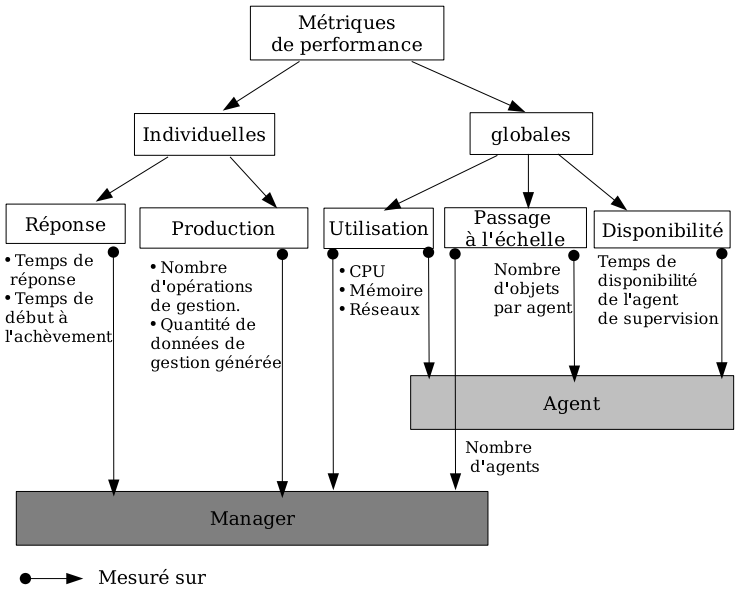

FIG. 2.6 – Un modèle de classification de métriques de performance des architectures de gestion.

2.3 Les paramètres de performance

Chaque évaluation de performance nécessite la définition d'un ensemble des paramètres que représentent les entrées du processus d'évaluation et influent considérablement les résultats de performance d'un système. La liste de ces paramètres peut être divisée en deux catégories : les paramètres du système et les paramètres de la charge (workload). Les paramètres du système comportent les paramètres matériels et logiciels que ne varient pas sous différentes configurations de système sous test. Les paramètres de la charge représentent les caractéristiques des requêtes de supervision, ou les fonctions de gestion. Ils varient d'une configuration à une autre du système sous test. Le choix de valeurs de ces paramètres est une étape importante dans tout projet d'évaluation. Dans les projets d'évaluation de performance d'architectures de gestion, les valeurs de certains paramètres de performance (taille de l'agent mobile, coût d'une transaction CORBA...) peuvent être estimées d'une façon précise. Cependant, d'autres paramètres dépendent fortement de l'application de gestion (types et caractéristiques des objets de gestion, nombre d'objets de gestion, nombre d'objets par requête, fréquence de scrutation...) et leurs valeurs sont souvent plus difficile à déterminer.

2.4 Les facteurs de performance

Les facteurs de performance sont un sous ensemble des paramètres de performance. Ce sous-ensemble contient les paramètres qui varient au cours du processus de l'évaluation de performance d'un système. Généralement, ce sont les paramètres qui influent le plus les performances du système sous test. Les valeurs de ces paramètres se caractérisent par des *niveaux*, représentant leur degré de variance. Certaines études de performance des applications de gestion que nous avons examinées, utilisent le nombre d'agents de supervision comme facteur de performance [3, 48]. D'autres [16], utilisent la fréquence de scrutation comme facteur. Pavlou et al [43] ont utilisé le nombre d'objets de gestion par méthode comme facteur pour évaluer les architectures de gestion basées sur les services web. Dans [2, 10], la topologie de réseau est utilisée comme facteur de performance. Les topologies les plus utilisées sont les LAN Ethernet et les topologies Internet avec une latence moyenne de 120 ms [33]. Le type de la charge ou les services demandés aux agents par le manager est aussi un facteur de performance utilisé par [19].

Nous identifions les facteurs suivants comme facteurs les plus utilisés dans l'évaluation des architectures de gestion :

- Le nombre d'agents de supervision : nous permet de mesurer l'impact du passage à l'échelle sur les performances d'un manager de supervision.
- Le nombre d'objets de gestion par requête : nous permet de mesurer son impact sur le temps de réponse et le volume de trafic généré.
- La fréquence de scrutation : nous permet de mesurer son impact sur le trafic de gestion généré.
- Le type de services sous test : nous permet de mesurer l'impact du service demandé au système sous test sur le temps de réponse et sur le volume de trafic de gestion.

Nous constatons que ces facteurs sont relatifs à certaines métriques de performance. La corrélation de l'influence de ces facteurs avec ces métriques de performance devra nous permettre d'avoir une compréhension claire de performance de système sous test. Nous pouvons dégager,ainsi, les axes d'évaluation suivantes :

- Le passage à l'échelle : inclut comme facteurs, le nombre d'agent de supervision et le nombre d'objets de gestion implanté dans l'agent,
- La résistance à la charge : inclut comme facteurs, la fréquence de scrutation et le nombre d'objets de gestion dans une requête,
- L'importance du type de l'objet de gestion,
- L'importance du type de service sous test (Get, Set, Notification),
- L'efficacité des implantations, essentiellement, des agents de supervision,

À partir de ces axes d'évaluation, nous pouvons déterminer quels sont les principaux facteurs à inclure pour évaluer les performances d'une architecture de gestion.

2.5 Le facteur d'homogénéité

Un facteur de performance important est l'hypothèse d'homogénéité. Cette hypothèse suppose que tous les domaines, les agents de gestion et les charges sont identiques, et que les activités de gestion sont symétriques tout au long de ces domaines. Ceci veut dire qu'on suppose que l'infrastructure de gestion est homogène. Les hypothèses suivantes caractérisent l'homogénéité d'une infrastructure de gestion :

1. Tous les domaines de gestion sont homogènes si leurs architectures de gestion respectives sont identiques.
2. Tous les sites sont homogènes si leurs charges respectives sont identiques.
3. Tous les agents de gestion sont homogènes si leurs fournisseurs d'implantations sont identiques.
4. Toutes les charges de gestion sont homogènes si leurs services et objets de gestion sont identiques.
5. Tous les services de gestion sont homogènes si leurs opérations de gestion sont identiques.
6. Tous les objets de gestion sont homogènes si :
   - Le nombre d'objets de gestion à collecter depuis les agents est le même.
   - Le type d'objets de gestion à collecter depuis les agents est le même.

Toutes les études de performance d'architectures de gestion (analytique et simulation) prennent en considération cette hypothèse. C'est aussi le cas de l'évaluation de performance par la technique de mesure, où l'hypothèse d'homogénéité est quasi générale. Quelques études, par exemple Pavlou et al [43], ont varié les plates-formes de services web et d'agents SNMP utilisés lors de l'évaluation de performances d'une approche de gestion basée sur les services web.

Cette hypothèse d'homogénéité, généralement implicite dans la majorité de cas, est lourde dans l'évaluation de performance des architectures de la gestion de réseaux, parce qu'elle ne reflète pas une plate-forme de supervision réelle qui est dans la plupart des cas hétérogène.

Nous définissons un nouveau facteur de performance pour l'évaluation des architectures de gestion. Ce facteur se présente sous la forme du degré d'homogénéité et caractérise cet aspect crucial de ces architectures. Le degré d'homogénéité d'un système est défini comme étant le produit des degrés d'homogénéité de ses groupes de composants. Le degré d'homogénéité d'un groupe r, est défini par la formule suivante :

$$GrHom(r) = \frac{1}{\text{nombre de classes d'équivalences différentes dans r}}.$$

Ainsi, le degré d'homogénéité d'un système est :

$$SysHom = \prod_{1 \leq i \leq n} GrHom(r_i).$$

Un système est homogène si son degré d'homogénéité est égal à 1. Nous prenons comme exemple une infrastructure de gestion composée de 5 agents SNMP situés dans un seul site, avec une charge identique et deux agents parmi les 5 sont des agents Cisco et le reste sont des agents NET-SNMP. Le degré d'homogénéité du groupe d'agents de gestion est de $\frac{1}{2}$ (nous avons deux types d'agents différents), le degré d'homogénéité des sites de gestion est 1 (un seul site) , le degré d'homogénéité des charges est 1, ce qui nous ramène à un degré d'homogénéité de l'infrastructure de gestion égal à $\frac{1}{2}$.

2.6 La charge

La charge (workload) représente l'ensemble de services demandés au système sous test, au cours du processus d'évaluation de ses performances. La charge est caractérisée selon la technique d'évaluation utilisée. Dans les modèles analytiques, elle se caractérise par un modèle représentant les probabilités d'arrivée des requêtes sur le système sous test. Dans les études basées sur les simulations, nous pouvons utiliser soit des traces de requêtes mesurées sur un système réel ou un modèle représentant ces requêtes (caractérisation du trafic de supervision). Dans la technique de mesure, nous pouvons utiliser soit une charge de gestion réelle basée sur un trafic opérationnel, soit des injecteurs de requêtes demandant des services au système sous test. Dans tous les cas, il est important que la charge soit représentative d'une utilisation réelle du système. Deux types de charges existent : les charges synthétiques et les charges réelles. Une charge réelle corresponde à une charge observée sur un système réel au cours de son utilisation pour des opérations ordinaires. Ce type de charge ne peut être régénéré en raison de son caractère variable au cours de temps. En effet, il ne convient pas comme charge de test pour les études d'évaluation de performance. Cependant, la charge synthétique qui possède les mêmes caractéristiques qu'une charge réelle, peut être régénérée plusieurs fois d'une façon contrôlée. C'est ce type de charge qui convient le mieux pour les études d'évaluation de performance. Une raison majeure d'utiliser ce type de charge pour

l'évaluation de performance, est qu'elle représente un modèle d'une charge réelle avec la caractéristique d'être facilement modifiée.

La charge d'une application de gestion de réseau, peut être représentée sous la forme de l'ensemble de requêtes envoyées par les algorithmes de gestion vers les agents. Par exemple Pattinson [42] a évalué les performances de SNMPv1 avec une charge qui se caractérise par des tailles respectives de requête et de réponses SNMPv1 de 90 et 100 octets. Il a utilisé un intervalle de scrutation variant entre 450 et 60 secondes. Le choix de ces valeurs d'intervalles n'est pas arbitraire, puisqu'elles sont utilisées par des systèmes de gestion réels (SUN's Network manager et tkined).

Un autre type de charge important dans l'évaluation de performance de gestion de réseau est la charge fonctionnelle du service supervisé. Cette charge caractérise la partie fonctionnelle d'un service sur lequel agit le système de gestion. À titre d'exemple, la charge caractérisant un serveur applicatif supervisé par une architecture de gestion. Certaines études [43, 9, 29] de performance n'ont pas considérées ce type de charge. Elles supposent que la seule charge dans le système sous test est celle de l'application de supervision. D'autre études, comme celle de Pattinson [42] ont considéré un trafic HTTP comme charge du service fonctionnel. La coexistence, de ce deux types de charge permet de mesurer l'impact des algorithmes de supervision sur les services fonctionnels supervisés. Une fonction caractérisant cet impact sera présentée dans le chapitre suivant.

2.7 Examen et analyse des études d'évaluation de performance

Dans cette section nous présentons une synthèse des méthodologies utilisées par certaines études publiées relatives à l'évaluation de performance des architectures de gestion. Les tableaux 2.8 et 2.9 présentent les éléments de base des études d'évaluation de performance que nous avons examinées. Ces éléments de base incluent, les techniques utilisées, l'architecture de gestion sous test, le facteur d'hétérogénéité des agents de gestion (nous avons examiné, essentiellement, l'hétérogénéité des implantations des agents sous test) et les métriques de performance choisies. Les tableaux 2.10 et 2.11 présentent l'ensemble de facteurs et de paramètres de performance utilisés par ces études. Cet ensemble inclut les caractéristiques des sites de gestion (nombre d'agents et de managers), les paramètres de la charge (nombre d'objets de gestion, nombre d'objets par requête, fréquence de scrutation), le modèle de communication (caractéristiques du réseau) et la fonction de gestion sous test (monitorage, résolution de problèmes) et son mode d'interaction (*pull-driven* ou *push-driven*).

Dans la figure 2.7, nous présentons une classification des méthodologies proposées par ces études selon leurs techniques d'évaluation, leurs métriques de performance et les architectures de gestion qui ont été analysées. Sur cette figure chaque étude est marquée par une étiquette qui correspond à l'auteur et l'année de publication de l'étude. Dans le cas d'une étude portant sur la comparaison de performances de deux ou plusieurs architectures de

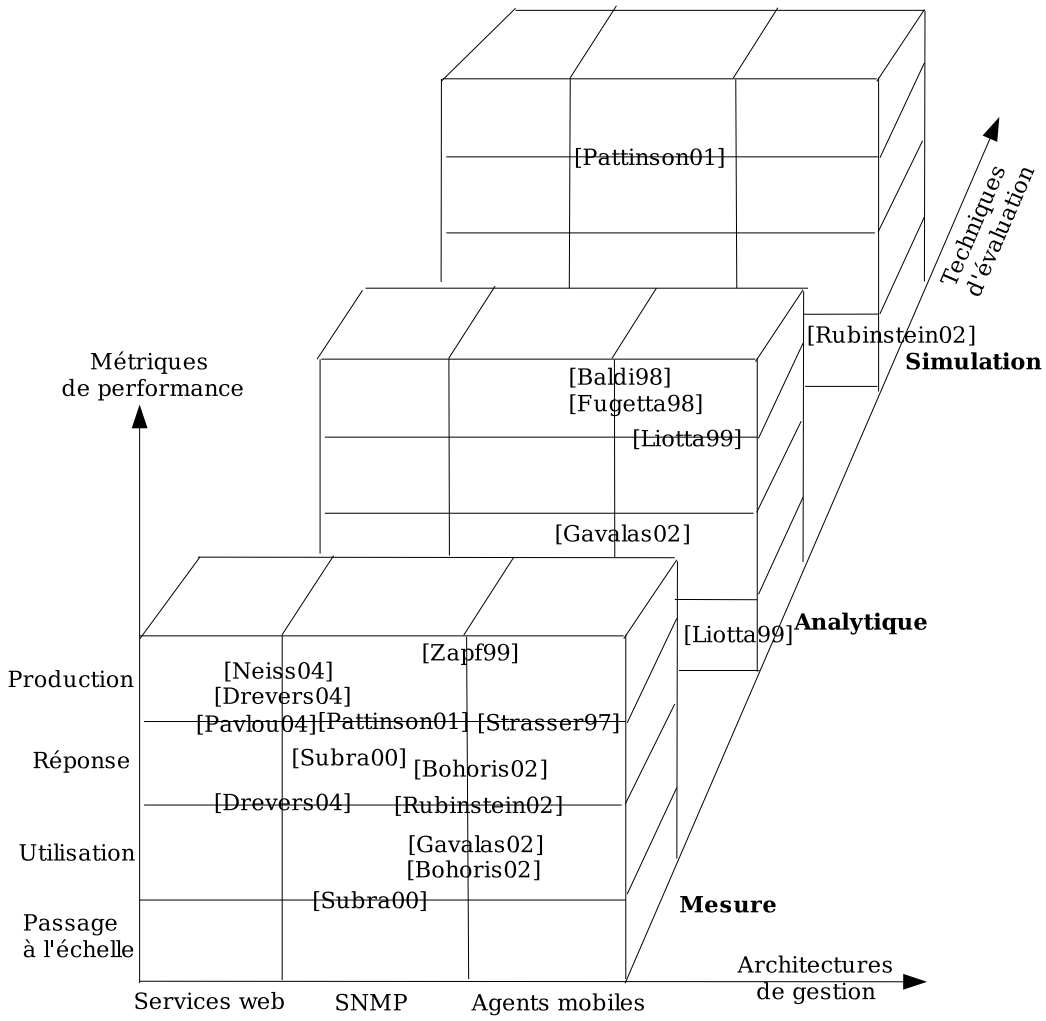

FIG. 2.7 – Classification des études d'évaluation de performance de certaines architectures de gestion.

gestion, elle est placée sur le croisement des cubes correspondants à ces architectures (par exemple l'étude de Gavalas [Gavalas02]).

Nous remarquons qu'il y a une forte tendance dans la communauté de gestion à s'appuyer sur des techniques de mesure plus que sur les deux autres techniques (simulation et analytique) pour évaluer les performances des architectures de supervision de réseaux. Ceci est dû au fait que des architectures de gestion comme SNMP ou les agents mobiles ont été implantées et déployées à de multiples échelles, ce que favorise la mesure de leurs performances sur des plates-formes ou des prototypes réels. Ce recours à la technique de mesure, nous permet de constater qu'il y existe un manque de modélisation des architectures de gestion en utilisant des approches analytiques standard telles que les files d'attentes, les Réseaux de Pétri ou les chaînes de Markov, etc. En effet, il n'existe pas de modélisation commune des architectures de gestion mais seules les implantations sont comparables. Néanmoins, nous pouvons expliquer ceci par le fait que les gens qui maîtrisent ce genre d'approches analytiques ne sont pas forcément des gens qui maîtrisent la gestion de réseaux et vice versa. Nous constatons également une forte variété de paramètres et de facteurs de performance entre ces différentes études. Par exemple, les variables de gestion varient d'un simple objet [48] à des tables SNMP de la MIB-II [19]. Parmi toutes les études que nous avons examinées, deux études [16, 9], seulement, ont utilisé le même objet de gestion (table *ifTable* de la MIB-II de SNMP) dans leurs projets d'évaluation de performance. Ainsi, il n'existe pas un ensemble de paramètres de performance prédéfinis communs à tous les projets d'évaluation de performances des architectures de gestion.

2.8 Conclusion

Dans ce chapitre nous avons examiné les études portant sur l'évaluation de performance des architectures de gestion. Nous avons dégagé l'ensemble des éléments de base (techniques, métriques, paramètres, facteurs et charge) de l'évaluation de performance utilisés par ces études. Nous avons également mis en évidence la diversité des ces études vis à vis de ces éléments de base. Dans un futur proche il sera nécessaire de proposer un ensemble de recommandations portant sur le choix de ces éléments de base afin d'uniformiser les études futures.

Élements de base				
Étude	Technique	Architecture de gestion	Hétérogeneité des agents	Métriques de performance
[M.J.Katchabaw et al, 97,96]	Mesure	Architecture de gestion OSI:manager/agent	None	- Métriques de réponse: temps de réponse d'une requête,temps d'un cycle (start-to-completion), temps d'initialisation - Métriques de productivité: volume du travail par unité de temps. - Métriques d'utilisation: ressources consommées. - Métriques de pannes et d'erreurs: taux d'erreur, taux de panne
[F.Sheikh et al, 97]	Analytique et simulation	Architecture de gestion basée sur CORBA	None.	- Temps de réponse - Passage à l'échelle
[M.Strasser et al, 97]	Analytique et mesure	Architecture de gestion basée sur les agents mobiles (MOLE) avec une mobilité faible	None	- Trafic généré - Temps d'exécution
[G. Luderer et al, 97	Mesure	- Architecture de gestion basée sur des agents intelligents JAVA - Modèle Client/serveur - Comparer avec CMU-SNMPv2	- Agents Java - Agents SNMP	Temps de réponse mesuré au niveau de manager
[M.Baldi et al, 98]	Analytique et quantitatif	- Architecture de gestion basée sur les agents mobiles: mobilité faible. - Comparer avec SNMPv1	None	Trafic de gestion généré
[A.Fugetta et al, 98]	Analytique	- Architecture de gestion basée sur les agents mobiles: mobilité faible/forte. - Comparer avec SNMP.	None	Trafic de gestion généré
[M.Zapf et al, 99]	Mesure	- Architecture de gestion basée sur les agents mobiles: AMETAS (mobilité faible) - comparer avec SNMPv1	None	Trafic de gestion généré
[A.Liotta et al, 99]	Analytique	Architecture de gestion basée sur les agents mobiles: mobilité faible	None	- Trafic de gestion généré - Délai de monitorage: nombre d'unités de temps nécessaires pour réaliser une opération de monitorage (requête/réponse) - Passage à l'échelle en fonction d'entités gérées, fréquence de scrutation et diamètre de réseau
[R.Subramanyan et al, 2000]	Mesure	Architecture de gestion Hiérarchique basée sur SNMP:SIMONE	None	- Délai de monitorage - Consommation CPU - Bande passante consommé - Passage à l'échelle
[H.T.Ju et al, 2000]	Mesure	Architecture de gestion basée sur les serveurs web embarqué et légés (EWS) et SNMP:Applet Java	None	- Taille de code (empreint mémoire) - Mémoire utilisée lors de l''exécution - CPU utilisée - Nombre de connections supportées
[T.Chen et al, 2001]	Analytique	-Architectures de gestion Distribuées. (SNMPV1: Client/serveur, Hiérarchique statique, Mobilité faible, Mobilité forte).	None	Temps moyen de changement du valeur d'une variable, trafic généré, temps moyen d'utilisation du CPU,Mémoire moyenne consommée.

FIG. 2.8 – Éléments de base retenus par les études d'évaluation de performance des architectures de gestion de réseaux-(1)

Étude	Technique	Architecture de gestion	Hétérogeneité des agents	Métriques de performance
[R.Stadler et al, 2001]	Analytique et mesure	Architecture de gestion basée sur les patterns de navigation: STAR pattern et ECHO pattern	None	-Complexité de trafic: volume de trafic sur le réseau pour réaliser une opération de gestion globale - Complexité de temps:Temps nécessaire pour réaliser une opération de gestion globale
[C.Pattinson, 2001]	Analytique et simulation	Architecture de gestion basée sur SNMPv1	None	- Trafic de gestion généré - Temps de réponse
[D.Gavalas et al, 2002,2000]	Analytique et mesure	Architecture de gestion basée sur les agents mobiles:Mobilité forte - Comparer avec SNMPv1: AdventNet	None	Bande passante utilisée, en fonction de l'intervalle de scrutation
[M.G.Rubinstein et al, 2002,2000]	Mesure et simulation	Architecture de gestion basée sur les agents mobiles(Mole): Mobilité faible - Comparer avec SNMPv1	- AdventNet SNMPv1 release 2.2 - Linux UCD-SNMP	Bande passante moyenne consommée par le manager - Temps de réponse moyen pour récupérer une variable de gestion depuis N agents - Passage à l'échelle
[C.Bohoris et al, 2002,2000]	Mesure	Architecture de gestion basée sur les agents mobiles (Grasshopper) et CORBA avec une mobilité faible - Comparer avec SNMPv1	- AdventNet SNMPv1 release 2.0	- Trafic de gestion généré au niveau TCP - Temps de réponse : RTT - Mémoire consommée
[J-K.Chun et al, 2002]	Analytique et mesure	Architecture de gestion TMN basée sur les agents mobiles(mobilité faible ou forte)	- Agent mobile Grasshopper.	Temps de réponse mesuré sur le manager
[A.Bivens et al, 2004,2002,99]	Mesure et simulation	Architecture de gestion hybride basée sur les agents mobiles:DOORS (mobilité faible) et SNMPv1	- SNMP d'un routeur cisco 2500	Bande passante utilisée, Temps de collection de données de gestion, Utilisation CPU, Temps de traitement d'une requête - Temps de service des requêtes des clients
[P.vilà et al, 2004]	Analytique et simulation	Architecture de gestion basée sur les systèmes multi-agents:P2P	None	- Taux de passage à l'échelle
[Q.Gu et al, 2004]	Mesure	Architecture de gestion basée sur CORBA - Comparer avec SNMPv1/v2	- UCD-SNMP - Plate-forme CORBA: TAO	-Volume de trafic de gestion - Latence et temps de réponse
[R.Neiss et al, 2004]	Mesure	Architecture de gestion basée sur les services web et SNMPv1	None	Volume de trafic de gestion
[G.Pavlou et al, 2004]	Mesure	Architecture de gestion basée sur les services web - comparer avec SNMPv1/v2 Et CORBA	- 3 plates-formes des services web: WASP, Apache Axis, gSOAP - 2 plates-formes SNMP: NET-SNMP et AventNet SNMP - Plateforme CORBA: Orbacus	- Volume de trafic de gestion - Temps de réponse mesuré sur la manager
[R.State et al, 2004]	Mesure	Architecture de gestion basée sur SyncML	None	- Volume de trafic de gestion - Temps de réponse d'un agent
[T.Drevers et al, 2004]	Mesure	Architecture de gestion basée sur les services web -Comparer avec SNMPv2	None	- Volume de trafic de gestion d'une opération de gestion (GET, GETBULK) - Temps de réponse d'une opération de gestion - Mémoire utilisée par opération de gestion

FIG. 2.9 – Éléments de base retenus par les études d'évaluation de performance des architectures de gestion de réseaux-(2)

Paramètres de performance									
Étude	#sites	Objets de gestion		Workload			Communication	Fonction de gestion	Mode d'intéraction
		#objets	#objets/req	Fréquence de polling	Données auxiliaires	#série de tests			
[M.J.Katchabaw et al, 97,96]	- 12 noeuds: AIX, RISC/OS SunOS,DYNIX/ptx - 1 manager	1 objet de gestion par processus géré (DCE server, DCE client,..)	N/A	N/A	None	N/A	- 2 LAN	Monitorage	-Pull-driven -Push-driven
[F.Sheikh et al, 97]	- 1 manager - 3 manager - deux sites avec 3 managers chacun	5000 à 50000 objets	N/A	None	N/A	N/A	WAN avec une latence de 20ms	Monitorage	Pull-driven
[M.Strasser et al, 97]	- 5 noeuds. - 2 sites: Un manager au USA et un cluster de 4 agents en allemagne	N/A	N/A	N/A	- Taille du l'agent mobile: 10Ko - Taille de données: 32 Ko	3 scénarios de tests	- Ethernet LAN:10 Mbits/s - Connexion WAN entre le cluster et le manager	Monitorage	- Pull-driven
[G. Luderer et al, 97	- 4 noeuds: Unix, Win95, MAC - 1 manager	Variables des groupes *ip* et *system* de la MIB	1 groupe/req	N/A	None	N/A	- Ethernet LAN: 10Mbits/s - Agents Java utilisent TCP	Monitorage	- Pull-driven
[M.Baldi et al, 98]	- 12 noeuds: AIX, RISC/OS,SunOS,DYNIX/ptx - 1 manager	5 objets de la MIB-II	5 objets/req	N/A	-Taille requête SNMP est 48 octets - Taille réponse SNMP est 66 octets - Taille code mobile est 5.6Ko	None	- 1 LAN - gestion à distance d'1 LAN -interconnexion de N LANs - Agents mobiles utilisent TCP	Monitorage	Pull-driven
[A.Fugetta et al, 98]	1 manager et N agents	Variable: charge réseau sur chaque interface	Q instructions SNMP pour récuperer la valeurs de variables	N/A	None	None	Réseau uniforme avec une latence constante	Monitorage	Pull-driven
[M.Zapf et al, 99]	100 noeuds	10 variables de la MIB HP-UNIX	10 objets/req	5 secondes	-Taille requête SNMP est 86 octets - Taille réponse SNMP est 90 octets - Taille code mobile est 2.5Ko	17280 collecte par jour	Ethernet LAN: 10 Mbits/s	Monitorage	Pull-driven event-driven
[A.Liotta et al, 99]	N noeuds	N variables	N/A	1 à 5 requete par seconde	Taille de la requete et de la réponse est identique	None	Graphe connecté avec un modèle de communication de type: *All Ports-Full Duplex*	Monitorage	Pull-driven
[R.Subramanyan et al, 2000]	- 128 machines SUN avec Solaris 5.6 - émulation de 1024 agents	Une variable de la MIB: Temps CPU consommé par un processus	N/A.	2 à 3 secondes	None	20 à 25 séries	Ethernet LAN	Monitorage	Pull-driven
[H.T.Ju et al, 2000]	9 machines avec des serveurs web embarqués propriétaire - Routeur propriétaire	Variables du MIB SNMP	N/A	N/A	N/A	N/A	Ethernet LAN	Monitorage	Pull-driven
[T.Chen et al, 2001]	N noeuds	Une variable	1 objet/req	Dynamique	None	None	N/A	Monitorage	Pull-driven

FIG. 2.10 – Facteurs de performance retenus par les études d'évaluation de performance des architectures de gestion de réseaux-(1)

Paramètres de performance									
Étude	# sites	Objets de gestion		Workload			Communication	Fonction de gestion	Mode d'intéraction
		# objets	# objets/req	Fréquence de polling	Données auxiliaires	# série de tests			
[R.Stadler et al, 2001]	N noeuds	N/A	N/A	N/A	Taille des messages de gestion est de 1024 octets	None	- Réseau sous forme de graphe avec des liens bidirectionels. - Ethernet LAN:10Mbits/s. - Internet avec une latence de 120ms et 34 hops	Moritorage	Pull-driven
[C.Pattinson, 2001]	1 manager,120 agents et 5 serveurs possédant chacun 2 interfaces.	- Utilisation de chaque interface d'un agent. - Débit du protocole IP pour un agent. - Paquets perdus et taux d'erreurs au niveau IP et TCP pour un agent	1 variable/requête	- 450 secondes - 60 secondes.	Taille moyenne d'une requete/réponse SNMP est entre 90 et 100 octets	Temps total de test est de 60 minutes	4 Ethernet LAN 10 Mbits interconnectés par des liens de 1.544Mbits	-Monitorage - Fault finding	Pull-driven
[D.Gavalas et al, 2002,2000]	3 LANs: 50 noeuds	Table des interfaces de la MIB-II: *ifTable*	1 objet/req	Varient ente 10 à 40 secondes	- Taille de messages de requete et de réponse est:90 octets - Taille de l'agent: 1.25Ko - Taille de la table des interfaces est 8x21 entrée	10 séries	- Ethernet LAN: 10Mbits/s - WAN: 10Mbits/s et 64 Kbits/s	Moritorage	Pull-driven
[M.G.Rubinstein et al, 2002,2000]	1 manager et 250 noeuds	Variable *ifnErrors* de la MIB-II	1 objet/req	N/A	- Taille de l'agent mobile: 1.5kbps - Taille de la requete SNMP est 42 octets - Taille de la réponse SNMP est 51 octets	10 séries	- Ethernet LAN: 10Mbits/s: Latence de 10us - Topologie Internet: 2Mbps	Moritorage	Pull-driven
[C.Bohoris et al, 2002,2000]	1 manager et 1 agent	1 tableau d'objets de type double contenant 25,50,75 et 100 nombres	1 tableau/req	N/A	NA	100 séries	- Ethernet LAN: 100Mbits/s - Réseau faiblement chargé	Moritorage	Pull-driven
[J-K.Chun et al, 2002]	1 manager et 84 agents	50 variables d'une MIB privée	N/A	N/A	None	20 séries	Réseau de communication PC basé sur ICPS/AICPS	Moritorage	Pull-driven
[A.Bivens et al, 2004,2002,99]	- Simple réel AS avec 1 manager. - Multiple AS simulé avec plusieurs manager	N/A	N/A	3,5,7 et 10 secondes	-Taille de la requête SNMP est 161 octets - Taille de la réponse SNMP est 175 octets - Taille d'une transaction DOORS est 186 octets	Temps total de tests est 1000 secondes	- Ethernet LAN: 10Mbits/s - Routage OSPF	Moritorage	Pull-driven
[P.vilà et al, 2004]	N noeuds	N/A	N/A	N/A	None	N/A	Réseaux MPLS et ATM	-Monitorage - contrôle de la congestion	Pull-driven
[Q.Gu et al, 2004]	1 manager et 1 agent	- 1 objet - 3 objets - table de routage BGP de taille varie entre 1 et 2000 entrées	- 1 objet/req - 3 objets/req - GET-BULK sur la table de routage BGP	N/A	- Taille d'une requete SNMPv1 (GET-NEXT) varie entre 87 et 105 octets - Taille d'une requete (GET-BULK) SNMPv2 varie entre 511 et 986 octets	100 séries de tests	Ethernet LAN:100Mbits/s et 10Mbits/s	Moritorage	Pull-driven
[R.Neiss et al, 2004]	1 manager, une gateway et 1 agent	1 à 80 objets	1 objet/req	None	None	1 série de test	Ethernet LAN	Moritorage	Pull-driven
[G.Pavlou et al, 2004]	1 manager et 1 agent	Table de connexions TCP	- 1 attribut/method - 8 attributs/method - 1 MO/methode - 40 MO/methode	None	None	1 série de test	Ethernet LAN:100Mbits/s	Moritorage	Pull-driven
[R.State et al, 2004]	1 manager et 1 agent: Une seule machine	Objets de gestion aléatoires	- 1 message SyncML par requête - 1 message SyncML par ensemble de requête	None	Taille de l'agent SyncML est 23Ko	1 à 700 opérations de gestion (Get/Set)	Test en Loopback	Moritorage	Pull-driven
[T.Drevers et al, 2004]	1 manager et 1 agent	Table *ifTable* de la MIB SNMP	- 1 objet/req (GET). - n objets/req (GET). - GET-BULK sur la table *ifTable*	None	None	- 10 tests pour le trafic de gestion - 10000 mesures pour déterminer le temps de réponse - 1000 mesure pour déterminer la mémoire consommée	Ethernet LAN: 10 Mbits/s	Moritorage	Pull-driven

FIG. 2.11 – Facteurs de performance retenus par les études d'évaluation de performance des architectures de gestion de réseaux-(2)

Chapitre 3

Méthodologie d'évaluation de performance des systèmes de supervision

Dans le chapitre précédent, nous avons examiné les méthodologie proposées par les études relatives à l'évaluation de performance des systèmes de supervision. Nous avons constaté que ces méthodologies sont hétérogènes au niveau des techniques, paramètres, facteurs et charge de performance. Ainsi, afin de combler cette hétérogénéité, nous proposons une méthodologie pour l'évaluation de performance de systèmes de supervision. Cette méthodologie est basée sur ces études que nous avons examinées et elle représentera un support pour notre travail de recherche dans ce domaine.

3.1 Choix de techniques d'évaluation

L'objectif de cette section est de définir les critères de choix d'une ou plusieurs techniques adéquates pour l'évaluation de performance d'une architecture de supervision. Comme on vient de le voir dans le chapitre précédent (section 2.1), il existe trois techniques candidates d'évaluation de performance : mesure, analytique et simulation. L'utilisation d'une seule technique pour évaluer les performances d'un système n'est pas recommandée. Elle sera insuffisante pour évaluer de façon pertinente et crédible les performances d'une architecture de gestion. Il faut envisager d'utiliser au moins deux techniques d'une façon séquentielle, pour que l'une puisse valider les résultats de l'autre. Idéalement, la technique utilisée, devrait répondre aux critères suivants :

- Le coût de la technique : ce coût ne doit pas être considérable en terme de temps de réalisation des tests et de leurs coût financier. La technique de mesure est la plus coûteuse puisqu'elle nécessite l'existence d'un prototype de l'architecture de gestion

à évaluer. La technique analytique est la moins coûteuse en terme de coût financier puisqu'elle ne nécessite pas le développement ou le déploiement d'un prototype du système de supervision. La technique de simulation possède un coût variable. Son coût dépend de type de simulation utilisé (*trace driven, execution driven, event driven, complete or components system simulation*) [22], de l'existence ou non d'un modèle de l'architecture de gestion à simuler dans l'outil de simulation utilisé.

- La réalisabilité de la technique : le choix de la technique dépend de l'environnement du système de supervision sous test. Certains environnements peuvent défavoriser l'utilisation d'une technique par rapport à une autre. L'évaluation de performance d'un système de supervision embarqué dans un satellite défavorise l'utilisation de la technique de mesure. Les mesures vont altérer et dégrader le fonctionnement du système (consommation excessive des batteries). Dans ce cas, la technique de simulation ou analytique sont plus adéquates. En revanche, la non disponibilité des modèles communs pour les architectures de gestion favorise l'utilisation de la technique de mesure, étant donné l'existence de prototypes de ces architectures.
- La flexibilité de la technique : le modèle utilisé par une technique d'évaluation doit être facile à modifier et à étendre. Les architectures de gestion et leurs environnements sont constamment en évolution et sous spécification. Ainsi, la technique d'évaluation utilisée, doit offrir la possibilité d'une intégration facile de ces évolutions dans les modèles élaborés.
- La précision de la technique : elle se présente sous la forme du degré de précision des résultats obtenus par une technique d'évaluation de performance d'un système de supervision. la techniques analytique est généralement la moins précise. Elle nécessite des simplifications du système sous test afin de construire un modèle calculable sous forme des formules mathématiques pour donner une estimation des métriques de performance. Les modèles élaborés par la technique de simulation incluent plus de détails du système sous test et nécessitent moins de simplifications. Ainsi les résultats seront potentiellement plus précis. Ces modèles sont plus proches du système réel de supervision. La précision de la technique de mesure est variable. Le degré de précision de résultats obtenus peut être fort comme faible. Cela est du au choix de l'ensemble des paramètres, des facteurs et de la charge de performance qui sont relatifs à l'expérimentation réalisée par cette technique.
- L'impact de la technique : la technique choisie ne doit pas altérer les performances du système de supervision sous test et le système supervisé.

Nous constatons que ces critères peuvent être en conflit, lors du choix d'une technique adéquate pour évaluer les performances d'un système de supervision. Par exemple, la technique analytique est la moins coûteuse mais elle est la moins précise. Systématiquement, on peut se baser sur les deux critères suivants pour choisir une technique parmi les trois :

- Si le système sous test existe et qu'il est accessible avec un effort raisonnable, nous pouvons utiliser la technique de mesure.

- Si le système sous test n'est pas implanté ou s'il est trop compliqué l'accès, alors un modèle de performance doit être développé en utilisant la technique analytique ou en construisant un modèle de simulation pour analyser ses performances.

3.2 Choix de métriques d'évaluation

Le but ultime des métriques de performance des architectures de gestion est de fournir aux opérateurs et aux équipementiers de gestion une compréhension commune de performance des architectures qu'ils utilisent. Pour atteindre ce but, ces métriques doivent satisfaire des critères qui sont recommandable de les appliquer avant toute évaluation de performance d'une architecture de gestion. S'inspirant des travaux proposés par le groupe du travail IPPM [20] au sein de l'IETF, portant sur les performances du protocole IP, nous définissons cet ensemble de critères de choix des métriques d'évaluation de performances des systèmes de supervision :

- Les métriques doivent être concrètes et bien définies : il faut essentiellement éviter les métriques stochastiques (probabilités) et se focaliser sur des métriques déterministes. Par exemple, il est recommandé d'éviter des métriques de genre la probabilité de réponse d'un agent de supervision à des requête envoyées par un superviseur. Mais définir une métrique de genre taux de réponse de l'agent en terme de nombre de requêtes correctes servies par seconde. Par exemple, une mesure selon la définition stochastique de cette métrique nous donne une valeur de genre 0.80, et la deuxième définition nous donne une valeur de genre 80 requêtes correctes servies sur 100. En effet, une définition déterministe nous permet d'éviter certaines hypothèses stochastiques qui peuvent être implicites par une définition probabiliste.
- La méthodologie de mesure d'une métrique doit avoir le caractère répétitif : si une méthodologie de mesure d'une métrique est utilisée plusieurs fois sous des conditions identiques, les résultats obtenus doivent être quasi identiques.
- Les métriques doivent être capable de refléter les performances de différentes opérations de supervision sous test. Par exemple, une comparaison de performance entre les opérations *getAttribute* et *getAttributes* fournies par l'approche de gestion JMX, nécessite l'utilisation d'une métrique de production mesurant le nombre d'attributs scrutés par seconde par le superviseur et non pas le nombre de requêtes par seconde. Le choix de cette métrique est dû au fait, ce deux opérations n'ont pas le même nombre d'attributs dans une requête.
- Les métriques ne doivent pas être aberrantes pour des architectures de gestion implantées par des technologies identiques. Elles doivent être valables pour ces différentes architectures.
- Les métriques doivent fournir une compréhension claire des architectures de gestion implantées par des technologies non identiques. Une comparaison de performance, par exemple, entre une architecture basée sur SNMP et une autre basée sur JMX, nécessite l'utilisation d'une métrique de réponse mesurant le temps de réponse de la scrutation d'un même objet de gestion implanté avec ces deux technologies.

– Les métriques doivent être utiles aux opérateurs et aux équipementiers pour maîtriser les performances des architectures de gestion qu'ils utilisent.

Cet ensemble de critères est à appliquer sur l'ensemble des métriques que nous avons définies dans le chapitre précédent (voir section 2.2). Ceci nous permet de choisir parmi elles, celles les plus adéquates pour évaluer les performance des systèmes de supervision, ou de les comparer entre eux.

3.3 Paramètres et facteurs de performance

Dans le chapitre précèdent, nous avons donné une définition des termes paramètres et facteurs de performance des architectures de supervision. Dans cette section, nous allons proposé un ensemble non exhaustif, de ces paramètres et facteurs relatifs à l'évaluation de performances de ces architectures. Le modèle superviseur/agent et information de gestion sur lequel se basent généralement, les architectures de supervision, nous servira comme référentiel pour identifier cet ensemble de paramètres et facteurs. Dans notre identification des ces paramètres, nous allons tenir compte, des autres modèles de base de la gestion de réseaux : le modèle de l'information, le modèle fonctionnel et le modèle de la communication. Les paramètres et les facteurs de performance seront relatifs à l'un ou l'autre de ces modèles. Des paramètres relatifs à la charge concernant, essentiellement le modèle de la communication seront examinés dans la section suivante. Dans une première étape, nous dégageons une liste de paramètres de performance d'une architecture de gestion :

1. Paramètres relatifs au superviseur : ces paramètres englobent, essentiellement, le nombre d'agents sous le contrôle du superviseur et les fonctions et algorithmes de gestion qui définissent le comportement du superviseur (FCAPS[1]). Ces fonctions de gestion représentent le modèle fonctionnel de l'architecture de supervision.

2. Paramètres relatifs au réseau : l'ensemble de ces paramètres caractérisent le réseau du système de supervision. Dans cette catégorie, nous identifions les paramètres suivants :
   – La bande passante minimale dans le réseau
   – Le taux de la bande passante minimale dans le réseau, attribué aux activités de gestion.
   – Les distances entre le superviseur et les agent de supervision : ce paramètre peut aussi être représenté par le degré d'éloignement entre le superviseur et un agent. Ce degré d'éloignement est proportionnel à la distance en terme de sauts de routage entre le superviseur et l'agent.

3. Paramètres relatifs à l'agent de supervision : cet ensemble contient les paramètres suivants : la technologie d'implantation d'un agent (SNMP, JMX, NetConf), le nombre d'objets de gestion contenus dans un agent (le nombre de modules MIBs pour SNMP,

[1] Gestion des fautes, de la comptabilité, de la configuration, de la performance et de la sécurité

le nombre de MBeans pour JMX) et le support de la sécurité dans un agent (contrôle d'accès[2], authentification et confidentialité [3])

4. Paramètres relatifs aux objets de gestion ou au modèle de l'information : ces paramètres essentiellement caractérisent les objets de gestion dans l'agent. À titre indicatif cet ensemble de paramètres contient : le modèle d'implantation de ces objets de gestion (modules de MIBs, MBeans, modules NetConf).

Cette liste de paramètres de performance est donnée à titre indicatif. Lors de l'évaluation d'un système de supervision, elle doit être raffinée et complétée par d'autres paramètres, que l'on jugez utiles pour clarifier les performances du système. Dans une deuxième étape, il faut examiner cet ensemble de paramètres et identifier un ensemble de facteurs (les paramètres qui varient au cours du processus d'évaluation) qui peuvent avoir un impact sur la performance du système de supervision sous test.

3.4 Charges d'évaluation de performance

La charge est un élément crucial dans toute projet d'évaluation de performance et son choix affecte considérablement les résultats d'évaluation. Les quatre paramètres à considérer lors de la caractérisation d'une charge d'évaluation sont les suivants : le service exercé par la charge, le niveau de détails de la charge et sa représentativité d'une application réelle de supervision.

3.4.1 Paramètres de sélection de la charge

L'un de moyen utile pour choisir la charge est de considérer le *système sous test* comme étant un fournisseur de service. L'identification claire du système sous test dans tout projet d'évaluation est importante, afin de bien définir les services exercés par la charge. Par exemple, le service de scrutation dans un système de supervision est basé sur des requêtes et c'est leur fréquence qui représentera la charge du système. Il faut noter que la fréquence seule comme paramètre de la charge n'est pas suffisante dans cas. Il faut, aussi, spécifier le type de la requête de gestion. Pour une architecture SNMP par exemple ce type de requête est l'un des suivants : *Get, Get-Next, Get-Bulk.* Il faut, aussi, définir la charge selon son niveau d'interfaçage avec le système sous test. Deux architectures de gestion qui sont comparées au niveau de la couche IP, auront une charge représentée par des paquets IP. Cependant, si elles sont comparées au niveau application, ce sont les requêtes de gestion qui vont représenter la charge du système.

Un autre choix important est le niveau de détails de la charge. Il faut définir ce niveau pour chaque service fourni par le système sous test afin de pouvoir reproduire les requêtes

[2]Dans SNMP, le contrôle d'accès se caractérise par l'utilisation du modèle VACM (View-based Access Control Model)

[3]Dans SNMP, l'authentification et le chiffrement se caractérise par l'utilisation du modèle USM (User-base Security Model)

sollicitant ce service. Plusieurs alternatives existent [22] pour choisir le niveau de détail d'une charge. Parmi ces alternatives, nous citons les suivantes :

- Le type de requêtes de gestion les plus fréquemment utilisées : Par exemple, pour une architecture SNMP utiliser une charge de scrutation basée sur l'opération *Get*.
- La fréquence d'utilisation de chaque type de requêtes : dans ce cas, il faut spécifier pour chaque service de supervision fourni par le système sous test, les caractéristiques et les fréquences de chaque type de requêtes de gestion.
- Une séquence de requêtes prise sur un système réel : cette alternative repose sur une trace d'une charge réelle prise sur un système opérationnel de supervision. En revanche, l'utilisation de cette alternative pour caractériser une charge pose certains problèmes avec des techniques d'évaluation comme la simulation ou la modélisation analytique. Une trace d'une charge réelle fournie trop de détails. Ceci nécessite l'évaluation du comportement exact de chaque composant du système sous test et engendre, ainsi, un sur-coût important du projet d'évaluation de performance basée sur ce type de charge.
- Moyenne des objets de gestion demandées par les requêtes : la modélisation analytique, nécessite juste la spécification de ressources demandées par les requêtes de gestion et non pas la spécification de requêtes en elles-mêmes. Par exemple, il est suffisant de caractériser la charge d'un système de supervision et son évaluation par la technique analytique, en spécifiant qu'une requêtes envoyée par le superviseur sur l'agent sollicite un attribut de gestion et que son le temps d'exécution sur l'agent est de 1 ms. La différenciation de type de requêtes s'effectue en spécifiant pour chaque type son temps d'exécution respectifs. Généralement, la spécification du temps d'exécution n'est pas suffisante, mais il faut spécifier une distribution probabiliste du temps de demande des objets de gestion pour construire un modèle stochastique complet de la charge de supervision.

Nous notons que le choix d'une alternative ou d'une autre pour caractériser la charge dépend fortement de la technique d'évaluation utilisée et de métriques de performance choisie. Pour mesurer par exemple le passage à l'échelle d'un superviseur, une charge basée sur un programme synthétique générant un ensemble de requêtes de supervision et interrogeant un ensemble d'agents est suffisante.

Une autre considération dont il faut tenir compte lors de la spécification de la charge est sa représentativité du système réel de supervision. Cette représentativité se caractérise par le respect de la charge de trois critères suivants :

- Le taux d'arrivée des requêtes de supervision : ce taux doit être identique ou proportionnel à celui d'une application réelle.
- Les demandes des objets de gestion doivent, aussi, être proportionnelles ou identiques à celles d'une application réelle.
- Le profil d'utilisation des objets de gestion : ce profil est relatif au séquencement et au nombre des objets de gestion utilisés dans une application de supervision réelle.

Le niveau exercé par la charge sur le système sous test représente, aussi, un paramètre de sélection. Ce niveau prend des valeurs qui correspondent soit à une utilisation typique du système, une surcharge du système ou un système non chargé. Par exemple, dans le cas d'une

évaluation de performance ayant pour objectif de tester la résistance à la charge d'un agent de supervision [29], il faut surcharger l'agent et les taux d'arrivée de requêtes prennent des valeurs élevées différentes d'une utilisation typique d'une application réelle de supervision.

Dans ce paragraphe, nous avons donné un ensemble de recommandations à prendre en compte lors de la caractérisation d'une charge d'un système de supervision pour évaluer ses performances. Dans le paragraphe suivant nous allons définir les caractéristiques clés de la charge d'un système supervision. Ces caractéristiques seront utiles pour développer un modèle de charge d'une architecture spécifique de supervision.

3.4.2 Charge de la supervision

La caractérisation d'une charge de supervision consiste à observer ses caractéristiques clés dans un environnement d'utilisation réel. Ces caractéristiques serviront pour développer un modèle de cette charge qui puisse être utilisé d'une façon répétitive lors du processus d'évaluation. Les caractéristiques de cette charge se présentent sous la forme des services sollicités par les fonctions ou les algorithmes de gestion sur le système sous test. Ces fonctions et algorithmes [4] représentent les composants de la charge et ils se positionnent au niveau de l'interface du système sous test (voir figure 3.1).

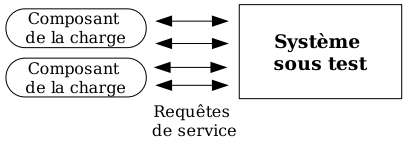

FIG. 3.1 – Les composants de la charge et leurs interfaces avec le système sous test

L'ensemble des caractéristiques clés de la charge présente ces paramètres. Des exemples de paramètres d'une charge de supervision sont : les types de requêtes de supervision, la taille de ces requêtes, le nombre d'objets de gestion dans une requête et les types de données de ces objets. Seuls, les paramètres qui ont un impact significatif sur les performances du système sous test doivent être inclus dans l'ensemble des paramètres de la charge. En revanche, on pourra négliger les paramètres qui ont un impact minimal sur les performances du système. Par exemple, si la taille des requêtes de gestion a un impact insignifiant sur le temps de réponse d'un agent, il doit être omis de la liste de paramètres de la charge. Pour déterminer les valeurs de ces paramètres, plusieurs techniques existent dans la littérature[22]. À titre d'exemple, nous pouvons utiliser la moyenne pour caractériser un paramètre de la charge, sur un nombre n de ses valeurs observées.

[4] Dans la littérature de l'évaluation de performance, ces fonctions et algorithmes de gestion représentent les utilisateurs du système sous test.

Paramètres d'une charge de supervision

La supervision de réseau (éventuellement d'applications ou des composants logiciels) se base, généralement, sur deux services qui sont la scrutation et la notification. Ce deux services peuvent être, éventuellement, complémentaires où une requête de notification enclenche une session de scrutation. Le service de scrutation consiste à l'interrogation régulière ou ponctuelle d'un ou plusieurs agents par un superviseur. Les requêtes d'interrogation portent sur un ou un ensemble d'objets de gestion et font appel à des opérations de gestion. Nous distinguons deux types d'opérations de gestion qui sont : *Get* pour récupérer un objet de gestion ou *Set* pour positionner sa valeur. Le service de notification[5] se base sur l'envoi , par un ou plusieurs agents vers un superviseur, des requêtes notifiant ce dernier de l'occurrence d'un évènement (changement du valeur d'un objet, dépassement de seuil, déclenchement d'une action de gestion). Certains paramètres de la charge de supervision sont communs à ces deux services, d'autres sont spécifiques à chacun d'eux. Les paramètres suivants sont identifiés, comme paramètres d'une charge de supervision :

1. La taille moyenne des requêtes de supervision.
2. Le type de requêtes les plus fréquemment utilisées sur les plates-formes de supervision (Get ou Set).
3. Le type d'objet de gestion le plus fréquent dans ces requêtes.
4. Temps d'exécution d'une requête sur un agent de supervision :
5. La fréquence de chaque type de requête.
6. La probabilité de réalisation d'un évènement de notification sur un agent portant sur une variable de supervision.
7. La fréquence de scrutation.

Cette liste de paramètres de la charge est donnée à titre indicatif et elle peut être complété par d'autres en cas de besoin. L'étude de Schönwälder [49] portant sur la caractérisation des MIBs SNMP pourra servir de base pour déterminer les valeurs de certains de ces paramètres, essentiellement les types de requêtes et les objets de gestion les plus fréquemment utilisés.

Dans le paragraphe suivant, nous allons proposer un modèle du trafic de supervision entre un ensemble d'agents de supervision et un manager. Ce modèle est basé sur le phénomène d'auto-similarité (*self-similarity*) détaillé dans [41].

3.4.3 Modèles de trafics réseau de la supervision

Un facteur important de la charge d'évaluation de performance des architectures de gestion est la modélisation du trafic réseau entre les différentes entités de l'architecture (le superviseur et les agents de supervision). Dans certaines études d'évaluation de performance [42, 7] portant sur ces architectures, le modèle poissonnien est le plus utilisé. Il est généralement le plus utilisé dans l'évaluation de performance de systèmes informatiques. Le

[5]Le terme notification est spécifique au standard JMX. Dans le standard SNMP, les notifications sont appelées *trap*

modèle poissonnien s'est imposé comme modèle le plus répandu grâce a sa simplicité et à son utilisation par la majorité des outils de simulation (NS2, OPNET). En revanche, des études récentes ont montrées la limitation de ce modèle pour caractériser certains trafics réseaux (SNMP, WWW, FTP) [45, 27]. Des nouveaux modèles sont apparus [32, 56, 8] pour modéliser les trafic réseaux à plusieurs niveaux (TCP, Ethernet,WWW) et qui diffèrent des modèle classiques Markovien et poissonnien. Ces modèles sont basés sur le phénomène d'auto-similarité (*Self-similar traffic*) et la fractalisation des trafics réseaux. Ils visent essentiellement à capturer leurs *burstiness* [6] à l'échelle de temps. Le couplage de ces modèles d'auto-similarité avec un modèle de files d'attentes de sources de type ON/OFF [51] [7] permet d'avoir une caractérisation réel d'un trafic réseau [21]. Nous proposons d'utiliser ce dernier modèle (auto-similarité et le modèle ON/OFF) pour caractériser le trafic de la supervision de réseau. Les agents de supervision seront modélisés par des sources de type ON/OFF. Chaque agent envoi des attributs de gestion pendant les périodes ON et il est silencieux pendant les périodes OFF. Les périodes OFF correspondent aux intervalles de temps entre les transmissions des réponses au superviseur, la réception d'une nouvelle requête ou le temps restant avant le début d'un nouveau cycle de polling (voir figure 3.2). Plusieurs distributions sont utilisées pour caractériser de façon analytique les périodes ON/OFF. Par exemple, la distribution géométrique des périodes ON/OFF pour un trafic VBR (traffic vidéo) [1], la distribution Pareto pour un trafic web [8]. Nous pensons que les modèles (self-similar et les

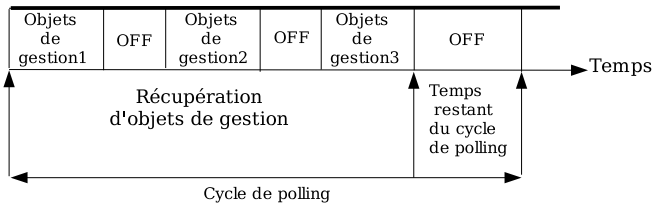

FIG. 3.2 – Un modèle ON/OFF de trafic de supervision observé sur un agent.

processus ON/OFF) que nous avons présenté précédemment, s'impose comme des modèles candidats pour modéliser le trafic de la supervision. La validation de ces modèles sur des architectures de gestion doit être effectuée pour vérifier leur validité pour un tel trafic.

3.5 Impact de la supervision sur un système supervisé

Dans les sections suivantes nous avons mis en évidence un ensemble des recommandations à prendre en compte lors de l'évaluation de performance d'un système de supervision de réseau afin de quantifier son coût. Mais n'oublions pas que l'un des buts (à part la quan-

[6] Dans l'ingénierie de trafic, le terme burstiness signifie la variabilité du trafic sur l'échelle de temps.

[7] Le modèle ON/OFF se base sur des périodes de ON durant lesquels il y a transfert des données sur des intervalles régulier et les périodes OFF durant lesquels ne se passe aucun transfert des données.

tification du coût de la supervision) de l'évaluation de ces systèmes de supervision est de mesurer l'impact qu'ils introduisent sur les systèmes supervisés. Dans cette section, nous allons proposer une fonction analytique qui caractérise l'impact de la supervision sur le système supervisé.

S'inspirant du travail de Parekh et al [40] portant sur l'impact des outils d'administrations[8] sur un système, nous avons défini une fonction analytique pour caractériser l'impact de la supervision sur un système supervisé sous un facteur d'impact k. Cette fonction se base sur la productivité [25] du système supervisé et sa maintient par rapport à une valeur de référence sous la forme d'un facteur d'impact relatif au système de supervision. Cette fonction est définie de la façon suivante et elle prend des valeurs comprises entre 0 et 1 :

$$Impact_{supervision}(k, k_1) = 1 - \frac{\textit{productivité}(\mathrm{k})}{baseline(k_1)} \tag{3.1}$$

Si la fonction $Impact_{supervision}$ est égale à 0, cela signifie que $k = k_1$ et le système est dans sa configuration de base. Si la fonction $Impact_{supervision}$ est égale à 1, la productivité du système supervisé est quasiment nulle et l'impact de la supervision est devenu considérable. La valeur de k_1 est fixée à une valeur de référence et la productivité du système supervisé est calculée pour cette valeur. La productivité, ainsi, calculée représentera la fonction *baseline* et k prendra des valeurs supérieures à k_1. Dans la suite, nous noterons la fonction *Impact* comme étant $Impact(k)$. Un moyen pour déterminer la valeur de *baseline* est de suspendre toute activité de gestion sur le système supervisé et de calculer sa valeur. Elle représentera la base du performance de système supervisé. La fonction *productivité* représente la productivité du système supervisé avec une activité de gestion. Cette activité de gestion est caractérisée par la variable k. Elle représente le facteur d'impact de la supervision sur le système supervisé. À titre d'exemple, un facteur d'impact peut être la variabilité du nombre de requêtes de supervision arrivant sur l'agent implanté dans une application supervisée[9] Les fonctions *productivité* et *baseline* sont basées sur la fonction puissance (*power function*) définie par Giessler et al [17]. Cette fonction a été utilisée dans les travaux de woodside et al [25] portant sur le passage à l'échelle des systèmes distribués, où ils ont défini la fonction de productivité d'un système distribué. Dans notre contexte, la fonction productivité du système supervisé, met en relation les trois métriques de performance suivantes : la production, l'utilisation et la réponse. Ainsi, la fonction productivité est égale à :

$$\text{productivité(k)} = \lambda(k).\frac{f(k)}{C(k)} \tag{3.2}$$

La variable λ quantifie la production du système supervisé avec une métrique bien appropriée, appartenant à la catégorie des métriques de production ; par exemple le nombre de

[8] Un exemple de ces outils : système de fichier, outil de backup d'une base de donnée, un garbage collector dans une machine virtuelle Java...

[9] Dans une architecture JMX, généralement, l'agent de supervision est intégré dans l'application supervisé et les objets de gestion font parties de son environnement d'exécution.

réponse du système par seconde. L'utilisation $C(k)$ est une fonction d'un ensemble des métriques d'utilisation pondérées par des coefficients selon leurs importances dans le système supervisé. Un exemple de cette fonction est le suivant : $C(k) = fonction(CPU, Mem, Network) = \alpha.CPU + \beta.Mem + \gamma.Network$, où CPU représente la consommation CPU du système supervisé, Mem représente sa consommation mémoire, $Network$ représente son utilisation du réseau et α, β et γ représente les coefficients de pondération. Les valeurs de la fonction productivité sont calculées en variant un seul facteur de performance du système de supervision qui représentera, ainsi, le facteur d'impact. Il faut mesurer ensuite les valeurs des trois métriques correspondantes (production, utilisation et réponse) sur le système supervisé. La figure 3.3 représente les différentes allures de la fonction $Impact(k)$ en fonction du facteur d'impact. Ces allures reflètent la variation des trois métriques (production, coût et réponse) du système supervisé en variant un facteur de productivité du système de supervision. Nous

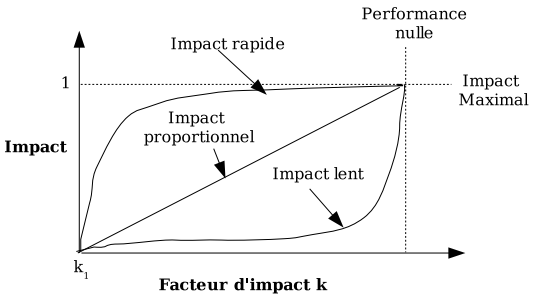

FIG. 3.3 – Les différentes allures de courbes de la fonction impact de la supervision sur le système supervisé en fonction d'un facteur d'impact.

distinguons trois allures des courbes de la fonction *Impact* : impact rapide des activités de supervision sur le système supervisé, impact proportionnel des activités de la supervision sur le système supervisé et un impact lent de ces activités sur le système supervisé. Nous envisageons d'étudier cette fonction dans des cas avec différents modèles de déploiements des agents de supervision dans un système supervisé [28], ainsi qu'avec différents scénarios d'activité de supervision : monitorage, contrôle et suivi (*tracking*) [39].

3.5.1 Modèles de déploiement des agents de supervision

Dans [28], Krerger et al ont présenté trois modèles de déploiement d'un agent de supervision dans un système supervisé (une application, un composant logiciel, un équipement réseau). Les modèles présentés sont les suivants :

L'agent comme démon

Dans ce modèle l'agent se présente comme étant un processus séparé dans le système supervisé. Le superviseur se connecte sur l'agent en spécifiant les informations de gestion

à collecter et les actions à prendre. L'avantage de ce type de modèle que le coût engendré par la communication entre le superviseur et l'agent est éliminé, puisque l'agent ne partage pas forcément les ressources du système supervisé. Un autre avantage de ce modèle est que l'agent est hors porté de l'application supervisé. Ceci implique qu'il puisse contrôler le cycle de vie de l'application en effectuant d'opérations du démarrage, d'arrêt, la supervision de sa disponibilité ou son recouvrement en cas de panne. Cependant, l'un de désavantage de ce modèle est qu'il nécessite la communication entre l'agent et l'application supervisé. Ceci engendre des problèmes de découverte de l'agent et l'implantation des adaptateurs adéquats au sein de l'application.

L'agent comme composant

Dans ce modèle l'agent est embarqué dans le système supervisé. Il se présente comme un composant du système qui expose les interfaces de gestion. Ce type de modèle fournit une granularité fine des données d'instrumentation de l'application. Un autre avantages de ce modèle est que l'application supervisée contrôle complètement l'intanciation de son agent de supervision. Si l'application ne possède pas un agent, elle pouvait instancier un sans avoir le souci de sa duplication.

L'agent comme noyau

Ce modèle se ressemble au modèle composant. En revanche, dans ce cas l'agent de supervision est le noyau de l'application et n'est pas seulement un composant. L'application supervisée est conçue et implantée autour de l'agent qui se charge de lancer les différents composants applicatifs. Le désavantage de ce modèle est qu'il nécessite une conception dès le début, de l'application supervisée autour de l'agent de supervision. Ceci est souvent compliqué, surtout pour des applications existantes. Un exemple typique de ce modèle est le serveur applicatif JBoss [24] qui est implanté autour du support JMX.

3.5.2 Scénarios des activités de supervision

Généralement, deux scénarios d'activités de supervision sont envisagés par certaines études lors de l'évaluation de performance des architectures de supervision [42, 7]. Ces scénarios sont le monitorage et la résolutions des problèmes qui se produisent sur les équipements réseaux. Le premier scénario (monitorage) consiste à collecter des informations de gestion depuis un équipement de façon régulière afin de s'assurer de son bon fonctionnement (*health check activity*) ou de détecter des éventuels problèmes. Le deuxième scénario est plutôt réactif et il consiste à collecter les informations pour identifier, isoler et rectifier les causes d'un problème apparu sur le réseau. À titre d'exemple, la perte de la connectivité dans un réseau, nécessite la collecte d'informations additionnelles pour localiser le problème d'une façon précise. D'autres scénarios d'activités de gestion plus générique ont été proposés par [39] qui raffinent ce deux scénarios présentés précédemment et recouvrent aussi, la supervision des composants logiciels (services et processus applicatif) en plus de la supervision des

équipements réseaux. En effet, les trois scénarios d'activités de gestion proposées sont les suivants :

- Monitorage : cette activité permet de capturer l'état actuel ou passé d'un équipement ou d'un composant logiciel particulier en utilisant les services de scrutation ou de la notification. Un exemple de cette activité est le monitorage de performance d'un routeur réseau.
- Suivi (tracking) : cette activité consiste à observer les aspects d'une seule unité de travail (*Unit of Work*) à travers plusieurs équipements ou composants logiciels. Par exemple, le suivi d'un message depuis son émetteur au récepteur sur un backbone des messages.
- Contrôle : cette activité permet d'altérer le comportement du composant géré au cours de son fonctionnement. Par exemple, changer le niveau de *logging* d'une application.

Le deux premier scénarios sont de type proactifs puisqu'ils visent à détecter certains problèmes sur les équipements et les services. Le troisième scénario est réactif et il vise généralement à corriger ces problèmes. Ces trois scénarios sont considérés comme les activités de gestion le plus courantes pour gérer des composants logiciels ou des équipements réseaux. Nous allons nous baser sur ces trois scénarios pour évaluer la fonction *Impact* pour chacun d'eux.

3.5.3 Exemple d'utilisation de la fonction Impact

Dans le cas d'utilisation de la fonction Impact que nous allons examiner, nous supposons que la valeur de base de facteur d'impact k est égale à 1. La fonction réponse $f(k)$ représente une valeur moyenne de la qualité de chaque réponse en fonction du facteur d'impact k. Dans la suite, cette fonction réponse est définie de la façon suivante [25] :

$$f(k) = \frac{1}{(1 + (\frac{T(k)}{\bar{T}}))} \tag{3.3}$$

où T(k) est le temps de réponse moyen du système supervisé sous le facteur d'impact k, comparer à son temps de réponse de base $\bar{T}$[10]. En effet, la fonction Impact aura comme valeur :

$$Impact(k, k_1) = 1 - \frac{\lambda.C_1.(T_1 + \bar{T})}{\lambda_1.C.(T + \bar{T})} \tag{3.4}$$

étude d'un cas : impact de la supervision avec un coût et une production croissants du système supervisé

Dans ce cas nous considérons que l'agent de supervision et le système supervisé partagent les mêmes ressources (CPU, Mémoire et réseaux). Cela signifie que le système supervisé

[10]La valeur de $\bar{T}$ représentera le temps de réponse moyen du système supervisé sans activité de supervision.

implante les fonctionnalités de gestion en plus de ces fonctionnalités opérationnelles[11] (voir figure 3.4).

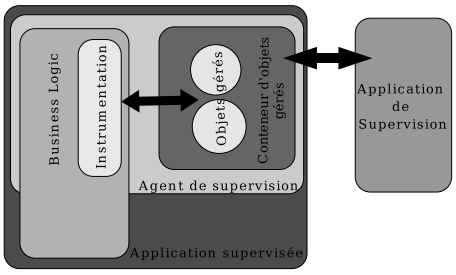

FIG. 3.4 – Le système supervisé et le système de supervision partagent les mêmes ressources.

Dans sa configuration de base, la productivité du système supervisé est définie par λ_1 qui représente sa production en terme du nombre de réponses par seconde, C_1 représente sa consommation des ressources et T_1 son temps de réponse. Nous obtenons, ainsi, comme valeur de la fonction $baseline = \frac{\lambda_1}{C_1.(1+\frac{T_1}{\bar{T}})}$. Dans une configuration du système supervisé, sous un facteur d'impact k, nous supposons que l'utilisation (ou coût) est proportionnelle à k, ainsi, $C = k.C_1$. La production augmentante de k, puisque le système répond aussi aux requêtes de la supervision, ainsi, $\lambda = k.\lambda 1$. Le temps de réponse augmente aussi et il est de $T = k.T_1$. En remplaçant ces valeurs dans l'équation 3.4, Nous obtenons comme fonction $Impact$:

$$Impact(k) = 1 - \frac{T_1 + \bar{T}}{(k.T_1 + \bar{T})} \tag{3.5}$$

Si $T_1 \simeq \bar{T}$, il s'agit alors d'une configuration de base du système supervisé sans activité

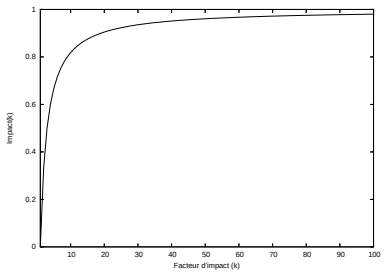

FIG. 3.5 – Impact rapide de la supervision sur un système supervisé.

[11] Autrement, cela signifie que les objets fonctionnels (business logic) du système supervisé implante aussi les interfaces de gestion.

de gestion par exemple (nous supposons que $\bar{T}$ est le temps de réponse du système sans supervision). Dans ce cas, la valeur de la fonction Impact est de :

$$Impact(k) = 1 - \frac{2}{1+k} \tag{3.6}$$

La figure 3.5 montre qu'on se trouve dans une situation d'un impact rapide. Cela est logique, étant donné que le facteur d'impact augmente l'utilisation des ressources sur le système supervisé d'une façon proportionnelle à l'utilisation de base tout en diminuant les réponses du système et leur qualité (augmentation du temps de réponse). Nous trouvons les mêmes résultats pour le même système mais avec une production descendante ($\lambda = \lambda 1/k$).

Une étude détaillée de la fonction impact de la supervision sur un système supervisé, avec une utilisation sur un cas réel a fait l'objet d'un papier que nous avons soumis au workshop DSOM (*IFIP/IEEE International Workshop on Distributed Systems : Operations and Management*) de l'année 2005.

Chapitre 4

Conclusion

L'évaluation de performance et la maîtrise du coût des architectures de gestion et de la supervision de réseaux est un nouveau défi dans ce domaine. Pour mener un projet d'évaluation de performance d'un système, et en particulier une architecture de gestion de réseau ou de service, il faut faire le choix sur un ensemble d'éléments de base. Ces éléments incluent, essentiellement les techniques d'évaluation de performances, les métriques à mesurer pour quantifier ces performances, et la charge de test. Concernant le choix de la technique d'évaluation, une seule technique n'est pas suffisante pour évaluer les performances d'un système. L'utilisation de deux ou même des trois techniques est plus avantageuse pour mieux analyser les performances d'un système. Une majorité des études d'évaluation de performance que nous avons examinées se sont basées sur la technique de mesure. Cette technique souvent considérée comme coûteuse en terme de temps de processus d'évaluation et en terme d'infrastructure, est insuffisante pour modéliser les performances d'un système. Elle nécessite d'être complétée par une autre étude (analytique ou simulation).

Un autre choix important pour l'évaluation est celui des métriques de performance. Ces métriques représentent les indicateurs de performance d'une architecture de gestion. Aussi important que les deux autres choix, la charge caractérise et représente les services d'une infrastructure de gestion dans ses différents cas réels. Dans l'évaluation de performance d'une architecture de gestion, nous avons identifié deux types de charge : la charge de la partie fonctionnelle du service utilisateur et la charge de la partie non fonctionnelle ou la charge de la gestion. Si plusieurs travaux ont caractérisé le premier type de charge (trafic HTTP, trafic des serveurs applicatifs,...) en proposant des modèles de leurs trafics, nous n'avons pas trouvé de modèles spécifiques aux charges des infrastructures de gestion. Ceci se présente aussi comme un verrou à soulever pour toute évaluation de performance des architectures de gestion basées sur les techniques de modélisation (analytique et simulation).

Nous avons également, constaté une hétérogénéité des études d'évaluation de performance des architectures de gestion. En effet, Il n'existe pas un ensemble de choix d'éléments de base pour ces études, mais une grande variété des paramètres, des facteurs et des métriques de performances. Nous sommes persuadés qu'il est nécessaire d'avoir une méthodolo-

gie standard spécifiant cet ensemble de choix de base pour toute étude future de performance portant sur les architectures de gestion. Dans ce rapport, nous avons défini une méthodologie d'évaluation de performance des architectures de gestion en spécifiant un ensemble de recommandations pour ces choix de base.

Nous envisageons maintenant d'appliquer cette méthodologie sur des architectures de gestion. Notre première cible est l'ensemble des implantations d'architectures de gestion basée sur JMX (Java Management eXtension) [53], devenue un standard de gestion des applications et des services basées sur la technologie Java.

Bibliographie

[1] Ian F. Akyildiz, Jorg Liebeherr, and Ioanis Nikolaidis. Multi-level rate-based flow control for ABR traffic. *Performance Evaluation*, 31(1-2) :107–131, 1997.

[2] Mario Baldi and Gian Pietro Picco. Evaluating the tradeoffs of mobile code design paradigms in network management applications. In *Proceedings of the 20th international conference on Software engineering*, pages 146–155. IEEE Computer Society, 1998.

[3] Alan Bivens, Rashim Gupta, Ingo McLean, Boleslaw Szymanski, and Jerome White. Scalability and performance of an agent-based network management middleware. *Int. J. Netw. Manag.*, 14(2) :131–146, 2004.

[4] C. Bohoris, A. Liotta, and G. Pavlou. Software agent constrained mobility for network performance monitoring. In *6th IFIP Conference on Intelligence in Networks :SmartNet*, 2000.

[5] Gunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor S. Trivedi. *Queueing Networks and Markov Chains - Modeling and Performance Evaluation with Computer Science Applications*. Jhon Wiley & Sons, New York, 1998.

[6] Anne-Louise Burness, Richard Titmuss, Caroline Lebre, Katie Brown, and Alan Brookland. Scalability evaluation of a distributed agent system. *Distributed Systems Engineering*, 6(4) :129–134, 1999.

[7] Thomas M. Chen and Stephen S. liu. A model and evaluation of distributed network management approaches. *IEEE Journal on Selected Areas in Communications*, 20(4), May 2002.

[8] Mark Crovella and Azer Bestavros. Self-Similarity in World Wide Web Traffic : Evidence and Possible Causes. In *Proceedings of SIGMETRICS'96 : The ACM International Conference on Measurement and Modeling of Computer Systems.*, Philadelphia, Pennsylvania, May 1996. Also, in Performance evaluation review, May 1996, 24(1) :160-169.

[9] Thomas Drevers. Performance of web services based network monitoring. Technical report, University of Twente, Enschede, The Netherlands, January 2004.

[10] Markus Strasser et Markus Schwehm. A performance model for mobile agent systems. In *Int. Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'97)*, volume 2, pages 1132–1140, 1997.

[11] K. Fall and K. Varadhan. Ns notes and documentation. Technical report, The VINT Project, January 1999.

[12] Olivier Festor and Laurent Andrey. *Chapitre 6 : JMX : un standard pour la gestion Java.* Hermes Science Publications, 2003. ISBN : 2-7462-0443-6.

[13] R. Franks and C.M. Woodside. Performance of multi-level client-server systems with parallel service operations. In *WOSP98*, pages 120–130, 1998.

[14] Svend Frolund and Pankaj Garg. Design-time simulation of a large-scale, distributed object system. *ACM Trans. Model. Comput. Simul.*, 8(4) :374–400, 1998.

[15] Alfonso Fuggetta, Gian Pietro Picco, and Giovanni Vigna. Understanding Code Mobility. *IEEE Transactions on Software Engineering*, 24(5) :342–361, 1998.

[16] D. Gavalas, D. Greenwood, M. Ghanbari, and M. O'Mahony. Advanced network monitoring applications based on mobile/intelligent agent technology. *Computer Communications, Elsevier Science*, 23(8) :pp. 720–730, April 2000. ISSN : 0140-3664.

[17] A. Giessler, J. Haenle, A. Koenig, and E. Pade. Free buffer allocation — an investigation by simulation. *Computer Networks*, 2(3) :191–208, July 1978. ISSN : 0376-5075.

[18] The Open Group. Application Response Measurement. http ://www.opengroup.org/tech/management/arm/.

[19] Qiang Gu and Alan Marshall. Network management performance analysis and scalability tests : Snmp vs. corba. In *IEEE/IFIP Network Operations & Management Symposium, Seoul, Korea*, April 2004.

[20] IETF. http ://www.ietf.org/html.charters/ippm-charter.html, April 1995.

[21] R. Jain and S. Routhier. Packet trains-measurements and a new model for computer network traffic. *IEEE Journal of Selected Areas In Communications*, SAC-4(6) :986–995, September 1986.

[22] Raj Jain. *The art of Computer Systems Performance Analysis.* John Wiley & Sons, Inc, 1991. ISBN : 0-471-50336-3.

[23] P Jardin. Supporting scalability and flexibility in a distributed management platform. *Distributed Systems Engineering*, 3(2) :115–123, 1996.

[24] JBoss. The professional open source company. http ://www.jboss.org, 1999.

[25] Prasad Jogalekar and Murray Woodside. Evaluating the scalability of distributed systems. *IEEE Trans. Parallel Distrib. Syst.*, 11(6) :589–603, 2000.

[26] L. Kleinrock. *Queueing Systems — Theory*, volume 1. Wiley-Interscience, New York, New York, 1975.

[27] Sunita Kode, Mukta Nandwani, Jiten Maheswary, and Shilpa Sureh. Traffic characterization for heterogeneous applications. Technical report, The Bradley Dept. of Electrical & Computer Engineering Virginia Tech, 2001.

[28] H. Kreger, W. Harold, and Leigh Willamson. *Java and JMX : Building Manageable Systems.* January 2003.

[29] Abdelkader Lahmadi, Laurent Andrey, and Olivier Festor. Performances et résistance au facteur déchelle d'un agent de supervision basé sur JMX : Méthodologie et premiers résultats. In *Colloque GRES 2005 : Gestion de REseaux et de Services, Luchon, France*, volume 6, pages 269–282, Mar 2005. ISBN : 2-9520326-5-3.

[30] Kevin Lai and Mary Baker. Measuring bandwidth. In *IEEE INFOCOM*, volume 1, pages 235–245, New York, USA, March 1999.

[31] L. C. Lee, H. S. Nwana, D. T. Ndumu, and P. De Wilde. The stability, scalability and performance of multi-agent systems. *BT Technology Journal*, 16(3) :94–103, 1998.

[32] Will E. Leland, Murad S. Taqq, Walter Willinger, and Daniel V. Wilson. On the self-similar nature of Ethernet traffic. In *ACM SIGCOMM*, pages 183–193, San Francisco, California, 1993.

[33] Koon-Seng Lim and Rolf Stadler. A navigation pattern for scalable internet management. In *IFIP/IEEE International Symposium on Integrated Network Management,Seatle, Washington*, pages 405–420, May 2001.

[34] Bruce Lindsay. A conversation with bruce lindsay. *ACM Queue*, 2(8), November 2004.

[35] Antonio Liotta, Graham Knight, and George Pavlou. On the performance and scalability of decentralized monitoring using mobile agents. In *DSOM*, pages 3–18, 1999.

[36] G.F Lucio, M. Paredes-Farrera, E. Jammeh, M. Fleury, and M.J. Reed. Opnet modeler and ns-2 : comparing the accuracy of network simulators for packet-level analysis using a network testbed. *WSEAS Transactions on Computers*, 2(3), July 2003.

[37] Gottfried W. R. Luderer, H. Ku, B. Subbiah, and A. Narayanan. Network management agents supported by a java environment. In *Proceeding of the V IFIP/IEEE International Symposium on Integrated Network Management*, volume 86, pages 790–790, May 1997. ISBN : 0-412-80960-5.

[38] SUN microsystems. The netbeans profiler project. http ://profiler.netbeans.org/.

[39] Justin Murray. Using jmx design patterns to enhance application manageability. Technical report, HP Invent Online, February 2005.

[40] Sujay Parekh, Kevin Rose, Joseph Hellerstein, and Sam Lightstone. Managing the performance impact of administrative utilities. In *IFIP/IEEE International Workshop on Distributed Systems : Operations and Management, (DSOM'03)*, volume 14, pages 132–142, October 2003.

[41] Kihong Park and Walter Willinger. *Self-Similar Network Traffic and Performance Evaluation.* John Wiley & Sons, Inc., 2000.

[42] Collin Pattinson. A study of the behaviour of the simple network management protocol. In *DSOM (International Workshop on Distributed Systems : Operations & Management)*, October 2001.

[43] G. Pavlou, P. Flegkas, S. Gouveris, and A. Liotta. On management technologies and the potential of web services. *IEE, Communications Magazine*, 42 :58–66, July 2004. ISSN : 0163-6804.

[44] K. Pawlikowski, H-D. Joshua Jeong, and J-S. Ruth Lee. On credibility of simulation studies of telecommunication networks. *IEEE Communications Magazine*, pages 132–139, January 2002.

[45] Vern Paxson and Sally Floyd. Wide area traffic : the failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3) :226–244, 1995.

[46] The GNU project. The GNU profiler, gprof. http ://www.gnu.org/software/binutils/manual/gprof-2.9.1/gprof.html.

[47] Jerome Rolia and Ken Sevcik. The method of layers. In *IEEE Transactions on Software Engineering*, volume 21, pages 689–700, August 1995.

[48] Marcelo G. Rubinstein, Otto Carlos Muniz Bandeira Duarte, and Guy Pujolle. Reducing the response time in network management by using multiple mobile agents. In *MMNS*, pages 253–265, 2000.

[49] Jürgen Schönwälder. Characterization of SNMP MIB Modules. In *Proceeding of the IX IFIP/IEEE International Symposium on Integrated Network Management*, May 2005.

[50] F. Sheikh, J. Rolia, P. Garg, S. Frolund, and A. Shephered. Layered performance modeling of a corba-based distributed application design. In *First World Congress on Computer Simulation*, pages 247–254, September 1997.

[51] Khosrow Sohraby. On the theory of general on-off sources with applications in high-speed networks. In *INFOCOM*, pages 401–410, 1993.

[52] Rajesh Subramanyan, José Miguel-Alonso, and José A. B. Fortes. A scalable snmp-based distibuted monitoring system for heterogeneous network computing. In *Supercomputing '00 : Proceedings of the 2000 ACM/IEEE conference on Supercomputing*, page 14. IEEE Computer Society, 2000.

[53] SUN. Java^{TM} management extensions, instrumentation and agent specification, v1.2. http ://jcp.org/en/jsr/detail ?id=3, october 2002. Maintenance Release 2.

[54] OPNET Technologies. The OPNET modeler. http ://www.opnet.com/products/modeler.

[55] Pere vilà, José L.Marzo, Antonio Bueno, Eusebi Calle, and Lluis Fàbrega. Distributed network resource management using a multi-agent system : scalability evaluation. In *SPECTS (International Symposium on Performance Evaluation of Computer and Telecommunication Systems)*, July 2004.

[56] Walter Willinger, Murad S. Taqqu, Robert Sherman, and Daniel V. Wilson. Self-similarity through high-variability : statistical analysis of Ethernet LAN traffic at the source level. *IEEE/ACM Transactions on Networking*, 5(1) :71–86, 1997.

[57] C.M. Woodside, J.E. Neilson, D.C. Petriu, and S. Majumdar. The stochastic rendezvous network model for performance of synchronous client-server-like distributed software. In *IEEE Transactions on Computer*, volume 44, pages 20–34, January 1995.

[58] M. Zapf, K. Herrmann, K. Geihs, and J. Wolfang. Decentralized SNMP management with mobile agents. *Proceedings of the VI IFIP/IEEE IM conference on network management*, page 623, 1999.

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399