Gestion intégrée et multi-échelles des systèmes répartis : Architecture et canevas intergiciel orientés composants

Mehdi Kessis

UNIVERSITÉ DE GRENOBLE

N° attribué par la bibliothèque

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE DE GRENOBLE

Spécialité : « Informatique : Systèmes et Logiciels »

préparée aux laboratoires LABORATOIRE INFORMATIQUE de GRENOBLE (LIG) – IMAG

et MAPS/AMS – France Télécom

dans le cadre de l'École Doctorale

« Mathématiques, Sciences et Technologies de l'Information, Informatique (MSTII) »

présentée et soutenue publiquement par

Mehdi KESSIS

Le 10.09.2007

Titre :

Gestion intégrée et multi-échelles des systèmes répartis : Architecture et canevas intergiciel orientés composants

Directeurs de thèse :
Mme. Claudia RONCANCIO et M. Alexandre LEFEBVRE

JURY

M. Jacques MOSSIÈRE,	Président
M. Eric GRESSIER-SOUDAN,	Rapporteur
M. Franck MORVAN,	Rapporteur
M. Daniel HAGIMONT,	Examinateur
M. Levent GÜRGEN,	Examinateur
M. Pascal DECHAMBOUX,	Examinateur
Mme. Claudia RONCANCIO,	Directeur de thèse
M. Alexandre LEFEBVRE,	Co-directeur de thèse

A ma mère, mon père et ma femme

Remerciement

Je tiens tout d'abord à remercier Jacques Mossière, directeur de l'École Nationale Supérieure d'Informatique et de Mathématiques Appliquées de Grenoble, de me faire l'honneur de présider ce jury.

Je remercie Eric Gressier-Soudan, professeur au Conservatoire National des Arts et Métiers, et Franck Morvan, maître de conférences à Université Paul Sabatier, Toulouse III d'avoir accepté d'être rapporteurs de cette thèse et d'avoir effectué une évaluation approfondie de mes travaux.

Je remercie également Daniel Hagimont, professeur à l'Institut National Polytechnique de Toulouse, Levent Gergen, chercheur au Commissariat à l'Énergie Atomique et Pascal Déchamboux, résponsable Système d'Information à France Télécom, d'avoir accepté d'être examinateurs de mes travaux.

Je remercie très chaleureusement Claudia Roncancio, ma directrice de thèse. Sa vision scientifique, et ses précieux conseils m'ont guidé tout au long de ces années de thèse. Elle a toujours été disponible et curieuse pour discuter de mes travaux, et elle m'a toujours encouragé et motivé.

Je ne saurais assez remercier Alexandre Lefebvre, responsable d'unité de recherche à France Télécom R&D et mon co-directeur de thèse, pour sa motivation, son ouverture et son encouragement. Il a créé pour moi un environnement de travail très confortable, avec un bon équilibre entre la recherche académique et industrielle. C'est une grande chance que j'ai eue de pouvoir travailler sous sa direction d'un chef aussi passioné.

Au delà du jury, je tiens à remercier tous ceux qui ont permis à ces travaux d'aboutir, par leurs conseils, leurs contributions et leurs encouragements.

- Merci à tous les membres de l'équipe MAPS/AMS à France Télécom pour leur accueil et pour avoir été toujours disponibles pour répondre à mes questions sur la plateforme ODIS. Je remercie particulièrement Yoann Bersihand, Olivier Beyler, Franck Milleville, Wei Monin, Kathleen Milsted pour leurs encouragements et leurs conseils.
- Merci à Pascal Déchamboux pour m'avoir aiguiller vers des pistes de travail intéressantes.
- Merci à tous les membres de l'équipe HADAS et SIGMA du laboratoire LIG-IMAG pour toutes les discussions intéressantes que nous avons eu durant la conception de DASIMA.
- Merci à tous les doctorants et post-docs de France Télécom Levent, Naga, Ahmed, Mourad, Pierre-Charle et Ada de l'intérêt qu'ils ont montré envers mon travail et pour leur sympathie et leurs services.
- Merci à Thomas Saillard, le travail qu'il a réalisé durant sont stage de fin d'études m'a permis de faire mes premiers tests de DASIMA avec des applications M2M.
- Merci à toutes les personnes sympathiques et intéressantes que j'ai eu l'occasion de côtoyer aux cours de ces dernières années (au sein de Bull, de France Télécom, de l'Inria, des laboratoires LIG et MAPS/AMS ou ailleurs) et qui voudront bien

m'excuser de ne pas les nommer.

Enfin, je garde une place toute particulière pour ma famille et mes proches, pour tout ce qu'ils ont pu m'apporter. Merci à mes soeurs Imène et Ines pour leur grand soutien et leur encouragement. Merci à ma femme Tiphaine pour avoir encouragé, soutenu et supporté un mari souvent très débordé. Merci pour sa patience et pour sa présence inestimable à mes côtés. Enfin, merci à mes parents pour m'avoir fourni toutes les bonnes conditions pour réussir mes études et pour m'avoir apporté, en plus de leur amour, le goût du travail et de la persévérance.

Résumé

Gestion intégrée et multi-échelles des systèmes répartis : Architecture et canevas intergiciel orientés composants

Les deux dernières décennies ont été marquées par un essor remarquable de l'informatique répartie (réseaux pair à pair, grilles de calcul, informatique pervasive, etc.) et des services de télécommunication. Désormais, les nouvelles technologies de l'information et de télécommunication font partie intégrante du quotidien des entreprises et des particuliers. Malgré ces avancées technologiques, les systèmes informatiques restent souvent vulnérables, menacés de pannes, de dysfonctionnements et d'utilisation anarchique ou intempestive des leurs ressources. De plus, les systèmes informatiques deviennent de plus en plus complexes, et entraînent une complexité de leur gestion. Cette vulnérabilité et complexité sont fortement amplifiées par l'extension géographique des réseaux et des systèmes informatiques, leur hétérogénéité et l'intégration toujours plus importante de l'informatique et des télécommunications.

Dans ce contexte, les outils d'administration sont devenus un instrument incontournable de planification, d'organisation et de gestion des systèmes informatiques dans leur ensemble. Ils sont même devenus aussi importants que les systèmes administrés eux-mêmes. Toutefois, malgré la multitude des acteurs concernés par la problématique d'administration (éditeurs, constructeurs, organismes de standardisation, etc.) ces systèmes présentent souvent des limitations sur le plan technologique et architectural. D'où le besoin de développer de nouvelles approches et d'explorer de nouvelles technologies mieux adaptées aux défis d'échelles et d'hétérogénéité des ressources administrées.

Cette thèse s'inscrit au cœur de cette problématique d'administration. Elle aborde la question d'administration intégrée dans des contextes multi-échelles.

Afin de répondre à cette question, nous nous appuyons sur une approche pluridisciplinaire : architecture logicielle à base de composants, middleware, systèmes et approches d'administration réseaux. Ainsi, nous proposons une approche d'administration multi-échelles basée sur le concept de Domaine d'administration, et orientée par l'architecture des systèmes administrés. Pour la validation de l'approche nous proposons le canevas Dasima, un canevas d'administration à base de composants logiciels dotée d'une double flexibilité (architecturale et comportementale) et d'une capacité à d'auto-introspection. Ces propriétés le distinguent des systèmes d'administration existants et favorisent une administration plus souple dans des contextes multi-échelles.

Dans le cadre de la validation de notre approche, le canevas Dasima a été mis en œuvre pour l'administration d'une application M2M d'Orange. Plusieurs expérimentations à différentes échelles, géographiques et numériques, ont été menées sur la plateforme expérimentale Grid5000 afin de valider la capacité de passage à l'échelle de l'approche proposée.

Mot-clés : administration, composants, architecture, multi-échelles, hétérogénéité

Abstract

Integrated Multi-scale of Networked Systems Management : Architecture and Components-Oriented Middleware

TINY devices, smart objects, home gateways, sensor networks, have become an integral part of our everyday life and of our socio-economic ecosystem. Managing efficiently such environments is just as important as the devices themselves. However, new management challenges such as management in different scales (multi-scale), heterogeneity management or extreme managed resources distribution make this task fiendishly complex.

Together, these challenges result in a new management complexity that breaks current paradigms based on ad-hoc or centralized and rigid monolithic management applications. Traditional management systems, that are mainly ad-hoc or centralized, reach their limits in such complex management contexts.

This PhD work aimes to bring some responses to overcome these difficulties by proposing a new management approch based combining advanced software engineering techniques, component based systems, middlewares, and network management systems concepts. Our proposal breaks with current management paradigms mainly related to quite rigid and monolithic systems.

To validate our approach, we have developed DASIMA framework (Domain-based Architecture for Scalable Integrated MAnagement middleware). DASIMA middleware framework, implements a new management approach combining domain-based and architecture-based management. It was implemented as a fully component-based management middleware intended to scale-up and down according to the number and the distribution of managed resources.

DASIMA has been experimented with a real industrial application in the Machine to Machine (M2M) context. Our experiments were done in reuced context (local network) and large scale deployement context (GRID 5000) and they confirmed the interest of developing adaptable middleware to manage networked systems in multi-scale contexts.

Keywords : software architecture, management, monitoring, distributed systems, component-based systems, scalability

Lorsque je tends vers un but
Je me fais porter par l'espoir
Oublie toute prudence
Je n'évite pas les chemins escarpés
Et n'appréhende pas la chute
Dans les flammes brûlantes.
Qui n'aime pas la montagne
Vivra éternellement au fond des vallées.
La volonté de vivre,
Abou El Kacem Echebbi (24 Février 1929 - 9 Octobre 1934)

Table des matières

1 Introduction générale **23**

1.1 Contexte et motivations 23

1.2 Vers une administration multi-échelles 24

1.2.1 Exemple des passerelles domestiques 24

1.2.2 Cas des plateformes M2M France Télécom 25

1.3 Evolution des besoins des systèmes d'administration 27

1.4 Les défis de l'administration dans un contexte multi-échelles 28

1.4.1 Le contexte multi-échelles 28

1.4.2 Une architecture logicielle adaptée 29

1.4.3 Hétérogénéité des ressources administrées 30

1.5 Contributions 31

1.6 Organisation du manuscrit 33

2 Administration des systèmes et réseaux **35**

2.1 Historique 36

2.2 Concepts de base de l'administration intégrée 37

2.2.1 Administration et administration intégrée 37

2.2.2 Le modèle gestionnaire-agent 38

2.2.3 Les fonctions d'administration 39

2.2.4 Les niveaux d'administration 40

2.3 Les standards d'administration 41

2.3.1 SNMP 41

2.3.2 JMX 44

2.3.3 CIM/WBEM 47

2.3.4 WSDM . . . 51
2.3.5 Conclusion sur les approches standardisées . . . 52
2.4 Les systèmes classiques d'administration . . . 53
2.4.1 IBM Tivoli . . . 53
2.4.2 HP-OpenView . . . 57
2.4.3 Nagios . . . 62
2.4.4 Conclusion : Réalité des plates-formes de gestion . . . 65
2.5 Travaux connexes de recherche . . . 66
2.6 Conclusion . . . 67

3 Les systèmes d'administration à base de composants 69
3.1 Les composants : définition et genèse . . . 70
3.2 Composants pour l'administration . . . 70
3.2.1 Administration à base de composants . . . 70
3.2.2 Administration fondée sur l'architecture des systèmes administrés . . . 71
3.2.3 Administration à base de composants et fondée sur l'architecture . . . 72
3.3 Administration à base de composants . . . 72
3.3.1 Processus de fabrication des systèmes de gestion à composants . . . 72
3.3.2 Liaison . . . 73
3.3.3 RMTool . . . 75
3.3.4 CBDM . . . 78
3.4 Administration fondée sur l'architecture . . . 80
3.4.1 Rainbow . . . 80
3.4.2 SmartFrog . . . 82
3.5 Administration à base de composants et orientée architecture . . . 85
3.5.1 JADE . . . 85
3.5.2 GridKit . . . 87
3.6 Composants et administration intégrée . . . 89
3.6.1 Discussion . . . 91
3.7 Discussion . . . 92
3.8 Conclusion . . . 93

4 Présentation générale des contributions **97**

4.1 Défis à relever . . . 97

4.2 Synthèse de l'état de l'art . . . 99

4.2.1 Les systèmes d'administration classiques . . . 99

4.2.2 Systèmes d'administration à base de composants . . . 100

4.3 Positionnement . . . 101

4.4 Contributions . . . 103

4.4.1 Une approche architecturale d'administration muti-échelles . . . 103

4.4.2 Un canevas intergiciel d'administration orienté composants . . . 103

4.4.3 Evaluation . . . 104

4.5 Organisation des chapitres de la contribution . . . 104

5 Approche architecturale d'administration multi-échelles **105**

5.1 Introduction . . . 106

5.2 Approche d'administration . . . 106

5.2.1 Administration fondée sur l'architecture . . . 106

5.2.2 L'élément administré . . . 107

5.2.3 Administration à base de domaines . . . 108

5.3 Combinaison de l'architecture et des domaines . . . 108

5.3.1 Processus d'administration fondée sur l'architecture . . . 109

5.3.2 Processus d'administration à base de domaines . . . 109

5.4 Administration fondée sur l'architecture des systèmes administrés . . . 111

5.4.1 Cycle de vie d'un élément administré . . . 111

5.4.2 Mise à jour des éléments administrés . . . 112

5.5 Administration basée sur les domaines . . . 112

5.5.1 Choix du critère de découpage des domaines . . . 113

5.5.2 Interactions et rôles des domaines . . . 114

5.5.3 Composition des domaines . . . 114

5.5.4 Stratégies d'administration à base de domaines . . . 116

5.5.5 Dimensionnement des domaines . . . 119

5.5.6 Approvisionnement des domaines . . . 120

5.5.7 Auto-administration des domaines 120
5.6 Conclusion 120

6 DASIMA : un canevas à base de composants pour la gestion intégrée multi-échelles 123

6.1 Architecture globale de DASIMA 124
6.2 Briques de base de l'architecture fonctionnelle 125
6.2.1 L'élément administré 125
6.2.2 Le domaine d'administration 126
6.3 Architecture fonctionnelle d'un domaine 126
6.3.1 Architecture interne d'un domaine 127
6.3.2 Conteneur des applications d'administration 128
6.3.3 Conteneur des éléments administrés 128
6.4 Les services communs d'administration 129
6.4.1 Le service de découverte des ressources 130
6.4.2 Le service de nommage des ressources 130
6.4.3 Service de gestion de bindings 131
6.4.4 Service de monitoring 131
6.4.5 Service de gestion des notifications 133
6.5 Le domaine des domaines 135
6.5.1 Fabrique des domaines 135
6.5.2 Service de paramétrage global 135
6.6 Synthèse 136

7 Implémentation et mise œuvre 137

7.1 Introduction 137
7.2 Implémentation du canevas Dasima 138
7.2.1 Choix du modèle à composants 138
7.2.2 Implémentation du canevas Dasima 140
7.3 Etude de cas : La plateforme Orange M2M 140
7.3.1 La plateforme Orange M2M 141
7.3.2 Modélisation des ressources à administrer 142

7.3.3 Instrumentation applicative avec JMX . . . 143
7.3.4 Instrumentation système avec CompositeProbe . . . 144
7.3.5 Instanciation de la plateforme M2M . . . 145
7.4 Evaluation quantitative . . . 147
7.4.1 Evalution à échelle réduite . . . 147
7.4.2 Evaluation à large échelle . . . 150
7.5 Evaluation qualitative . . . 156
7.5.1 Compatibilité avec les standards . . . 157
7.5.2 Pertinence de l'approche . . . 157
7.6 Synthèse . . . 158

8 Conclusion : bilan et perspectives 159
8.1 Rappel des objectifs . . . 159
8.1.1 Administration multi-échelles . . . 159
8.1.2 Hétérogénéité des ressources administrées . . . 160
8.1.3 L'administration intégrée . . . 160
8.2 Contributions . . . 160
8.2.1 Une approche d'administration intrégrée multi-échelles . . . 161
8.2.2 Un canevas d'administration à base de composants . . . 161
8.2.3 Implémentation du canevas DASIMA . . . 161
8.2.4 Etude de cas en contexte industriel . . . 162
8.2.5 Evaluation et recommandations dans un contexte large échelle . . . 162
8.3 Perspectives . . . 162

Table des figures

1.1 Evolution du nombre des abonnés France LiveBox 25

1.2 Situation du haut débit par DSL en France au 01/07/2006 26

1.3 Evolution des systèmes d'administration 30

2.1 Exemple d'administration intégrée (extrait de [MKP99]) 38

2.2 Le modèle architectural gestionnaire-agent 39

2.3 Les niveaux d'administration 40

2.4 Architecture d'administration SNMP 42

2.5 Structure hiérarchique d'une MIB 43

2.6 Architecture globale de JMX [JMX04] 45

2.7 Agent JMX hiérarchiques 47

2.8 L'architecture WBEM 48

2.9 Extrait du modèle noyau de CIM [CIM99] 49

2.10 MUWS : Architecture de gestion orientée services 52

2.11 Architecture d'IBM Tivoli [GBS+05] 54

2.12 IBM Tivoli : Architecture de gestion à large échelle 56

2.13 Architecture distribuée et services de HP NNM 58

2.14 HP OpenView : Strarégies d'administration centralisée 59

2.15 Startégie d'administration répartie 60

2.16 Startégie d'administration hiérarchique 61

2.17 Nagios : Architecture et déploiement distribué 63

3.1 Processus de construction des applications de gestion à composants (extrait de [KYM+00]) 73

3.2 Architecture globale du canevas Liaison 75

3.3 RMTool et RETOS : Gestion des réseaux de capteurs à base de composants 76
3.4 CBDM : composants et architecture de gestion hiérarchique/dynamique . . 79
3.5 Architecture interne de Rainbow [CHSS04] 81
3.6 cfJMX : Déploiement des infrastructures JMX avec SmartFrog 84
3.7 Jade : Modèle et architecture à composants 86
3.8 Le modèle à composants OpenCOM [BCU+04] 88
3.9 Architecture du canevas de gestion des ressources de Gridkit 90
3.10 Architecture de gestion intégrée à base de composants 91

4.1 Positionnement par rapport à l'état de l'art 102

5.1 Exemple d'élément administré (ME) : passerelle de service 107
5.2 Classification des domaines selon leur contenu 109
5.3 Processus d'administration à base de domaines 110
5.4 Cycle de vie d'un élément géré 112
5.5 Modèles d'interaction entre ressources administrées et éléments administrés 113
5.6 Rôles d'un domaine 115
5.7 Relations inter-domaines 115
5.8 Exemple de stratégie d'administration centralisée à base de domaines 116
5.9 Exemple de stratégie d'administration répartie 117
5.10 Exemples de stratégies d'administration hiérarchique 118
5.11 Exemples de topologies d'administration hybrides 119

6.1 Architecture globale pour l'administration multi-échelles 124
6.2 Modélisation d'un élément administré 125
6.3 Modélisation d'un domaine 126
6.4 DASIMA : Architecture fonctionelle d'un domaine 127
6.5 Processus d'activation des applications d'administration 128
6.6 Services communs d'administration 129
6.7 Services de découverte 130
6.8 Service de gestion de binding 131
6.9 Architecture fonctionnelle du service de monitoring 132

6.10 Les deux processus de supervision de DASIMA 133
6.11 Format du message 134
6.12 Service de gestion des notifications 134
6.13 Fabrique des domaines 135
6.14 Gestion des éléments administrés dupliqués 136

7.1 Le modèle à composants Fractal : exemple de composants imbriqués 138
7.2 Vue Globale du workflow applicatif M2M 141
7.3 Une modélisation abstraite des concepts M2M 142
7.4 Une modélisation des concepts M2M en terme de ME DASIMA 142
7.5 Intégration de l'instrumentation JMX avec DASIMA 143
7.6 Instrumentation JMX de la plateforme M2M 144
7.7 Administration avec DASIMA et CompositeProbe 145
7.8 Processus d'instanciation d'un réseau de nœds M2M 146
7.9 Visualisation d'un réseau de nœds M2M 146
7.10 Platefome d'expérimentation à échelle réduite 147
7.11 Evaluation du temps de traiment du service gestion des évènements 148
7.12 Mesure des temps en mode push et pull 149
7.13 Evaluation du temps de traitement du service de gestion de binding 150
7.14 Distribution globale de Grid5000 151
7.15 Architecture de déploiement de 1000 nœuds M2M sur GRID5000 152
7.16 Courbe de performance du domaine pour 1 à 1000 nœuds M2M 153
7.17 Performance du domaine pour 1 à 1000 nœuds M2M avec temps services DASIMA 154
7.18 Architecture de déploiement de 1.000.000 nœuds M2M sur GRID5000 . . . 155
7.19 Courbe des performances du domaine intermédiaire (1.000.000 nœuds) M2M 156

Liste des tableaux

2.1 Table services JMX 46
2.2 Table principaux éléments du modèle CIM 50
2.3 Table greffons Nagios 64

3.1 Table services Yasmin 74
3.2 Table Synthèse et comparaison des travaux d'administration orientée composants 95

4.1 Table synthèse positionnement 103

7.1 Table synthèse des informations concernant le code de DASIMA 140
7.2 Table de paramétrage des tests du service de gestion des évènements 149

Chapitre 1

Introduction générale

Contents

1.1 Contexte et motivations 23
1.2 Vers une administration multi-échelles 24
1.2.1 Exemple des passerelles domestiques 24
1.2.2 Cas des plateformes M2M France Télécom 25
1.3 Evolution des besoins des systèmes d'administration 27
1.4 Les défis de l'administration dans un contexte multi-échelles . 28
1.4.1 Le contexte multi-échelles 28
1.4.2 Une architecture logicielle adaptée 29
1.4.3 Hétérogénéité des ressources administrées 30
1.5 Contributions 31
1.6 Organisation du manuscrit 33

1.1 Contexte et motivations

Les deux dernières décennies ont été marquées par un essor remarquable de l'informatique répartie (e.g. réseaux pair à pair, grilles de calcul, informatique « pervasive ») et des services de télécommunication. Désormais, les nouvelles technologies de l'information et de télécommunication envahissent aussi bien les entreprises que les foyers et font partie intégrante de notre vie quotidienne.

Malgré l'avancée technologique, les systèmes informatiques sont vulnérables, menacés de pannes, de dysfonctionnements et d'utilisation anarchique ou intempestive des leurs ressources. De plus, les systèmes informatiques deviennent de plus en plus complexes, et leur gestion un véritable challenge. Cette vulnérabilité et complexité évoluent d'avantage avec leur taille et leurs niveaux d'hétérogénéité. Cette complexité ne cesse d'augmenter avec l'extension géographique des réseaux et des systèmes informatiques, et l'intégration toujours plus importante de l'informatique et des télécommunications.

Des exemples sont apportés par l'ampleur des parcs des équipements à gérer pour les entreprises et les opérateurs de télécommunication. Aujourd'hui, une entreprise gère, en moyenne, une dizaine voir des milliers d'équipements alors qu'un opérateur ou un fournisseur de services de télécommunication gère des milliers voir des millions d'équipements

[IEC07]. L'opérateur Orange par exemple, à la date de rédaction de ce manuscrit, gère un parc de plusieurs millions de passerelles domestiques de modèles hétérogènes et dispersées sur le territoire français. Cette complexité met l'opérateur face à des défis techniques et architecturaux de gestion de qualité de service dans un environnement à large échelle (géographique et numérique).

Le besoin d'appliquer une *gestion intégrée* complexifie l'activité de gestion. En effet, autrefois, la gestion pour l'entreprise se limitait à la gestion de son infrastructure systèmes et réseaux. Aujourd'hui l'activité d'administration a gagné du terrain, elle couvre aussi bien l'aspect infrastructure que l'aspect applicatif (au niveau du système d'information de l'entreprise). Les outils de gestion sont devenus un instrument incontournable de planification, d'organisation et de gestion de l'informatique et du système d'information dans son ensemble. Les informations liées à l'infrastructure (système et réseau) sont gérées au même titre que les informations liées aux services et aux applications de l'entreprise.

Par ailleurs, les différents acteurs qui alimentent le secteur de technologies de l'information et de la communication mènent aujourd'hui une course d'innovation et de commercialisation exacerbée. Victimes de leurs succès, ces acteurs ont été pris de court par l'évolution de la taille et l'hétérogénéité de leurs parcs et leurs services. Quant aux organismes de standardisation, dont le rôle est d'apporter un soutient à ces acteurs, ils arrivent difficilement à suivre le rythme des mutations technologiques à cause des longs délais de validation des standards.

L'ensemble des facteurs présentés ci-dessus a pris de court la conception technique des outils d'administration, souvent restée rudimentaire au plan des technologies et des architectures. D'où le besoin de développer de nouvelles approches et d'explorer de nouvelles technologies afin de pallier ces limitations.

Notre travail de recherche s'inscrit au cœur de cette problématique. Nous proposons dans une approche de gestion basée sur le concept de domaine de gestion, issu du monde la gestion des systèmes et réseaux et sur les composants logiciels, qui proviennent du monde du génie logiciel. Nous présentons dans la suite de ce chapitre un aperçu des questions de recherche auxquelles nous proposons des éléments de réponse.

1.2 Vers une administration multi-échelles

Cette section illustre à travers deux cas d'applications issues du monde des télécommunications les nouveaux besoins d'administration dans des contextes à échelles, numériques et géographiques, potentiellement variables.

1.2.1 Exemple des passerelles domestiques

Afin d'illustrer par des exemples concrets les problématiques sous-jacentes à la gestion multi-échelles, nous introduisons le cas des passerelles de service de France Télécom (LiveBox). Outre l'accès haut débit à Internet (fonction de modem routeur), ces passerelles permettent d'offrir plusieurs services tels que la téléphonie sur IP ou la télévision. Avec l'évolution croissante du nombre des passerelles de services à gérer, France Télécom se retrouve face à un parc de millions de ressources hétérogènes à gérer. La figure 1.1 illustre

cette évolution croissante de la taille du parc des LiveBox dont les caractéristiques sont :

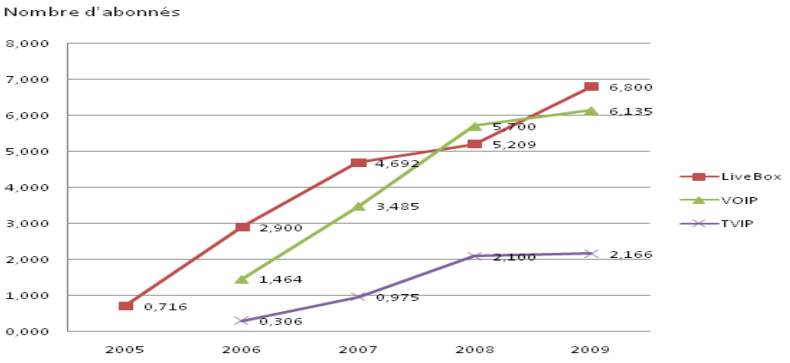


FIG. 1.1 – Evolution du nombre des abonnés France LiveBox

Trés forte volumétrie. En 2009, la taille du parc de LiveBox de France Télécom était de plus de 5 millions de passerelles de service dont 3,4 millions en France.

Très forte distribution géographique. A titre indicatif, la carte de la figure 1.2 présente la situation géographique des offres du haut débit au 30 juin 2006. Cette carte nous donne un aperçu sur l'étendu géographique du parc de ressources à gérer. Par ailleurs, les LiveBox sont déployées sur tout le territoire français mais également dans plusieurs pays d'Europe.

Forte hétérogénéité des ressources à administrer Il existe plusieurs variantes de ces passerelles (e.g. LiveBox, LiveBox boosted, LiveBox mini, Livebox professionnal). De plus, les passerelles sont liées à leur tour à d'autres équipements (e.g. PDA, téléphone mobile, réveil, lapin Nabastag) qui sont susceptibles d'être administrés par France Télécom. La figure 1.1 illustre cette hétérogénéité des services proposés au niveau de la passerelle avec plus de 4 millions d'abonnements téléphonie sur IP et plus d'un million d'abonnements Télévision sur IP à gérer.

Complexité orgnisationnelle Dans la figure 1.2, la couleur orange caractérise les zones où seul France Télécom a installé ses équipements haut débit et pour lesquelles les opérateurs alternatifs ont recours aux offres de bitstream. Les zones bleues indiquent les zones où des opérateurs alternatifs ont déployé leurs propres équipements d'accès haut débit en utilisant les seules offres de dégroupage de France Télécom. Cette gestion partagée des ressources rajoute une complexité organisationnelle étant donné que France Télécom et les opérateurs alternatifs doivent se partager la gestion des parc LiveBox à différents niveaux(e.g. infrastructures et services). Les opérateurs doivent également donner aux fournisseurs de services et opérateurs alternatifs un accès partiel ou total aux passerelles ou aux équipements accessibles via les passerelles.

1.2.2 Cas des plateformes M2M France Télécom

Avec la prolifération et la miniaturisation sans cesses croissantes des équipements informatique est apparu le besoin de faire communiquer ces machines entre elles. On parle alors de M2M (Machine-to-Machine) pour désigner les solutions permettant l'association des technologies de l'information et de la communication (TIC) avec des objets intelligents et

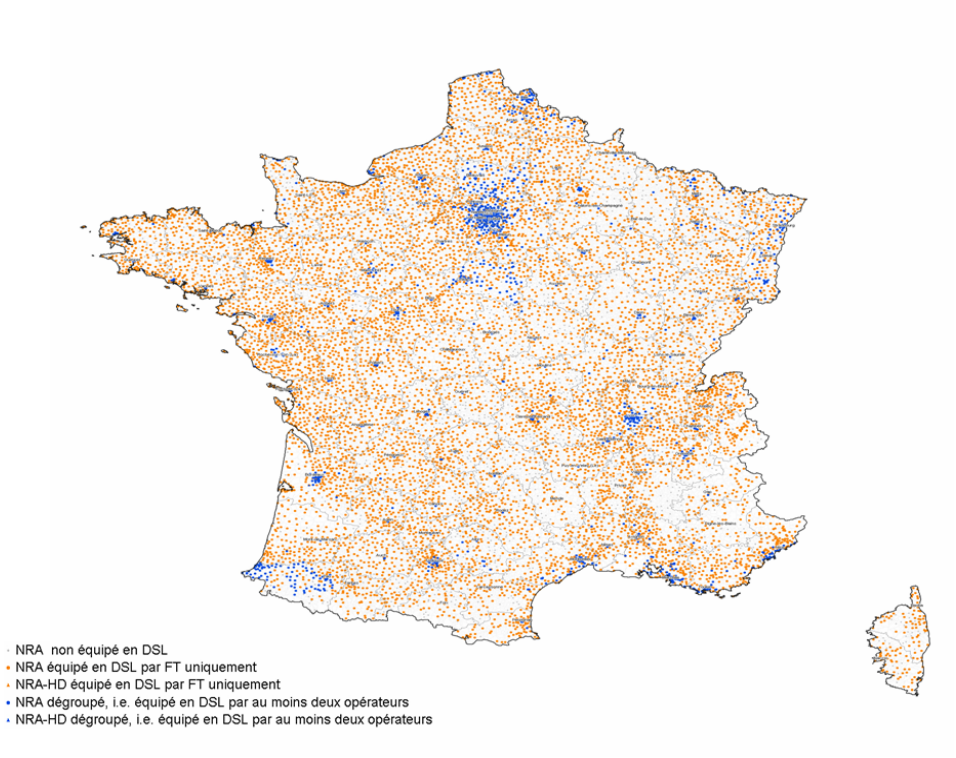


FIG. 1.2 – Situation du haut débit par DSL en France au 01/07/2006

communicants, dans le but de donner à ces derniers les moyens d'interagir sans intervention humaine avec le système d'information d'une organisation ou d'une entreprise [M2M06].

Les plateformes M2M automatisent des échanges entre machines et objets communicants de toute sorte tels que les ampoules, caméras de surveillance, capteurs RFID, capteurs de température, vêtements intelligents. L'offre M2M de France Télécom comprend par exemple une plateforme applicative (Orange M2M Connect) et diverses solutions applicatives telles que la gestion de flotte de véhicules et l'analyse des déplacements (Fleet Management) ou encore la connectivité des terminaux sur le réseau (Fleet Connect).

Ci-dessous quelques exemples concrets d'applications M2M proposées par France Télécom ou ses partenaires :

Gestion des patients aux CHU de Grenoble et Toulouse. En 2002, France Télécom R&D a lancé, en partenariat avec les CHU de Grenoble et de Toulouse, une expérimentation dénommée Gluconet, destinée à prendre en charge à distance des patients diabétiques. Les données de glycémie, lues quotidiennement par un équipement spécifique, sont transmises automatiquement via un téléphone mobile à un centre de gestion accessible aux médecins par Internet. Ces derniers renvoient un avis médical au patient par SMS ou par message vocal. Les consultations sont anticipées. Le patient et le médecin sont informés à temps de toute complication éventuelle.

Entretien des photocopieurs et des imprimantes d'entreprise Konica Minolta. En 2005, Konica Minolta a décidé de mettre en place une plateforme M2M pour la gestion de ses équipements (e.g. multifonctions, imprimante, télécopieur et photocopieurs). Le résultat escompté est disposer d'une meilleure qualité de service, assurer

une continuité de fonctionnement des équipements et d'un délai optimal d'intervention et de réapprovisionnement. En 2005, le système a été installé sur toutes les machines livrées et compatibles, au rythme de 500 par mois. Konica Minolta a décidé d'équiper un parc de plus de 50 000 systèmes d'impressions répartis sur tout le territoire français.

Gestion de la flotte de taxis des aéroports de Paris. L'aéroport de Paris a déployé un service permettant d'optimiser le flux des taxis sur les terminaux 1 et 3 de l'Aéroport de Paris-Charles de Gaulle. En moyenne, 30 % des taxis parisiens, sur un total de 15 200 véhicules, se trouvent à chaque instant sur la plateforme de l'Aéroport de Paris-Charles de Gaulle. Les taxis sont équipés de badges RFID de type télépéage, permettant la détection automatique des véhicules aux entrées et sorties des points de stationnement et des zones de prise en charge. Une application comptabilise le nombre de taxis dans les différentes zones et affiche les temps d'attente sur des panneaux électroniques destinés aux chauffeurs.

Depuis quelques années, France Télécom explore ce domaine de la communication inter-machines hétérogènes et se heurte à plusieurs problématiques, notamment au niveau l'administration de ces systèmes.

Nombre potentiellement élevé de ressources à administrer. Le nombre de ressources M2M à gérer peut varier de quelques centaines de patients au CHU de Grenoble à 50 000 photocopieuses et imprimantes. La généralisation du M2M dans le contexte domestique (gestion des compteurs, des appareils de chauffage, de la régulation de l'éclairage) implique des millions de ressources.

Distribution géographique importante. La distribution géographique des ressources M2M peut varier d'une échelle locale (e.g. cas des taxis parisiens) à nationale (e.g. cas des photocopieuses Konica.).

Très forte hétérogénéité des ressources administrées. Les ressources à gérer sont généralement plus hétérogènes par rapport aux parcs de ressources classiques. On peut gérer à la fois différents types de lecteurs et étiquettes RFID (e.g. cas des taxis de Paris), que des imprimantes et des photocopieuses (e.g. cas Konica), des téléphones mobiles (e.g. cas CHU de Grenoble) ou encore des équipements domestiques (e.g. compteurs, chaudières, volets)

1.3 Evolution des besoins des systèmes d'administration

Dans un contexte où l'évolution des besoins des entreprises est en perpétuel changement, et où les mutations technologiques se font à un rythme exponentiel, les systèmes d'administration actuels montrent leurs limites. Les nouveaux besoins que nous avons identifiés incluent les points suivants :

Besoins de plus de robustesse et de disponibilité. D'après une enquête [1], les utilisateurs des platesformes de supervision ont dénoncé le manque de flexibilité des plateformes de gestion, l'inadéquation entre leurs demandes et les capacités des frameworks, et la nécessité de reconstruire leur serveur de supervision à chaque correction d'erreurs.

[1] *Data Communication*,[www.data.com/issue/990921]

Des ressources à gérer de plus en plus hétérogènes. Avec la convergence des services de télécommunication, la multiplication des offre de plateformes M2M et les offres « Triple Play »(télévision, téléphone et internet), les opérateurs de télécommunication sont amenés à gérer d'avantage de ressources hétérogènes et à un volume très important. Les technologies et les approches techniques, bien qu'elles s'appuient souvent sur des standards au niveau protocolaire, sont quelques peu artisanales restent pour la plus part centralisées. Tel est le cas de la plateforme *CARMA* développée et gérée par Thales pour l'administration des LiveBox France Télécom.

Administration à différentes échelles. Un exemple concret de ce nouveau phénomène est la gestion des passerelles de services des opérateurs de télécommunication. En effet, l'administration de ces passerelles implique aujourd'hui un élargissement de champ d'intervention des opérateurs et des fournisseurs de services. Il s'agit d'une administration qui dépasse les frontières du réseau de l'opérateur pour s'étendre au réseau domestique. Chacun de ces deux réseaux requiert un système d'administration adapté à ses ressources. Un réseau d'opérateur nécessite des sondes dotées d'une forte capacité de collecte et de traitement de données, alors qu'un réseau domestique requiert des sondes basiques et une intrusivité moindre, vu que les ressources de calcul dont il dispose sont généralement assez limitées. C'est ce qu'on désignera dans le reste du document par *contexte d'administration multi-échelles*. L'idéal dans ce genre de contexte serait de disposer d'une plateforme d'administration capable de s'adapter en fonction de son contexte de déploiement constitué des ressources d'administration qui lui sont attribuées et des ressources administrées dont elle a la charge.

Besoin d'une gestion de plus en plus consolidée. La convergence des services, la diversité accrue des ressources administrées et la multiplicité des échelles d'administration ont fait naître un nouveau besoin d'administration chez les opérateurs et les fournisseurs de services : il s'agit de la gestion consolidée des ressources. On désignera ce besoin dans le reste du document par *l'administration intégrée*.

Besoin d'une automatisation des plateformes d'administration. Un autre besoin est la gestion automatisée ou « autonomique »[OM03]. Il s'agit d'injecter de l'intelligence dans les applications d'administration afin de les doter de plus d'autonomie. Des infrastructures d'administration qui s'auto-configurent, s'auto-optimisent ou s'auto-adaptent à l'échelle d'administration sont des exemples de comportements autonomiques possibles. Ce besoin est la conséquence des constations suivantes : coût élevé des plateformes d'administration, rareté des compétences d'administration sur le marché, besoin de réactivité dans des temps records, besoin de réduire le temps de reprise après pannes.

Besoin d'une architecture logicielle adaptée La plupart des systèmes d'administration classiques étaient, jusqu'à récemment, exclusivement basés sur des architectures prédéfinies et monolithiques. Cette caractéristique représente un obstacle majeur qui freine leur adaptabilité et leur reconfigurabilité dans un contexte multi-échelles.

1.4 Les défis de l'administration dans un contexte multi-échelles

1.4.1 Le contexte multi-échelles

L'ampleur des réseaux peut varier grandement. Que l'on parle d'un réseau d'un opérateur et fournisseur ou bien que l'on parle du réseau interne d'une petite entreprise, la

supervision doit pouvoir offrir des outils performants, adaptables aussi bien à la taille des réseaux qu'à leur grande diversité technologique. Nous définissons le concept d'administration dans un contexte multi-échelles de la manière suivante :

Définition 1.1. *Un contexte d'administration multi-échelles est un contexte dont l'échelle d'administration considérée peut subir des variations plus ou moins importantes au cours du temps.*

Les variations de l'échelle d'administration peuvent inclure, à titre d'exemples, le nombre de ressources à administrer, leurs emplacements géographiques, le nombre d'utilisateurs du système d'administration, la volumétrie des alertes et des échanges entre le système d'administration et les ressources à administrer.

Ces variations, plus ou moins rapides, du contexte d'administration ajoutent des difficultés supplémentaires pour les systèmes d'administration. Une nouvelle dimension (la variation) vient se rajouter à la liste des paramètres à prendre en compte lors de la mise en place d'un système d'administration. Non seulement les futurs systèmes d'administration doivent assurer leur rôle dans des contextes réduits et/ou fortement développés, mais en plus, ils doivent gérer automatiquement ces variations de contextes, ou, au moins, offrir les moyens de les prendre en considération.

1.4.2 Une architecture logicielle adaptée

L'architecture logicielle représente la moelle épinière d'un système d'administration. Or, la plupart des systèmes d'administration classiques [Ope, Tiv, Nag07] étaient, jusqu'à récemment, exclusivement basés sur des architectures prédéfinies et monolithiques (voir figure 1.3). Cette caractéristique est un obstacle majeur qui freine leur adaptabilité et leur reconfigurabilité dans un contexte multi-échelles.

Par ailleurs, aujourd'hui l'ingénierie logicielle propose des modèles à *composants* de plus en plus décentralisés, dynamiques et découplés. On parle alors de composants et de services pour distinguer l'entité logicielle elle même (le composant) des services qu'elle offre (le service). Ainsi, les systèmes se présentent comme un « puzzle »dont les pièces sont des unités sujettes à composition par une tierce personne. Les sytèmes conçus et implémentés à base de composants sont caractérisés par un couplage faible entre leurs différents modules logiciels.

Nous estimons que les modèles à composants offrent d'excellentes caractéristiques pour les applications d'administration.

Bien que l'origine de ces systèmes remonte aux années soixante [Mcl68], leur application dans le domaine de l'administration est récente. Depuis la fin des années 90, plusieurs travaux de recherches [Der97, JS00, KYM+00, RK05, GGL+03, JMX04, CHSS04, SBH05] ont consenti un effort considérable pour introduire les composants dans l'administration. Les multitudes d'approches appliquées (e.g. systèmes à composants pour la gestion, systèmes de gestion orientée composants, systèmes de gestion à composants pour la gestion orientée composants) et la diversité des domaines d'application concernées (e.g. réseaux de capteurs, grilles de calcul, systèmes et réseaux, cluster J2EE) montrent bien l'intérêt que représentent les composants pour les systèmes d'administration.

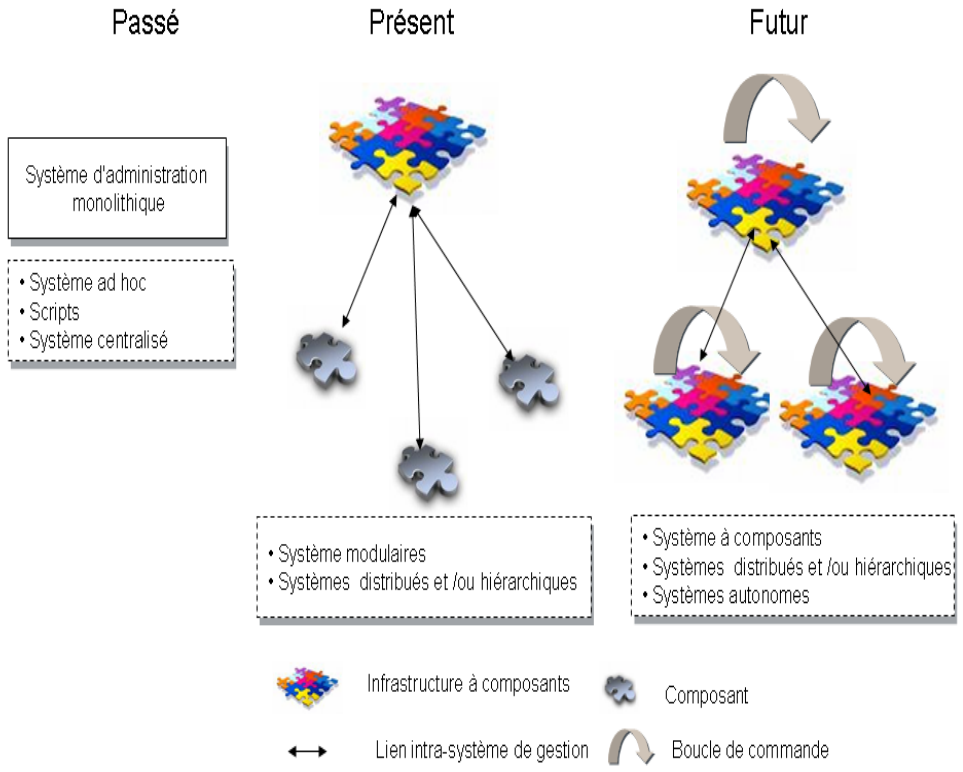


FIG. 1.3 – Evolution des systèmes d'administration

Nous explorons, dans ce manuscrit, la piste des architectures à composants afin d'obtenir un système d'administration souple et flexible offrant des possibilités d'adaptation à la fois comportementales et structurelles dans un contexte multi-échelles.

La complexité des systèmes d'administration requiert, également, que ceux-ci puissent également être administrés. On parle alors d'administration reflexive. Il importe de noter que certaines infrastructures à composants [BCL+06] offrent cette possibilité avec un faible coût d'intrusivité.

1.4.3 Hétérogénéité des ressources administrées

Avec la prolifération des réseaux informatique, la convergence des services de télécommunication et l'accroissement du nombre des logiciels et des services déployés sur les réseaux, s'est accréditée la notion d'hétérogénéité. Il s'agit de l'association dans un seul réseau de matériels et de logiciels provenant d'une multiplicité de fournisseurs.

Dans le cadre de ce travail, nous considérons comme ressources à administrer l'ensemble des équipements réseaux (e.g. routeurs, PC, PDA, Set-Up Box) et des ressources logicielles (e.g. tout ou partie de systèmes d'exploitation, middleware, applications, services). Ces ressources présentent hétérogénéité à différents niveaux (matériel et logiciel). Elles sont réparties et interconnectées par des réseaux. Cette diversité est exacerbée dans contexte multi échelles contenu du nombre et des types des ressources et considérées.

Aujourd'hui, pour prendre en compte l'hétérogénéité des ressources administrées, trois approches sont communément appliquées mais aucune d'entre elles ne répond à notre besoin :

Multiplication des systèmes d'administration. L'approche la plus simple, mais pas la moins onéreuse, consiste à utiliser des systèmes d'administration différents pour chaque type de ressource gérée. Tout d'abord, ce genre de solutions est loin d'être économique (e.g. coût des licences, coût de la formation des administrateurs). De plus, disposer de plusieurs systèmes d'administration implique une complexité supplémentaire de la maintenance des outils de gestion.

Gestion via des adaptateurs. La seconde approche consiste à définir des adaptateurs pour faciliter la prise en charge du système administré par le système d'administration. Cette approche a été adoptée par plusieurs systèmes d'administration, notamment [Nag07, Tiv, Ope]. L'avantage de cette approche est sa simplicité. Toutefois, les adaptateurs proposés sont généralement assez élémentaires et ad-hoc ou basés sur une technologie propriétaire, et par conséquent difficilement extensibles par les administrateurs.

Gestion via un modèle d'information générique. La troisième approche de gestion de l'hétérogénéité des ressources administrées consiste à appliquer des modèles d'information génériques, tels que le modèle CIM (voir section 2.3.3). Cette approche n'exclut pas l'approche de gestion via les adaptateurs qui peut être appliquée pour l'acquisition et la communication des données. La principale limitation de cette approche est la complexité du modèle CIM et la difficulté de sa prise en main. De plus, il est difficile d'exprimer certaines relations entre les ressources administrées telles que la hiérarchie ou le partage des ressources.

Ce travail aborde la problématique de la gestion des ressources hétérogènes dans un contexte multi-échelles. Afin de relver ce défi, nous explorons la piste de l'encapsulation par composants et de la découverte dynamique des ressources administrées.

1.5 Contributions

Face aux nouveaux besoins, les systèmes de gestion traditionnels [Tiv, Ope, Nag07] ont montré leurs limitations. En effet, bien que de nombreuses plates-formes disponibles soient développées en utilisant des architectures ouvertes , dans la pratique, leurs limitations en termes d'extensibilité opérationnelle ou fonctionnelle sont fortement ressenties par les utilisateurs. En terme opérationnel, il est reconnu que l'accroissement de la taille d'un réseau géré impose, au mieux, l'augmentation des capacités de traitement des systèmes sur lesquels sont installées les plates-formes de gestion, ainsi que l'augmentation des capacités de transfert des données, et, au pire, le changement des systèmes existants pour des systèmes plus puissants. Fonctionnellement parlant, les faibles niveaux d'abstraction utilisés par les systèmes d'information sous-jacents ont pour conséquence d'entraîner des délais et des coûts importants lorsqu'il s'agit de développer de nouveaux services, difficiles à admettre par les clients.

Approche architecturale d'administration à base de domaines. Le principal enjeu de l'administration intégrée multi-échelles est d'offrir une solution de gestion unique permettant de gérer des ressources hétérogènes à différentes échelles.Il existe plusieurs pistes pour répondre à cette problématique, telle que les architectures logicielles, les

protocoles d'administration, les méta-modèles et modèles d'information génériques. Dans le cadre de ce travail, nous avons privilégié l'approche architecturale. Nous estimons que, même si on dispose du protocole d'administration le plus intelligent ou du modèle d'information le plus générique possible, si l'architecture du logiciel d'administration n'a pas été pensée initialement pour évoluer à différentes échelles nous nous retrouverons toujours dans une impasse. Nous proposons donc une approche [MkL06a, MkL06b] qui s'appuie sur :

- **L'administration fondée sur l'architecture des systèmes administrés.** L'administration fondée sur l'architecture des systèmes administrés est l'approche que nous proposons pour la gestion des ressources hétérogènes. Ce choix est motivé par la capacité des descriptions architecturales à offrir une vue homogène commune des ressources hétérogènes.
- **La gestion par domaines.** Nous définissons un domaine d'administration comme un regroupement de ressources administrées qui partagent une ou plusieurs propriétés communes. La plupart des systèmes d'administration classiques peuvent être considérés comme « mono-domaine » : il s'agit de systèmes d'administration centralisés avec un seul domaine qui regroupe l'ensemble des ressources administrées. Dans un contexte multi-échelles, cette approche peut rencontrer des limitations pour passer à l'échelle. Dès lors, le domaine d'administration peut servir de mécanisme d'administration pour fragmenter l'ensemble des systèmes administrés d'un parc de ressources en ensembles plus réduits. Cette fragmentation simplifie considérablement les tâches d'administration.
- **Stratégies d'administration à base de domaines.** Le choix de la stratégie d'agencement des domaines seront agencés et les fonctions d'administration exercées par plusieurs domaines sont déterminants pour le passage à l'échelle des infrastructures d'administration. Il n'existe pas de meilleure stratégie dans l'absolu. Le choix d'une stratégie ou d'une autre est effectué au cas par cas en fonction d'un certain nombre de paramètres. Nous présenton alors un échantillon de ces paramètres et une classification des différentes stratégies d'administration à base de domaines.

Un canevas intergiciel à base de composants pour l'administration multi-échelles. Dans le cadre ce travail, nous avons implémenté le canevas DASIMA afin de valider notre approche architecturale d'administration à base de domaines. Il s'agit d'un canevas intergiciel pour l'administration intégrée multi-échellles [MkL09].

DASIMA propose des fonctions de médiation qui présentent, d'une part, des fonctions d'observation des ressources à administrer, et, d'autre part, des mécanismes d'action sur ces ressources. Ce canevas d'administration est au service à la fois des administrateurs (personnes physiques) et d'un ensemble d'applications et de processus d'administration.

Afin de doter DASIMA d'un ensemble de propriétés (e.g. reconfigurabilité, modularité, administrabilité, extensibilité) facilitant son adaptation dans un contexte multi-échelles nous nous sommes appuyés sur le modèle à composants Fractal.

Notre canevas d'administration permet de pallier les limites des systèmes d'administration classiques existants. Il résulte d'une approche combinant systèmes répartis et approches à composants qui donne une solution à la fois souple et flexible. Ces deux propriétés sont fortement recommandées dans un contexte d'administration multi-échelles.

Mise en œuvre de l'administration reflexive Afin de doter DASIMA d'une capacité d'administration reflexive, pour faciliter son administration, nous nous sommes ap-

puyés sur FractalJMX[2]. Cette introspection des domaines DASIMA a été effectuée grâce à l'injection d'un Agent FractlJMX dans le domaine principal. Ce faisant, nous avons pu administrer aussi bien les éléments administrés que le système d'administration lui même, via l'introspection et l'administration de tous les sous-domaines et services DASIMA.

Evaluation et recommandations dans un contexte large échelle. Pour la validation de notre approche d'administration nous avons choisi une étude de cas réelle issue du monde des télécommunications. Il s'agit d'une application de l'opérateur Orange utilisée dans un contexte M2M pour la gestion des communications entre des équipements et des ressources potentiellement fortement hétérogènes. Dans le cadre de nos campagnes de test, nous avons déployé cette application à différentes échelles (réseau local) et sur une grille de cacul fortement hétérogène et distribuée à l'échelle nationale (GRID5000). Plusieurs expérimentations ont été effectuées afin d'évaluer les performances du canevas DASIMA [MkL09]. A l'issue de ces expérimentations, nous avons obtenu les informations nécessaires pour le calibrage de la plateforme pour des déploiements dans des conditions matérielles similaires aux conditions des expérimentations.

1.6 Organisation du manuscrit

Ce document s'organise en deux parties. La première partie (chapitres 2 et 3) présente l'état de l'art des domaines considérés dans ce manuscrit. La seconde partie du manuscrit (chapitre 4,5,6,7) s'appuie sur les failles des systèmes d'administration décelées et les besoins actuels en terme d'administration pour proposer notre approche et plateforme d'administration.

Le chapitre 2 rappelle succinctement les principes de la gestion des systèmes et réseaux. Nous présentons ensuite un survol des principaux standards du domaine de l'administration, et nous introduisons les outils de gestion les plus aboutis dans ce domaine. Nous constatons que ces standards et ces technologies ne sont plus adaptés à la révolution que connaît depuis quelques années le monde des systèmes et des réseaux informatiques. Au regrad, des recherches menées pour diminuer l'écart entre les besoins et les possibilités techniques, nous identifions les principaux problèmes reconnus dans ce domaine. La fin de ce chapitre est une projection en avant dans l'évolution des technologies utilisées ou développées dans le monde de l'administration, pour constater que la technologie des composants logiciels pourrait représenter une piste prometteuse pour pallier certaines limitations des approches existantes.

Le chapitre 3 est consacré aux systèmes de gestion à base de composants. Nous présentons dans un premier temps le concept de composant logiciel. Nous dressons ensuite un panorama des applications de gestion à base de composants. Nous étudions ces systèmes à travers 3 familles de travaux dans ce domaine : les systèmes à base de composants, les systèmes de gestion fondée sur l'architecture et les systèmes à base de composants pour la gestion fondée sur l'architecture. A l'issue de chapitre, nous dégageons les lacunes de ces systèmes et celles des systèmes d'administration existants.

[2] http : //fractal.objectweb.org/current/doc/javadoc/fractal-jmx/

Le chapitre 4 est un chapitre de transition, qui rappelle les défis à relever, présente une synthèse des travaux de l'état de l'art, et positionne notre contribution.

Le chapitre 5 introduit notre approche d'administration multi-échelles. Cette approche est basée sur le concept de *domaine d'administration*. Nous définissons dans un premier temps ce concept et nous justifions son utilité dans un contexte multi-échelles. Nous proposons, ensuite, des exemples d'usage et de fédération des domaines d'administration.

Le chapitre 6 dessine les principaux traits de l'architecture fonctionnelle et technique de la solution intergicielle DASIMA [MkL09] que nous proposons pour mettre en œuvre notre approche de gestion.

Le chapitre 7 est dédié à l'évaluation de notre travail. Nous présentons une évaluation qualitative ainsi que les résultats quantitatifs des expériences menées dans cette thèse.

La conclusion synthétise ce qui a été réalisé dans cette thèse et décrit les perspectives de notre travail.

Chapitre 2

Administration des systèmes et réseaux

Contents

2.1 Historique 36
2.2 Concepts de base de l'administration intégrée 37
2.2.1 Administration et administration intégrée 37
2.2.2 Le modèle gestionnaire-agent 38
2.2.3 Les fonctions d'administration 39
2.2.4 Les niveaux d'administration 40
2.3 Les standards d'administration 41
2.3.1 SNMP 41
2.3.2 JMX 44
2.3.3 CIM/WBEM 47
2.3.4 WSDM 51
2.3.5 Conclusion sur les approches standardisées 52
2.4 Les systèmes classiques d'administration 53
2.4.1 IBM Tivoli 53
2.4.2 HP-OpenView 57
2.4.3 Nagios 62
2.4.4 Conclusion : Réalité des plates-formes de gestion 65
2.5 Travaux connexes de recherche 66
2.6 Conclusion 67

L'objectif de ce chapitre est d'introduire les préceptes de base de l'administration des réseaux et des applications logicielles. Il brosse un aperçu des différentes solutions d'administration actuelles et met l'accent sur quelques standards dominants, ou émergeants, prônés par les organismes internationaux de standardisation de la gestion des réseaux informatique et de télécommunication. Ce chapitre débute par un bref historique des systèmes d'administration et introduit des concepts de base de la gestion. Il poursuit avec un survol des standards d'administration les plus populaires. Il enchaîne par la suite avec la présentation des principaux systèmes d'administration commerciaux et libres disponibles sur le marché. Et il termine avec un balayage des plus récents travaux de recherche dans le domaine.

2.1 Historique

L'administration des systèmes et des réseaux comprend l'ensemble des activités qui permettent de surveiller, configurer et commander l'utilisation et le fonctionnement des ressources du réseau, dans le but de fournir des services aux usagers avec le meilleur niveau de qualité et de coût. Les origines de cette pratique remontent à plus d'un siècle. En effet, pour les réseaux de télécommunication, la première opération d'administration d'envergure connue date de 1884. Il s'agit d'une étude pour déterminer le tarif optimal qui a été effectué par *Moris Tayler* pour la *Southern New England Telephone Company* six ans après la première mise en service commerciale du téléphone.

Ce n'est que dans les années 80 avec le développement des systèmes informatiques et surtout avec leur mise en réseau que la communauté scientifique et technique a commencé à s'intéresser à la gestion des systèmes et des réseaux en tant que discipline à part entière. C'est à cette époque que les premiers systèmes et standards d'administration ont commencé à apparaître. Cette période fut l'âge d'or de l'administration avec la naissance des standards ISO tels que CMIS/CMIP (*Common Management Information Services and Protocols*). C'est à cette époque également que le protocole d'administration le plus répandu de tout les temps, SNMP [JCD90], a vu le jour grâce aux travaux de l'IETF.[1]

Une dizaine d'années plus tard, le réseau des réseaux, Internet, est apparu. De nouvelles technologies telles que le protocole HTTP (*Hypertext Transfer Protocol*) ou le langage XML (*Extensible Markup Language*) se sont imposées comme outils indispensables pour l'échange des données sur la toile. Parallèlement, les travaux menés par la communauté génie logiciel ont donné naissance au paradigme orienté objet, qui consiste en la définition et l'assemblage de briques logicielles appelées objet. Ces technologies ont été très rapidement adoptées par le monde de l'administration. Le standard CIM/WBEM [wbe], qui a été validé par le DMTF en 1997, est le résultat de la combinaison de ces technologies.

Aujourd'hui la toile Internet continue son succès. Les entreprises et les organisations ont développé des nouveaux concepts et technologies pour échanger des informations ou pour offrir des *services* à travers ce réseau. C'est autour de ce concept de service qu'une nouvelle approche architecturale connue sous le nom d'« *architecture orientée service* »a vu le jour. Cette approche facilite la structuration des infrastructures de gestion dans des contextes caractérisés par leur hétérogénéité et leur dynamisme. De nouveaux standards basés sur cette technologie ont été définis en 2005 et 2006, WSDM [WSD05b, WSD05b] et WS-Management [wsm06].

Par ailleurs, le marché des infrastructures d'administration, estimé a 5,1 milliards de dollars en 2002 [2] convoitent les éditeurs logiciels et les constructeurs de matériels les plus prestigieux de la planète. Les offres de logiciels d'administration se sont multipliées. Ces logiciels sont concurrencés par des logiciels gratuits et libres tels que Nagios [Nag07], qui couvrent un spectre moins large des besoins fonctionnel de l'administration, mais permettent une gestion convenable des petites et moyennes entreprises.

Dans ce chapitre, nous passons en revue l'ensemble de ces travaux. Mais avant de présenter les architectures d'administration et les infrastructures de gestion, nous allons tout d'abord introduire quelques concepts de base indispensables pour comprendre la suite

[1] *The Internet Engineering Task Force.*
[2] Source Giga Group

du manuscrit.

2.2 Concepts de base de l'administration intégrée

Nous présentons dans cette section un ensemble de concepts indispensables pour assimiler la suite du manuscrit.

2.2.1 Administration et administration intégrée

Nous considérons dans le cadre de ce travail les termes *gestion* et *administration* synonymes. Afin de définir le concept d'administration nous nous référons aux travaux de recherche autour l'administration des systèmes et réseaux, dont les racines remontent à deux décennies. La définition suivante introduit le concept d'administration, tel que nous le comprenons.

Définition 2.1. *L'***administration d'un système** *consiste à le surveiller et le contrôler afin qu'il satisfasse les demandes des utilisateurs et les contraintes du propriétaire. [Slo94]*

Cette définition restreint l'administration à sa composante technologique, qui est celle sur laquelle nous avons développé ce travail de recherche. Elle fait clairement apparaître deux processus de base de l'activité d'administration : *la surveillance* (observation) et *le contrôle* (action).

Aujourd'hui, avec l'évolution des besoins d'administration, *la convergence des services*, le développement du marché des M2M, les entreprises, en général, et des opérateurs de télécommunications, en particulier, sont de plus en plus confrontés à la problématique de l'administration des ressources hétérogènes. Un opérateur comme France Télécom, par exemple, peut être amené à gérer un équipement réseau au même titre qu'un système informatique, un programme informatique, un service déployé sur ses passerelles domestiques ou encore le trafic sur un réseau d'où l'avènement de l'**administration intégrée** que nous définissons comme suit :

Définition 2.2. *La* **administration intégrée** *consiste à mutualiser, au sein d'une même infrastructure d'administration, des activités d'administration communes à un ensemble de ressources hétérogènes.*

La figure 2.1 illustre le schéma général d'une administration intégrée. Elle montre un exemple de consolidation de données d'administration en provenance des différents systèmes de surveillance et d'administration disponibles dans une entreprise.

Cette approche d'administration est particulièrement utile dans un environnement de ressources hétérogènes et interdépendantes. Des exemples possibles sont les applications d'entreprise et les systèmes s'exploitation. L'état d'une application dépend fortement de l'état du système d'exploitation qui la supporte. A partir du moment où la surveillance d'une simple application ne peut donner une vue correcte de l'ensemble des problèmes possibles, il devient nécessaire d'intégrer les informations en provenance de tous les composants impliqués dans le système.

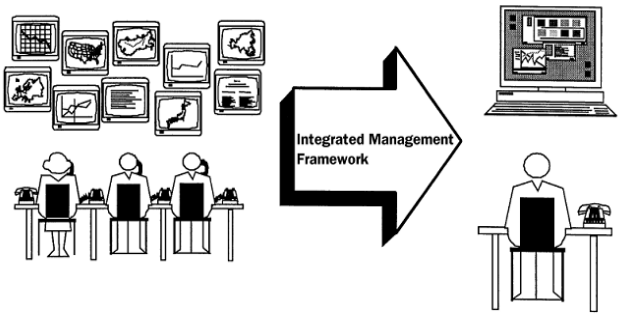


FIG. 2.1 – Exemple d'administration intégrée (extrait de [MKP99])

Pour chaque système géré, les fonctions d'administration peuvent différer [X.792] mais les objectifs restent les mêmes : **surveiller** et **contrôler**. Nous présentons dans cette section les principales fonctions d'administration, le modèle global des applications d'administration ainsi que les différents niveaux d'administration.

2.2.2 Le modèle gestionnaire-agent

En dépit d'une grande variété, l'essentiel des technologies/standards/produits utilisés dans le domaine de l'administration repose généralement sur un modèle générique, appelé le modèle gestionnaire-agent [X.792]. La figure 2.2 illustre les entités constituant ce modèle à savoir :

- Le **gestionnaire** peut être soit un administrateur humain soit un logiciel contenant la logique requise pour effectuer des opérations d'administration bien identifiées. Pour se faire, il interagit avec les agents pour collecter des informations concernant les ressources administrées. Un gestionnaire n'interagit jamais directement avec un système administré mais au travers d'un agent, généralement proche des ressources administrées.
- L'**agent** est une entité logicielle ou matérielle embarquée dans une ressource administrée. Il représente le délégué local du gestionnaire pour les *objets administrés* dont il consulte ou modifie l'état. Un système peut ainsi être administré par simple exposition d'information au travers d'un agent.
- La **MIB (Management Information Base)** représente la connaissance partagée des données d'administration sur les ressources administrées. Elle repose sur un *modèle d'information* qui définit l'organisation des informations de gestion.
- Le **protocole d'administration** définit les interactions entre le gestionnaire et les agents.

Le principe du modèle agent-gestionnaire est de bien distinguer le rôle du système d'administration et celui des entités administrées. Il a été spécifié pour satisfaire plusieurs fonctions de l'administration telles que la configuration, la sécurité, la performance ou la facturation. Ces fonctions seront détaillées dans la section suivante.

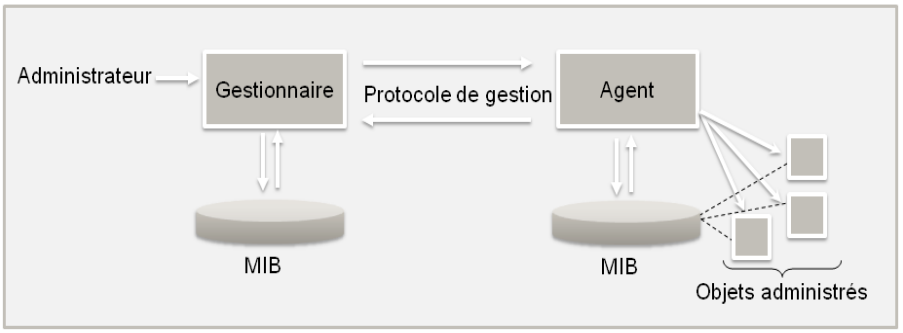


FIG. 2.2 – Le modèle architectural gestionnaire-agent

2.2.3 Les fonctions d'administration

Le premier travail qui a décrit les fonctions [X.792] de gestion a été effectué dans les années 80 par l'organisation OSI dans le cadre de la gestion des réseaux ISO (*Interconnexion de Systèmes Ouverts*). Ce travail tente, entre autre, de séparer la tâche d'administration en différentes fonctions administratives. Les principales fonctions définies, couramment désignées sous le nom de *FCAPS*[3], sont les suivantes :

La gestion des pannes. L'ensemble des ressources gérées et disponibles sur le réseau ne sont pas à l'abri d'une défaillance. La fonction gestion des pannes consiste à faciliter la détection et l'identification des éventuels dysfonctionnements qui peuvent se produire durant le cycle de vie des ressources administrées. Idéalement, une fois localisée, la faute est confinée pour ne pas paralyser le système administré, par exemple ou pour éviter sa propagation. Enfin, le dysfonctionnement est traité l'administrateur.

La gestion de configuration. La gestion de la configuration a pour mission d'initialiser le système et permettre la modification de l'état des ressources composant le système. Un exemple d'action possible consiste à modifier les tables de routage pour rediriger le trafic sur le réseau pendant les heures de pointe. Dans le cas de ressources logicielles, la fonction de configuration peut permettre, par exemple, la modification de son état en cours d'exécution.

La gestion des performances. Cette fonction sert essentiellement à évaluer en permanence les performances du réseau afin de mesurer la qualité des services offerts aux utilisateurs et d'identifier les éventuelles dégradations de performance. Les principales tâches effectuées sont l'observation de l'activité des systèmes et des réseaux et la collecte des indicateurs de mesure de performance (le débit efficace, le taux de perte d'information, le taux d'utilisation, le temps de réponse de la base de données, etc.). Un débit réseau excessif est un exemple de dégradation au niveau du réseau. Un temps de réponse inacceptable d'un serveur Web est également une dégradation de la qualité du service web qui peut paralyser l'activité commerciale d'une entreprise de vente en ligne et occasionner des pertes financières considérables.

La gestion de la sécurité. Les principales tâches de la fonction de gestion de sécurité consistent à gérer la protection des informations et à surveiller et contrôler l'accès aux ressources matérielles et logiciels disponible sur le réseau. Elle concerne également la génération, la distribution et la maintenance des mots de passe et des clés de cryptage.

[3]FCAPS : acronyme de gestion de Fautes/ Configuration/ Audit/ Performance/ Sécurité.

Audit et facturation. Il est indispensable pour certaines entreprises, tels que les opérateurs de télécommunication, de surveiller l'utilisation de leurs services (la consommation téléphonique par exemple). La fonction de gestion de facturation permet de répondre à ce besoin. Elle consiste principalement à la définition des coûts d'utilisation pour chaque ressource facturable. Ses principales tâches consistent à collecter les données nécessaires à la facturation et à mettre à la disposition des utilisateurs des ressources les informations de tarification relatives à leur utilisation.

Les fonctions FCAPS ne sont pas indépendantes ni disjointes. La tâche de gestion des pannes peut être fortement liée à celle de la gestion de configuration. En pratique, les fonctions d'administration sont appliquées à différents niveaux d'abstraction des systèmes administrés. Ces niveaux d'abstraction feront l'objet de la section suivante.

2.2.4 Les niveaux d'administration

Dans le contexte des systèmes répartis, on peut distinguer quatre niveaux d'abstraction possibles. La figure 2.3 illustre ces différents niveaux. Chaque niveau est représenté par un ensemble de ressources matérielles ou logicielles ayant des propriétés communes.

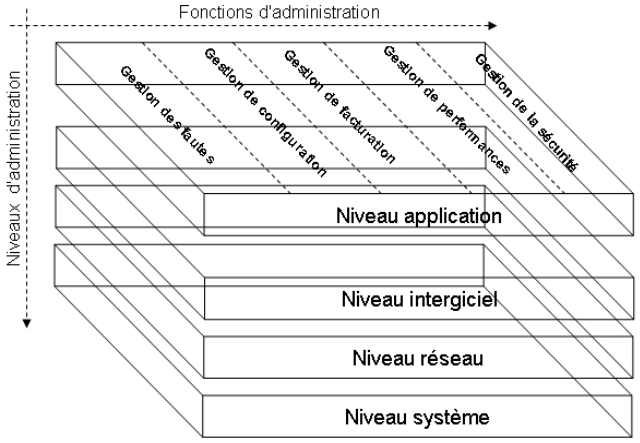


FIG. 2.3 – Les niveaux d'administration

- **Le niveau système**. Ce niveau se base sur les concepts systèmes de bas niveau. Il concerne, entre autre, l'utilisation des diverses ressources du système d'exploitation tels que l'allocation de mémoire ou du disque. Il peut concerner également la gestion des utilisateurs du système et des politiques d'affectation des ressources mises à leur disposition.
- **Le niveau réseau**. Ce niveau regroupe la gestion de la communication et des équipements intervenant dans les couches basses du réseau comme les routeurs ou les ponts. Les constructeurs de ces équipements embarquent généralement des agents logiciels ou matériels afin de permettre leur administration. Dresser et mettre à jour la topologie du réseau ou la supervision des informations qui circulent sur le réseau sont des exemples typiques des opérations d'administration effectuées à ce niveau.
- **Le niveau intergiciel**. Le niveau intergiciel est constitué de l'ensemble des intergiciels déployés dans une entreprise. Les serveurs d'application ou les intergiciels de

communication sont des exemples typiques d'intergiciels qui constituent l'épine dorsale de l'infrastructure informatique répartie d'un grand nombre d'entreprises.
- **Le niveau application**. Ce niveau représente l'ensemble des applications disponibles telles que les SGBD ou les applications de gestion de travail collaboratif. Ces applications s'appuient très souvent sur des infrastructures intergicielles.

2.3 Les standards d'administration

Avec la prolifération des réseaux informatiques et l'accroissement de la complexité des logiciels s'est accentuée. Il s'agit de l'association dans un seul réseau de matériels et de logiciels provenant d'une multiplicité de fournisseurs. Afin de faire face à cette hétérogénéité, il fallait relever un double défi : d'une part inventer les techniques d'administration et d'autre part normaliser les mécanismes utilisés par les différents concepteurs de solutions d'administration. Cet effort considérable est actuellement mené au travers de nombreux organismes coopérants au plan international. Nous balayons dans les sections suivantes quatre parmi les standards les plus aboutis dans le monde de l'administration des systèmes et réseaux : a) SNMP [JCD90] l'un des premiers travaux sur l'administration des systèmes et réseaux, b) JMX [JMX04] un de facto standard d'administration du monde Java/JavaEE, c) CIM/WBEM [wbe] un standard DTMF qui a marqué l'histoire de l'administration et finalement d) WSDM [WSD05a, WSD05b] un standard émergeant qui réglemente l'application des services Web dans l'administration.

2.3.1 SNMP

SNMP (pour *Simple Network Management Protocol*) [Sta04] est issu du protocole SGMP (*Simple Gateway Monitoring Protocol*), un protocole qui n'a jamais été déployé. La première version de protocole a été proposée par l'organisme de standardisation IETF [4] en 1988. Cette version est aujourd'hui la version la plus utilisée dans le domaine de d'administration de *réseaux IP*. Aujourd'hui, l'utilisation de SNMP dépasse largement la simple gestion des réseaux et des équipements. Elle couvre également la gestion des services et des applications (logiciels, serveurs, bases de données, etc.). Le protocole doit son succès principalement à sa simplicité. Cette propriété constitue à la fois sa force et sa faiblesse.

2.3.1.1 Architecture

SNMP s'inspire des concepts introduits par le modèle gestionnaire-agent. Son architecture illustrée dans la figure 2.4 se décline principalement en quatre éléments :

- Une **station d'administration** qui joue le rôle du gestionnaire et qui héberge les applications d'administration. C'est via ces applications que l'administrateur va interagir avec les différentes ressources administrées.
- Un **agent** d'administration sur chaque équipement qu'on souhaite administrer (station de travail, routeur, répéteur, etc.) La mission de cet agent consiste, d'une part, à répondre aux requêtes de gestion qui lui sont adressées et d'autre part remonter

[4] The Internet Engineering Task Force

à la station de gestion des éventuels alarmes et notifications relatifs aux ressources administrées.
- Une **MIB** qui est une base de données administrative. Elle regroupe l'ensemble des objets administrés de la station administrée ainsi que leurs attributs. Un objet administré dans SNMP est une abstraction d'une ressource réseau administrable.
- Un **protocole d'administration** définit le dialogue entre un gestionnaire SNMP et un agent d'administration. Il se base sur un échange de messages entre le gestionnaire et l'agent.

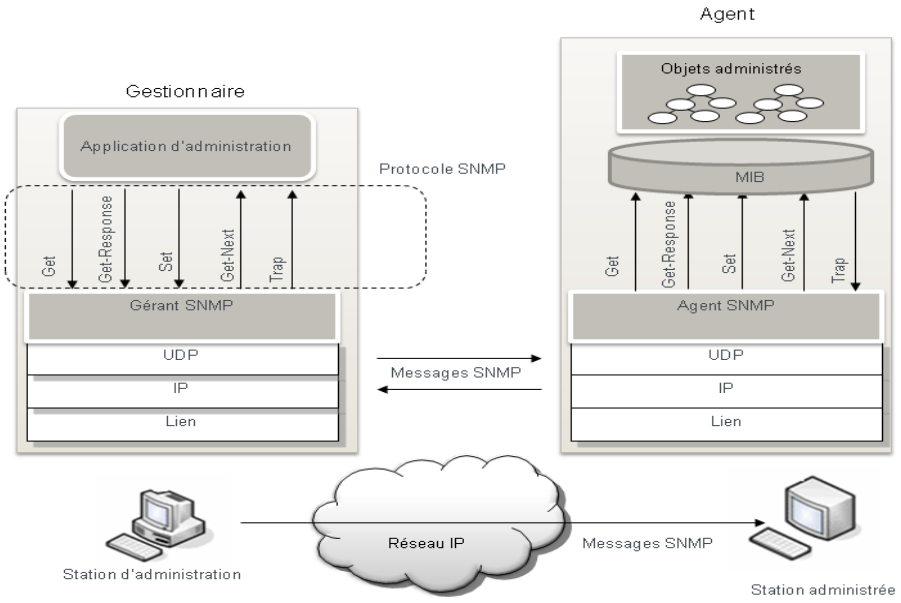


FIG. 2.4 – Architecture d'administration SNMP

2.3.1.2 Le protocole SNMP

On ne peut pas parler de SNMP sans évoquer son protocole de gestion. Ce protocole se base sur le protocole UDP (*User Datagram Protocol*) pour le transport des informations. Les messages appelés PDU (*Process Data Unit SNMP*) sont échangés entre la station d'administration et l'agent sans garantie d'arrivée. En cas de retard de la réponse à une requête d'administration, la station d'administration relance l'agent en réémettant la même requête d'administration. Le protocole SNMP décrit l'ensemble des opérations entre l'application d'administration et le système administré. Ces opérations sont des requêtes, des réponses et des alertes. Il existe principalement quatre types de requêtes :

- `GetRequest` qui envoie une requête, demandant la valeur de l'entrée de la MIB passée en paramètre,
- `GetNextRequest` qui permet de faire une requête sur l'entrée de la MIB suivant celle passée en paramètre,
- `SetRequest` qui change l'entrée passée en paramètre par la valeur donnée elle aussi en paramètre,
- `GetResponse` qui permet de récupérer une réponse à une requête précédemment émise.

2.3.1.3 La MIB SNMP

Par analogie aux bases de données classiques, la MIB est peut être vue comme une mini base de données normalisée contenant les différentes informations de gestion. Les agents SNMP jouent le rôle de système de gestion de base de données (*SGBD*). La MIB regroupe l'ensemble structuré d'objets administrés. La syntaxe et l'encodage de ces objets sont tous les deux définis grâce au langage de description ASN.1 [JCD90]. L'ISO définit des identificateurs d'objets (`OID`) afin de désigner les objets de la MIB d'une façon unique. Les OID sont organisés dans un arbre d'enregistrement. Des règles normalisées régissent l'affectation des numéros d'identificateurs à ces variables. La figure 2.5 illustre un extrait d'une MIB SNMP. Chaque identificateur est constitué d'une séquence d'entiers séparés par des points. Cette séquence reflète l'aspect hiérarchique d'organisation des variables dans une MIB. La structure hiérarchique de la MIB est représentée sous une forme arborescente dont la racine n'est pas numérotée.

Si on veut connaître, par exemple, le temps depuis lequel une machine est allumée, il suffit de regarder dans la MIB à quoi correspond en notation en nombres pointés le (*uptime*) : .1.3.6.1.2.1.1.3.0

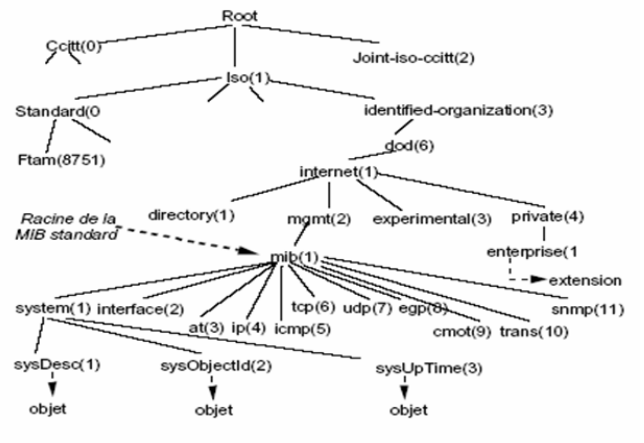


FIG. 2.5 – Structure hiérarchique d'une MIB

2.3.1.4 Synthèse

SNMP est un protocole pionnier dans le monde de l'administration des systèmes et réseaux. Le secret de sa réussite et de sa pérennité c'est sa simplicité. C'est pour cette raison que les constructeurs matériels peuvent instrumenter leurs appareils avec un minimum de coût et que les administrateurs peuvent le prendre en main avec un minimum d'effort. De plus, les ressources nécessaires pour mettre en place une infrastructure d'administration basée sur SNMP sont peu importantes. Toutes ces raisons font que SNMP est très largement répandu et continue à être utilisé intensivement. Toutefois, la simplicité de SNMP implique aussi plusieurs inconvénients :

Vulnérabilité. La sécurité est l'un des problèmes les plus importants dans SNMP. En effet l'authentification reste très simple et donc non sécurisée ce qui rend SNMP assez vulnérables aux intrusions. Un intrus peut, avec un minimum de connaissance

en réseau, accéder au contenu des informations d'administration qui transitent sur le réseau. Pire encore, l'intrus peut prendre le contrôle des équipements disponibles sur le réseau à cause de ces multiples failles.

Manque de fiabilité. En plus des failles de sécurité, SNMP souffre également d'un sérieux problème de fiabilité. En effet, les alarmes ne sont pas acquittées et un agent n'est jamais sûr de bien avoir averti sa station d'administration. Par ailleurs, SNMP a été conçu initialement pour l'administration des réseaux d'entreprises. C'est pour cette raison qu'il souffre de quelques limitations pour passer à l'échelle

Absence de communication entre gestionnaires. La première de ces lacunes consiste dans le choix du modèle d'administration. Un seul modèle est applicable : il s'agit du modèle centralisé. Aucune communication de station d'administration à station d'administration n'est possible. Il est donc impossible à une station d'administration de faire des requêtes à un périphérique administré par une autre station en passant par son intermédiaire.

Verbosité. Une autre limitation qui risque d'écrouler le réseau si on applique SNMP dans un contexte large échelle réside dans le fait que SNMP est un protocole très verbeux. Chaque réponse reçue correspond une requête et le transfert de données à travers le réseau s'avère donc assez important, ce qui surcharge ce dernier par rapport à ce qu'il aurait été en cas de non administration.

Non-adaptation à l'échange de grands volumes de données. Le protocole n'est pas très pratique pour manipuler une quantité de données considérable comme une table de routage par exemple.

Toutes ces raisons font que SNMP souffre d'un manque de fiabilité, de sécurité et de capacité de passage à l'échelle. Ces lacunes deviennent de plus en plus problématiques au fur et à mesure que le nombre des ressources à administrer devient important.

2.3.2 JMX

JMX (*Java Management Extensions*) [JMX04] est un de facto standard, développé dans le cadre du *Java Community Process* [Mic06] qui supporte le service de gestion pour des ressources Java [GJ+05]. Il définit une extension du langage Java qui permet à toute ressource Java d'être administrable.

2.3.2.1 Architecture

La spécification JMX définit une architecture ainsi qu'un ensemble de services permettant d'administrer aussi bien des applications Java que des équipements réseaux. L'architecture proposée est illustrée dans la figure 2.6. Elle se décline en trois niveaux :

- Le **niveau instrumentation** définit un mécanisme connu sous le nom de beans gérés ou MBeans (*Managed Beans*) permettant d'instrumenter les ressources.
- Le **niveau agent** est définit par un ensemble d'agents d'administration. Ces agents contrôlent les MBeans et les rendent accessibles aux applications d'administration distantes. Chaque agent a la responsabilité d'un ensemble de MBeans. Toutes les opérations d'administration requises sur un MBean doivent passer par son agent.
- Le **niveau services distribués** représente les gestionnaires qui communiquent avec les applications via les connecteurs. Il spécifie également des composants particuliers,

appelés connecteurs ou adaptateurs de protocoles, permettant aux applications d'administration d'accéder à distance aux agents JMX en respectant divers protocoles de communication.

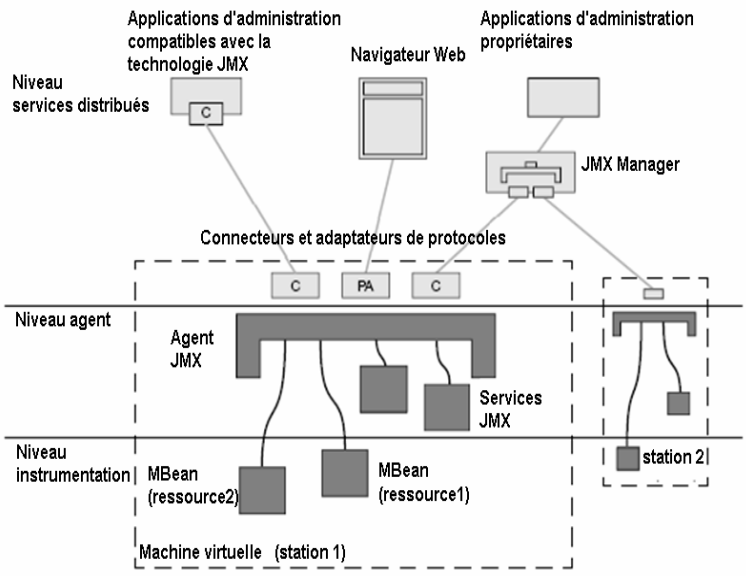


FIG. 2.6 – Architecture globale de JMX [JMX04]

2.3.2.2 La couche agent

La couche agent joue un rôle central dans l'architecture JMX. C'est via cette couche que transitent toutes les opérations d'administration. L'agent JMX est similaire à l'agent SNMP dans le sens où il constitue le représentant des ressources administrées vis-à-vis des applications d'administration. Le rôle principal d'un agent JMX est d'enregistrer les MBeans. Il permet également d'explorer des MBeans en fournissant les moyens de consulter et modifier leurs attributs ou d'invoquer les opérations définies dans leurs interfaces. Un agent JMX intercepte aussi toutes les notifications émises par les MBeans pour les transférer aux applications de gestion concernées.

En plus du rôle de la médiation, un agent JMX définit un certain nombre de services, implémentés sous forme de MBeans, dont les plus importants sont résumés dans le tableau 2.1.

2.3.2.3 La couche d'instrumentation

Les MBeans représentent la brique de base de l'architecture JMX. Ils fournissent le moyen d'instrumentation des ressources logicielles ou matérielles qu'on souhaite administrer. Un MBean se présente comme un objet Java qui expose une interface d'administration et qui se conforme à un certain nombre de règles de conception (*Design Pattern*). Il est définit par un ensemble d'attributs représentant son état ainsi qu'un ensemble d'opérations permettant de manipuler son état et de l'administrer.

TAB. 2.1 – Table services JMX

Service	Description
Le service m-let	Il permet le chargement dynamique des classes depuis une URL (*Uniform Resource Locator*). Il peut être utile par exemple pour déployer sur le réseau une applet d'administration JMX dont le *ClassLoader* viendra chercher les classes sur le serveur HTTP (*Hypertext Transfer Protocol*) de l'agent.
Le service de monitoring	Ce service permet, dans le cadre d'une gestion réseau, de surveiller des variables déportées. Il permet en fait de s'enregistrer auprès d'un MBean distant pour être notifié des changements de valeurs.
Le service de compteur	Ce service envoie un événement lorsqu'un délai est écoulé.
Le service de gestion de dépendances	permet de définir des relations entre MBeans et de s'assurer de la validité de ces relations. Il peut être utile lorsqu'un MBean dépend d'un autre et qu'on souhaite le charger.
Les adaptateurs de protocoles	Ils permettent d'accéder aux agents JMX depuis des clients éventuellement distants ou basés sur une autre technologie, par exemple depuis un client HTML (Hypertext Markup Language)

Un MBean joue un double rôle vis-à-vis des applications d'administration. D'une part, il permet d'abstraire l'état d'une ressource et de modifier cet état. D'autre part, il émet des notifications suite à la production de certains évènements liés à la ressource. Il existe deux types de MBeans :

- Les **MBeans standards** (*Standard MBean*). Ils se basent sur une interface d'administration fixée au moment de la compilation. Les attributs et les méthodes d'administration associés à une ressource sont déterminés de manière statique.
- les **MBeans Dynamiques** (*Dynamic MBean*). Cette catégorie de MBeans permet de déterminer, en cours d'exécution, les attributs et les opérations d'administration associés à une ressource.

Chaque MBean est cratérisé par un ensemble de notifications qu'il peut émettre suite à la réalisation d'un incident qui concerne la ressource qui lui est associée. Pour ce faire, JMX définit un modèle de notification de type *publish/subscribe* et s'appuie sur les classes et les interfaces suivantes.

- Une classe `Notification` qui représente la classe de base et implante la nature de l'événement.
- Une interface `NotificationBroadcaster` qui permet d'émettre les événements. Elle définit principalement deux méthodes (`addNotificationListener` et `removeNotificationListener`) permettant respectivement à un objet de s'abonner à un événement ou de s'en désabonner.
- Une interface `NotificationListener` qui permet aux entités qui désirent recevoir des notifications issues des MBean. La méthode `handleNotification` implante le traitement à faire à la réception d'un événement.

2.3.2.4 Synthèse

JMX est un standard d'administration des programmes Java. Toutefois son utilisation n'est pas limitée à ce contexte. Il s'étend aujourd'hui à d'autres contextes ; équipements réseau, équipement de télécommunication, système (CPU, mémoire, etc.), etc. JMX propose une architecture claire organisée en trois couches : le niveau agent, le niveau instrumen-

tation et le niveau applications d'administration. Ses points forts sont principalement la simplicité de l'instrumentation avec les MBean, la prise en main très rapide et l'ensemble des services de gestion élémentaires qu'il propose.

Cependant, l'utilisation de JMX comporte certains inconvénients :

- **L'absence de modèle d'information**. La modélisation des systèmes administrée par JMX reste insuffisante puisqu'elle ne permet pas de refléter la structure de l'application sous-jacente. En effet, la communication entre les MBeans n'est pas définie par le modèle et n'est pas exposée à l'administrateur de manière explicite. De plus, le modèle des MBeans est plat et il n'y a pas de notion de hiérarchie qui est très importante pour représenter des architectures logicielles.
- **Le déploiement limité à une seule machine virtuelle**. Le standard ne définit pas le déploiement distribué sur plusieurs machines virtuelles Java ce qui limite sa capacité de passage à l'échelle.
- **Un modèle de gestion centralisée**. Tout comme son prédécesseur SNMP, il ne spécifie pas (ou pas encore) la connexion entre un gestionnaire et plusieurs agents ce qui met en cause sa capacité de passage à l'échelle. Des travaux encours menés par Sun Microsystems traitent cet aspect de hiérarchisation des agents MBean.

La figure 2.7 [Mcm07] décrit un scénario de hiérarchisation des agents JMX dans les futures versions de JMX. L'idée de base de cette hiérarchisation consiste à importer les MBeans des Agents JMX esclaves (*Subagent1* et *Subagent2*) en tant que *MBeans virtuels* au niveau de l'agent JMX maître (*Master Agent*).

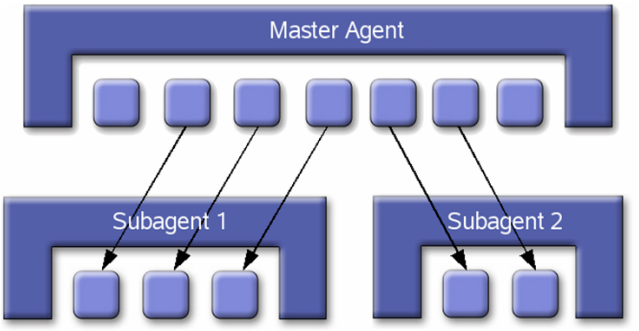


FIG. 2.7 – Agent JMX hiérarchiques

2.3.3 CIM/WBEM

CIM/WBEM [wbe] est un standard d'administration qui a été crée par le consortium DMTF (*Distributed Management Task Force*) en 1998. L'objectif de ce standard est de spécifier une solution d'administration intégrée, des utilisateurs, des applications et du réseau, basée sur le Web. L'originalité de WBEM consiste à consolider et à unifier les données fournies par des sources hétérogènes (SNMP, CMIP, DMI, JMX, etc.). En effet, ce standard se base sur une représentation commune des informations de gestion. Nous détaillons dans les sections suivantes les principaux éléments cette architecture.

2.3.3.1 Architecture

WBEM est fondée sur une architecture multi tiers. Cette architecture, illustrée dans la figure 2.8, se décline en quatre composants principaux :

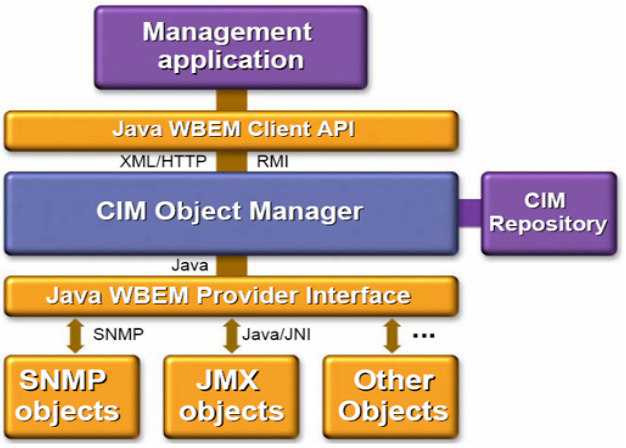


FIG. 2.8 – L'architecture WBEM

- **Les applications d'administration**. Le premier niveau de l'architecture est constitué par l'ensemble des applications clientes qui sollicitent les services du serveur CIM/WBEM via la Java WBEM Client API.
- **Les Providers (Java WBEM Provider Interface)**. Les providers jouent le rôle de fournisseurs de l'information de gestion et communiquent directement avec les ressources gérées. Ils peuvent également communiquer directement avec des ressources SNMP ou JMX. La mission des providers est la capture des informations et des évènements provenant des ressources administrées et de les communiquer au CIMOM.
- **Le CIMOM (CIM Object Manager)**. Le CIMOM est le composant cœur de l'architecture WBEM. Son rôle est similaire à celui de l'agent JMX. Il gère l'interaction entre les applications d'administration et les providers. Le CIMOM assure différents services tels que la gestion des notifications remontées par les providers, le traitement des requêtes provenant des clients et l'accès distant aux objets administrés.
- **Le référentiel (CIM Repository)**. Le référentiel est une base persistante dans laquelle le CIMOM stocke les informations concernant les providers et les ressources administrées. Un référentiel CIM est l'équivalent d'une MIB SNMP.

2.3.3.2 Le modèle CIM

CIM (*Common Information Model*) [CIM99] est un modèle et une méthodologie destinés à décrire les données de gestion dans un environnement informatique et réseaux. L'objectif de ce modèle est de faciliter la gestion des réseaux et des systèmes en proposant une approche de modélisation des ressources à administrer (applications, systèmes, réseaux, capteurs, etc.). Pour ce faire, le modèle se base sur des techniques de structuration et de conception selon le paradigme orienté objet. Les caractéristiques du modèle à objets telles que l'encapsulation et l'héritage font de CIM un modèle fortement générique et extensible. C'est grâce à cette flexibilité que le modèle est sans cesse enrichi par ses utilisateurs avec de

nouveaux concepts liés à des nouveaux domaines d'applications. CIM s'appuie sur un *méta-modèle* (une définition formelle du modèle CIM) qui est à son tour subdivisé en plusieurs niveaux conceptuels :

- **Le modèle noyau** est le premier niveau conceptuel issu du schéma CIM. Il représente les entités de base pour la construction de la hiérarchie d'objets CIM. Il définit les objets conceptuels qui permettent de capturer les notions transversales à toutes les activités de gestion. Il représente le point de départ pour les autres modèles qui sont dérivés. La figure 2.9 présente le diagramme de classe correspondant au modèle noyau.
- **Le modèle commun** capture les notions communes à une activité de gestion. Les domaines communs sont les systèmes, les applications, les bases de données, les réseaux et les équipements. Les objets qui sont définis dans ce modèle peuvent être étendus afin d'enrichir le modèle et de répondre à des besoins spécifiques.
- **Les schémas d'extensions** sont toutes les extensions du schéma CIM et qui sont spécifiques à une technologie particulière, par exemple un système d'exploitation, une application, une technologie réseau particulière, etc.

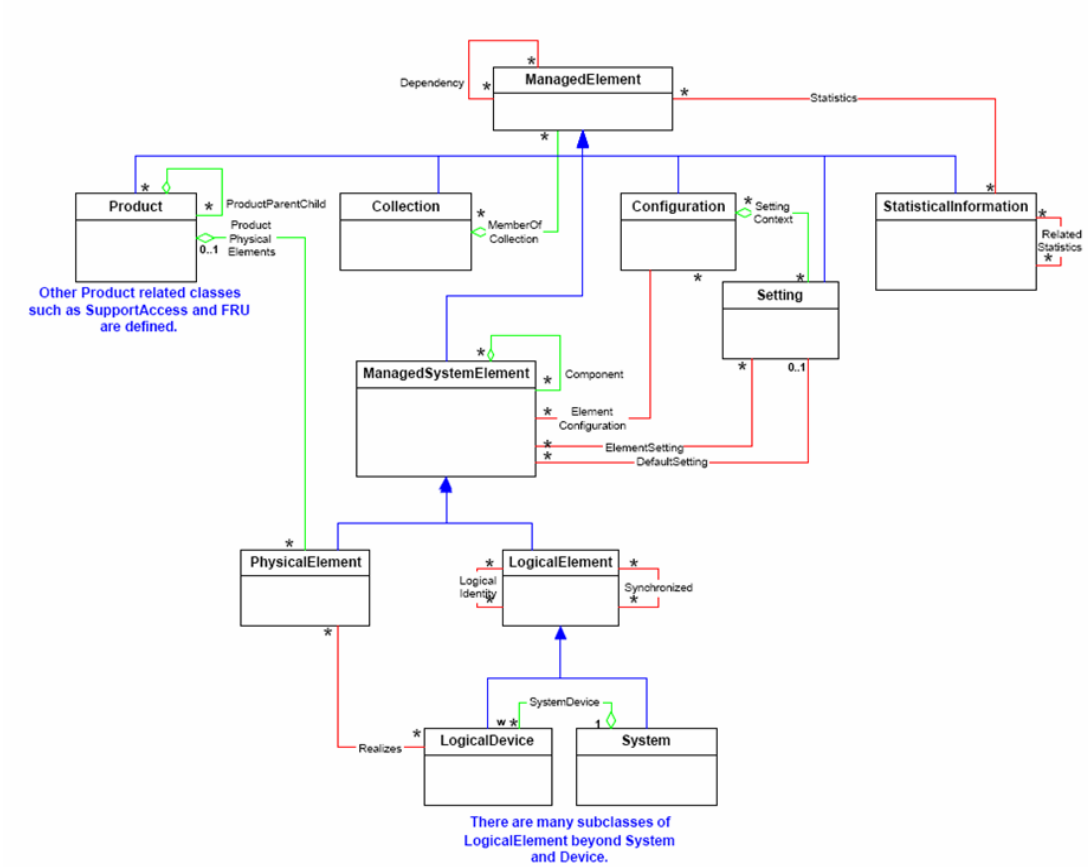


FIG. 2.9 – Extrait du modèle noyau de CIM [CIM99]

La figure 2.9 illustre un extrait du modèle noyau de CIM. Les principaux éléments de ce modèle sont résumés dans le tableau 2.2

TAB. 2.2 – Table principaux éléments du modèle CIM

Elements du modèle CIM	Description
`ManagedElement`	est la racine de la hiérarchie des classes CIM. Elle représente l'ensemble des éléments administrables logiques (tels que les programmes) ou physiques (occupent un espace physique : équipements réseau, PC, passerelle de service, capteur, etc.). Toutes les classes qui représentent ces éléments sont des descendants de la classe *ManagedElement*.
`ManagedSystemElement`	représente une spécialisation de la classe abstraite *ManagedElement*. Elle représente les éléments physiques administrables (PC, capteur, équipement réseau, etc.)
`ManagedLogicalElement`	représente les éléments logiques administrables tels que les processus système ou les bases de données.
`Settings`	définissent des paramètres prédéfinis applicables à un ou plusieurs *Managed Elements*.
`StatisticalInformation`	est une classe abstraite qui représente tout type d'informations statistique concernant les éléments gérés.
`Collections`	représente un groupement de ressources gérées (*Managed Element*).

2.3.3.3 Synthèse

CIM/WBEM est un standard qui a contribué fortement à l'avancement du domaine de l'administration intégrée. Le point fort de ce standard est incontestablement son modèle d'information CIM. Il s'agit d'un modèle orienté objets, extensible et générique. Un modèle générique orienté objets est la solution adoptée par ce standard pour faire face au problème de l'**hétérogénéité des ressources**. Le point fort concerne la prise en compte de la **volatilité des ressources** et des informations de gestion. CIM/WBEM stocke les informations de gestion au niveau du *CIM repository*. Il fait la distinction entre les *informations statiques* et les *informations dynamiques*. Pour satisfaire les requêtes d'administration, le CIMOM vérifie d'abord si les informations souhaitées sont statiques ou dynamiques. Dans le cas ou l'information demandée est statique, le CIMOM se contente de récupérer cette information à partir de son référentiel. Autrement, il va chercher l'information directement auprès des providers concernés par la requête.

Le standard CIM/WBEM souffre de plusieurs limitations :

- **Modèle très complexe**. La prise en main du modèle CIM reste une tâche ardue. CIM est un modèle très générique et comporte plusieurs niveaux d'abstraction. De plus, comme le modèle est extensible, plusieurs modèles sont étendus régulièrement et les dépendances entre ces modèles sont importantes et en perpétuelle évolution. Ceci rend la prise en main un peu délicate. Par ailleurs, le modèle ne cesse de subir des extensions et des modifications (même au niveau du noyau) ce qui implique des problèmes de compatibilité entre les versions de CIM.
- **Langage de description de bas niveau**. Un autre problème lié à CIM est le langage de description des ressources MOF. En effet, le MOF est un langage assez proche des langages à bas niveau de description, peu outillé, et peu utilisé dans d'autres contextes.
- **Modèle d'administration centralisée**. De point de vue du **passage à l'échelle**, tout comme le standard SNMP, CIM/WBEM s'est focalisé sur un modèle d'administration centralisé, qui limite les possibilités de passage à l'échelle. Le standard ne

se prononce pas quant à la possibilité de communication entre plusieurs CIMOM répartis.

2.3.4 WSDM

Avec la démocratisation des services web et l'essor des architectures orientées services, certains organismes comme l'OASIS (*Organization for the Advancement of Structured Information Standards*) et le DMTF ont proposé des standards d'administration basés sur ces nouvelles technologies et concepts. Dans cette section nous présentons WSDM (pour *Web Services Distributed Management*) [WSD05a, WSD05b], l'un de ces travaux. WSDM est un standard OASIS qui a été validé en mars 2005. L'objectif de ce travail est l'utilisation des services web pour exposer des informations de gestion sur des ressources, pouvant elles-mêmes être des services web. Il spécifie deux standards d'administration liés aux services web :

- **L'administration à base de services web** (*MUWS*, pour *Management Using Web Services*) [WSD05a]. Cette spécification définit comment administrer des ressources à l'aide de Web Services. Cette spécification couvre les mécanismes de base permettant d'interagir avec ces ressources selon le schéma *publier-découvrir-interagir*.
- **L'administration des services web** (*MOWS* pour *Management Of Web Services*) [WSD05b]. Cette spécification définit un modèle pour l'administration des Web Services eux-mêmes (et non des ressources qu'ils représentent). MOWS repose sur MUWS pour définir un modèle dédié à l'administration des Web Services.

Étant donné qu'on s'intéresse aux différentes approches génériques d'administration, et qu'on ne s'intéresse pas à l'administration d'une ressource particulière telle que les services web, nous proposons dans la section suivante de développer la proposition du WSDM concernant l'administration à base de services web MUWS. Nous recommandons la référence suivante [WSD05b] aux lecteurs souhaitant avoir plus de détails sur l'administration des services web.

2.3.4.1 Architecture

L'architecture de MUWS propose un modèle d'administration qui s'appuie sur une architecture conforme à l'approche SOA (pour *Architecture Orientée Services*). SOA définit un modèle d'interaction applicative mettant en œuvre des connexions en couplage lâche entre divers composants d'une application. Cette approche architecturale a été propulsée par le succès des services web. D'ailleurs, tout comme les services web, MUWS repose sur les standards classiques des Web Services - XML, WSDL, UDDI - permettant d'invoquer des services selon le schéma "*publier-découvrir -interagir*". La publication consiste à publier dans un registre (*registry* ou *repository*) les services disponibles aux utilisateurs, tandis que la notion de découverte recouvre la possibilité de rechercher un service parmi ceux qui ont été publiés. Le principal standard utilisé est UDDI (*Universal Description Discovery and Integration*), normalisé par l'OASIS. La figure 2.10, extraite de la spécification de MUWS, illustre l'architecture globale d'une infrastructure MUWS.

A ce titre, une ressource administrable (`Mangeable Ressource`) est décrite par une interface (*Web service endpoint*). Cette interface contient l'ensemble des opérations d'administration disponibles. Elle est publiée par un fournisseur dans un annuaire de fa-

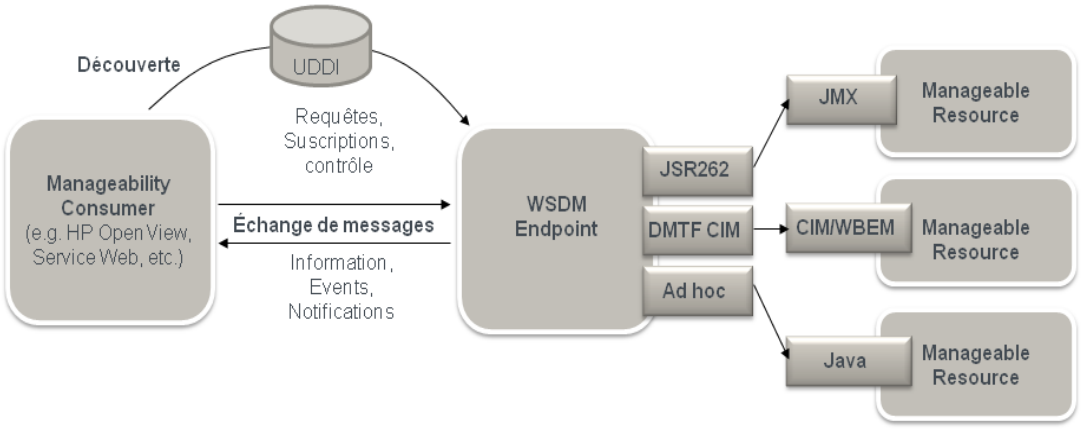


FIG. 2.10 – MUWS : Architecture de gestion orientée services

çon à ce que des clients puissent la localiser afin d'interagir. Selon le modèle SOA, le `ManageabilityConsumer`, qui joue le rôle du gestionnaire selon le modèle gestionnaire-agent, découvre l'ensemble des `ManageableResources` disponibles dans le registre des services. Ensuite, il interagit avec chaque `ManageableResource` en échangeant des messages (notifications, des commandes de contrôle, des requêtes, etc.).

2.3.4.2 Synthèse

WSDM est un standard qui s'appuie sur les architectures orientées services et propose une architecture d'administration entièrement basée sur les services web. L'idée maîtresse de WSDM est d'utiliser les services comme un moyen pour exposer des informations de gestion sur des ressources, pouvant elles-mêmes être des services web. Cette approche procure aux applications d'administration une indépendance vis-à-vis de la technologie d'administration sous-jacente. Cette facilité permet l'exploitation conjointe de composants logiciels hétérogènes.

Le couplage lâche, l'une des caractéristiques principales de ce type d'architecture, permet de rajouter ou d'extraire des agents à tout moment durant le processus d'administration. Le MUWS ne vise pas à remplacer les technologies d'administration existantes (SNMP, JMX, CIM/WBEM, etc.) mais plutôt de servir de passerelle vers celles-ci dans le cadre d'une architecture d'intégration. Comme illustré dans la figure 2.10, l'objectif est d'offrir aux applications d'administration une interopérabilité orientée services avec les systèmes administrés, et une neutralité en termes de plateformes et de technologie d'administration sous-jacente.

2.3.5 Conclusion sur les approches standardisées

Dans cette section nous avons fait remarquer que les standards actuels, bien qu'indispensables à la gestion des équipements et à l'interconnexion des réseaux, n'apportent pas encore la solution idéale attendue par les utilisateurs. Les procédures de normalisation elles-mêmes sont remises en cause, car, étant beaucoup trop longues, elles amènent

les constructeurs à proposer des solutions de gestion propriétaires lorsqu'il s'agit de gérer de nouveaux types d'équipements [JLM99]. Ces solutions posent ensuite d'importants problèmes pour leur intégration dans les plates-formes de gestion. Un autre problème est lié à la conception manager/agent décrite par les standards et plus particulièrement au rôle que joue l'agent. Toutes les approches présupposent que l'agent n'est là que pour renvoyer des informations de gestion au système qui lui, au contraire, est chargé de gérer le réseau et par conséquent présupposé avoir toute la connaissance de la gestion. L'utilisation d'une telle conception interdit à l'agent la possibilité de prendre des initiatives pour corriger la cause d'une alarme qu'il a détectée. Plus généralement, les standards de gestion de réseaux proposent des solutions :

- qui sont très sensibles au facteur d'échelle, la principale raison étant leur approche manager/agents.
- qui présentent des systèmes d'informations trop complexes (OSI, CIM) ou trop spécifiques aux équipements et constructeurs (MIBs SNMP).
- en décalage avec la réalité du marché. Les délais de normalisation sont incompatibles avec la demande du marché, et de plus ces standards ne sont pas toujours suivis.
- qui ne sont que partiellement implémentés. Les frameworks n'implémentent qu'une partie des standards et diminuent de ce fait leur interopérabilité, et donc l'intérêt des standards.

Après avoir étudié les standards d'administration, nous dressons, dans la section suivante, un panorama des systèmes d'administration les plus connus sur le marché.

2.4 Les systèmes classiques d'administration

Il existe principalement deux grandes familles d'infrastructures de gestion. La première correspond à une conception ad hoc de solutions pour résoudre un problème spécifiques. Elle comprend, dans la plupart du temps, des scripts Shell (langage de script pour les systèmes Linux) ou des scripts Perl. Ces scripts représentent des solutions d'administration temporaires. Ils peuvent suffire dans certains cas, dans le cadre de parcs ou de systèmes de petite taille (une dizaine d'entités). Toutefois, quand le nombre des ressources gérées devient considérable et que leur hétérogénéité croit. Les scripts peuvent devenir très complexes et difficile à maintenir et à faire évoluer. Pour pallier cette lacune, la deuxième famille des infrastructures de gestion propose des systèmes avec une conception plus lourde basée sur les standards d'administration. Dans cette section, nous nous focalisons sur la deuxième catégorie d'applications de gestion.

2.4.1 IBM Tivoli

Tivoli [Tiv] est le nom d'une gamme de logiciels d'administration éditée par IBM. A l'origine, ces produits étaient développés, depuis 1989, par l'éditeur américain Tivoli Systems, devenu depuis 1996, la branche d'IBM dédiée aux logiciels d'administration de réseaux, de systèmes et d'applications. Dans les sections suivantes, nous détaillons l'architecture d'un système de supervision IBM Tivoli (version 6.X) [GBS+05] et nous présentons deux exemples de passage à l'échelle avec cette infrastructure.

2.4.1.1 Architecture

Un environnement de supervision IBM Tivoli est constitué d'un ensemble de logiciels paramétrés et configurés pour offrir un service de gestion cohérent. La figure 2.11 illustre les composants typiques d'une infrastructure IBM Tivoli. Les principaux composants de cette infrastructure peuvent être classés en quatre parties :

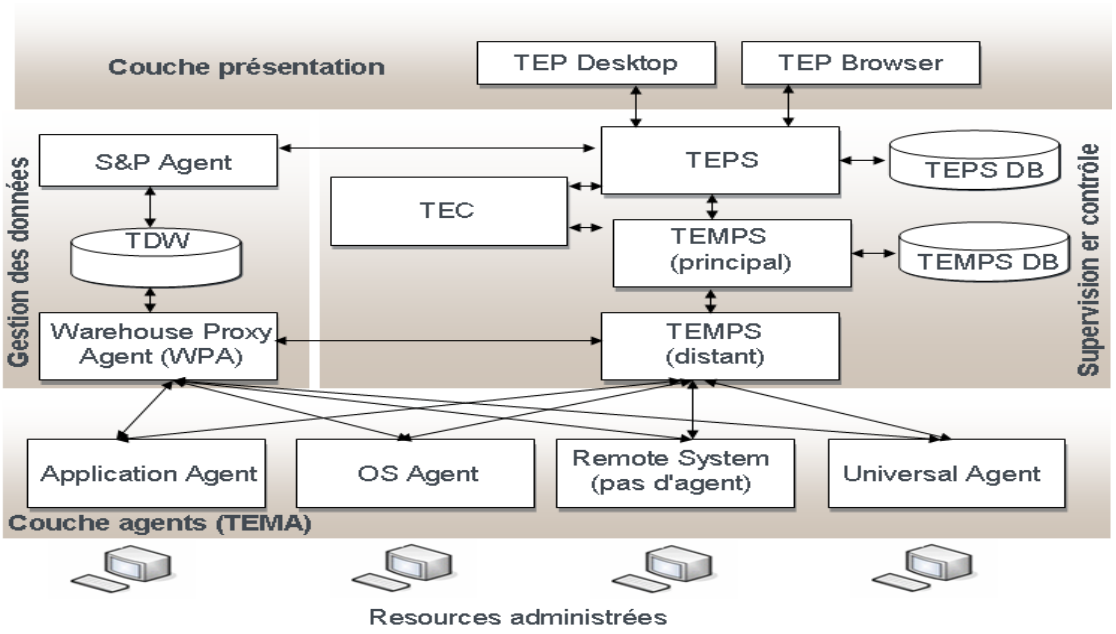


FIG. 2.11 – Architecture d'IBM Tivoli [GBS+05]

La couche présentation. Cette couche regroupe principalement des clients de type *Tivoli Enterprise Portal (TEP) Browser / Desktop*. Un TEP est un client Java qui établit une connexion entre un TEPS Tivoli *Tivoli Enterprise Portal Server (TEPS)* et les consoles d'administration. Le TEPS est responsable de l'affichage et de la présentation des données de supervision. Il requiert l'installation d'un serveur web propriétaire IBM pour afficher les pages web présentant les données de supervision. Il établit une connexion permanente avec le TEMS afin de disposer des données de surveillance. De plus, il requiert une connexion à l'entrepôt de données de IBM Tivoli (*Tivoli Data Warehouse*) pour accéder aux données historiques liées aux ressources administrées.

La couche supervision et contrôle. Cette zone de l'infrastructure Tivoli est primordiale pour la collecte des données de gestion. Elle inclut au moins un TEMS (*Tivoli Enterprise Monitoring Server*) et optionnellement un TEC (*Tivoli Enterprise Console event synchronization*). Le TEC est le composant qui assure la fonction de gestion des évènements (synchronisation, publication, etc.). Quant au TEMPS, il constitue l'épine dorsale de l'architecture d'IBM Tivoli. Il représente le point de collecte et de contrôle des informations de gestion envoyés par les agents. Il assure le suivi des notifications de présence des agents qui lui sont connectés. Il permet également de stoker *les règles de gestion*. Un TEMPS peut coexister avec d'autres de ses semblables. Des TEMS distants peuvent être rajoutés pour faciliter le passage à l'échelle.

Gestion des données. Cette partie de l'architecture englobe les composants responsables de la gestion de la persistance et de l'archivage des données de gestion. Le composant TDW *Tivoli Data Warehouse* est le composant moteur de ce module. Il représente l'entrepôt de données dans lequel sont sauvegardées toutes les données. Se rajoutent à ce composant deux agents, un agent WPA (*Warehouse Proxy agent*) et un agent S&P (*Warehouse Summarization and Pruning agent*). Un WPA a pour mission la collecte et la consolidation des informations de gestion et les transférer à l'entrepôt de données, alors qu'un agent S&P s'occupe de l'agrégation et la purification des données de gestion. Il est recommandé d'installer l'agent S&P sur la même machine que l'entrepôt de données à cause de l'éventuel volume important de données à échanger et à traiter.

La couche agents TEMA. Les agents TEMA (*Tivoli Enterprise Management Agent*) sont installés au niveau des systèmes administrés. Ils assurent la collecte des données de gestion et initient la notification du serveur d'administration pour indiquer leur présence. Ils communiquent directement avec le TEMS mais peuvent également, si l'administrateur le souhaite, communiquer directement les données de supervision à l'agent Warehouse Proxy. Transférer les donnés de supervision d'un TEMPS distant à l'entrepôt de données se fait rarement et est conseillé par IBM comme un dernier recours.

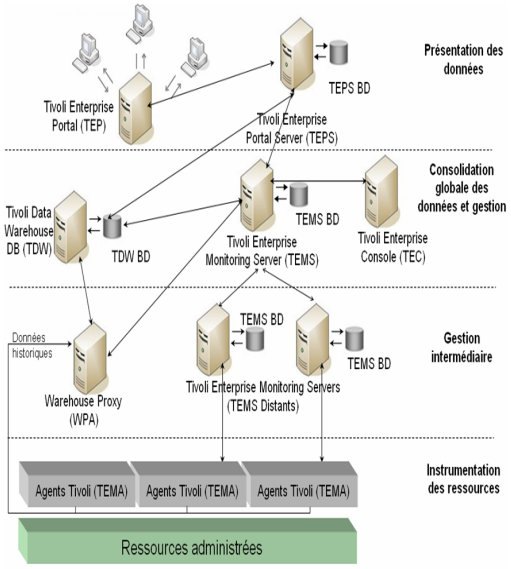
Il existe plusieurs types d'agents TEMA. On peut trouver des agents spécialisés par systèmes d'exploitation (Windows, Linux, Unix, etc.) ou des agents applicatifs (DB2 Agent, Oracle Agent, MS SQL Agent, etc.). Tivoli propose aussi des agents d'intégration (adaptateurs) qui permettent de récupérer des données de gestion à partir de versions antérieures d'agents IBM Tivoli. Il existe également les agents dits "universels", qui permettent à des stations de gestion Tivoli de communiquer avec des systèmes patrimoniaux.

L'assemblage de ces trois modules forme l'environnement de supervision IBM Tivoli. Cet environnement est généralement recommandé pour des petites et moyennes installations (environ 250 ressources administrées). Pour de telles configurations IBM préconise deux agents par ressource administrée, à savoir 500 agents. Nous allons voir dans la section suivante si on peut dépasser ce chiffre.

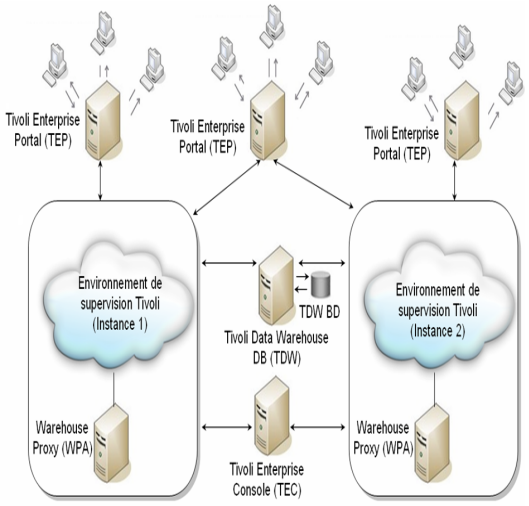
2.4.1.2 Passage à l'échelle

Dans cette section, nous présentons deux exemples de passage à l'échelle de l'infrastructure Tivoli, avec respectivement 1500 puis 3000 ressources administrées. Dans les deux scénarios présentés, nous considérons qu'une instance de l'environnement Tivoli est composée des éléments suivants :

- un niveau d'instrumentation représenté par les agents Tivoli,
- un niveau de consolidation intermédiaire des informations de gestion représenté par les TEMS distants,
- un niveau de consolidation global représenté par le TEMS,
- un niveau d'archivage et d'historisation des informations de gestion représenté par le serveur TDW,
- un niveau de gestion des évènements représenté par le TMS/TEC.



(a) Gestion d'un parc de moins de 1500 ressources



(b) Gestion d'un parc de 3000 ressources

FIG. 2.12 – IBM Tivoli : Architecture de gestion à large échelle

2.4.1.3 Gestion d'un parc de moins de 1500 ressources

. Un environnement d'administration Tivoli peut supporter la gestion de parcs allant jusqu'à 1500 unités administrées. La figure 2.14 décrit le modèle globale de déploiement dans de tels scénarios. L'interconnexion d'un TEMS principal et des TEMS distants produit une topologie d'administration hiérarchique. Les TEMS distants interagissent localement avec leurs agents, et diffusent ensuite les informations collectées au TEMS principal. Ce mécanisme permet d'avoir une vue synoptique de l'ensemble du réseau. Cette vue sera par la suite transmise au composant TEPS.

Pour réussir une installation large échelle de Tivoli, IBM recommande :

- une sélection rigoureuse des attributs pertinents des ressources à administrer,
- une installation de 500 agents (chiffre maximum recommandé par un serveur d'administration IBM Tivoli) et de 10 TEMs,
- de faire très attention à la sélection de l'infrastructure matérielle et réseau (e.g. topologie, latence, pare-feu, etc.). Une installation large échelle requiert une infrastructure matérielle plus lourde que celle requise pour une petite ou moyenne échelle.
- un déploiement sur des machines séparées des composants TDW et WPAgent, responsables respectivement de la collecte des données de gestion et leur archivage.

2.4.1.4 Gestion d'un parc de 3000 ressources

Une installation Tivoli pour gérer 3000 équipements requiert en moyenne 8000 agents. La figure 2.12(b) illustre les principaux composants ainsi que l'architecture globale de déploiement de IBM Tivoli dans tels scénarios. Nous pouvons distinguer dans la figure deux instances Tivoli (chacune de 1500 ressources). Le lien entre les deux instances est assuré par composant d'archivage TDW (entrepôt de données mutualisé) et par le module

de consolidation des évènements TMR/TEC. IBM recommande le partage de ces deux modules dans un contexte très large échelle.

2.4.1.5 Synthèse

IBM Tivoli est l'un des systèmes d'administration les plus complets sur le marché. Nous avons présenté dans cette section une des dernières versions de ce produit. Le produit dispose d'une architecture fonctionnelle modulaire qui distingue bien les composants de traitement de ceux de la présentation et de l'archivage. Concernant le passage à l'échelle, le produit donne le choix à l'administrateur de choisir le modèle de gestion (une gestion centralisé, centralisée hiérarchique et complètement distribuée) qui lui convient le mieux. Nous avons présenté dans cette section ces possibilités de passage à l'échelle à travers deux exemples de 1500 et de 3000 ressources administrées. Néanmoins, les principales limitations de Tivoli sont certainement le coût onéreux de ce produit et la difficulté d'extension par les utilisateurs. De point de vue de l'éditeur de la solution, les principales difficultés sont la complexité des API du produit et la difficulté de l'administration de l'outil lui-même au cours de l'exécution. De plus, Tivoli, ne propose pas d'outils pour s'administrer lui-même.

Afin d'avoir une vue plus complète sur le marché des outils d'administration, nous présentons dans la section suivante un autre classique des systèmes d'administration.

2.4.2 HP-OpenView

HP OpenView [Ope] représente une famille de produits intégrés pouvant être assemblés pour répondre à des besoins d'administration de bout en bout : depuis les réseaux, les systèmes et les postes de travail, jusqu'aux middlewares et aux applications d'entreprise. Comme nous ne pouvons pas présenter l'ensemble de ces produits dans le cadre de ce manuscrit, nous proposons dans les sections suivantes une étude de l'architecture du produit *Network Node Manager (HP NNM)* et de ses capacités de passage à l'échelle.

2.4.2.1 Architecture

L'architecture simplifiée de système NNM, résumé dans la figure 2.13, s'articule principalement autour de deux entités :

- **La station d'administration** qui joue le rôle du gestionnaire (selon le modèle gestionnaire-agent). Elle facilite l'accès aux fonctionnalités d'administration en permettant l'accès à une ou plusieurs consoles d'administration. Une station d'administration peut accueillir jusqu'à 15 consoles d'administration.
- **La station de collecte** qui joue le rôle d'un agent, de point de vue de la station d'administration avec laquelle elle communique. Cependant, elle peut également être considérée comme une station d'administration. Elle assure principalement une fonction de collecte de données : la supervision des statuts du réseau et des ressources administrées, cartographie du réseau, corrélation des évènements locaux, interception et transfert des évènements à une ou plusieurs stations d'administration. Pour pourvoir jouer mener à bien ces tâches, une station de collecte possède une architecture interne similaire à celle d'une station d'administration.

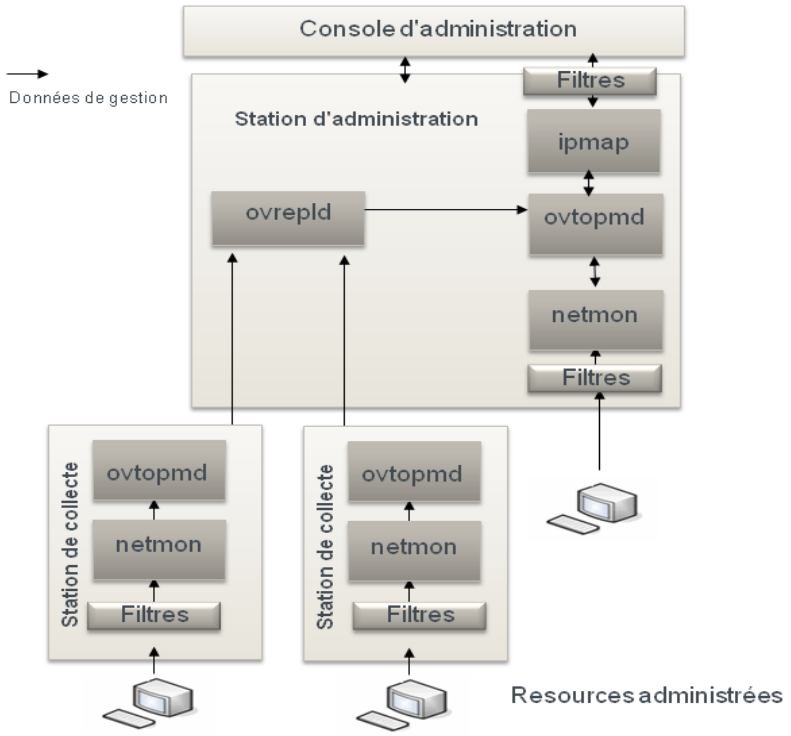


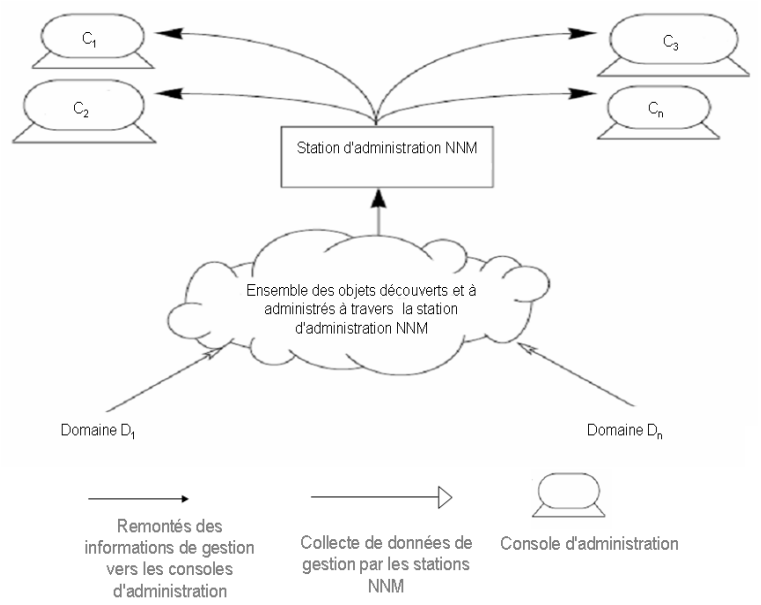
FIG. 2.13 – Architecture distribuée et services de HP NNM

De point de vue de l'architecture interne, une station d'administration se compose principalement de trois modules. Le premier module est *netmon*. Il assure la fonction d'extraction initiale de la topologie du réseau et des ressources administrées. Il assure également la supervision des ressources administrées (interrogation régulière des ressources, suivis de leurs statuts, etc.). Pour ce faire, il se base sur les protocoles SNMP et/ou ICMP. Ces protocoles sont utilisés respectivement pour l'extraction des informations de gestion et le suivi de certaines informations liées à la gestion de configurations. *Netmon* interagit directement avec *ovtopmd* (*OpenView Topology Manager Daemon*), un autre module interne à la station d'administration qui maintient une base de données locale de la topologie du réseau et la rafraîchit grâce aux informations communiquées par le service *netmon*. Le module *ovtopmd* utilise également des informations en provenance du service *overpld*. Ce dernier assure la construction de la cartographie globale du réseau à partir des fragments de identifiés par les stations de collecte. Il lit les informations liées à la topologie du réseau qui sont fournis par le service *ovtopmd* et il se met à l'écoute aux évènements liés aux changements de la topologie. Toutes les informations collectées sont présentées au service *ipmap* pour être accessibles par les consoles des administrateurs.

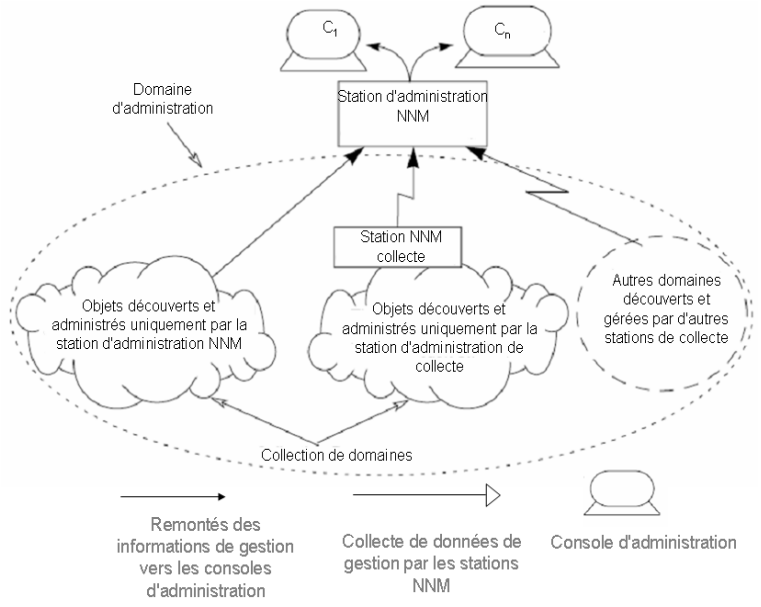
Après avoir étudié l'anatomie de l'infrastructure NMM, nous nous focalisons sur ses capacités de passage à l'échelle. Pour ce faire nous allons étudier les techniques de passage à l'échelle [JCD04] mises en œuvre avec NNM.

2.4.2.2 Les stratégies d'administration

HP NNM propose quatre stratégies d'administration résumées les figures 2.14(a),2.14(b),2.15 et 2.16. Le choix de la stratégie de gestion adéquate est primordial pour le passage à l'échelle.



(a) Startégie d'administration centralisée



(b) Startégie d'administration centralisée hiérarchique

FIG. 2.14 – HP OpenView : Strarégies d'administration centralisée

La stratégie d'administration centralisée. Ce modèle est le modèle d'administration le plus répandu. La figure 2.14(a) illustre les principaux éléments d'une stratégie d'administration centralisée. Il est souvent utilisé pour des parcs de petite ou moyenne taille. Selon ce modèle, une seule et unique station d'administration pilote, directe-

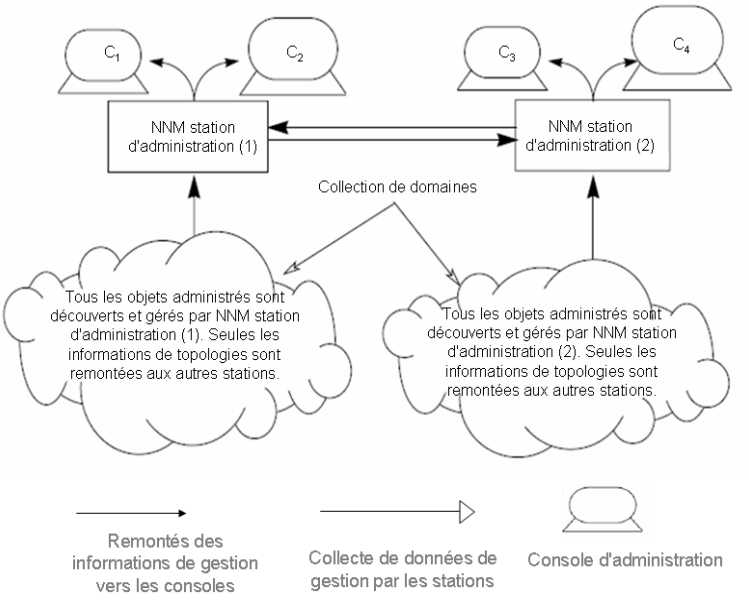


FIG. 2.15 – Startégie d'administration répartie

ment, un ensemble de ressources administrées. Dans la figure 2.14(a) cette station gère les domaines D1 et D2. Elle permet d'avoir une ou plusieurs consoles d'administration.

La stratégie d'administration centralisée hiérarchique. Ce modèle consiste à déployer d'une manière hiérarchique une station d'administration et un ensemble de stations de collecte. (voir figure 2.14(b)). Selon ce modèle, seule la station d'administration est capable d'offrir aux administrateurs la possibilité d'effectuer des opérations de supervision ou de contrôle. Avec ce modèle, on peut atteindre jusqu'à 25000 équipements et une quinzaine de consoles d'administration [JCD04].

La stratégie d'administration hiérarchique. Ce modèle est souvent appliqué dans des contextes à large échelle. Il est similaire au modèle centralisé hiérarchique de point de vue déploiement. Toutefois, dans ce modèle l'autorité d'administration n'est pas centralisée au niveau de la station d'administration principale (le haut de la hiérarchie des stations d'administration). Sur la figure 2.16 chaque station de collecte (Station NNM hybride (1) et (3)) joue un double rôle. Elle permet, d'une part, la collecte des informations de gestion et, d'autre part, aux administrateurs locaux d'effectuer des opérations de gestion au niveau des domaines de gestion sous leur responsabilité.

La stratégie d'administration répartie. Cette stratégie est recommandée dans le cas où deux ou plusieurs sites complètement indépendants et généralement géographiquement répartis requièrent une interaction afin d'échanger des informations de gestion. L'administration est faite localement au niveau de chaque site. La figure 2.15 illustre un exemple d'architecture de gestion répartie. Les deux stations de gestion NNM (1) et NNM (2) communiquent ensemble afin d'échanger des informations de gestion (objets découverts, requêtes de gestion, opération de mise à jour, etc.).

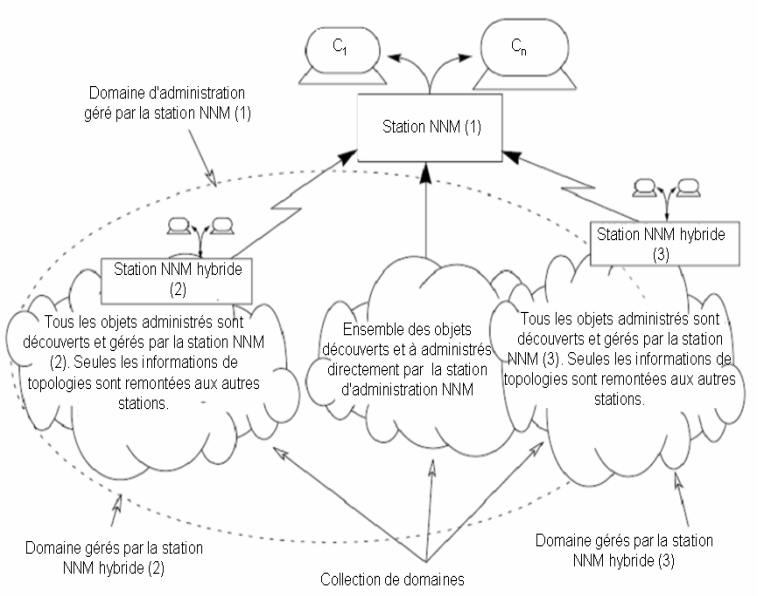


FIG. 2.16 – Startégie d'administration hiérarchique

2.4.2.3 Les techniques de filtrage

Les filtres sont applicables à différents niveaux de l'infrastructure d'administration. Ils peuvent être appliqués par exemple au niveau du processus de la découverte des ressources pour limiter les objets rajoutés à la base de données et à mettre à jour. Appliquer des filtres à ce niveau permet également de limiter le volume des informations échangées entre les stations de collecte et les stations d'administration. Globalement, NNM propose deux catégories de filtres :

- **Le filtrage sur les flux de données**. Selon cette approche, des filtres sont appliqués sur les flux d'information de gestion qui transitent entre le système de gestion et les systèmes administrés. Ces filtres ne laissent passer que les informations souhaitées. Le reste des informations est rejeté.
- **Le filtrage sur les ensembles**. Cette approche consiste à appliquer des filtres sur un ensemble d'objets candidats selon des critères bien déterminés. Contrairement à l'approche précédente, le changement des attributs (ou critères discriminatoires) de filtrage requiert une réévaluation de l'ensemble des objets retenus par le filtre.

2.4.2.4 Le découpage en domaines

NNM définit un domaine comme l'ensemble d'objets gérés pertinents pour l'administrateur. Dans la figure 2.16, le domaine d'administration géré par l'administrateur utilisant les consoles C1 et C2 est l'ensemble de tous les objets du réseau. Par contre, le domaine d'administration de l'administrateur utilisant la console connectée à la station NNM (2) est limité aux ressources visibles et administrées par cette station. NNM fait la distinction entre le *domaine d'administration* comme étant l'ensemble des ressources administrables et la *collection du domaine* qui représente l'ensemble des ressources découvertes (objets et

ressources pertinentes pour l'administrateur) et pertinentes pour l'administrateur.

2.4.2.5 La technique de cache

Afin d'éviter la saturation des ressources mémoires et CPU, HP NNM a prévu un mécanisme pour pallier le phénomène d'engorgement des représentations des ressources en mémoire. Avec le service *ipmap*, NNM crée une hiérarchie de "cartes" du réseau. Les cartes les plus fréquemment utilisées persistent en mémoire alors que les autres sont chargées suite aux demandes des administrateurs ou suite à des requêtes d'administration qui les concernent. L'un des avantages de cette approche est le gain de performances considérable, surtout lors de mise à jour des synchronisations des informations liées à la topologie.

2.4.2.6 Synthèse

HP Open View est l'une des infrastructures d'administration les plus avancées du marché des systèmes d'administration. Grace à une architecture modulaire et des composants fonctionnels d'administration bien identifiés, HP OpenView interconnecte ses modules pour construire un système d'administration adaptés aux petits ou aux grands réseaux. HP Open View a prévu le support de plusieurs stratégies d'administration (centralisé, centralisée hiérarchique, complètement distribuée). Il a prévu également des mécanismes de filtrage applicables à différents niveaux de son infrastructure ce qui facilite la prise en compte d'un grand nombre de ressources gérées et d'évènements. Quant aux systèmes cibles, HPOpenView couvre aussi bien la gestion du réseau, que des équipements ou des services. NNM, l'outil qu'on a étudié dans cette section, représente le module qui s'occupe de la gestion des équipements réseau.

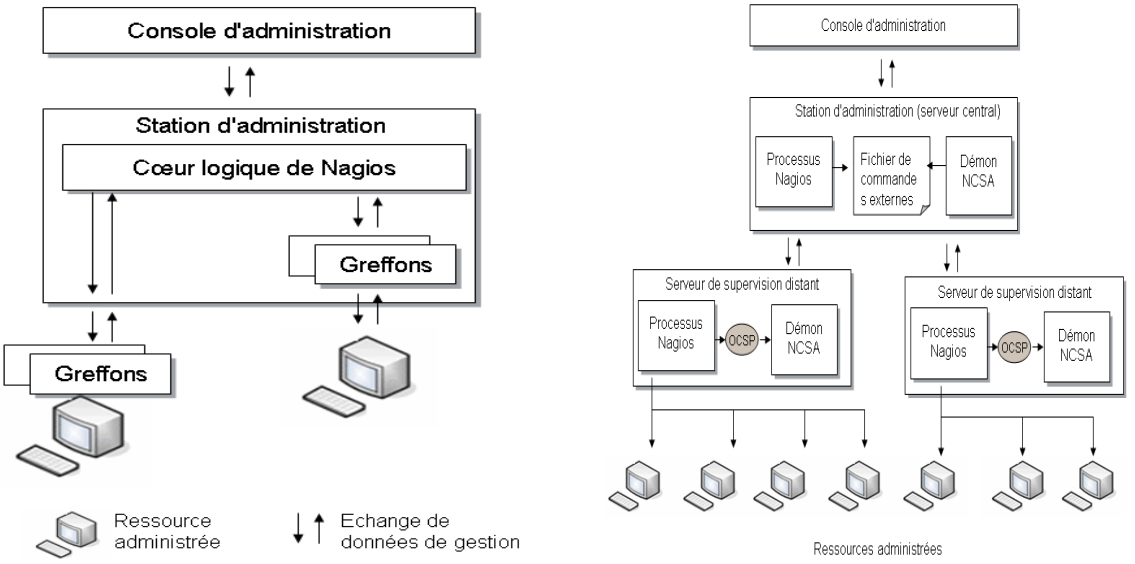
Toutefois, la principale limitation de HP OpenView est qu'il ne supporte pas la gestion multi-échelles. En effet, le support d'un parc de grande taille requiert une configuration matérielle avancée et une configuration logicielle assez minutieuse et personnalisée [JCD04]. De plus, aucun moyen n'est mis en place pour faciliter la supervision et l'observation de la disponibilité de l'infrastructure d'administration elle-même.

2.4.3 Nagios

À côté des grandes plates-formes d'administration telles que HP OpenView [Ope] ou Tivoli[Tiv] d'IBM, dont les coûts de licence sont parfois rédhibitoires, le monde du logiciel libre offre quantité d'outils de supervision. Leur couverture fonctionnelle est certes moins étendue, mais ils répondent aux besoins de la plupart des petites et moyennes structures. Nagios [Nag07], anciennement appelé Netsaint, est un *logiciel libre* de supervision, parmi les logiciels d'administration les plus utilisés par les entreprises. Pérenne - il a été développé en 1999 -, il est soutenu par une communauté très active.

2.4.3.1 Architecture

Nagios permet de superviser un système d'information complet (système et réseau). La figure **??** décrit l'architecture interne de Nagios. Nous pouvons distinguer trois modules



(a) Architecture globale de Nagios (b) Exemple de déploiement distribué avec Nagios

FIG. 2.17 – Nagios : Architecture et déploiement distribué

importants :

- **Un moteur de l'application** dont le rôle consiste à gérer et ordonnancer les tâches supervision ainsi que les actions à prendre sur les éventuels incidents (alertes, escalades, prise d'action corrective),
- **Une interface web** qui permet d'avoir une vue d'ensemble du système d'information et des possibles anomalies. Des vues de l'état des services, hôtes ou matériels réseau sont présentées au travers de cette interface Web, avec un accès aux informations éventuellement filtré selon les droits des utilisateurs.
- **Les greffons** qui sont donc des scripts ou programmes et qui sont la base des vérifications effectuées par Nagios.

A la différence de beaucoup d'autres outils de supervision, Nagios ne dispose pas de mécanismes internes pour vérifier l'état d'un service ou d'un hôte. A la place, il utilise des programmes externes appelés greffons (*plugins*) ou sondes pour réaliser cette tâche. Les greffons sont des scripts ou programmes, pour la plupart écris en *PERL* et en C, qui sont la base des vérifications effectuées par Nagios. Les greffons fonctionnent soit en local sur la machine supervisée, soit à distance. Ils interagissent avec les ressources administrées et renvoient le résultat et éventuellement un message décrivant le déroulement de l'exécution (aide au diagnostic en cas de problème). Nagios analyse le résultat provenant du plugin. Il prend ensuite les mesures nécessaires et décide de lancer par exemple une séquence d'actions programmées (lancer les gestionnaires d'évènements, envoyer des notifications, etc.).

Le projet Nagios fournit par défaut bon nombre de greffons de base élémentaires résumés dans le tableau 2.3 :

TAB. 2.3 – Table greffons Nagios

Greffons	Description
`check disk.c`	permet de tester l'espace disque disponible sur les partitions testées. Basé sur une utilisation de la commande */bin/df*.
`get load average.c`	permet de tester la charge d'un système en temps réel en se basant sur des commandes unix du type *uptime* ou *getloadaverage*.
`check swap.c`	permet de tester le swap disque
`check ping.c`	permet de vérifier qu'un hôte est vivant sur un réseau, basé sur ICMP.
`check udp.c`	permet de tester le bon fonctionnement de la couche transport en se basant sur UDP.
`check mysql.c`	permet de tester la disponibilité d'une base de données *MySQL*.

2.4.3.2 Supervision répartie avec Nagios

Selon la documentation de référence de Nagios [Nag07] « *la plupart des petites et moyennes entreprises n'auront pas réellement besoin de mettre en œuvre cet environnement...* ». Selon la même documentation un déploiement réparti devient indispensable si on souhaite superviser des centaines ou des milliers d'hôtes. L'objectif d'un tel déploiement est de déplacer l'excès de charge induit par les contrôles de services (sur la CPU, etc.) du serveur central de supervision sur un ou plusieurs serveurs répartis. Ces configurations de gestion englobent principalement deux entités :

- **Un serveur central**. Parmi ces tâches on peut citer : l'écoute des résultats des contrôles de service d'un ou de plusieurs serveurs répartis, l'envoi des notifications, l'exécution des scripts de gestionnaires d'événements, la détermination de l'état des hôtes, des envois des données de supervision à l'interface web.
- **Un serveur distant**. C'est une installation simplifiée de Nagios. Le rôle d'un serveur distant est de contrôler tous les services définis pour un groupe de ressources gérées données, en l'occurrence un groupe d'hôtes du réseau. Il n'est pas besoin d'installer l'interface web, d'envoyer des notifications, de faire tourner les scripts de gestionnaires d'événements ou de faire autre chose que l'exécution des contrôles de service.

La figure **??** décrit un scénario de déploiement distribué de Nagios. Chacun des processus Nagios qui s'exécutent au niveau des serveurs distants a en charge un ensemble de machines ou de services. Ces processus exécutent régulièrement la commande `oscp` afin d'effectuer des vérifications des services. Les résultats de la vérification sont par la suite acheminés vers le client `nsca` qui tourne sur l'hôte distant. Le client `nsca` se charge par la suite d'envoyer ces résultats de contrôle de service au serveur central. Au niveau du serveur central, un démon *nsca* qui fonctionne comme un démon autonome écoute les connections des programmes du client. Après avoir reçu l'information de contrôle de service de la part d'un client, le démon envoie l'information de contrôle à Nagios (sur le serveur central) en insérant une commande `PROCESS_SVC_CHECK_RESULT` dans le fichier de commandes externes, avec les résultats du contrôle. Nagios contrôle régulièrement les fichiers de commandes externes. Ces informations de contrôle sont traitées alors par Nagios et transférées à l'interface web d'administration.

2.4.3.3 Synthèse

Nagios est un système d'administration libre très utilisé par les petites et moyennes entreprises. Il représente un bon rapport qualité/prix par rapport aux solutions d'administration qui existent sur le marché. Nagios est simple à installer. Il est généralement déployé pour le suivi de quelques centaines de serveurs. De plus, Nagios est peu gourmand en ressources, il est loin de nécessiter un serveur puissant. Cela permet de tirer parti de vieux serveurs, au lieu de devoir investir dans une nouvelle machine.

Toutefois, Nagios, ne fournit pas, par exemple, de découverte automatique des éléments présents sur le réseau, n'effectue pas de suivi de la charge de ce dernier - il n'est pas prévu pour traiter nativement des informations SNMP -, ou encore ne réalise pas d'historiques. Cependant, il est possible de combler ces lacunes par le biais d'autres logiciels libres, pour constituer une solution « à la carte »répondant aux besoins de l'entreprise. De plus, pour assurer le suivi en temps réel du fonctionnement des services et matériels surveillés, Nagios utilise uniquement trois états : *normal*, *avertissement* et *critique*. Il n'est donc pas possible d'affiner la supervision par des états intermédiaires.

2.4.4 Conclusion : Réalité des plates-formes de gestion

En analysant l'état de l'art des systèmes d'administration des systèmes et réseaux, nous avons constaté plusieurs limitations :

Le coût onéreux des licences. Ce coût est d'autant plus élevé que le parc est important.

Le manque de modularité. La plupart des systèmes d'administration classiques sont monolithiques. Les applications d'administration tels que HP Open view et IBM Tivoli en sont un bon exemple. Ces systèmes sont en effet très souvent le fruit de plusieurs années de travail, d'intégration et de fusion de différents systèmes d'administration avec des philosophies de programmation différentes. Par ailleurs, les applications d'administration ad hoc de type script ne présentent aucune modularité. Ils se présentent comme des morceaux de codes qui sont conçus pour être utilisé dans un contexte particulier et ne sont pas conçus pour être réutilisé.

La sensibilité à l'échelle. Pour la plupart des systèmes d'administration, le passage à l'échelle devient aujourd'hui un facteur de pérennité incontournable, dans un contexte où les systèmes d'administration deviennent de plus en plus volumineux en termes de ressources à administrer, d'utilisateurs et de volumes de données échangés. Les systèmes de gestion les plus aboutis tels que OpenView et Tivoli offrent un support pour le passage à l'échelle à travers le support de différentes stratégies de gestion, des techniques de filtrage des flux de données et des approches de fragmentation des ressources gérées.

L'absence de solution de gestion administrable. Administrer le système d'administration au même titre que les ressources administrées (réseaux, applications, systèmes, etc.) permet de renforcer la robustesse et la qualité du service d'administration. L'administrabilité des infrastructures de gestion peut permettre par exemple de remédier à certaines défaillances en restructurant les modules du système de gestion ou en modifiant son modèle de déploiement. Toutefois, aucune des infrastructures de gestion

étudiées ne fournit des mécanismes de supervision et de contrôle de son propre état et de son comportement.

L'absence des solutions de déploiement partiel. Aujourd'hui, la plupart des systèmes de gestion requièrent un déploiement intégral sur les environnements cibles. Cette approche a plusieurs limitations. Tout d'abord, au moment du déploiement, il en résulte un transfert de données inutiles et effectué à travers le réseau. De plus, au moment de l'exécution, la totalité de l'application de gestion ou de l'agent est chargé en mémoire. Ce genre de déploiement génère donc une utilisation non optimale des ressources de calcul et des ressources réseaux disponibles.

La difficulté de la gestion de l'hétérogénéité des ressources. Aujourd'hui, ils existent deux solutions pour gérer l'hétérogénéité des ressources. La première solution consiste à multiplier et dupliquer des systèmes de gestion différents pour chacune des ressources gérées. Tout d'abord, ce genre de solutions est loin d'être économique (coût des licences, coût de la formation des administrateurs, coût d'exploitation, etc.). De plus, disposer de plusieurs systèmes d'administration implique une complexité de l'outillage et de la maintenance des outils de gestion mis à la disposition des administrateurs. Ce qui nous emmène à une seconde solution, plus élégante qui consiste à appliquer des modèles d'information génériques tels que le modèle CIM. La principale limitation de cette approche est la complexité du modèle CIM et la difficulté de sa prise en main. De plus, il est difficile d'exprimer des relations telles que la hiérarchie ou le partage des ressources, des relations qu'on peut retrouver au niveau des ressources administrées et qu'on souhaite modéliser au niveau du modèle d'information.

2.5 Travaux connexes de recherche

Afin de pallier ces limitations, plusieurs travaux de recherche ont étudié les problématiques soulevées ci-dessus. Nous discutons dans cette section d'administration de l'hétérogénéité, de la flexibilité et la modularité, l'administrabilité, l'autonomie et la gestion de l'échelle.

Administration Autonomique [Jen09] Ce courant de recherche vise l'automatisation des décisions d'administration vient en renfort, voire en remplacement, des administrateurs humains. Plusieurs travaux se sont penchés sur cette question [RM06] et sont inscrits dans un courant recherche baptisé l'*autonomic computing* [OM03]. Ces travaux ont pour ambition d'« injecter de l'intelligence »dans les programmes informatique. Deux techniques sont généralement appliquées. La première technique se base sur des travaux issus de l'intelligence artificielle [RSS, Mar08] pour enrichir une base de connaissances. La seconde technique consiste à rajouter des règles actives ou règles ECA (Evenement, Condition, Action) prédéfinis à l'avance [DBB+08, BJ09]. Par ailleurs, des travaux récents [SY07] se sont penchés sur l'administration autonomique basée sur les web-services afin de pallier à la fois les problématiques d'hétérogénéité et d'administration automatisée.

Passage à l'échelle de l'administration Plusieurs travaux se sont penchés récemment sur la problématique de passage à l'échelle des systèmes d'administration. Certains d'entre ces travaux proposent des architectures hiérarchiques [LGO08] ad-hoc alors que d'autres explorent des pistes similaires mais en se basant sur des standards de gestion par web services [CA09a] pour gérer à la fois le passage à l'échelle et les

éventuels problèmes d'interopérabilité à large échelle. D'autres travaux combinent les web services avec des middlewares orientés messages pour améliorer la capacité de passage à l'échelle des plateformes d'administration [Tal05]. S'ajoutent à ces travaux des travaux qui se basent sur réseau overlay pair-à-pair [Ada07] ou une architecture décentralisée et des approches probabilistes pour optimiser le comportement de la plateforme d'administration et réguler la charge d'administration en optimisant le surcoût d'administration[BM09].

Administration intégrée Avec la convergence des services, l'administration intégrée est au cœur des problématiques à résoudre pour améliorer la qualité de service. Des travaux récent se sont penchés sur la question et ont produit de nouveaux canevas d'administration intégrée pour la gestion réseaux et serveurs d'applications [Wen08], la gestion de bout en bout des services multimédia [CA09b] ou réseaux et terminaux [O.08], ou la gestion intégrée des réseaux de capteurs sans fils [KM09] (gestion de l'infrastructure, fonctionnalités, du firmware,). De nouveaux méta-modèles des informations d'administration ont récemment vu le jour [SM08].

2.6 Conclusion

Dans ce chapitre, nous avons présenté les différents concepts de base de l'administration des systèmes, des réseaux et des applications. Nous avons présenté dans un premier temps les principaux standards qui constituent le paysage système et réseaux sur lequel vient se poser les systèmes d'administration classiques. L'étude de ces travaux nous a révélé un certain nombre de limitations telles que le manque de flexibilité, la difficulté d'extensibilité, la sensibilité à l'échelle.

Dans le chapitre suivant, nous explorons une nouvelle approche construction de systèmes de gestion. Il s'agit de l'approche à composants. Cette approche consiste à construire des systèmes d'administration comme des *puzzle* à base de composants unitaires composables. L'objectif est de construire des systèmes d'administration souples, flexibles, extensibles, modulaires et avec un moindre coût que les systèmes actuels.

Chapitre 3

Les systèmes d'administration à base de composants

Contents

3.1 Les composants : définition et genèse 70
3.2 Composants pour l'administration 70
3.2.1 Administration à base de composants 70
3.2.2 Administration fondée sur l'architecture des systèmes administrés 71
3.2.3 Administration à base de composants et fondée sur l'architecture 72
3.3 Administration à base de composants 72
3.3.1 Processus de fabrication des systèmes de gestion à composants . 72
3.3.2 Liaison 73
3.3.3 RMTool 75
3.3.4 CBDM 78
3.4 Administration fondée sur l'architecture 80
3.4.1 Rainbow 80
3.4.2 SmartFrog 82
3.5 Administration à base de composants et orientée architecture 85
3.5.1 JADE 85
3.5.2 GridKit 87
3.6 Composants et administration intégrée 89
3.6.1 Discussion 91
3.7 Discussion 92
3.8 Conclusion 93

Ce chapitre introduit la notion de composant logiciel (section 3.1). Il passe en revue l'essentiel des travaux qui s'appuient sur les composants pour répondre à des besoins d'administration des systèmes et réseaux (sections 3.2 à 3.5) en les classifiant en 4 catégories. Il enchaîne avec une mise en exergue l'enjeu que représente le paradigme de composants pour la gestion intégrée à large échelle (section 3.6) pour terminer avec une discussion sur les différents types de travaux analysés (sections 3.6, 3.7).

3.1 Les composants : définition et genèse

Le concept de composant logiciel est apparu pour la première fois dans les années soixante avec *Malcolm Douglas Mcllroy* [Mcl68]. A cette époque, l'auteur a décrit sa vision d'un marché de composants logiciels réutilisables et facilement combinables afin de construire rapidement des systèmes complexes.

L'objectif principal de la programmation par composants est de faciliter la réutilisation de code sous la forme de composants logiciels génériques. Elle devrait permettre de construire une application comme *un puzzle* dont les pièces seraient des unités sujettes à composition par une tierce personne. Grâce à une description explicite des services ouverts (interfaces) et requis (dépendances contextuelles), de tels composants peuvent être plus facilement distribuées, puis intégrées dans d'autres applications. De ce fait, un composant peut être vu comme un fragment de code (bibliothèque, classe, canevas, etc.) associé à des méta-données qui explicitent (sous forme de contrats) la manière dont il s'utilise, se configure, s'étend, etc. Plus formellement, un composant peut être défini comme :

Définition 3.1. *Un composant est une unité de composition dont les interfaces et les dépendances contextuelles sont spécifiées sous forme de contrats explicites. [Szy97]*

De nombreuses technologies ont été proposées pour réaliser des composants sur étagère. On peut citer pour les modèles industriels : COM (Component Object Model) de Microsoft (avec ses variantes), CCM [DK01] du consortium OMG et les EJBs de Sun [DK06]. Sur le plan académique, on retrouve le modèle à composants Fractal [BCL+06], OpenCOM [BCU+04] ou encore Droplet [Der95].

Bien que le concept de composant remonte à quelques décennies, son application dans le domaine de l'administration est très récente. En effet, pendant plusieurs années la communauté de gestion des systèmes et réseaux a développé des systèmes de gestion ad hoc et monolithiques puis des systèmes basés sur le paradigme *orienté objet*.

Depuis la fin des années 90, certains travaux de recherche ont étudié et appliqué le concept de composant logiciel dans le domaine de la gestion des systèmes et réseaux. Nous dressons dans les sections suivantes un panorama de ces différents travaux.

3.2 Composants pour l'administration

Nous nous focalisons dans cette section sur les travaux qui ont appliqué le concept de composant dans le domaine de l'administration. Nous avons groupé ces travaux en trois catégories : a) les systèmes de gestion à composants, b) les systèmes de gestion fondée sur l'architecture et c) les systèmes à composants pour la gestion fondée sur l'architecture.

3.2.1 Administration à base de composants

Définition 3.2. *Les systèmes de gestion à base de composants sont des systèmes de gestion conçus et surtout développés à base de composants logiciels.*

Les systèmes de gestion à composants s'appuient sur composants pour fabriquer des systèmes de gestion modulaires et faiblement couplés. En plus de cette modularité, ces systèmes sont caractérisés par une aisance d'extensibilité et de maintenance grâce aux capacités d'introspection et aux possibilités de reconfiguration dynamique, proposées par certains modèles à composants.

Ces systèmes ont été appliqués dans plusieurs contextes tels que la gestion des systèmes et réseaux [Der97, JS00, KYM+00], ou la gestion des réseaux de capteurs [JC07]. Le point commun entre ces travaux est l'application des composants pour la conception et l'implémentation des canevas d'administration. Quant à la représentation des ressources administrées elle a conservé un aspect ad hoc ou orienté objet.

Nous présentons dans la section suivante une autre catégorie de travaux qui se focalisent sur l'application des composants pour la représentation des ressources gérées. On parle alors de *gestion fondée sur l'architecture des systèmes administrés.*

3.2.2 Administration fondée sur l'architecture des systèmes administrés

Dans un contexte où les ressources à gérer sont de plus en plus hétérogènes *l'administration fondée sur l'architecture* [DvdHT02] semble être une alternative prometteuse pour la gestion transparente de cette hétérogénéité. Nous définissons ce concept de la manière suivante :

Définition 3.3. *Un système de gestion fondée sur l'architecture est un système de gestion qui s'appuie sur les modèles d'architecture des systèmes administrés comme base pour la construction des fonctions d'administration.*

Selon cette approche, le système d'administration base ses actions sur une connaissance de l'architecture du système administré. L'architecture d'un système administré décrit ce système en tant qu'un assemblage de composants. Cette description se base en général sur un modèle servant à définir l'organisation du système et son fonctionnement. Le modèle d'un système sert à mieux le comprendre, à prédire son comportement et sert également à sa conception et à son implantation. La notion d'architecture des systèmes a été surtout exploitée pour la conception des systèmes et ce n'est que récemment qu'on s'est rendu compte qu'elle pourrait servir de base pour l'administration [Ber06]. La gestion guidée par l'architecture des systèmes administrés est de plus en plus utilisée grâce aux efforts récents consentis par l'OMG (*Object Management Group*) pour développer des architectures dirigées par les modèles (*Model-Driven Architecture*).

Dans le cadre ce travail nous nous intéressons à une catégorie particulière des travaux de gestion fondée sur l'architecture. Il s'agit des systèmes de gestion fondée sur l'architecture qui utilisent les composants logiciels comme moyen pour modéliser l'architecture des systèmes gérés [JMX04, GGL+03, CHSS04]. Ce modèle sert généralement au système d'administration comme représentation intermédiaire des ressources administrées. Avec cette approche, les composants logiciels apparaissent comme une alternative à la technologie orientée objet pour représenter et modéliser ces ressources. Ce choix est justifié par plusieurs raisons. La première raison est la capacité d'abstraction des composants. En effet, les composants offrent une couche d'abstraction permettant de produire une vue homogène à partir d'un ensemble de ressources hétérogènes. La seconde raison est expliquée par les

possibilités d'introspection et d'extension et/ou de reconfiguration dynamique des composants. Ces deux raisons font que les composants sont très appréciés et gagnent du terrain dans le domaine de l'administration et en particulier dans la gestion du déploiement.

Plus récemment, une autre approche de gestion inspirée par les composants a vu le jour. Cette approche combine la gestion à base de composants et celle fondée sur l'architecture des systèmes administrés. Nous présentons cette approche dans le paragraphe suivant.

3.2.3 Administration à base de composants et fondée sur l'architecture

La gestion à base de composants et fondée sur l'architecture des systèmes administrés est une solution hybride qui mélange les deux premières approches de gestion inspirées par les composants. On peut définir cette catégorie de travaux de la façon suivante :

Définition 3.4. *Les systèmes à composants pour la gestion fondée sur l'architecture des systèmes administrés sont des systèmes développés à base de composants et dont l'approche de gestion est guidée par l'architecture des systèmes administrés.*

Cette approche cumule donc à la fois la modularité des systèmes à composants et la capacité d'abstraction des systèmes de gestion fondée sur l'architecture des systèmes administrés.

Le travail de recherche mené autour de la *gestion autonomique*[1] avec Jade [DBB+08], qui sera présenté dans les sections suivantes, est l'un des travaux qui ont adopté cette approche. Un autre travail dans le même cadre a permis d'appliquer cette approche à la gestion de déploiement des serveurs J2EE [Ber06]. Ce travail se présente comme un système d'administration à base de composants qui gère les ressources réelles à travers leurs représentations exprimées en termes de composants Fractal [BCL+06].

3.3 Administration à base de composants

Dans cette section nous mettons l'accent sur les systèmes dont l'architecture interne est conçue et développée à base de composants. Pour commencer, nous présentons un travail de recherche sur l'automatisation de la construction des systèmes d'administration à base de composants [KYM+00]. Nous présentons ensuite le canevas Liaison [Der97], un prototype de gestion de réseaux développé par IBM. Nous introduisons, par la suite, RMTool [JC07] un canevas de gestion des réseaux de capteurs.

3.3.1 Processus de fabrication des systèmes de gestion à composants

Le processus de construction des systèmes de gestion à base de composants peut être résumé en 6 étapes. Kawabata et al. [KYM+00] décrit ce processus (voir figure 3.1). Tout d'abord, il faut construire une bibliothèque de composants (étape 1). Un composant peut représenter un protocole de gestion (SNMP, CMIP, etc.), un protocole de communication

[1] Un système capable se s'auto-défendre, s'auto-configurer, s'auto-réparer et s'auto-optimiser [OM03].

(HTTP par exemple) ou bien une entité de traitement (agent de collecte, filtre, etc.). Ensuite, il faut sélectionner les composants pertinents pour la construction de l'application de gestion (étape 2). L'étape suivante (étape 3) consiste à extraire les spécifications des composants par introspection. Après, il faut assembler les composants pour fabriquer l'outil de gestion (étape 4). L'outil est ensuite sauvegardé (étape 5) et instancié (étape 6).

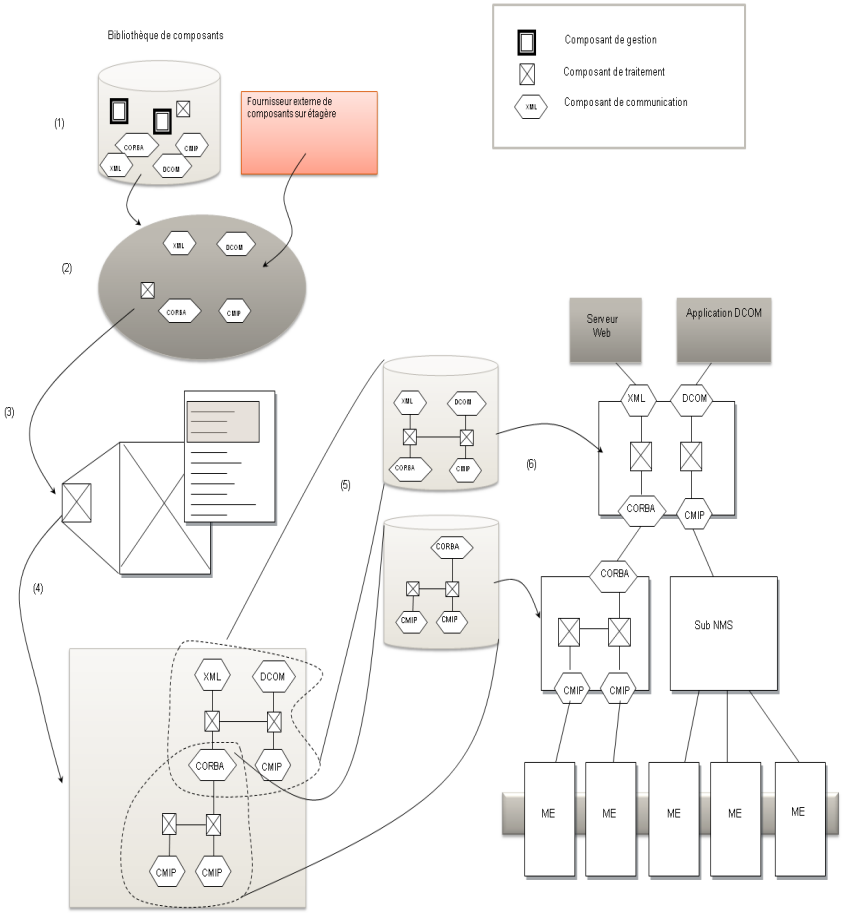


FIG. 3.1 – Processus de construction des applications de gestion à composants (extrait de [KYM+00])

Nous présentons dans les sections suivantes un ensemble de travaux qui ont suivi cette approche pour la construction de système de gestion à base de composants.

3.3.2 Liaison

Liaison [Der97] est un canevas d'administration de réseaux qui a été développé par IBM dans le cadre d'un travail de recherche autour des systèmes d'administration à composants. Ce canevas se base sur un modèle à composants propriétaire IBM nommé Droplet [Der95]. Nous présentons dans cette section ce modèle à composants ainsi que l'architecture d'administration du canevas Liaison.

3.3.2.1 Le modèle à composants Droplet

Le modèle Droplet considère le composant comme une entité :

- capable de supporter le chargement dynamique dans une application durant l'exécution,
- qui peut être remplacée par une autre au cours de l'exécution,
- disposant d'un ensemble d'interfaces lui permettant de communiquer avec ses semblables,
- capable de traiter des requêtes concurrentes.

Afin de mettre en œuvre ces composants, IBM a fourni le canevas Yasmin [Der97]. Ce canevas représente un environnement d'exécution pour les applications à base de composants Droplets. La figure 4.1 présente l'architecture interne du canevas Yasmin (voir le module *Yasmin Kernel services*). Les principaux modules de cette architecture sont résumés dans le tableau 3.1.

TAB. 3.1 – Table services Yasmin

Service Droplet	Description
Event Manager	gère l'échange des messages concernant les Droplets et les services Yasmin.
Collaboration Service	s'occupe de la collaboration entre les Droplets.
Service Manager	joue le rôle d'un annuaire de services. Il permet d'exposer les services fournis par les Droplets.
Resource Manager	surveille le cycle de vie des Droplets et informe le *Droplet Manager* si un composant Droplet expire. Il permet également de gérer les ressources systèmes, telles que les *threads* et la mémoire vive, nécessaires à l'exécution des Droplets.
Droplet Manager	gère principalement les versions des Droplets ainsi que leur chargement et le déchargement. Il collabore avec le *Service Manager* pour le notifier des nouveaux services disponibles.
Le Communication service	permet aux applications à base de Droplets de communiquer avec des entités distantes, la communication interne étant assurée par le mécanisme de gestion d'évènements.

Tous ces services sont destinés aux applications développées à base de composants Droplet. Dans le paragraphe suivant, nous présentons un exemple de ces applications destiné à l'administration des réseaux.

3.3.2.2 Architecture d'administration

Le canevas Liaison est une application de gestion de réseaux développée à base de composants Droplets. Il s'agit d'un canevas fortement modulaire et extensible qui s'appuie sur des composants dynamiquement interchangeables. Ce canevas dispose d'une architecture qui allie modularité et couplage lâche. Cette architecture se décline en deux niveaux : a) le *niveau présentation* et b) le *niveau noyau*. La figure 4.1 illustre l'architecture globale de ce canevas.

Le niveau présentation offre un ensemble d'interfaces de communication orientées CORBA et orientées Web. Ce niveau assure la liaison entre le canevas de gestion et les consoles d'administration. Quant au niveau noyau, il propose une multitude de services de gestion pour

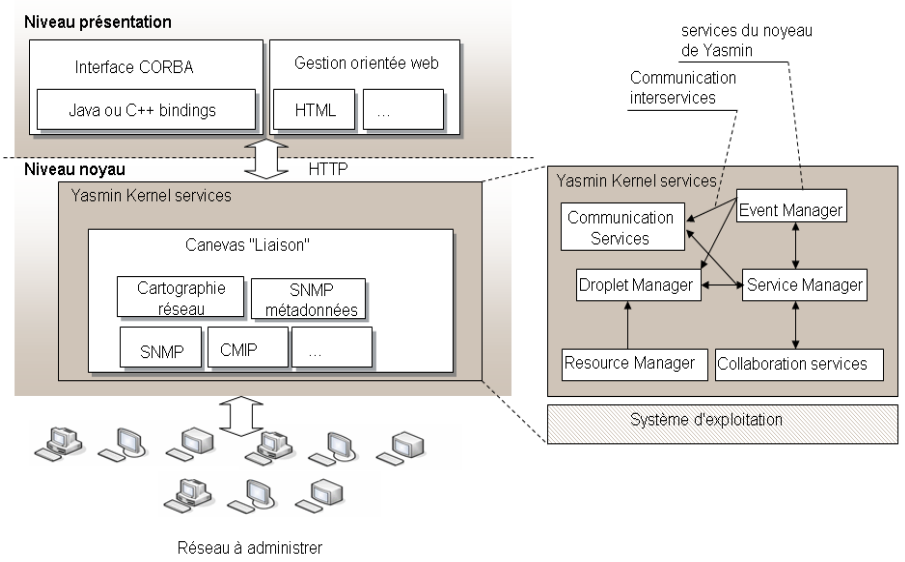


FIG. 3.2 – Architecture globale du canevas Liaison

le support de la gestion des réseaux. Il offre, par exemple un service de cartographie pour localiser l'ensemble des ressources administrées, la gestion des opérations de gestion basées sur les protocoles CMIP ou SNMP et la gestion des métadonnées SNMP. Tous ces composants sont implémentés en tant que composants Droplets.

3.3.2.3 Synthèse

Le canevas Liaison est proposé IBM pour la gestion des réseaux à base de composants Droplets. Il s'agit d'un canevas modulaire et extensible mais qui s'appuie sur un modèle à composants propriétaire et basique. En effet, Droplets ne dispose d'aucun support pour l'assemblage des composants. De plus, aucune facilité d'introspection des composants n'est fournie.

Le point fort de ce travail est la mise en place d'un système d'administration qui supporte le chargement et le déchargement dynamique des modules de gestion. Ce travail propose également une solution à l'**hétérogénéité des protocoles** de gestion en enveloppant les protocoles SNMP et CMIP dans des composants. De point de vue architecture, la séparation entre les composants de présentation, le noyau de gestion et les protocoles de gestion est clairement identifiée.

Toutefois, ce travail n'aborde pas la problématique de gestion de l'hétérogénéité des ressources administrées, ni celle de l'imprévisibilité des ressources administrées ou celle du passage à l'échelle des systèmes d'administration.

3.3.3 RMTool

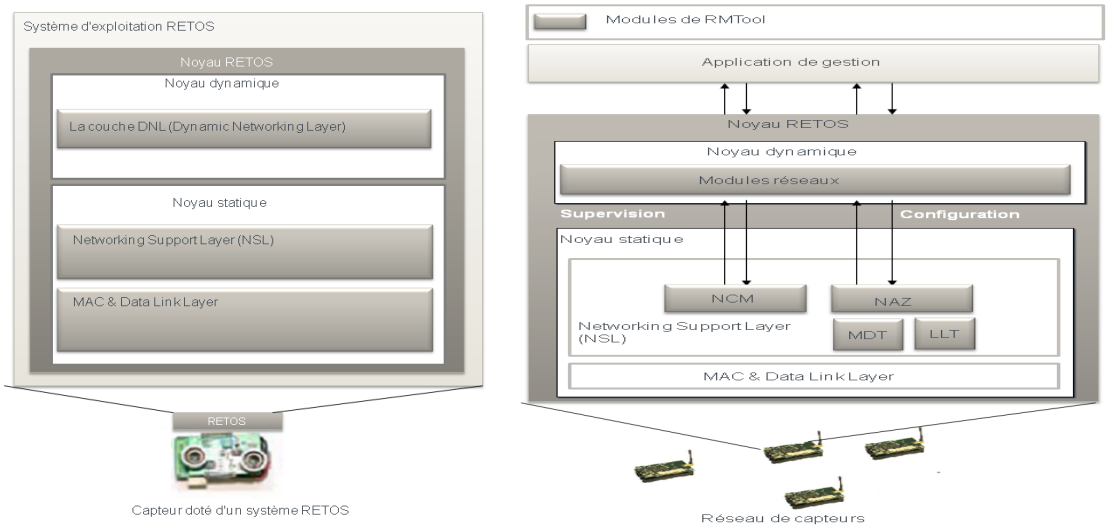
Nous nous intéressons dans cette section à un autre exemple de système de gestion à base de composants dans un contexte très restreint de point de vue des ressources de calcul

mises à la disposition de l'infrastructure de gestion. Cet exemple est également intéressant parce qu'il nous montre cette fois l'utilité des composants dans le domaine des réseaux de capteurs sans fil. Nous présentons dans le reste de cette section le système d'exploitation des capteurs RETOS [CCJ+07] et le canevas de gestion RMTool [JC07].

3.3.3.1 RETOS : Un système d'exploitation modulaire pour les réseaux de capteurs

RETOS est un système d'exploitation pour les réseaux de capteurs sans fil. Il s'agit d'un canevas implémenté en C avec un modèle à composants ad hoc. L'architecture globale de RETOS est illustrée dans la figure 3.3(a). Elle se décline principalement deux couches :

- **Le noyeau dynamique** (*Dynamic Kernel*). Il englobe les bibliothèques communes utilisées par l'ensemble des applications déployées au dessus du système d'exploitation. Ce niveau contient également la couche DNL (*Dynamic Neworking Layer*) qui gère les algorithmes des protocoles de communication et fournit une API permettant aux administrateurs de développer de nouvelles applications.
- **Le noyeau statique** (*Static Kernel*). Cette couche inclut les parties du système qui sont très dépendantes de la couche matérielle. Elle comporte à son tour deux niveaux. Le premier niveau est représenté par la couche NSL (*Neworking Support Layer*). Cette couche gère entre autre une table LLT (*Logical Link Table*) qui joue le rôle d'un modèle d'information statique et qui regroupe l'ensemble des attributs du capteur à observer. La couche NSL gère également une MDT (*Module Description table*) pour décrire et recenser l'ensemble des applications déployées au niveau du capteur (applications au dessus du noyau). Cette table permet à l'administrateur de retrouver des informations telles que le type ou les versions des applications disponibles dans une région spécifique du réseau des capteurs. Le second niveau est représenté par la couche MLL (*MAC& Data Link Layer*). Cette couche permet de contrôler les équipements réseaux et gère les connections physiques et la transmission de paquets de données.



(a) Architecture interne du système RETOS (b) Architecture du RMTool : extension de RETOS

FIG. 3.3 – RMTool et RETOS : Gestion des réseaux de capteurs à base de composants

Afin d'administrer les capteurs dotés d'un système d'exploitation RETOS, il fallait développer un système de gestion à base de composants qui se chargent comme le reste des modules du système RETOS. Ces composants doivent respecter une contrainte majeure à savoir la limitation des ressources de calcul et d'autonomie disponible au niveau du capteur. Face à cette pénurie de ressources de traitement, l'application de gestion doit optimiser sa taille et ses exigences en termes de ressources de calcul. C'est dans cette optique que le système de gestion RMTool [JC07] a été conçu.

3.3.3.2 Architecture

L'outil de gestion RMTool [JC07] est un canevas d'administration pour les réseaux de capteurs, un contexte où les ressources de calcul et de traitement se font rares. Ce canevas a été développé dans le but de gérer des capteurs qui fonctionnent avec le système d'exploitation RETOS. RMTool comporte plusieurs composants déployés au niveau des différentes couches du système RETOS (voir figure 3.3(a)). Les principaux composants de RMTool sont les suivants :

- **Le composant application de gestion**. Ce composant représente l'application d'administration de haut niveau du réseau de capteurs.
- **Les modules réseaux**. Ces composants regroupent l'ensemble des services de gestion que peut offrir un capteur. Ils exposent également des API de gestion pour les applications de haut niveau.
- **Network Analyzer (NAZ)**. Ce composant fait partie du noyau statique de RETOS. Il assure le service de surveillance du réseau et fournit aux développeurs des informations relatives à l'état du réseau. Comme l'illustre la figue 3.3(b) le NAZ interagit avec la table MDT. C'est ce qui lui donne la possibilité de charger et d'échanger des composants représentants des applications de gestion ou des fragments d'applications de gestion. Ce composant permet également aux applications de gestion de changer ou de sélectionner la liste des attributs à gérer en manipulant le *LLT*.
- **Network Configuration Manager (NCM)**. Ce module fait partie du noyau statique de RETOS. Il permet de reconfigurer à la volée les fonctions du noyau ou des applications des capteurs et d'y injecter de nouvelles applications.

3.3.3.3 Synthèse

RMTool est un canevas de gestion à base de composants pour les réseaux de capteurs. Il représente une extension du système d'exploitation RETOS, un système d'exploitation à base de composants pour les capteurs sans fils.

Le point de départ de ce travail est le système d'exploitation RETOS. Ce système est un conçu et développé comme un système modulaire à composants avec des possibilités de chargement dynamique des composants souhaités. Le canevas RMTool vient se greffer sur RETOS comme un ensemble de modules supplémentaires.

Ce travail vise à fournir une infrastructure de gestion à composants capable d'intégrer un environnement d'exécution à faibles ressources de calcul et d'autonomie. Il met également l'accent sur les possibilités d'extension dynamique de l'infrastructure de gestion. RMTool permet par exemple le rajout, à la volé, de nouvelles applications ou modules de

gestion au niveau des capteurs. L'un des points forts de ce travail est le support de l'administrabilité de l'infrastructure de gestion avec la table MDT (*Module Description table*) qui recense l'ensemble des applications déployées au niveau d'un capteur donné.

Toutefois, le canevas de gestion RMTool est un canevas ad hoc, développé pour un besoin précis et non applicable dans d'autres contextes de gestion. De plus, l'approche à composants appliquée dans le cadre de ce travail s'appuie sur des composants ad hoc. Aucun modèle à composants standard ou académique n'a été appliqué pour la conception et l'implémentation de RMTool. Par ailleurs, RMTool traite la question de la gestion de l'hétérogénéité des ressources administrées d'une manière ad hoc en implémentant des adaptateurs pour les capteurs supportés. Quant au passage à l'échelle, RMTool se base sur un modèle de gestion centralisée qui n'est pas adapté à a gestion à large échelle.

3.3.4 CBDM

CBDM (*Component-Based Distributed Network Management*) [JS00] est une approche de gestion des systèmes et réseaux à base de *composants mobiles*. Ce travail combine la mobilité et la composition dans le but de fournir des systèmes de gestion à la fois dynamiques et modulaires. Nous détaillons dans cette section le modèle à composants adopté par ce travail ainsi que l'approche et l'architecture de gestion proposées.

3.3.4.1 Le modèle à composants EJB

Les composants dans le cadre de ce travail sont considérés comme des entités qui :

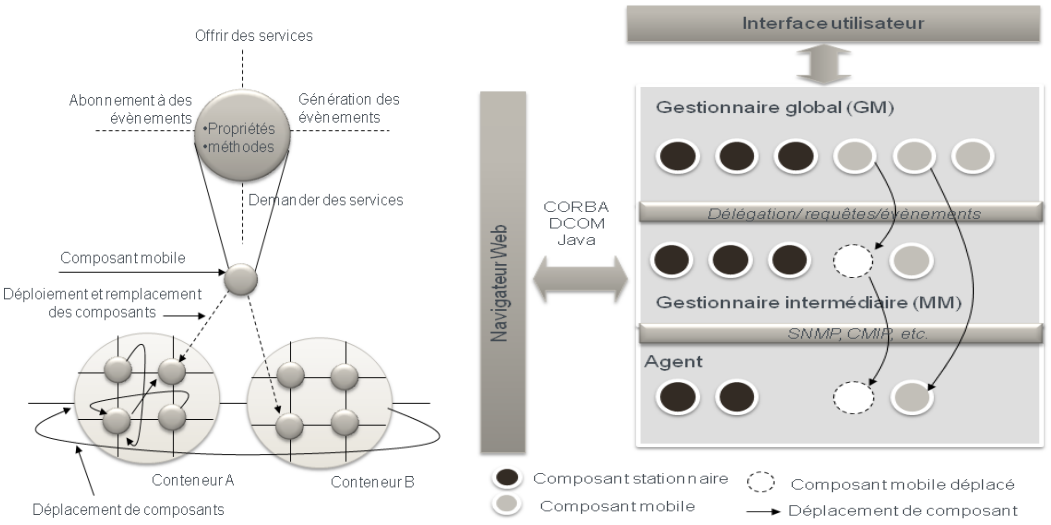
- possèdent des propriétés et des méthodes,
- interagissent avec les autres composants à travers leurs interfaces pour offrir et bénéficier de certains services,
- sont capables de s'abonner à des évènements et d'en produire.

Les EJB (Enterprise JavaBeans) [DK06] est la technologie de composants utilisée pour réaliser cette vision de composant dans le cadre de l'approche CBDM. Les EJB sont des composants Java portables, réutilisables et déployables qui peuvent être assemblés pour créer des applications de type Web. Un composant EJB s'exécute dans un conteneur EJB qui lui fournit des services non fonctionnels tels que les transactions ou la persistance et la sécurité. On distingue trois types de composants EJB :

- *Les EJB de type Session* qui modélisent une session ; c'est-à-dire une suite d'interactions avec les données,
- *Les EJB de type Entity* qui modélisent une entité persistante,
- *Les EJB de type Message-Driven* qui représentent des composants réactifs à des événements. Ils permettent le traitement des messages asynchrones.

3.3.4.2 Component-Based Distributed Network Management (CBDM)

Les composants mobiles sont le résultat de la combinaison du concept des composants avec les agents mobiles. Ce mélange a donné naissance au concept d'*architecture dynamique*. De telles architectures s'appuient sur des composants mobiles pour doter les applications



(a) architecture d'un canevas de gestion à base de composants

(b) architecture canevas

FIG. 3.4 – CBDM : composants et architecture de gestion hiérarchique/dynamique

de gestion d'une flexibilité supplémentaire. CBDM est une approche de gestion à base de composants mobiles. La figure 3.4 illustre deux scénarios possibles de mobilité. La figure 3.4(a) décrit un exemple de *mobilité physique*. Selon cette approche les composants changent de conteneur (environnement d'exécution) au cours de l'exécution. Quant à la figure 3.4(b), elle illustre une double mobilité physique mais aussi fonctionnelle. Les composants se déplacent d'un rôle de gestion (agent, gestionnaire et gestionnaire intermédiaire) à un autre afin de suivre l'évolution des besoins de gestion. Ces composants sont classés en deux catégories :

- Les **composants stationnaires** qui sont des modules statiques qui ne changent pas d'emplacement physique au cours de l'exécution de l'application d'administration. Ils représentent des modules d'administration indispensable au fonctionnement de l'application d'administration.
- Les **composants mobiles** matérialisent un ou plusieurs agents mobiles. Ces agents se déplacent entre les différentes entités qui participent à l'activité d'administration (gestionnaire global, gestionnaire intermédiaire et agent). Chaque déplacement des composants peut être vu comme une délégation [Gol96, MF02] des tâches d'administration à une entité afin d'en décharger une autre.

Parler de la délégation nous emmène naturellement vers la troisième contribution de ce travail, à savoir une architecture de gestion à la fois centralisée, hiérarchique et dynamique (voir figure 3.4(b)). Cette architecture permet de s'attaquer aux défis liés à une administration centralisée en déléguant des tâches d'administration à des gestionnaires intermédiaires. Cette architecture se base sur trois entités :

- *Un gestionnaire global* (GM) : qui comporte à son tour trois partie ; a) une interface utilisateur, b) une application d'administration et c) un service de composants mobiles. Cette entité englobe la logique et l'intelligence d'administration. Elle délègue cette intelligence, en partie, aux gestionnaires intermédiaires.

- *Un gestionnaire intermédiaire* (*Middle Level Manager* (MM))qui interagit avec les agents installés au niveau des ressources administrées afin d'opérer des opérations d'administration. Cette interaction est basée sur des protocoles d'administration classiques tels que le protocole SNMP.
- *Un agent* est une entité logicielle installée au niveau des équipements gérés et qui exécute les tâches d'administration demandées par les GMs et les MMs.

3.3.4.3 Synthèse

CBDM est un canevas de gestion à base de composants mobiles. Il combine la technologie des composants et celle des agents mobiles dans le but d'offrir une infrastructure de gestion flexible et dynamique. Toutefois, CBDM ne traite pas les problématiques de l'hétérogénéité des ressources administrées, ni celle de l'administration différentes échelles.

3.4 Administration fondée sur l'architecture

Dans cette section, nous nous intéressons aux systèmes d'administration fondée sur l'architecture des systèmes administrés. Nous étudions dans un premier temps le canevas Rainbow [CHSS04], dont l'objectif est la gestion autonome des systèmes. Ensuite, nous présentons le canevas SmartFrog [GGL+03] qui a pour mission la gestion des applications dans un contexte de grilles.

3.4.1 Rainbow

Rainbow [CHSS04] est un canevas conçu pour la construction de systèmes auto-adaptables. Le point focal dans ce travail est la réalisation de *la boucle de commande* [OM03] telle qu'elle a été introduite dans les travaux de recherche sur les systèmes autonomes par IBM. Pour ce faire, il s'appuie sur une architecture orientée composants et une approche orientée architecture. Nous détaillons dans le reste de cette section le modèle à composants adopté et l'architecture d'administration résultante.

3.4.1.1 Le modèle à composants de Rainbow

Le modèle de composants utilisé dans ce travail est le modèle à composants spécifique au projet Rainbow [CHSS04]. Il s'agit d'un modèle qui permet de représenter le système administré d'une manière hiérarchique. Les liaisons entre les composants sont représentées sous forme de connecteurs. Dans le cas de composants composites (hiérarchiques), ces liaisons permettent d'exposer des interfaces internes au niveau d'un composant composite (parent).

Le principe de la gestion orientée par les composants consiste à modéliser d'une façon abstraite les ressources à administrer sous forme de composants Rainbow. Le modèle abstrait est ensuite utilisé pendant la conception du système à l'aide d'un fichier ADL (*Architecture Description Langage*). L'ADL est également consulté à l'exécution pour procéder à l'adaptation du système.

3.4.1.2 Architecture d'administration

De point de vue architecture, Rainbow distingue la couche d'administration de la couche du système administré pour permettre l'intégration de politiques d'administration indépendamment des mécanismes (e.g. sondes, canaux d'observation). Cette séparation permet également de réutiliser des parties de la couche d'administration pour différents types de systèmes. La figure 3.5 résume les principaux éléments qui composent cette architecture.

- **Infrastructure de la couche système**. A ce niveau, des interfaces d'accès sont implantées. Des sondes (*probes*) observent et mesurent les différents états du système et des actionneurs (*effectors*) permettent la mise en place des modifications souhaités par l'administrateur. De plus, un service de découverte peut être sollicité pour allouer des ressources à administrer suivant le type des ressources ou d'autres critères bien définis.
- **Infrastructure de la couche d'architecture**. Des jauges (*gauges*) permettent d'agréger les informations envoyées par les sondes et de mettre à jour les propriétés appropriées dans le modèle d'architecture. Un gestionnaire de modèle (*model manager*) traite les informations et accède au modèle d'information. Un évaluateur de contraintes (*constraint evaluator*) vérifie périodiquement le modèle et détecte le besoin d'une adaptation si une contrainte est violée. Une machine d'adaptation (*adaptation engine*) détermine l'action à effectuer et met à jour l'adaptation correspondante.
- **Infrastructure de traduction**. Cette couche établit la correspondance entre le modèle et le système d'administration. Un dépôt dans l'infrastructure maintient la liste de correspondance entre la représentation en composants et l'implantation système. Par exemple, le système peut faire correspondre l'identificateur d'un composant à une adresse IP.

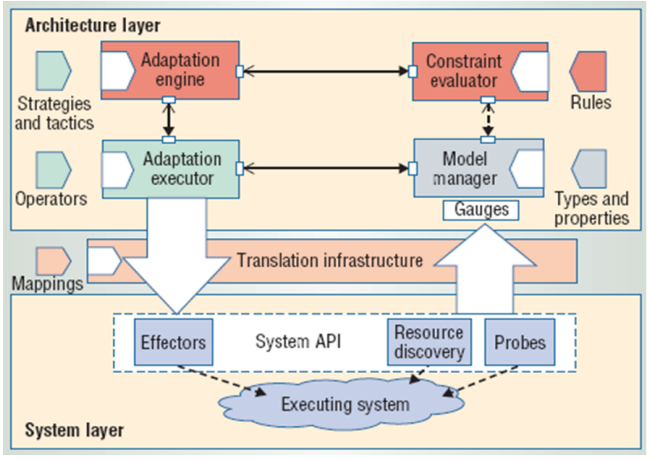


FIG. 3.5 – Architecture interne de Rainbow [CHSS04]

3.4.1.3 Synthèse

Rainbow est un canevas conçu pour la construction des systèmes auto-adaptables. Pour ce faire, il s'appuie sur une architecture basée sur les composants et une approche de gestion orientée architecture. Ce travail propose une vision avancée de la boucle de commande telle

qu'elle a été proposée dans les travaux menés par IBM dans le domaine de l'*autonomic computing*.

L'architecture d'administration proposée fait bien la distinction entre la couche qui englobe l'intelligence de l'administration (couche d'architecture) et les couches qui gèrent les mécanismes d'interaction avec les ressources réelles administrées (couche de traduction et couche système).

Néanmoins, Rainbow présente quelques limitations. En effet, le canevas se focalise sur l'automatisation de l'adaptation et n'exploite pas, par exemple, l'ADL pour le déploiement des systèmes distribués. De plus, la partie instrumentation et la création des actionneurs se font d'une manière ad hoc. Le modèle de composants est exploité pour la représentation du système administré et non pas pour la construction du système d'administration lui-même. Certes l'automatisation des décisions d'administration est indispensable dans un contexte large échelle. Toutefois, avec un modèle de gestion centralisé, l'efficacité du moteur de décisions dans un contexte large échelle risque d'être faible.

3.4.2 SmartFrog

SmartFrog (*Smart Framework for Object Groups*) [GGL+03] est un canevas d'administration dédié à la description, le déploiement, la configuration et la gestion d'applications Java distribuées. Il a été développé par HP Labs et a été exploité en production. En particulier, SmartFrog a été exploité pour l'administration des grilles de calcul.

Nous présentons dans cette section le modèle à composants sur le quel s'appuie ce travail et nous étudions les fonctions d'administration qu'il propose ainsi que les possibilités de passage à l'échelle qu'il offre.

3.4.2.1 Le modèle à composants SmartFrog

Afin d'êtrer gérées par SmartFrog, les applications à déployer doivent respecter un modèle de composants spécifique. Ces composants étendent une classe Java, ce qui leur confère un cycle de vie commun. De plus, chaque composant fournit deux types d'interfaces : les *interfaces fonctionnelles* et *les interfaces de gestion*. Par ailleurs, il y a deux types de composants, *primitifs* et *composites*. De point de vue implémentation un composant primitif est un objet Java qui implémente les interfaces de gestion et des interfaces fonctionnelles SmartFrog. Un composant composite regroupe un ensemble de composants primitifs.

Pour décrire la structure et les interactions de ses composants, SmartFrog fournit un langage de description qui permet de définir une description du système à déployer d'une manière déclarative. Une description d'un composant est une collection ordonnée d'attributs. Un attribut a la forme d'une paire (nom, valeur). Il existe trois types de valeurs :

- des valeurs simples (entiers, chaînes de caractères, etc.),
- une référence vers un autre attribut,
- une autre description de composant.

Le dernier cas permet de définir une relation d'inclusion (hiérarchie) entre composants. SmartFrog fournit un ensemble de services pour faciliter le déploiement et l'administration de ces composants. Parmi ces services on peut citer le service de désignation de noms ou

de celui de la découverte dynamique des ressources.

Nous nous focalisons dans cette section sur la fonction la plus importante fournie par SmartFrog, à savoir la gestion du déploiement.

3.4.2.2 Gestion du déploiement

La principale fonction d'administration remplie par SmartFrog est la gestion du déploiement. Pour ce faire, SmartFrog dispose d'un descripteur de déploiement basé sur un langage propriétaire et un gestionnaire qui se base sur les descriptions des composants pour orchestrer les différentes tâches liées au déploiement. Le gestionnaire est un système distribué et les différentes parties de ce système se coordonnent entre elles pour accomplir les tâches de déploiement. Ces tâches utilisent les interfaces de gestion fournies par les composants. Par ailleurs, le gestionnaire reste actif pendant l'exécution de l'application et se charge d'un certain nombre d'opérations telles que l'observation du système, la réaction aux fautes et l'arrêt des applications. De plus, pour palier les éventuels problèmes de sécurité, SmartForg offre un canevas de gestion de sécurité qui organise les différentes machines cibles sous la forme de domaines de sécurité, ce qui permet une répartition des responsabilités. De plus, la communication entre les nœuds est sécurisée.

3.4.2.3 Gestion à large échelle

Les infrastructures et les applications SmartFrog sont la plupart du temps réparties. Typiquement, elles peuvent être installées sur des dizaines ou des centaines de machines, d'où le besoin d'administrer cette infrastructure fortement répartie. JMX peut être utilisé afin d'administrer à la fois l'infrastructure et les applications SmartFrog. Toutefois, les implémentations actuelles de JMX ne permettent pas gérer des milliers de MBeans déployés sur une centaine de machines. La combinaison de JMX et de SmartFrog pourrait permettre à la fois l'administration des infrastructures SmartFrog et aussi la gestion des configurations JMX à large échelle. sfJMX (SmartFrog -JMX) est une extension de JMX qui matérialise cette combinaison [Gui02]. Elle permet la gestion des infrastructures et des applications SmartFrog en offrant une instrumentation transparente et peut intrusive dans l'infrastructure et les applications SmartFrog. D'autre part, sfJMX permet la gestion des installations JMX à large échelle en offrant les moyens de descrir, configurer et déployer des installations JMX réparties.

Chaque élément de l'infrastructure JMX (Mbean, Agent, etc.) peut être décrit avec le langage de description d'architecture fourni par SmartFrog. C'est ainsi qu'on peut décrire des topologies de déploiement avancées qui répondent aux besoins des applications d'administration. De telles topologies peuvent contenir plusieurs niveaux de hiérarchie. Elles peuvent également englober des interactions entres les Agents JMX. La figure 3.6 [Gui02] illustre un exemple de déploiement complexe des infrastructures JMX avec SmartFrog. Le *MBean Deployer* est un MBean sfJMX qui assure la gestion et découverte des agents JMX distants.

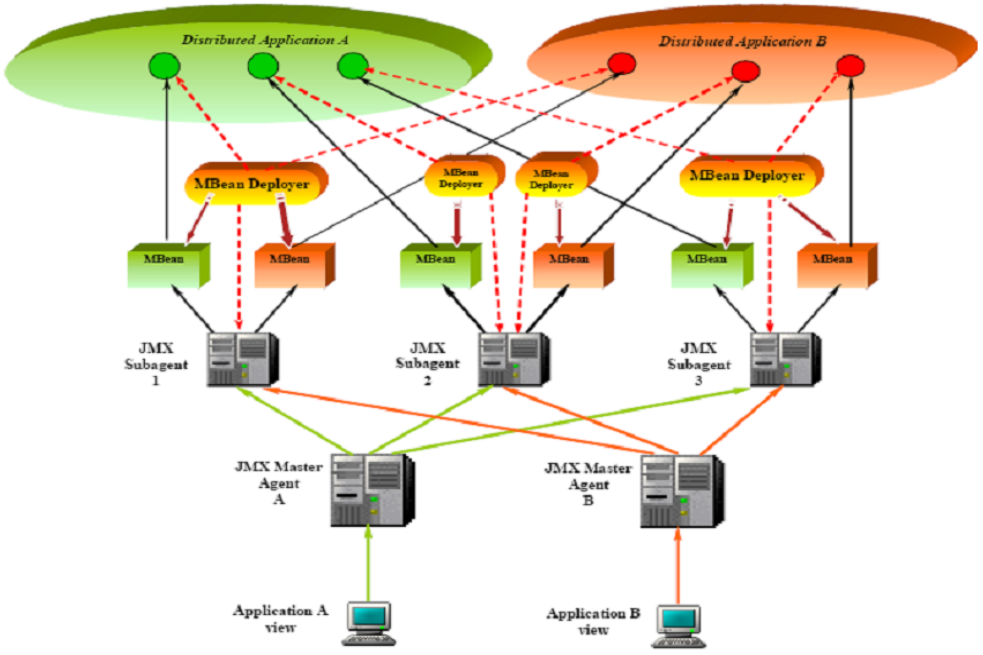


FIG. 3.6 – cfJMX : Déploiement des infrastructures JMX avec SmartFrog

3.4.2.4 Synthèse

SmartFrog est un canevas logiciel conçu pour la gestion de déploiement des applications réparties. Pour ce faire, il s'appuie sur un modèle à composants et un langage de description de déploiement. Le système est donc modélisé sous la forme d'un ensemble de composants interconnectés et imbriqués. Cette encapsulation dans des composants suivant le modèle SmartFrog permet aux applications patrimoniales d'être intégrées dans SmartFrog. Pour déployer ces composants, SmartFrog se base sur un gestionnaire définissant les différentes tâches de déploiement et un ensemble de services comme le service de découverte et d'adressage. SmartFrog a été utilisé dans un environnement industriel durant plusieurs années. Il a été utilisé, en particulier, dans le contexte de gestion des grilles de calcul.

Toutefois, ce canevas présente quelques limitations. Il suppose l'existence d'un environnement d'exécution déjà déployé sur les machines cibles. D'autre part, les fonctions de reconfiguration sont très limitées et sont restreintes au cycle de vie d'un composant. Enfin, le système d'administration est faiblement reconfigurable et ne profite pas, par exemple, du modèle à composants pour la gestion de son cycle de vie.

Afin de pallier ces limitations, des travaux récents [Gui02] ont mis en place un système d'administration qui combine la capacité d'administration d'une manière efficace, transparent et peu intrusive avec JMX et la faculté de déploiement à large échelle de SmartFrog.

3.5 Administration à base de composants et orientée architecture

Dans cette section, nous présentons à travers Jade [DBB+08] et GridKit [GCB+04] deux travaux qui combinent à la fois l'usage des composants et l'approche d'aministraion orientée par l'architecture.

3.5.1 JADE

Jade [DBB+08] est un canevas logiciel hautement adaptable et modulaire. Il a été expérimenté dans le cadre d'un travail de recherche sur l'administration autonome [OM03] des clusters J2EE (*Java Enterprise Edition*). Dans le reste de cette section, nous présentons le modèle à composants Fractal, sur lequel s'appuie Jade. Ensuite, nous présentons l'architecture d'administration de Jade.

3.5.1.1 Modèle à composants Fractal

Fractal [BCL+06] est un modèle à composants qui a été conçu dans le cadre d'une collaboration entre France Télécom et l'INRIA (*Institut National de Recherche en Informatique et Automatique*) dans le but de faciliter la conception et le développement des infrastructures logicielles à composants. Ces infrastructures bénéficient de plusieurs propriétés intéressantes telles que la modularité, la flexibilité et re-configurabilité dynamique.

De point de vue de son architecture interne, un composant Fractal est défini par une *membrane* et un contenu. Le contenu d'un composant peut être du code exécutable, dans ce cas on parle de *composant primitif*, ou un autre composant imbriqué d'une façon hiérarchique, et dans ce cas on parle de *composant composite*.

La figure 3.7(a) illustre les différents éléments qui constituent un composant Fractal ainsi qu'un exemple d'imbrication de composants.

Les composants Fractal communiquent avec leurs semblables via des interfaces. C'est ce qui constitue la membrane. Il existe principalement deux catégories distinctes d'interfaces :

- les interfaces métier qui constituent les points d'accès externes au composant,
- les interfaces de contrôle (ou contrôleurs) qui prennent en charge des propriétés non fonctionnelles du composant.

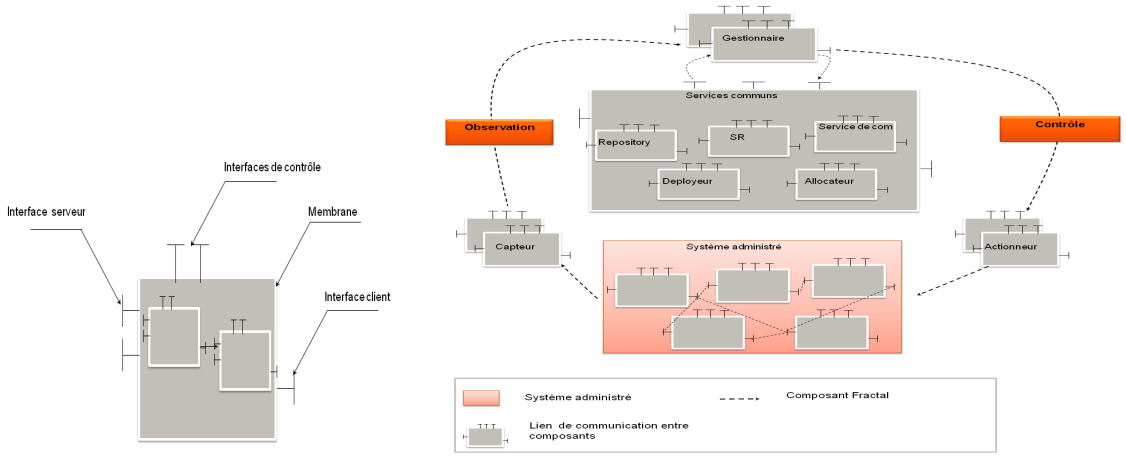
Fractal met à disposition un ensemble d'interfaces de contrôle qui sont la gestion du cycle de vie (*life cycle controler*), des liaisons (*binding controler*), des attributs (*attribute controler*), du contenu (*content controler*) pour les composites. Le modèle étant extensible, il est possible de définir ses propres interfaces de contrôle. Une infrastructure « fractalisée »typique comporte un ensemble de composants Fractal interconnectés et inter reliés ou tout simplement assemblés. L'assemblage des composants est effectué de deux manières dans Fractal. D'une façon programmatique ou en s'appuyant sur un langage de description d'architecture nommé l'*ADL*.

Ce modèle à composants a fortement inspiré les concepteurs et les développeurs de Jade. C'est ce que nous allons voir dans la section suivante.

3.5.1.2 Architecture

Jade est un canevas d'administration qui concrétise des travaux de recherche sur l'auto-administration. D'ailleurs, son architecture globale reflète bien cet aspect de gestion autonomique. La figure 3.7(b) illustre cette architecture qui se présente sous la forme d'une *boucle de commande* complète. Cette boucle est constituée des éléments suivants :

- les capteurs collectent des informations du système administré, véhiculées via des canaux de communication à des composants gestionnaires ;
- les actionneurs sont des composants co-localisés généralement avec le système administré et qui exécutent les actions décidés par les gestionnaires ;
- les gestionnaires implantent les politiques d'administration et décident des actions à effectuer sur le système ;
- les services communs tels que le déploiement, le *repository* ou le service de communication. Ces services sont utiles aux gestionnaires pour la gestion des ressources administrées.



(a) Modèle à composants Fractal (b) Boucle de commande de Jade

FIG. 3.7 – Jade : Modèle et architecture à composants

Afin de mettre en œuvre cette boucle de commande, le canevas Jade s'appuie sur l'hypothèse que le système administré est construit à l'aide de composants Fractal et utilise l'ADL Fractal pour décrire l'architecture à déployer. Toutes les ressources du système (nœuds, entités logicielles, etc.) sont des composants Fractal et leurs dépendances sont exprimées à l'aide de propriétés d'encapsulation et de liaisons. Dans le cas de composants non écrits en Fractal, il est possible de les encapsuler en tant que composants primitifs Fractal. L'architecture est par conséquent explicite et dynamiquement reconfigurable.

3.5.1.3 Gestion fondée sur l'architecture

Le SR (*System Representation*) est un élément central dans Jade. Il sert de méta-niveau et réifie l'architecture logicielle interne du système administré. Il représente la vue architecturale du système administré de point de vue Jade. D'ailleurs, le système administré et le SR sont causalement liés : toute modification de l'un se répercute sur l'autre. La connaissance de cette architecture facilite la mise en œuvre de politiques de reconfigura-

tions ; elle permet notamment d'identifier les éléments à reconfigurer et de les introduire en conformité avec le reste de la structure du système administré. Par ailleurs, Jade a une structure réflexive, c'est-à-dire qu'en cours d'exécution, le système administré avec Jade utilise une représentation de lui-même : le SR.

3.5.1.4 Synthèse

Jade est un canevas de gestion qui a été appliqué et étudié dans le cadre de l'administration des clusters JavaEE. L'originalité de ce travail réside dans sa capacité à introspecter dynamiquement son état et l'état des systèmes qu'il gère à travers une représentation système en mémoire. En effet, Jade est un canevas intergiciel réflexif qui prend conscience de son état et de sa disponibilité au même titre que les systèmes qu'il gère. Un autre point fort de Jade est son mécanisme de déploiement qui s'appuie sur une description architecturale de type ADL. De point de vue passage à l'échelle, Jade adopte une approche d'administration centralisée. C'est ce qui limite ses capacités de passage à l'échelle. L'autre limitation qu'on peut reprocher à Jade au niveau de point de vue passage à l'échelle est sa stratégie de collecte de données de supervision. En effet, Jade se limite a une approche de collecte de type « pull »[2]. Il ne propose pas donc des possibilités de collecte en mode « push »[3]. Ce mode est particulièrement intéressant dans certaines situations. Il permet d'économiser des échanges de données et augmenter ainsi la capacité de passage à l'échelle d'une plateforme d'administration.

3.5.2 GridKit

GRridKit [GCB+04] est un autre exemple de système de gestion à composants pour la gestion fondée sur l'architecture. Cette fois, il s'agit d'appliquer l'approche de gestion à base de composants dans un contexte similaire aux clusters JavaEE. Il s'agit du contexte de la gestion des ressources dans les grilles de calcul. Nous présentons dans un premier temps le modèle à composants adopté dans le cadre de ce travail. Nous introduisons par la suite un des services clés de GidKit, à savoir le service de gestion des ressources.

3.5.2.1 Le modèle à composant OpenCOM

Le canevas GridKit se base sur le modèle à composants OpenCOM [BCU+04]. Il s'agit d'une évolution du modèle COM [COM] de Microsoft en vue de la mise en place d'intergiciels *réflexifs* [Cou07] et reconfigurables. Les contributions de ce modèle à COM commencent par la définition des services requis (ce qui est optionnel dans COM) et par l'identification des connexions comme étant des éléments architecturaux. Ce modèle est basé sur des interfaces avec deux éléments : *les réceptacles* et *les connexions*. Les réceptacles identifient les services requis par un composant. Quant aux connexions, elles permettent d'établir des liens entre les composants.

OpenCOM définit un méta-niveau pour tous les composants. Ce dernier fournit essentiellement des services d'introspection et de reconfiguration dynamique des composants.

[2] Activation régulière des mécanismes de collecte de données à des intervalles de temps réguliers et prédéfinis.

[3] envois de données de supervision à l'initiative des ressources administrées.

Pour ce faire, chaque composant OpenCOM doit fournir les interfaces suivantes pour permettre à son environnement d'exécuter sur lui des opérations d'introspection et d'interception (voir figure 3.8) :

- *IMetaInterface* fournit des opérations d'introspection des types d'interfaces et des réceptacles de composants.
- *IConnections* permet de modifier les connexions établies sur les réceptacles du composant.
- *ILifeCycle* fournit les opérations de gestion de cycle ce vie telles que les opérations *Start* et *Stop* qui sont appelées par l'environnement de déploiement au moment du chargement et du déchargement du composant.
- *IMetaArchitecture* permet d'identifier les connexions établies entre les réceptacles du composant et les interfaces d'autres composants. Cette interface permet entre autres de reconstituer le graphe d'interconnexion d'un système OpenCOM.

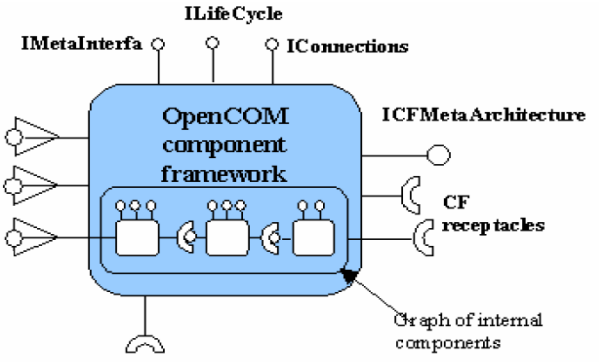


FIG. 3.8 – Le modèle à composants OpenCOM [BCU+04]

3.5.2.2 Les services Gridkit

GridKit [GCB+04] est une infrastructure intergicielle qui propose un ensemble de services de médiation dans un contexte de grilles de calcul. Ces services sont développés sous forme de composants OpenCOM « pluggables »et sont au nombre de quatre.

- *Le service Binding.* Il fournit des APIs génériques permettant aux programmeurs des applications GridKit de créer et utiliser différents modes d'interaction entre les services et les utilisateurs.
- *Le service Resource Discovery.* Il s'agit d'un service de découverte des ressources. Il regroupe des implémentations de protocoles de découverte standards tels que SLP (Service Location Protocol) ou UPnP (Universal Plug and Play). Il inclut d'autres protocoles spécifiques aux grilles de calcul comme *GRAM* [CFK+98] pour la découverte des *CPU* dans un contexte de grilles.
- *Le service Resource management.* La gestion de ressources comprend à la fois une gestion à une grosse granularité, tout comme le fait GRAM, et une gestion à une granularité plus fine (*buffers*, *threads*, canaux de communication, etc.)
- *Le service Grid Security.* Ce service contient des services qui gèrent les communications sécurisées.

Nous nous focalisons dans la suite sur le service de gestion des ressources [RK05]. Si le lecteur est intéressé par les autres services de GridKit, nous lui recommandons la référence suivante [GCB+04].

3.5.2.3 Gestion des ressources

Plusieurs applications complexes et demandeuses de ressources de calcul sont déployées sur les grilles. Le service de gestion des ressources de GridKit a pour objectif l'allocation des ressources disponibles sur la grille pour effectuer des calculs et les traitements requis par ces applications. Pour mener à bien cette mission, ce service se base sur une architecture de gestion hiérarchique composée de deux principaux éléments (voir figure 3.9) :

- **Le gestionnaire global des ressources** (*Global Resources Management*) qui s'occupe de la coordination de la gestion des ressources déployées sur plusieurs nœuds de la grille.
- **Le gestionnaire local des ressources** (*Local Resources Management*) qui gère l'allocation des ressources au niveau de chaque nœud.

Afin de gérer une allocation efficace des ressources, ce service requiert que l'application à départager entre les ressources soit implémentée en termes de composants OpenCOM. Initialement, l'application à distribuer entre les nœuds est décrite en XML (*Application Description*). Cette description est traitée par le composant *Virtual Cluster Builder* qui communique avec le *Decomposer* et le *QoS Mapper* afin de décomposer l'application en tâches et la distribuer entre les nœuds, en fonction de leur capacité et disponibilité. Ce *Virtual Cluster Buidler* communique également avec le composant *Finding* pour retrouver les ressources disponibles. Une fois que cette phase est terminée, le *Virtual Cluster Builder* communique avec les gestionnaires locaux des nœuds sélectionnés. Ces gestionnaires contiennent un composant *Ressource Manager* qui communique avec le composant *Task Manager* pour gérer l'affectation des tâches aux ressources disponibles au niveau du nœud. Le composant *Monitor* surveille la consommation des ressources et leur disponibilité et déclenche en cas de violation de la qualité de service exigée.

3.5.2.4 Synthèse

GridKit est un canevas de médiation a été appliqué et étudié dans le contexte des grilles de calcul. Le canevas propose quatre services de bases pour la gestion de la découverte, la gestion des ressources, la gestion de la sécurité et la gestion des *bindings*.

Ce canevas se base sur une architecture d'administration hiérarchique afin de pallier les problèmes liés au passage à l'échelle. Quant à l'hétérogénéité, GridKit encapsule les applications, qu'il permet de déployer sur la grille, dans des composants OpenCOM.

3.6 Composants et administration intégrée

Après avoir défini la gestion intégrée dans la première section du second chapitre, nous analysons dans cette section un travail de recherche effectué autour de la gestion intégrée à base de composants [MKP99].

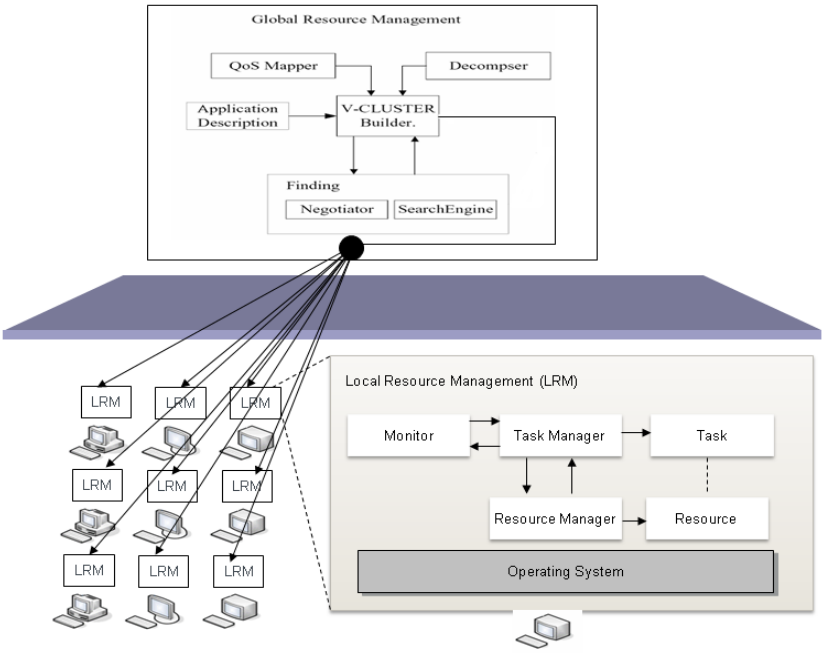


FIG. 3.9 – Architecture du canevas de gestion des ressources de Gridkit

3.6.0.5 Architecture d'administration intégrée

De point de vue architecture globale, ce travail définit un environnement à composants indispensable pour les applications de gestion à base de composants. Cet environnement se décline en trois couches (voir figure 3.10) :

- **Le conteneur des composants**. Cette couche assure la gestion du cycle de vie des composants et représente leur environnement d'exécution.
- **La couche d'infrastructure**. Elle propose des mécanismes pour assister l'administration des composants et la gestion de leurs communications. Elle inclut, entre autre, des services de gestion de migration et de déploiement des composants.
- **La couche infrastructure de répartition**. Cette couche inclut les fonctionnalités de communication et de distribution. Elle représente la couche middleware et peut être implémenté avec les technologies telles que CORBA ou DCOM.

Dans ce qui suit, nous nous focalisons sur le conteneur des composants et plus précisement sur un canevas de gestion intégrée proposé dans le cadre de ce travail. Ce canevas se base sur une architecture à composants qui englobe des composants frontaux *FrontEnd client* pour établir le lien entre les utilisateurs du système d'administration et les ressources administrées. Ces composants permettent aux utilisateurs d'accéder à la base de données des informations de gestion et aux données d'historique. Ces informations sont stockées grâce aux composants de base de données qui gèrent les données de communes de gestion. Ces composants peuvent être implémentés sous forme de base de données relationnelles ou orientées objet.

De la même façon qu'un composant client frontal assure le lien entre l'administrateur et les ressources administrées, le composant *interface d'administration* gère l'interaction entre la ressource administrée et le système de gestion. Ce composant offre une interface

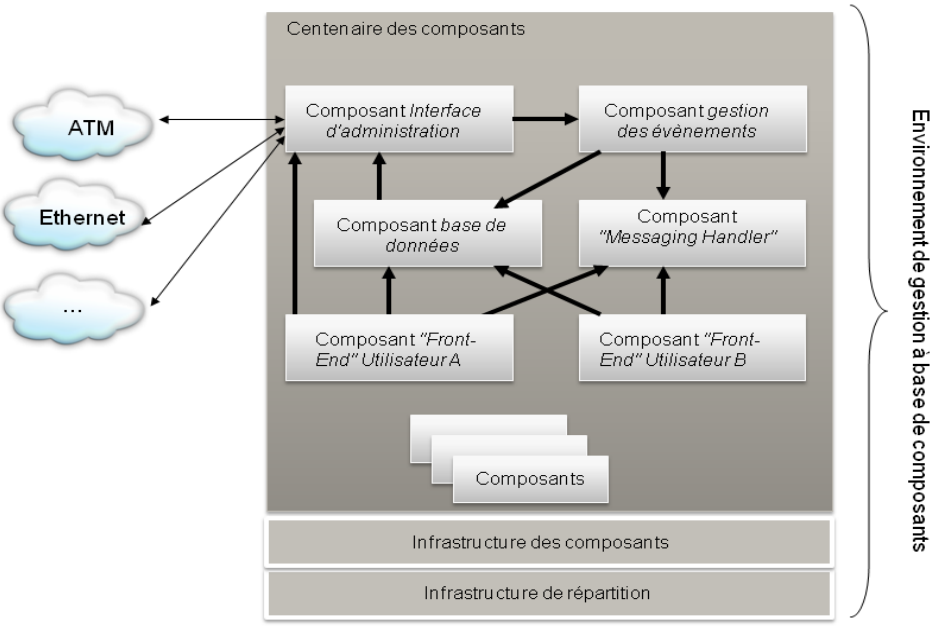


FIG. 3.10 – Architecture de gestion intégrée à base de composants

de gestion permettant de communiquer avec la ressource administrée. On peut trouver un composant interface d'administration par ressource gérée ou un composant par groupe de ressources gérées.

Le composant *gestionnaire des évènements* intercepte les évènements produits par les interfaces de gestion et déclenche les actions de gestion qui lui sont associées. Il peut par exemple se contenter de décider d'activer une intervention directe au niveau de la ressource à gérer ou envoyer une simple notification au composant gestionnaire des notifications (*Messaging Handler*) qui diffuse les informations de gestion aux administrateurs de différentes façons : e-mail, fax, SMS, etc.

3.6.1 Discussion

Le travail proposé par [MKP99] représente un canevas d'administration intégrée à base de composants. Les principales caractéristiques ce travail sont : a) une architecture fonctionnelle clairement définie, b) un canevas flexible conçu et développé à base de composants et c) une integration via les proxies pour la prise en compte des ressources administrées. Toutefois, [MKP99] n'aborde pas la problématique de gestion multi-échelles. En effet, aucune information n'a été communiquée concernant la volumétrie de ressources administrées concidérées.

3.7 Discussion

En examinant l'ensemble des travaux d'administration influencés par les composants, nous avons dégagé quelques préoccupations communes à certains d'entre eux. Nous discutons dans cette section ces préoccupations qui sont par ailleurs résumées dans le tableau TAB.3.2.

La modularité. La préoccupation initiale des systèmes d'administration orientés composants est la fabrication des systèmes d'administration modulaires pour rompre avec les systèmes monolithiques existants. La preuve est que les premiers systèmes d'administration orientés composants n'ont pas ciblé des fonctions de gestion particulières. Ils ont juste mis l'accent sur l'importance de disposer d'un système d'administration modulaire et extensible.

Tous les modèles à composants fournis offrent des possibilités d'assemblage de composants. C'est ce qui permet de construire des applications de gestion ou des représentations des ressources administrées en assemblant des composants unitaires. Cette opération d'assemblage est souvent effectuée d'une manière statique et déclarative avec des descripteurs d'assemblage (ou d'architecture) de types ADL. Ces derniers décrivent l'architecture du système à représenter et représentent une cartographie du système à composants qui sera instancié.

L'abstraction. La seconde préoccupation concerne plutôt les systèmes d'administration guidés par l'architecture. Il s'agit d'utiliser les composants comme moyen pour matérialiser une abstraction des ressources réelles administrées. L'idée de base de cette approche consiste à extraire une représentation homogène, matérialisée par des composants, à partir d'un ensemble de ressources hétérogènes. Cela permet de faire face à l'hétérogénéité des ressources administrées et de leurs interactions. Cette approche permet également l'intégration des systèmes patrimoniaux en les enveloppant dans des composants.

L'autonomie. Une autre préoccupation, commune à plusieurs travaux de la gestion orientée composants [SBH05, CHSS04] est le support de l'autonomie des infrastructures de gestion. En effet, aujourd'hui, de plus en plus de travaux adressent la problématique de l'auto-gestion [OM03]. Loin d'être un phénomène de mode la gestion autonome est née d'un besoin réel de gérer des systèmes de plus en plus large et qui requiert d'avantage de réactivité. L'automatisation des tâches de gestion permet, dans certains cas, d'accroître la réactivité des systèmes de gestion. Cette propriété est très intéressante dans un contexte de gestion à large échelle.

La reconfigurabilité. Certains modèles à composants tels que Fractal [BCL+06] ou OpenCOM [BCU+04] proposent des possibilités de re-configuration dynamique des composants au cours de l'exécution. Ces fonctionnalités permettent de rendre les systèmes de gestion plus flexibles du point de vue structurel. Avec les systèmes à composants, un administrateur peut par exemple extraire ou rajouter des modules d'administration pour répondre à un nouveau besoin ou pour faire à un besoin temporaire de gestion. Ces opérations d'extension sont effectuées sans interruption de l'activité du système d'administration.

La réutilisabilité. Réutiliser des morceaux des applications de gestion au lieu de les dupliquer est l'un des objectifs des infrastructures de gestion à base de composants. Cette réutilisation occasionne à la fois des gains de performance (à travers l'optimisation de l'utilisation de ressources physiques de gestion) et des gains en termes de

coût de développement et de maintenance des systèmes d'administration.

Le déploiement partiel et incrémental. Certains travaux tels que [SBH05, JC07] se sont intéressés à la possibilité de déploiement partiel de morceaux de l'application de gestion au niveau de l'environnement cible. Cette approche occasionne des gains de performance considérables.

L'administrabilité. Systèmes d'administration développés avec des composants administrables deviennent naturellement administrables. Le travail le plus aboutis au niveau de cet aspect, parmi les travaux évalués, est Jade [SBH05]. Ce travail s'appuie sur une approche de type *middleware réflexif*. Jade gère une représentation interne (SR) qui lui permet d'identifier les éventuelles défaillances au niveau du système d'administration.

La dynamicité. Face à une dynamicité aléatoire et un comportement imprévisible des ressources administrées, une dynamicité des systèmes d'administration s'impose. Cette dynamicité se manifeste à deux niveaux : au niveau structurel et au niveau fonctionnel. Goldszmidt [Gol96] est le premier à avoir abordé cette propriété dans le domaine de l'administration en introduisant la notion de gestion par délégation et de gestion élastique. Quelques années plus tard, ces travaux ont été repris par [JS00] en combinant les agents mobiles et les infrastructures à composants. Les agents mobiles sont peut être utiles dans un contexte dynamique. Mais ils sont surtout vulnérables. Plusieurs études ont démontré la vulnérabilité de ces systèmes et les nombreuses failles de sécurité que peut engendrer leur mobilité.

3.8 Conclusion

Dans ce chapitre nous avons passé en revue les principaux travaux autour de la problématique d'administration à base de composants. Nous avons distingué trois principaux courants de recherche : a) les systèmes de gestion à composants, b) les systèmes de gestion fondée sur l'architecture et d) les systèmes à composants pour la gestion fondée sur l'architecture.

Le premier courant a pour objectif la fabrication de systèmes d'administration de plus en plus modulaires et extensibles pour rompre avec les systèmes monolithiques existants. Quant au second courant, il se focalise plutôt sur une gestion guidée par l'architecture du système administré, l'architecture étant considérée comme une abstraction des systèmes administrées qui peut être matérialisée en termes de composants. Le troisième courant quant à lui, il combine les deux premières approches.

En analysant ces travaux nous avons constaté qu'ils sont majoritairement centralisés. Ce choix d'approche de gestion limite leur capacité de passage à l'échelle. Nous avons également constaté que les fonctions de gestion offertes sont pour la plupart des travaux élémentaires limités aux simples opérations de collecte de données ou aux opérations de base de reconfiguration ou de déploiement.

Bref, plusieurs briques existent pour construire des applications flexibles, administrables, reconfigurables et avec un support de gestion à large échelle. Toutefois, aucun travail étudié ne propose une solution complète et cohérente pour gérer les systèmes imprévisibles et hétérogènes dans un contexte multi-échelles.

Ces constats nous conduisent à la seconde partie de ce manuscrit. Nous proposons

dans les chapitres suivants une approche de gestion basée sur les composants et fondée sur l'architecture pour la gestion intégrée dans un contexte multi-échelles.

TAB. 3.2 – Table Synthèse et comparaison des travaux d'administration orientée composants

Propriétés	Admin. à base de composants (A)					Admin. fondée sur l'architecture (B)		A & B	
-	Liaison	RMTool	CBDM	KAWABATA	MKP99	Rainbow	SmartFrog	GridKit	Jade
Autonomie	✗	✗	✗	✗	✗	✔	✗	✗	✔
Abstraction	✗	✗	✗	✗	✔	✔	✔	✔	✔
Administrabilité	✗	✔	✔	✗	✗	✗	✗	✗	✔
Reconfigurabilité	✗	✔	✔	✗	✗	✗	✗	✗	✔
Modularité	✔	✔	✔	✔	✔	✔	✔	✔	✔
Réutilisabilité	✗	✗	✗	✗	✔	✔	✔	✔	✔
Déploiement incrémental	✗	✗	✗	✗	✔	-	-	✔	✔
Dynamicité	✗	✗	✗	✗	✗	✔	✗	✗	✔
Architecture	C	C	C	C	C	C	H	H	C

C=Centralisé, H=Hiérarchique

Chapitre 4

Présentation générale des contributions

Contents

4.1 Défis à relever 97
4.2 Synthèse de l'état de l'art 99
4.2.1 Les systèmes d'administration classiques 99
4.2.2 Systèmes d'administration à base de composants 100
4.3 Positionnement 101
4.4 Contributions 103
4.4.1 Une approche architecturale d'administration muti-échelles . . . 103
4.4.2 Un canevas intergiciel d'administration orienté composants . . . 103
4.4.3 Evaluation 104
4.5 Organisation des chapitres de la contribution 104

Ce chapitre de transition, introduit les contributions de cette thèse. Il expose, dans un premier temps, les défis à relever et propose une synthèse des travaux de l'état de l'art pour y répondre. Il enchaîne avec un positionnement de notre contribution par rapport à ces travaux avant de conclure avec l'annonce de l'organisation de la seconde partie de ce manuscrit.

4.1 Défis à relever

L'analyse de l'état de l'art des systèmes d'administration nous a permis d'identifier plusieurs défis à relever au niveau de l'administration dans un contexte multi-échelles. Parmi ces défis nous pouvons citer les cinq points suivants :

L'administration multi-échelles. Ce défi englobe trois dimensions à considérer : a- *l'ampleur numérique*, qui concerne l'évolution du nombre des ressources administrées et le volume d'informations de gestion échangé, b- *la dimension organisationnelle*, qui dépend du nombre et des profils des administrateurs (définis par les droits d'accès qui leurs sont accordés) et c- *la distribution géographique* des systèmes administrés qui peut avoir un impact très important sur les performances d'une infrastructure d'administration et sur sa capacité de passage à l'échelle. En plus de ces trois dimensions,

un contexte multi-échelles peut être caractérisé par une importante hétérogénéité des ressources. Cela nous emmène au défi suivant.

L'administration de ressources hétérogènes. Très souvent, les contextes multi-échelles, tels que les réseaux d'opérateurs, sont caractérisés par une forte hétérogénéité des ressources à administrer. Les plateformes M2M (*Machine to Machine*) [M2M06] ou les offres de type *triple play* (combinaison de trois offres de téléphonie fixe, internet et télévision) sont des exemples concrets de contextes multi-échelles souvent rencontrés dans le monde des télécommunications. Malgré l'avènement de ces offres, leur commercialisation précède souvent la mise en place de plateformes garantissant leur qualité de service. L'administration des ressources hétérogènes devient alors un enjeu vital pour les opérateurs de télécommunication. Pallier cette limitation peut être un premier pas vers l'administration de la qualité de service de bout en bout.

L'administration de qualité de service de bout en bout. Il s'agit de surveiller un ensemble d'indicateurs de qualité, définis préalablement en fonction de leur pertinence et de leur représentativité de la qualité du service. Ce suivi est de bout en bout, autrement dit, il couvre l'ensemble de l'infrastructure sous-jacente au service. Cette problématique est déjà assez complexe dans le contexte des réseaux d'entreprise. En changeant d'échelle comme dans le cas des opérateurs de télécommunication, l'administration de qualité de service de bout en bout devient un véritable cauchemar pour les administrateurs. Tous les opérateurs, aujourd'hui, tentent d'assurer et de maîtriser la qualité des services qu'ils proposent. La complexité de ce défi réside dans l'aptitude à surveiller et contrôler toutes les couches logicielles et matérielles par lesquelles transitent les services de l'opérateur.

L'administration autonomique des réseaux et des services. Dans un contexte large échelle, l'administration manuelle devient quasiment impossible, vu le nombre de paramètres à configurer et l'ampleur numérique, géographique et organisationnelle des systèmes administrés. L'automatisation des décisions d'administration vient en renfort, voire en remplacement, des administrateurs humains. Plusieurs travaux se sont penchés sur cette question. Ils se sont inscrits dans un courant recherche baptisé l'*autonomic computing* [OM03]. Ces travaux ont pour ambition d'« injecter de l'intelligence »dans les programmes informatiques. Deux techniques sont généralement appliquées. La première technique se base sur des travaux issus de l'intelligence artificielle [RSS] pour enrichir une base de connaissances. La seconde technique consiste à rajouter des règles actives ou règles ECA (Evenement, Condition, Action) prédéfinis à l'avance [DBB+08]. Rares sont les travaux qui ont traité la problématique de l'administration multi-échelles.

Une architecture monolithique. L'architecture monolithique des systèmes d'administration est un obstacle majeur pour leur adaptabilité et leur re-configurabilité dans un contexte multi-échelles. Or, la plupart des systèmes d'administration classiques [Ope, Tiv, Nag07] sont dotés d'une architecture monolithique. Ce constat peut être expliqué par deux raisons. La première raison est la frontière étanche entre le domaine d'administration réseaux et celui du génie logiciel. La seconde raison est que les systèmes classiques, du moins pour les plus aboutis, sont le résultat de plusieurs fusions et évolutions au fur et à mesure des années. Cela rend leur maintenance assez ardue.

Nous nous focalisons dans ce travail sur trois de ces défis : **l'administration multi-échelles**, **la gestion de ressources administrées hétérogènes** et **l'architecture monolithique des systèmes d'administration**. Néanmoins, nous n'ignorons pas les autres

défis soulignés précédemment. Ce travail peut en effet servir comme base de réflexion pour relever les autres défis mentionnés. Avant d'introduire notre approche d'administration, nous passons en revue les réponses apportées par l'état de l'art, étudié dans les chapitres 2 et 3, à ces trois défis.

4.2 Synthèse de l'état de l'art

Cette section synthétise d'une manière succincte l'état de l'art des systèmes d'administration déjà introduit dans les chapitres 2 et 3. Après, une synthèse des systèmes d'administration classiques, les systèmes d'administration à base de composants sont présentés.

4.2.1 Les systèmes d'administration classiques

Nous considérons comme systèmes d'administration classiques ceux abordés dans le chapitre 2 [Ope, Tiv, Nag07]. L'objectif des sous-sections suivantes est d'étudier les réponses apportées par ces systèmes aux trois défis adressés.

4.2.1.1 Un modèle d'administration fortement centralisée

Les systèmes d'administration classiques, qui sont pour la plupart centralisés, ne sont pas adaptés à l'administration dans un contexte multi-échelles. Les éditeurs de ces solutions ainsi que les chercheurs industriels et académiques ont récemment pris conscience de la nécessité de développer de nouvelles approches d'administration pour relever le défi de l'administration quelque soit l'échelle d'administration considérée.

Il en résulte plusieurs modèles d'administration qui tentent de pallier les limitations du modèle centralisé. Le système d'administration Nagios [Nag07], par exemple, propose la possibilité de d'administration hiérarchique. IBM Tivoli [Tiv], quant à lui, propose une solution d'administration complètement distribuée pour la gestion de plus de 3000 ressources. Quant au système HP NNM [Ope], il propose plusieurs modèles d'administration en fonction des besoins d'administration ainsi que plusieurs catégories de filtres des flux de données d'administration pour maîtriser le volume des données échangées.

Malgré des efforts considérables, la capacité de passage à l'échelle reste réduite pour plusieurs raisons. Tout d'abord, le passage d'une configuration centralisée à une configuration distribuée ou hiérarchique peut s'avérer assez onéreux et complexe à mettre en œuvre avec des systèmes tels que IBM Tivoli et HP OpenView. De plus, tous les systèmes étudiés requièrent une intervention humaine importante pour régler les éventuels problèmes liés au passage à l'échelle.

4.2.1.2 Une architecture logicielle monolithique

L'une des failles majeures des systèmes existants, sur le plan architectural, est l'aspect monolithique des architectures logicielles. En effet, contrairement aux systèmes à base de composants, les systèmes d'administration classiques n'ont pas été conçus selon le principe de modularité, ce qui limite leur capacité d'adaptation en fonction du changement du

contexte d'administration adressé. L'aspect monolithique de systèmes classiques d'administration limite également les possibilités de configurabilité de ces systèmes.

4.2.1.3 Approches de gestion des ressources hétérogènes

L'hétérogénéité des ressources administrées peut exister dans des systèmes de différentes tailles. Toutefois, elle à tendance à être encore plus accentuée lorsque le parc administré est de grande taille. Nous pouvons observer cette hétérogénéité à plusieurs niveaux (protocoles, modèle d'information, type de ressources, etc.). Nous nous focalisons dans ce travail sur l'hétérogénéité des types de ressources. L'état de l'art étudié propose différentes approches pour répondre aux problèmes liés à la gestion des ressources hétérogènes (voir section 1.4.3 pour plus de détails) :

- Multiplication des systèmes d'administration,
- Gestion via des adaptateurs,
- Gestion via un modèle d'information générique.

4.2.2 Systèmes d'administration à base de composants

Nous considérons comme systèmes d'administration à base de composants, ceux abordés dans le chapitre 3 [JC07, Der97, JS00, KYM+00, GGL+03, CHSS04, GCB+04, SBH05, PVRY05, DBB+08]. L'objectif des sous-sections suivantes est d'étudier les réponses apportées par ces systèmes aux trois défis adressés.

4.2.2.1 Un modèle d'administration fortement centralisée

A notre connaissance, rares sont les travaux sur l'administration basée sur les composants ou dirigée par l'architecture qui ont adressé la problématique de passage à l'échelle des systèmes de gestion. La plupart des systèmes d'administration abordés dans le chapitre 3 adoptent l'approche d'administration centralisée [JC07, CHSS04]. Quelques travaux récents, notamment dans le domaine de l'administration des grilles de calcul, tels que SmartFrog [GGL+03], sfJMX [Gui02] et GridKit [GCB+04], ont commencé à explorer la piste de l'administration hiérarchique.

4.2.2.2 Une architecture logicielle flexible

La modularité de l'architecture et le couplage faible inter-modules sont parmi les principaux avantages des systèmes d'administration orientés composants en général, et des systèmes de gestion à composants et des systèmes à composants pour la gestion fondée sur l'architecture en particulier (voir chapitre 3 pour plus de détails sur ces systèmes). Cette modularité facilite, entre autre, la maintenance et le déploiement des systèmes d'administration. Toutefois, il importe de souligner que tous les systèmes d'administration orientés composants ne disposent pas eux-mêmes d'une architecture à composants. Nous avons vu dans le chapitre précédent que certains systèmes d'administration appliquent l'approche par composants uniquement pour la gestion de l'hétérogénéité mais sont construits sur une architecture monolithique [JMX04, GGL+03, CHSS04].

L'autre point fort des infrastructures d'administration orientées composants est la capacité de chargement et de déchargement dynamique de certains modules du système d'administration [Der97, JC07]. Cette facilité permet de rendre les systèmes d'administration plus flexibles de les adapter dynamiquement en fonction de l'évolution des besoins d'administration.

Toutefois, nous pouvons reprocher aux systèmes d'administration orientée composants étudiés leur **manque de généricité**. En effet, la plupart des ces systèmes ont été développés pour répondre à des besoins précis (administration de capteurs, de grilles, etc.) et ne sont donc pas directement applicables dans d'autres contextes d'administration. La plupart de ces systèmes adopte une approche ad hoc pour la gestion des ressources hétérogènes.

D'autres systèmes appliquent l'approche par composants uniquement pour la représentation du système administré [CHSS04] et non pas pour la construction du système d'administration lui-même.

4.2.2.3 Approches de gestion des ressources hétérogènes

En étudiant les travaux d'administration orientée composants, nous avons décelé principalement deux approches de gestion de l'hétérogénéité.

- Gestion ad hoc : RMTool [JC07], par exemple, adopte cette approche pour la gestion des réseaux de capteurs qui consiste à implémenter des adaptateurs spécifiques pour gérer les systèmes à administrer.
- Gestion par encapsulation des ressources : La seconde approche consiste a encapsuler les systèmes à administrer dans des composants [CHSS04, GCB+04]. L'abstraction des ressources à gérer à travers des composants permet de masquer l'hétérogénéité des ressources sous-jacentes. Elle offre une vue homogène des ressources. Le travail de [SBH05], suit cette approche. Les composants jouent le rôle d'intermédiaire entre la plateforme d'administration et les ressources administrées.

4.3 Positionnement

Nous proposons dans cette section un positionnement de notre travail par rapport à l'existant :

Activités d'administration. Le point focal de travail est l'activité de médiation (*middleware*) plutôt que le niveau applicatif des systèmes d'administration. Nous proposons, dans le chapitre suivant, un canevas générique applicable à tout type d'applications d'administration (e.g. l'administration des performances, de la sécurité, de la facturation, etc.). Nous nous focalisons sur les aspects liés à la supervision plutôt que le contrôle des systèmes administrés.

Systèmes d'administration classique. Bien que notre travail présente une rupture avec les systèmes classiques sur le plan architectural, nous partageons certains aspects communs, notamment au niveau des fonctions d'administration et des protocoles d'administration.

Au niveau architectural, nous proposons une architecture logicielle radicalement configurable et hautement modulaire. Cette approche architecturale nous distingue des

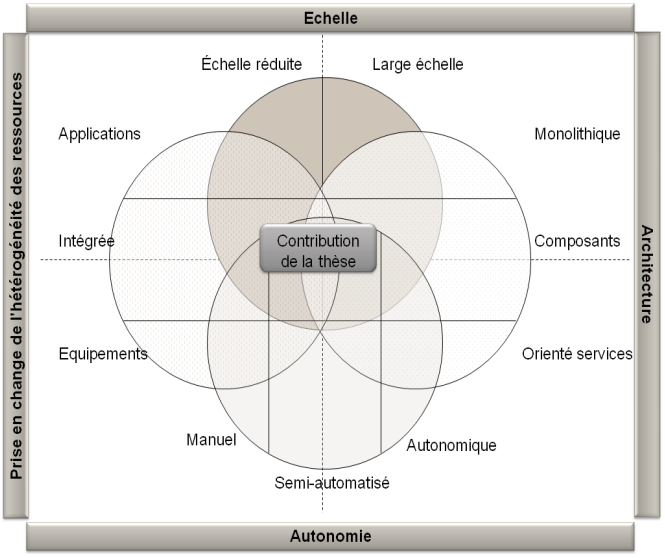


FIG. 4.1 – Positionnement par rapport à l'état de l'art

systèmes d'administration classiques monolithiques. Au niveau de la gestion de l'hétérogénéité, nous nous distinguons, avec notre approche d'encapsulation par composants, aussi bien des ressources administrées que du système d'administration lui même. Cette approche se caractérise par une faible intrusivité. Elle est donc assez facilement applicable aux systèmes patrimoniaux. Nous nous distinguons également par le support de différents modèles d'administration (centralisé, hiérarchique, hybride, etc.).

Quant aux protocoles d'administration, notre travail présente une complémentarité plutôt qu'une rupture avec les standards d'administration. Nous intégrons dans notre prototype des adaptateurs des protocoles standards tels que SNMP [JCD90]. Nous nous appuyons également sur JMX [JMX04] pour administrer les applicatifs que nous supervisons. Notre approche ouverte permet également d'ajouter facilement la prise en compte de tout autre protocole.

Systèmes d'administration orientés architecture et à base de composants. Notre travail s'inscrit dans le cadre des travaux d'administration à base de composants adoptant une approche d'administration fondée sur l'architecture. Nous nous distinguons des systèmes d'administration orientés composants étudiés par le support de différents modèles d'administration et par sa généricité. L'architecture que nous proposons peut être appliquée aussi bien à l'administration de grilles de calcul qu'à la gestion de réseaux de capteurs, par exemple.

Systèmes d'administration multi-échelles. Notre travail traite la problématique d'administration multi-échelles à travers une approche architecturale. Contrairement à certains travaux qui abordent cette problématique en se focalisant uniquement sur les modèles d'administration [YpD92, KB97, CL02], sur le protocole d'échange de données d'administration [Gol96, YD04] ou encore sur l'approche de rafraîchissement des données [LS05], nous abordons cette problématique selon un angle architectural. Nous estimons que l'architecture d'un système d'administration est complémentaire aux protocoles et aux modèles d'administration. Nous pensons que les avancées tech-

nologiques en matière d'architecture logicielle et les nouvelles approches de fabrication des systèmes d'administration peuvent contribuer au développement d'une nouvelle génération de systèmes plus flexibles et plus autonomes, et capables d'évoluer dans un contexte multi-échelles.

Systèmes d'administration autonomique. Ce travail propose une plateforme d'administration adaptée à l'administration autonomique multi-échelles. Nous expliquons dans les chapitres suivants comment l'implémentation orientée composants de notre système d'administration confère plus de possibilités de reconfiguration et d'introspection applicative lors de l'exécution. D'autres systèmes tels que Jade [SBH05, PVRY05] proposent de l'administration autonomique avec des plateformes orientées composants. Nous nous distinguons de ces travaux par plus de généricité (indépendance vis à vis du domaine d'application) et par la gestion de l'aspect large échelle qui n'a pas été abordé par ces travaux.

TAB. 4.1 – Table synthèse positionnement

Propriété	Systèmes classiques	Systèmes à composants	Notre approche
Multi-échelles	centralisé	centralisé	centralisé/distribué/hiérarchique/ hybride
Architecture	monolithique	flexible	flexible
Hétogénéité	modèle d'information générique/ systèmes d'administration multiples/ adaptateurs	ad hoc/ encapsulation	encapsulation

4.4 Contributions

Cette thèse présente trois contributions à l'administration des systèmes composés de ressources hétérogènes.

4.4.1 Une approche architecturale d'administration muti-échelles

Notre première contribution est une approche architecturale d'administration dans un contexte multi-échelles. Cette approche s'articule autour de deux principes :

- Le premier principe est *l'administration à base de domaines*. Un domaine peut être vu comme un moyen de regrouper des ressources administrées ou d'autres sous-domaines.
- Le second principe consiste à appliquer *l'administration fondée sur l'architecture des systèmes administrés* afin de pallier les difficultés d'administration liées à l'hétérogénéité des ressources. Cette contribution sera détaillée dans le chapitre suivant.

4.4.2 Un canevas intergiciel d'administration orienté composants

La seconde contribution de ce travail est le canevas DASIMA. Il s'agit d'un intergiciel pour l'administration intégrée dans un contexte multi-échelles basé sur notre approche architecturale. Nous nous sommes appuyés sur le modèle à composants Fractal [BCL+06]

pour la modélisation et la réalisation de ce canevas. Nous détaillons l'architecture fonctionnelle et l'architecture technique de ce canevas dans le chapitre 6.

4.4.3 Evaluation

Afin de valider notre approche de gestion, nous avons utilisé le canevas DASIMA pour administrer un système M2M[1] développé par Orange. Plusieurs expérimentations ont été effectuées pour évaluer les performances du canevas DASIMA dans un contexte multi-échelles. Ces expérimentations seront détaillées dans le chapitre 7.

4.5 Organisation des chapitres de la contribution

La suite de ce document s'organise de la façon suivante : le **chapitre 5** présente notre approche architecturale d'administration multi-échelles. Nous reprendrons les défis soulevés et montrons en quoi nos travaux répondent bien à ces défis. Le **chapitre 6**, décrit l'architecture globale du canevas d'administration que nous avons implémenté dans le cadre de ce travail. Nous présentons également dans ce chapitre les différents services d'administration supportés par notre canevas d'administration. Enfin, le **chapitre 7** présente l'évaluation de notre travail avec une application réelle de France Télécom développée dans le cadre du contexte M2M.

[1] Machine to Machine

Chapitre 5

Approche architecturale d'administration multi-échelles

Contents

5.1 Introduction 106
5.2 Approche d'administration 106
5.2.1 Administration fondée sur l'architecture 106
5.2.2 L'élément administré 107
5.2.3 Administration à base de domaines 108
5.3 Combinaison de l'architecture et des domaines 108
5.3.1 Processus d'administration fondée sur l'architecture 109
5.3.2 Processus d'administration à base de domaines 109
5.4 Administration fondée sur l'architecture des systèmes administrés 111
5.4.1 Cycle de vie d'un élément administré 111
5.4.2 Mise à jour des éléments administrés 112
5.5 Administration basée sur les domaines 112
5.5.1 Choix du critère de découpage des domaines 113
5.5.2 Interactions et rôles des domaines 114
5.5.3 Composition des domaines 114
5.5.4 Stratégies d'administration à base de domaines 116
5.5.5 Dimensionnement des domaines 119
5.5.6 Approvisionnement des domaines 120
5.5.7 Auto-administration des domaines 120
5.6 Conclusion 120

Ce chapitre introduit notre approche d'administration des systèmes hétérogènes dans un contexte multi-échelles. Nous présentons dans un premier temps cette approche qui s'articule autour de deux grandes lignes : (a) l'administration fondée sur l'architecture des systèmes administrés et (b) l'administration basée sur les domaines. Nous développons, par la suite, le processus à suivre dans le cadre de notre approche pour combiner ces deux grands axes, et détaillons ensuite chacun de ces deux axes.

5.1 Introduction

Parmi les principales caractéristiques des systèmes d'administration classiques abordés dans le chapitre 2 de l'état de l'art figure une administration principalement basée sur un modèle centralisé, une gestion ad hoc de l'hétérogénéité des ressources administrées et des architectures monolithiques.

Bien que les systèmes d'administration à base de composants, étudiés dans le chapitre 3 de l'état de l'art, pallient l'aspect monolithique des systèmes d'administration classiques en proposant des systèmes plus modulaires, ils restent, généralement, des systèmes centralisés. De plus, ces systèmes proposent, pour la plupart, des solutions ad hoc pour la gestion de l'hétérogénéité des ressources administrées.

Afin de pallier ces limitations, nous proposons dans ce chapitre une nouvelle approche architecturale d'administration intégrée multi-échelles. Nous introduisons dans les sections qui suivent les fondements de cette approche.

5.2 Approche d'administration

Notre approche s'articule autour de deux grands axes :
- **l'administration fondée sur l'architecture des systèmes administrés** et,
- **l'administration à base de domaines**.

5.2.1 Administration fondée sur l'architecture

L'administration fondée sur l'architecture des systèmes administrés est l'approche que nous retenons pour la gestion des ressources hétérogènes. Ce choix est motivé par la capacité des descriptions architecturales à offrir une vue homogène commune des ressources hétérogènes, l'une des caractéristiques des systèmes à large échelle. Cette approche s'appuie sur les modèles d'architecture comme base pour la construction des fonctions d'administration. Autrement dit, selon cette approche, la configuration du système à administrer est décrite en utilisant les éléments de son architecture. Afin d'atteindre cet objectif, nous appliquons deux principes qui ont été identifiés par [ABKS06]. Ces principes peuvent être appliqués à des granularités différentes selon le niveau d'adaptation souhaité et sont comme suit :

- Le modèle d'architecture du système à administrer doit être explicite en termes des ressources à administrer (les services, les nœuds, etc) ainsi que leurs relations (encapsulations et liaisons). Il doit être également présent et modifiable à l'exécution.
- Chaque ressource administrée doit être une unité séparée de configuration. Ceci est nécessaire en particulier pour la reconfiguration , ou afin de pouvoir remplacer une ressource administrée par une autre dans le modèle.

Dans les sections suivantes, nous parlerons plutôt d'éléments administrés qui représentent des instances des ressources administrées dans les modèles architecturaux. Ces entités peuvent être regroupés dans des *domaines*.

5.2.2 L'élément administré

Nous nous appuyons sur le concept d'élément administré (*managed element* (ME)) pour extraire une représentation homogène des ressources hétérogènes. Nous définissons ce concept de la manière suivante :

Définition 5.1. *Un* **élément administré** *est une représentation architecturale d'une ou plusieurs ressources administrées du monde réel.*

Il s'agit de la brique de base de l'approche d'administration fondée sur l'architecture des systèmes administrés. Nous proposons dans ce travail un modèle générique d'élément administré. Nous détaillons ce modèle dans le chapitre suivant.

Un élément administré interagit directement avec une ou plusieurs ressources qu'il représente. Il joue le rôle de médiateur entre l'infrastructure de gestion et les ressources administrées réelles. Cette fonction de médiation permet de pallier les problèmes liés à l'hétérogénéité des ressources administrées. En effet, les éléments administrés permettent de fournir une vue homogène d'un ensemble de ressources hétérogènes.

La figure 5.1 montre un exemple d'élément administré qui représente une passerelle domestique de service. Cette entité englobe un ensemble d'éléments administrés qui représentent les différents équipements connectés à la passerelle domestique tels qu'un PDA, un ordinateur portable ou une télévision. Chacune de ces entités est caractérisée par un ensemble de propriétés (e.g version du *firmeware*, services offerts, caractéristiques systèmes, CPU et mémoire, etc.) et un certain nombre d'évènements (e.g. arrêt de la passerelle de service, saturation des ressources mémoire ou CPU, etc.).

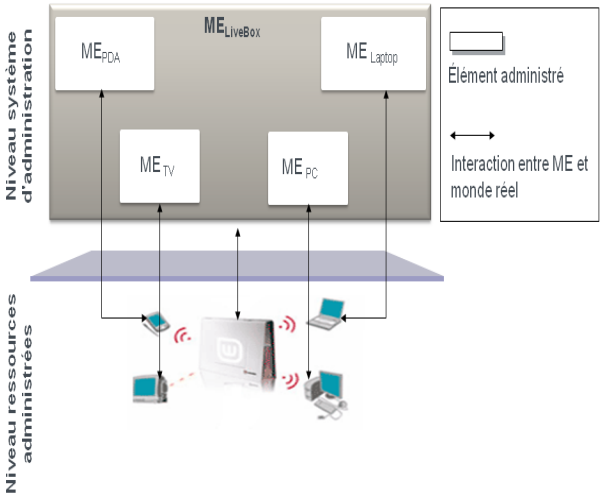


FIG. 5.1 – Exemple d'élément administré (ME) : passerelle de service

5.2.3 Administration à base de domaines

Le **domaine d'administration** est la pierre angulaire de notre approche architecturale d'administration. Nous définissons ce concept comme suit :

Définition 5.2. *Un* **domaine d'administration** *est un regroupement de ressources administrées qui partagent une ou plusieurs propriétés communes.*

La plupart des systèmes d'administration classiques peuvent être considérés comme « mono-domaine ». Il s'agit de systèmes d'administration centralisés avec un seul domaine qui regroupe l'ensemble des ressources administrées existantes. Dans un contexte multi-échelles, cette approche peut rencontrer des limitations pour passer à l'échelle. Dès lors, le domaine d'administration peut servir de moyen pour fragmenter l'ensemble des ressources administrées d'un système administré en ensembles plus réduits. Cette fragmentation peut simplifier considérablement la tâche d'administration. Elle met à la disposition de l'administrateur un mécanisme de gestion de groupes de ressources qui remplace la gestion individuelle des entités à gérer.

Dans le cadre de notre travail, un domaine peut contenir deux types d'éléments administrés : les (a) représentants des ressources administrées et (b) les représentants des domaines d'administration eux-mêmes. Cette classification nous mène à une spécialisation des domaines en fonction de leur contenu. Ainsi, nous distinguons entre :

- **Les domaines de ressources.** Cette catégorie de domaines gère tout type de ressources, sauf les ressources de type domaine. La figure 5.2 montre l'exemple d'un tel domaine (domaine D_A) qui contient un ensemble n de ressources administrées. Une ressource administrée est, par exemple, un serveur de base de données ou un équipement réseau.
- **Les domaines de domaines.** Contrairement au domaine des ressources, un domaine de domaines gère exclusivement des ressources de type domaine. Il permet de piloter un ensemble de domaines d'administration. Ce type de domaine peut servir pour la gestion du système d'administration lui-même. La figure 5.2 illustre l'exemple d'un domaine de domaines avec le domaine D_C.
- **Les domaines de domaines et de ressources.** Ces domaines hybrides gèrent les domaines au même titre que les autres ressources. (voir domaine D_B dans la figure 5.2).

5.3 Combinaison de l'architecture et des domaines

Nous avons défini dans la section précédente les notions de domaine d'administration et d'administration fondée sur l'architecture des systèmes administrés. Nous expliquons dans les prochaines sections le processus à suivre pour combiner ces deux approches. Nous rappelons que l'objectif est d'administrer des ressources hétérogènes dans un contexte multi-échelles.

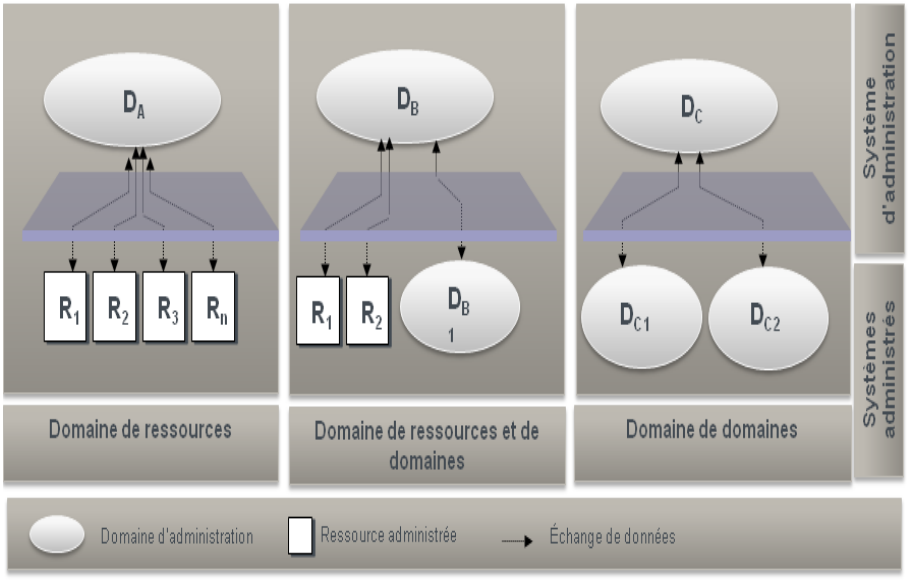


FIG. 5.2 – Classification des domaines selon leur contenu

5.3.1 Processus d'administration fondée sur l'architecture

Nous commençons par les étapes à suivre pour extraire des représentations architecturales correspondant aux ressources existantes.

- **Modélisation des ressources à administrer.** La première étape de notre approche consiste à modéliser les ressources à administrer. Cette étape permet d'identifier les informations et les éventuels évènements qu'on souhaite observer au niveau de la ressource à administrer. Le résultat de cette étape est un modèle abstrait de la ressource administrée.
- **Instrumentation des ressources.** Après avoir modélisé la ressource à administrer, il est indispensable d'instrumenter ces ressources afin de pouvoir alimenter le modèle abstrait avec des données concrètes.
- **Découverte des ressources et instanciation des représentations.** Cette étape consiste à découvrir l'existence d'une ressource à administrer et à instancier les représentations correspondantes au niveau du système d'administration. La phase de découverte des ressources peut être effectuée à l'initiative de la ressource administrée, qui déclare son apparition ou disparition au système d'administration. Mais elle peut être également réalisée à l'initiative du système d'administration (ou l'administrateur) qui peut initier une opération de recherche de ressources dans un périmètre bien spécifique (une plage d'adresse IP ou l'URL d'un serveur MBean, par exemple). Elle peut être déclarative : l'administrateur ou une application d'administration ajoute alors une ressource directement.

5.3.2 Processus d'administration à base de domaines

L'objectif de cette section est de nous montrer les différentes étapes à suivre afin de mettre en place un domaine d'administration. Ces étapes sont résumées dans la figure 5.3.

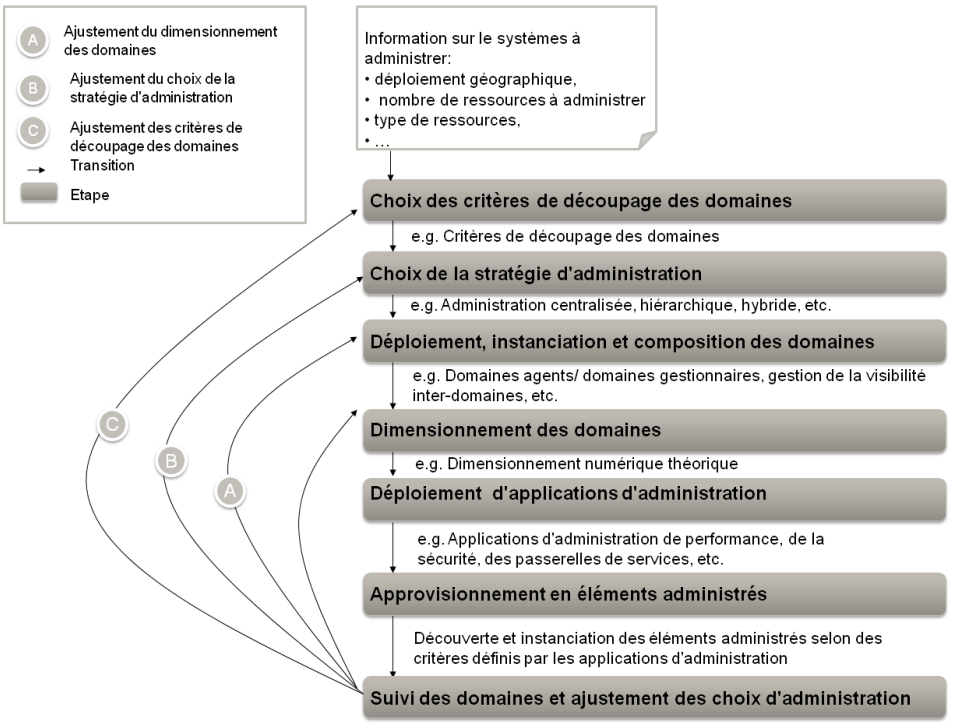


FIG. 5.3 – Processus d'administration à base de domaines

- **Le choix des critères de découpage des domaines.** Ce choix relève de la responsabilité de l'administrateur et dépend de plusieurs caractéristiques du système à administrer. Parmi ces caractéristiques nous pouvons citer la distribution géographique ou l'ampleur numérique du système administré. Nous présentons dans la section 5.5.1 les différents critères de découpage des domaines.
- **Le choix de la stratégie d'administration.** Le choix de la stratégie d'administration adéquate est déterminant pour le passage à l'échelle de l'infrastructure de gestion. Ce choix définit le modèle d'agencement des domaines et leur fonctions et périmètre d'administration. Plusieurs paramètres, tels le nombre des requêtes d'administration, le volume des données échangés ou la nature des requêtes échangées peuvent influencer ce choix. Ces paramètres peuvent concerner aussi bien l'environnement matériel que l'environnement logiciel ou réseau du système d'administration et du système à administrer. Nous présentons une classification de ces variables dans la section 5.5.4 de ce chapitre.
- **Le déploiement et la composition des domaines.** Afin de concrétiser la vue organisationnelle résultante de l'étape précédente, des domaines d'administration sont instanciés et déployés, et dans certains cas composés. L'instanciation consiste à la création des domaines. Le déploiement correspond à leur copie dans leur environnement d'exécution matériel. Quant à la composition, elle désigne la possibilité de tisser des liens et des relations entre les domaines (hiérarchie, agrégation, etc.).
- **Le dimensionnement des domaines.** Cette opération est indispensable pour le bon fonctionnement d'un domaine. Elle permet de dresser un profil de charge sim-

plifié du domaine. Le dimensionnement permet d'extraire des opérations telles que le nombre maximum de requêtes supportées, le nombre maximum de ressources administrées, etc.

- **Déploiement d'applications d'administration.** Une fois le domaine déployé et dimensionné, nous pouvons déployer les applications d'administration. Ces applications interagissent directement avec les domaines qui les intéressent afin d'effectuer des opérations d'administration.
- **L'approvisionnement en éléments administrés.** La première opération que les applications d'administration peuvent effectuer est l'approvisionnement des domaines en ressources administrées. Cette opération d'approvisionnement consiste en la sélection et puis l'affectation des ressources administrées aux domaines. A l'issue de cette opération, des instances d'éléments administrés abstraits sont créées au niveau des domaines. Ces éléments sont des représentations architecturales des ressources administrées du monde réel.
- **Le suivi et l'évolution de la stratégie d'administration.** Au cours de l'exécution, un suivi régulier de la plateforme d'administration doit être effectué. Ce suivi permet de planifier l'évolution de la plateforme d'administration et d'anticiper des éventuelles défaillances. Quant à son évolution, elle est radicalement simplifiée si on opte pour une approche à composants.

5.4 Administration fondée sur l'architecture des systèmes administrés

5.4.1 Cycle de vie d'un élément administré

Chaque élément administré évolue selon un cycle de vie bien défini et illustré dans la figure 5.4. Les étapes du cycle de vie sont comme suit :

- **Initialisé**. Un élément administré est initialement instancié et par la suite initialisé avec des informations de gestion recueillies auprès de la ressource qu'il représente. Cette phase comprend par exemple la définition des propriétés gérées et des ressources avec laquelle un élément administré interagit pour mettre à jour son état.
- **En exécution**. Un élément administré ne peut commencer à interagir avec les ressources qu'il représente qu'à partir du moment où il est initialisé et lancé en exécution.
- **Migré**. Cet état concerne les éléments administrés qui changent de domaine au cours de leur cycle de vie. Le domaine initial peut dans certaines situations avoir besoin d'informations concernant le devenir d'un élément administré qui a migré vers un autre domaine.
- **Vérouillé**. Un élément administré peut être verrouillé par une application d'administration afin de pouvoir effectuer des opérations de mise à jour ou de contrôle par exemple.
- **En attente**. L'activité d'un élément administré peut être suspendue à la demande de l'administrateur du ME via les applications d'administration. Cet état peut servir par exemple à contrôler la charge d'activité au niveau des domaines. L'activité d'un ensemble de ME peut être suspendue afin de libérer plus de ressources de calcul et éviter la défaillance du système d'administration suite à une surconsommation des ressources disponibles.

– **Effacé**. Un élément administré effacé n'existe plus dans le système d'administration. C'est l'étape finale du cycle de vie d'un élément administré.

Etape	Description
0	Début du cycle de vie
1	Instanciation du ME par la plateforme d'administration.
2	Activation du ME
3	Verrouillage du ME par la plateforme d'administration
4	Migration du ME d'un domaine à un autre
5	Déverrouillage du ME
6	Reprise de l'activité du ME
7	Suspension de l'activité du ME
8	Reprise de l'activité du ME
9	Suppression du ME
10	Fin du cycle de vie

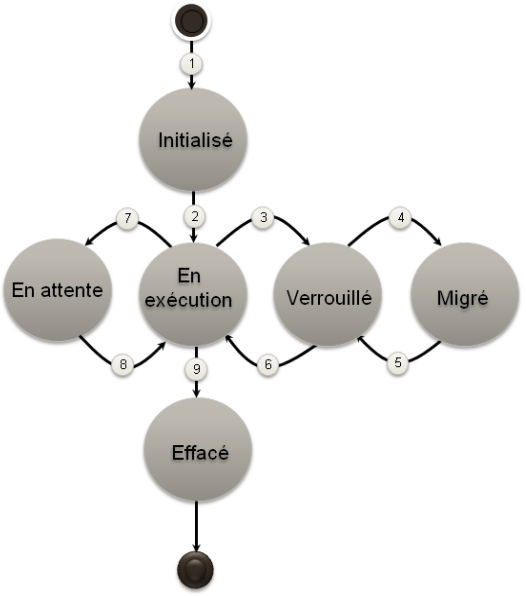


FIG. 5.4 – Cycle de vie d'un élément géré

5.4.2 Mise à jour des éléments administrés

Un élément administré interagit régulièrement avec les ressources qu'il représente afin de mettre à jour son état. Cette interaction peut suivre trois modèles :

– **Pull.** Un élément administré interagit avec la ressource gérée à l'initiative de l'administrateur ou de l'élément administré lui-même. (voir le mode 1 de la figure 5.5)

– **Pooling**. Ce modèle d'interaction est similaire au modèle précédent sauf que les requêtes sont exécutées à une fréquence régulière et prédéfinie à l'initiative de l'élément administré correspondant. Un *scheduler* est donc positionné au niveau de l'élément administré. Il gère la fréquence d'activation des requêtes. Ce mode requiert donc une configuration de l'élément administré afin de gérer l'alternance entre l'activation et la mise en attente de l'élément administré (voir mode 2 de la figure 5.5).

– **Push**. Selon ce modèle d'interaction, une ressource prend l'initiative d'informer un élément administré de chaque éventuel changement d'état. Le mode 3 de la figure 5.5 montre un exemple de ce mode d'interaction. Avec cette approche, nous pouvons économiser certaines interactions par rapport au modèle précédent. Cette économie est matérialisée dans la réduction des flux de données sur le réseau et du temps de traitement au niveau des infrastructures de gestion. La mise en œuvre de cette approche requiert une instrumentation de la ressource administrée pour qu'elle puisse prendre l'initiative et envoyer l'information sélectionnée à l'entité administrative concernée.

5.5 Administration basée sur les domaines

Après avoir présenté les éléments administrés, leur cycle de vie et les modèles de leurs mise à jour, nous détaillons dans la section suivante l'autre brique de base de notre ap-

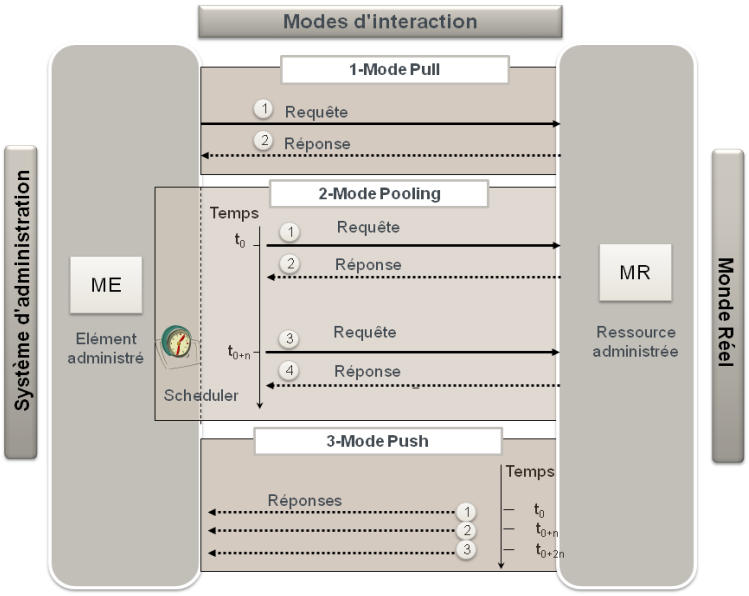


FIG. 5.5 – Modèles d'interaction entre ressources administrées et éléments administrés

proche. Il s'agit des domaines d'administration qui servent de conteneurs aux éléments administrés.

5.5.1 Choix du critère de découpage des domaines

Nous pouvons envisager plusieurs critères de découpage des domaines selon la classification des domaines d'administration. Ces critères peuvent être d'ordre fonctionnel, organisationnel, géographique, etc. Nous présentons dans cette section quelques exemples de critères à titre indicatif. D'autres critères peuvent être définis par l'administrateur, si besoin.

- **Le critère géographique.** En appliquant ce critère sur un parc d'équipements, par exemple, on peut obtenir un ou plusieurs domaines par site géographique. Ce découpage peut être intéressant dans le cas des parcs de ressources géographiquement éclatées. Les domaines locaux peuvent assurer une gestion locale des ressources avec des décisions à court terme et grâce à une proximité des ressources. Alors que des domaines plus globaux peuvent fournir une vue synoptique aux administrateurs afin de prendre des décisions à long et à moyen terme (planification de l'évolution de l'infrastructure, dimensionnement du réseau, etc.).
- **Le critère organisationnel.** Selon ce critère, les domaines sont calqués sur l'organisation du système réparti à administrer. Ce découpage peut être intéressant pour les système d'information des entreprises. Dans de tels cas, un domaine peut alors correspondre à une division entière de l'entreprise (domaine comptable, domaine ressources humaines, domaine production, etc.).
- **Le critère fonctionnel.** On peut également spécialiser les domaines selon leurs fonctions d'administration. Dans ce cas, un domaine qui gère la sécurité peut communiquer avec un domaine qui gère les performances pour détecter les éventuelles attaques par déni de services par exemple.

– **Le critère numérique.** Ce critère consiste à découper les domaines en fonction de leur capacité d'accueil des éléments administrés. Ce découpage est effectué durant les premières phases de paramétrage du système d'administration. Chaque domaine est donc configuré en fonction de ses capacités de traitement en termes de ressources de calcul disponibles (e.g. CPU, Mémoire, disque, etc.). Ce paramétrage n'est pas figé tout au long du cycle de vie des domaines. Il peut évoluer en fonction des besoins d'administration.

5.5.2 Interactions et rôles des domaines

Durant les interactions inter-domaines, un domaine peut endosser l'un de ces trois rôles :

Domaine agent. Un domaine agent joue le rôle d'un centre de collecte d'information. Il interagit directement avec les ressources gérées. Ensuite, il envoie les informations collectées à un domaine gestionnaire ou à un domaine intermédiaire. Un exemple de domaine agent est illustré dans la figure 5.6 par le domaine D_{Agent}.

Domaine gestionnaire. Un domaine gestionnaire, quant à lui, joue le rôle d'un centre de contrôle. En effet, c'est à ce niveau que sont prises les décisions d'administration. Pour ce faire, il procède en trois étapes :
– une étape de collecte d'information à travers les domaines agents et les domaines intermédiaires,
– une phase d'analyse et de décision qui requière une interaction avec un administrateur humain ou un système automatisé pour l'aide à la décision
– une phase d'action, pour appliquer les décisions prises.
La figure 5.6 décrit un exemple de domaine gestionnaire illustré par le domaine $D_{Gestionnaire}$.

Domaine gestionnaire intermédiaire. Un domaine intermédiaire joue le rôle de médiateur entre un domaine gestionnaire et un domaine agent. La figure 5.6 montre un exemple de domaine intermédiaire $D_{Intermediaire}$. Dans la plupart des cas, un domaine intermédiaire peut être à la fois gestionnaire et agent. Il joue un rôle très important pour le passage à l'échelle des systèmes d'administration grâce aux possibilités d'extension qu'il apporte au modèle classique gestionnaire-agent.

5.5.3 Composition des domaines

La composition de domaines désigne les liens que nous pouvons tisser entre les domaines. En observant les stratégies d'administration, nous avons distingués 3 relations possibles entre les domaines :
– **La disjonction.** Les domaines disjoints sont des domaines qui ne partagent aucun élément administré. Chaque domaine gère ses éléments indépendamment des autres domaines. La figure 5.7 à gauche montre un exemple de deux domaines D_1 et D_2 qui sont complètement disjoints.
– **L'intersection.** La figure 5.7, au centre, illustre un exemple de domaines qui s'intersectent et qui se partagent des éléments ou des fragments d'éléments administrés. Cette situation est fréquente dans le cas du découpage fonctionnel des domaines d'administration.

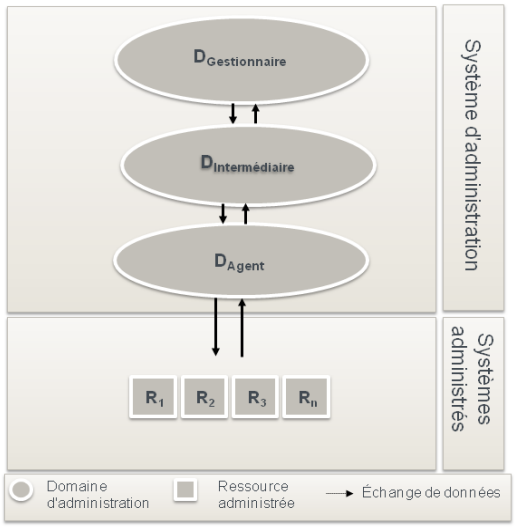


FIG. 5.6 – Rôles d'un domaine

– **L'inclusion.** Cette relation se traduit par une imbrication des domaines les uns dans les autres. C'est un cas fréquent dans le cadre de la gestion de grands systèmes distribués et des larges réseaux répartis. Un domaine de plus haut niveau collecte et agrège les données et les informations issues des domaines initiaux (au plus près des ressources administrées). La 5.7, à droite, montre un exemple de deux domaines imbriqués. Le domaine D_1 inclut tous les éléments du domaine D_2.

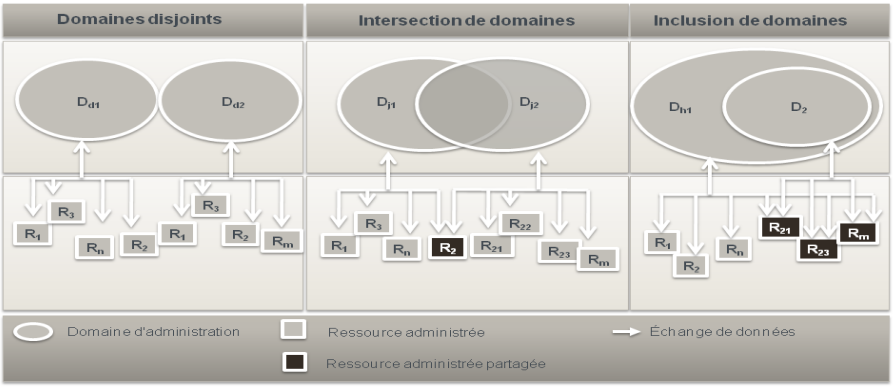


FIG. 5.7 – Relations inter-domaines

Les relations illustrées dans la figure 5.7 impliquent des interactions inter-domaines. Nous abordons plus en détail ces modalités d'interaction dans la section suivante. Le choix du critère de découpage des domaines est l'un des paramètres qui vont influencer le choix de la stratégie globale d'administration. Ce choix ainsi que des exemples de stratégies d'administration sont détaillés dans les sections suivantes.

5.5.4 Stratégies d'administration à base de domaines

Le choix de la stratégie selon laquelle les domaines seront agencés et les fonctions d'administration exercées par plusieurs domaines sont déterminants pour le passage à l'échelle des infrastructures d'administration. Il n'existe pas de meilleure stratégie dans l'absolu. Le choix d'une stratégie ou d'une autre est effectué au cas par cas. Il est influencé par un certain nombre de paramètres :

- **Paramètres liés à l'environnement matériel du système d'administration :** capacité de calcul, capacité de mémoire, capacité de stockage de données, etc.
- **Paramètres liés à l'environnement réseau du système d'administration :** Ces paramètres englobent, entre autre, la bande passante, les protocoles de sécurité du réseau, etc.
- **Paramètres liés aux systèmes administrés :** Le nombre des ressources administrées, leur autonomie ou encore leurs capacités de traitement sont des exemples possibles des paramètres liés aux ressources d'administration.

Nous présentons un exemple de choix de stratégie d'administration dans le chapitre d'expérimentation. La suite présente des stratégies d'administration possibles qu'on peut appliquer avec les domaines d'administration. Il s'agit de stratégies : centralisée, répartie ; hiérarchique et hybride.

5.5.4.1 Stratégie d'administration centralisée

Cette stratégie est la plus simple à mettre œuvre. La figure 5.8 illustre le modèle global d'une stratégie d'administration centralisée. Seul le domaine $D_{Gestionnaire}$ assure la fonction d'administration de l'ensemble de ressources $\{R_1,..., R_n\}$.

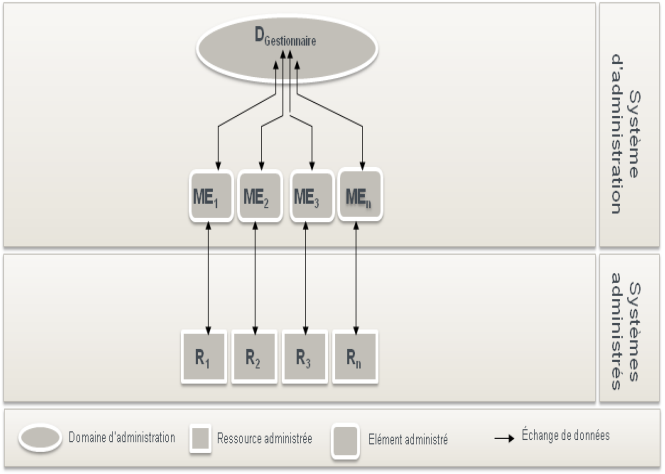


FIG. 5.8 – Exemple de stratégie d'administration centralisée à base de domaines

Cette stratégie est intéressante dans le cas où le nombre de ressources à gérer et de requêtes d'administration est restreint. Dans le cas contraire, le domaine unique devient un goulet d'étranglement. Afin d'éviter ce genre de situations, deux solutions sont envi-

sageables ; (a) une **extension des capacités matérielle** de la machine hôte de l'infrastructure de gestion ou (b) une **extension logicielle** de l'infrastructure d'administration. Nous proposons dans les sous-sections suivantes d'étudier différentes possibilités liées à la seconde approche.

5.5.4.2 La stratégie d'administration répartie

La **stratégie d'administration répartie** consiste à répartir le rôle de gestionnaire entre plusieurs entités. Dans notre cas ces entités sont des domaines. Contrairement à la stratégie d'administration centralisée, où il y a un point de contrôle et de décision unique, cette approche permet de promouvoir les capacités de passage à l'échelle des infrastructures de gestion.

Comme l'illustre la figure 5.9, les domaines peuvent jouer un rôle à la fois de gestionnaire et d'agent. Les gestionnaires ont la possibilité de communiquer entre eux afin d'échanger des informations et des requêtes de gestion diverses. La visibilité des ressources administrées n'est pas limitée au domaine gestionnaire. Elle s'étend aux domaines avec lesquels communique ce dernier. La figure 5.9 montre deux domaines D_A et D_B qui jouent à la fois le rôle gestionnaire et celui de l'agent. Les deux domaines communiquent afin d'échanger les informations et les requêtes de gestion.

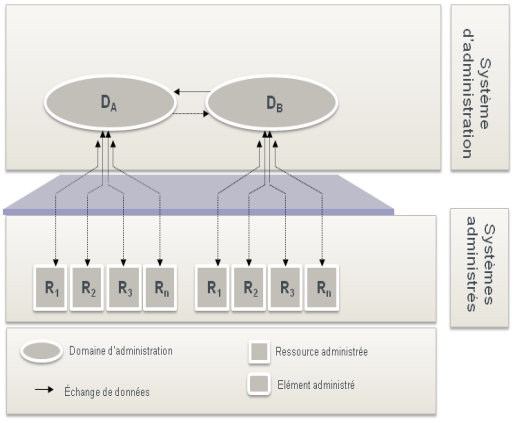


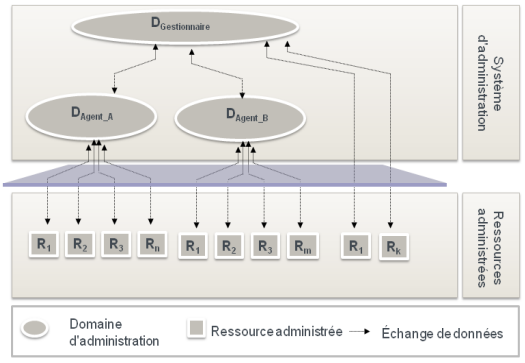
FIG. 5.9 – Exemple de stratégie d'administration répartie

Il importe de souligner qu'il est plus efficace pour le domaine D_A de traiter des requêtes sur les ressources qu'il gère en direct, que des requêtes sur des ressources gérées par le domaine D_B auquel D_A est connecté. D'où l'importance de combiner le choix de la stratégie d'administration avec un bon choix de découpage.

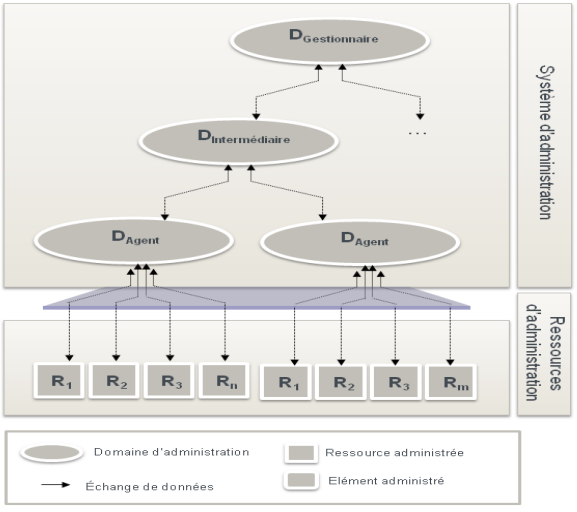
5.5.4.3 Stratégie d'administration hiérarchique

La **stratégie d'administration hiérarchique** est une variante de la stratégie de gestion répartie. La figure 5.10 illustre un exemple possible d'application de la stratégie d'administration hiérarchique.

La figure 5.10(a) illustre une topologie avec trois domaines. $D_{Gestionnaire}$ joue le rôle de gestionnaire global. Les deux autres domaines (D_{AgentA} et D_{AgentB}) jouent le rôle d'agent vis à vis du domaine gestionnaire et des ressources administrées. En effet, un domaine gestionnaire permet de contrôler à la fois les ressources qu'il gère et les ressources considérées dans le périmétres des autres domaines agents.



(a) Exemple de stratégie d'administration hiérarchique à deux niveaux de médiation



(b) Exemple de stratégie d'administration hiérarchique à 3 niveaux de médiation

FIG. 5.10 – Exemples de stratégies d'administration hiérarchique

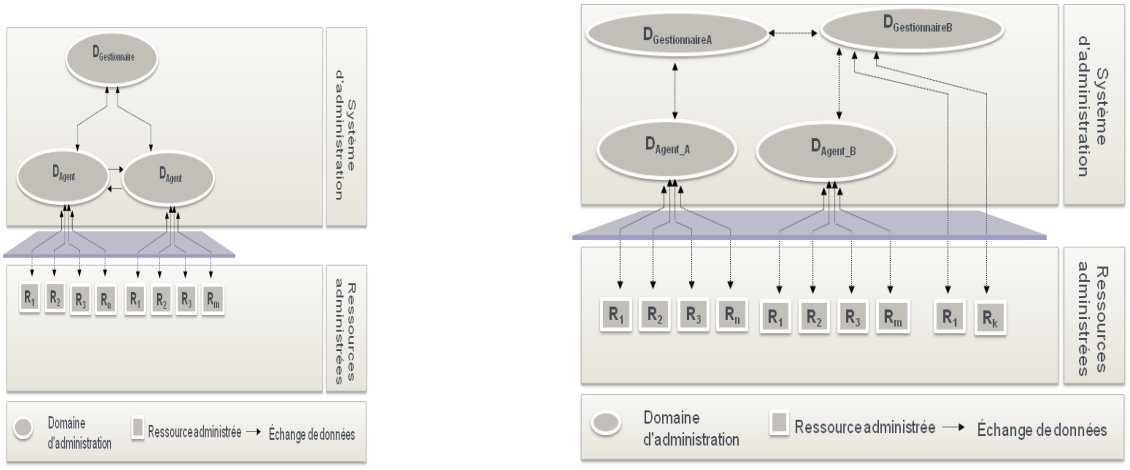
Le deuxième exemple (voir figure 5.10(b)) ajoute un autre niveau de médiation. Ce niveau peut être appliqué dans le cas où le nombre de domaines agents augmente considérablement. Ce domaine intermédiaire joue un rôle hybride (à la fois le rôle du gestionnaire et l'agent). Il collecte les informations de gestion au près des domaines D_{Agents} et les expose au domaine $D_{Gestionnaire}$. Il permet également d'exposer les ressources gérées par les domaines D_{agent} pour les rendre visibles (totalement ou en partie) par les gestionnaires.

5.5.4.4 Les stratégies de gestion hybrides

En combinant les trois stratégies présentées au début de cette section nous pouvons obtenir plusieurs stratégies hybrides. La figure 5.11 illustre deux variantes possibles de stratégies hybrides. La première stratégie est décrite dans la figure 5.11(a). Elle consiste à combiner l'administration centralisée (avec le point de contrôle unique du domaine $D_{Gestionnaire}$), la gestion hiérarchique (un domaine gestionnaire et deux domaines agents) et la gestion répartie (possibilité de communication entre domaines agents).

La figure 5.11(b) montre un autre exemple de stratégie de gestion hybride en combinant cette fois la gestion répartie et la gestion hiérarchique. Contrairement à l'exemple précédent, nous avons deux domaines gestionnaires au lieu d'un.

Les deux exemples requièrent des domaines agents dotés d'une capacité de traitement importante et requièrent un lien réseau fluide et disponible pour effectuer les opération de gestion et de synchronisation entre les deux domaines D_{Agent}.



(a) Administration hybride répartie

(b) Administration hybride centralisée

FIG. 5.11 – Exemples de topologies d'administration hybrides

5.5.5 Dimensionnement des domaines

Le dimensionnement des domaines consiste à définir et à positionner les paramètres permettant le déroulement des opérations d'administration telles que la gestion de la consommation des ressources systèmes (e.g. CPU, mémoire, etc.) du domaine. Nous pouvons par exemple définir des seuils de consommation de ressources du domaine au delà desquels la plateforme d'administration alerte l'administrateur ou adopte une stratégie de gestion prédéfinie (e.g. basculer en mode performances dégradés, filtrer les applications d'administration selon les priorités, etc.).

L'approvisionnement initial peut être par la suite ajusté en fonction des données et des informations de supervision de la plateforme d'administration.

5.5.6 Approvisionnement des domaines

L'approvisionnement des domaines d'administration consiste à alimenter les domaines d'administration en éléments administrés. Nous rappelons que l'une des fonctions d'un domaine est mettre en relation des applications d'administration et des ressources administrées représentées par des éléments administrées. Les domaines d'administration entament alors une phase de découverte et d'instanciation des éléments administrés. Cette étape consiste à retrouver les ressources administrées et à instancier, par la suite, les éléments administrés correspondants. Nous abordons cette étape plus en détail dans le chapitre suivant. Nous détaillons l'aspect fonctionnel et technique de la découverte des ressources et de l'approvisionnement des domaines en éléments administrés.

A ce niveau, le processus d'administration multi-échelles n'est pas complet. Il nous manque un aspect que nous avons défini comme primordial pour l'administration multi-échelles : il s'agit de l'administration du système d'administration lui-même (auto-administration). Cela nous amène à la dernière étape de notre processus d'administration, qui consiste à assurer le suivi et l'ajustement du système d'administration et des domaines d'administration.

5.5.7 Auto-administration des domaines

Cette étape permet le suivi et l'ajustement des choix d'administration et des domaines. Il s'agit en quelque sorte de l'administration du système d'administration par lui-même. Pour ce faire, les domaines d'administration sont surveillés durant l'exécution de leurs activités. Les informations remontées peuvent concerner par exemple la consommation des ressources de calcul du domaine (CPU ou mémoire par exemple) ou le nombre et la taille des ressources qu'il gère à un instant donné. Ces informations remontées peuvent déclencher des décisions d'ajustement et de rectification pour garantir la stabilité, la robustesse ou le passage à l'échelle de la plateforme d'administration. Ces décisions d'ajustement peuvent être prises par l'administrateur ou par une application d'administration automatisée. Elles peuvent concerner l'ajustement de la stratégie d'administration, la rectification du choix de découpage des domaines, la modification du choix de composition ou un redimensionnement des domaines.

Cette fonctionnalité peut être utile dans un contexte multi-échelles avec des ressources dynamiques et imprévisibles. Nous détaillons dans les chapitres suivants les choix techniques que nous avons mis en œuvre afin de permettre l'auto-administration.

5.6 Conclusion

Nous avons présenté dans ce chapitre notre approche architecturale d'administration multi-échelles. Elle peut se résumer en deux principes. Le premier principe est l'administration fondée sur l'architecture des systèmes administrés. Ce principe permet de pallier le problème de l'hétérogénéité des ressources administrées. Quant au second principe, il consiste en l'administration basée sur les domaines. Un domaine d'administration peut être considéré comme un groupe de d'éléments administrés. Une classification plus fine des domaines a été proposée ainsi que les différentes stratégies d'administration, les organisa-

tions possibles des domaines, et les mécanismes d'administration associés.

En introduisant notre approche, nous avons mis l'accent sur la nécessité de construire un système flexible et modulaire. Nous avons estimé que ces propriétés sont indispensables dans un contexte multi-échelles. Le chapitre suivant détaille l'aspect modulaire de notre solution. Il dresse le portrait fonctionnel de la solution d'administration que nous proposons.

Chapitre 6

DASIMA : un canevas à base de composants pour la gestion intégrée multi-échelles

Contents

6.1 Architecture globale de DASIMA 124
6.2 Briques de base de l'architecture fonctionnelle 125
6.2.1 L'élément administré 125
6.2.2 Le domaine d'administration 126
6.3 Architecture fonctionnelle d'un domaine 126
6.3.1 Architecture interne d'un domaine 127
6.3.2 Conteneur des applications d'administration 128
6.3.3 Conteneur des éléments administrés 128
6.4 Les services communs d'administration 129
6.4.1 Le service de découverte des ressources 130
6.4.2 Le service de nommage des ressources 130
6.4.3 Service de gestion de bindings 131
6.4.4 Service de monitoring 131
6.4.5 Service de gestion des notifications 133
6.5 Le domaine des domaines 135
6.5.1 Fabrique des domaines 135
6.5.2 Service de paramétrage global 135
6.6 Synthèse 136

Ce chapitre présente le canevas d'administration DASIMA qui nous avons mis en place afin de valider l'approche d'administration proposée dans le chapitre précédent. Il introduit, dans un premier temps, l'architecture globale du canevas et met en exergue l'ensemble des acteurs externes qui interagissent avec DASIMA. Ensuite, il présente les briques de base sur lesquels s'appuie la conception du canevas, à savoir l'élément administré et le domaine d'administration. Le chapitre détaille ensuite l'architecture fonctionnelle des domaines et des services communs proposés par DASIMA. Il se termine par la présentation du domaine des domaines d'administration, un élément qui joue un rôle important dans la gestion multi-échelles.

6.1 Architecture globale de DASIMA

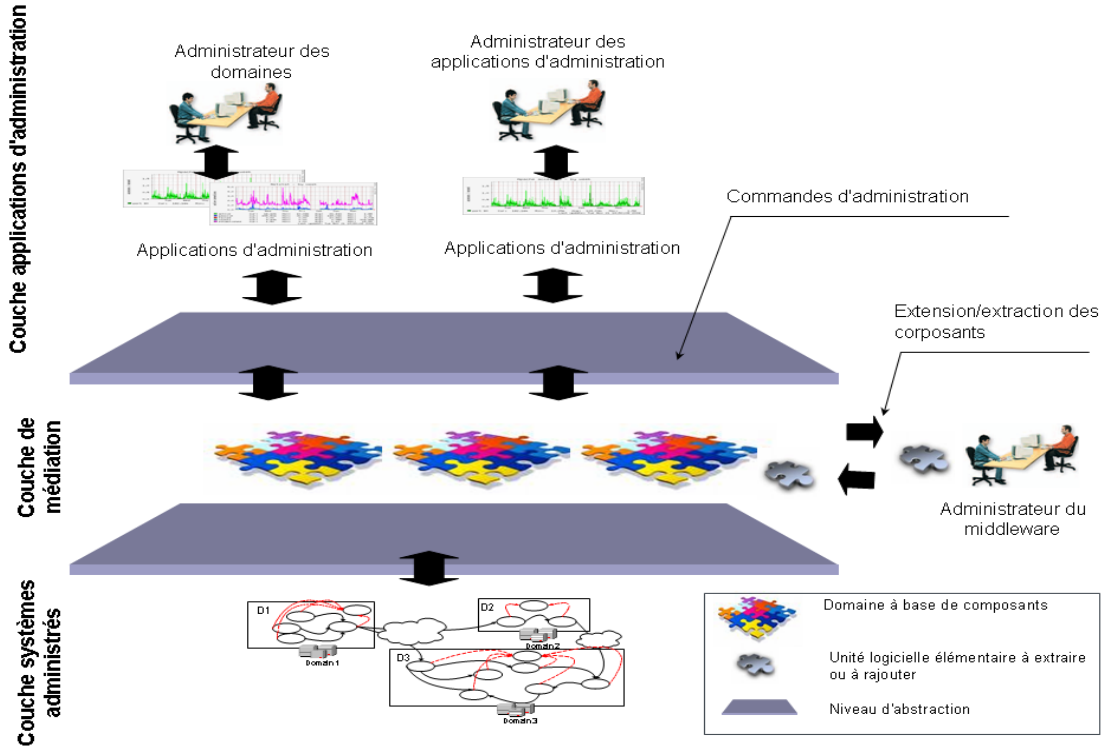


FIG. 6.1 – Architecture globale pour l'administration multi-échelles

Cette section présente le canevas d'administration DASIMA [MkL06a, MkL06b]. L'architecture de DASIMA s'appuie sur le patron architectural *Mediator* [GHVJ95]. Ce patron permet d'introduire une entité qui joue un rôle de médiation entre différents interlocuteurs. L'intérêt de l'application de ce patron architectural est double. D'une part, il permet de factoriser certains traitements entre les applications d'administration. Ces traitements incluent la collecte, la sauvegarde et la mise à jour des données d'administration. D'autre part, la couche de médiation permet d'extraire une vue homogène et cohérente à partir d'un ensemble hétérogène de ressources à gérer. La figure 6.1 présente une vue globale de cette architecture de médiation. Elle met en exergue trois niveaux d'abstraction :

- **La couche applications d'administration**. Cette couche représente l'ensemble des applications d'administration qui interagissent directement avec les systèmes administrés. Les applications d'administration peuvent varier des simples consoles de supervision à des systèmes d'administration plus sophistiqués tels que les systèmes d'administration « autonomiques ».
- **La couche de médiation**. Ce niveau d'abstraction englobe un ensemble de services communs d'administration (e.g. nommage, archivage, modèle d'information). Il est organisé en domaines d'administration (voire le chapitre précédent 5.2.3).
- **La couche systèmes administrés**. Cette couche regroupe l'ensemble des systèmes administrés du monde réel. Ces systèmes peuvent être des ressources matérielles (équipement réseau, PDA, ordinateur, capteur, etc.) ou logicielles (base de données, applications du système d'information, etc.).

La figure 6.1 met en exergue également l'ensemble des acteurs externes qui interagissent avec le canevas DASIMA. Ces acteurs sont comme suit :

- **L'administrateur de middleware**. Cet acteur gère l'ensemble du système d'administration. Il s'agit d'un super-administrateur qui gère à la fois les domaines et les applications d'administration.
- **L'administrateur de domaine**. Cet acteur a pour mission l'administration d'un domaine d'administration particulier. Ses principales tâches consistent à injecter ou extraire des applications d'administration. Elles incluent également l'approvisionnement un domaine avec l'ensemble de types de ressources qu'un domaine peut permettre d'administrer.
- **L'administrateur d'applications d'administration**. Cet acteur gère uniquement une application d'administration particulière.
- **Les ressources administrées**. C'est l'ensemble des ressources à administrer par les applications d'administration. Ces ressources interagissent avec les applications qui les gèrent via DASIMA. Elles reçoivent des requêtes d'administration et sont capables de leur envoyer des notifications concernant leur état.

6.2 Briques de base de l'architecture fonctionnelle

6.2.1 L'élément administré

L'élement administré introduit et défini dans la section 5.2.2 du chapitre 5 est une abstraction représentant une ou plusieurs ressources administrées. Il s'agit de la brique de base de notre approche concernant la gestion de l'hétérogénéité des ressources administrées. La figure 6.2 illustre note modélisation en UML du concept élément administré (à gauche) et sa déclinaison en composants fonctionnels (à droite).

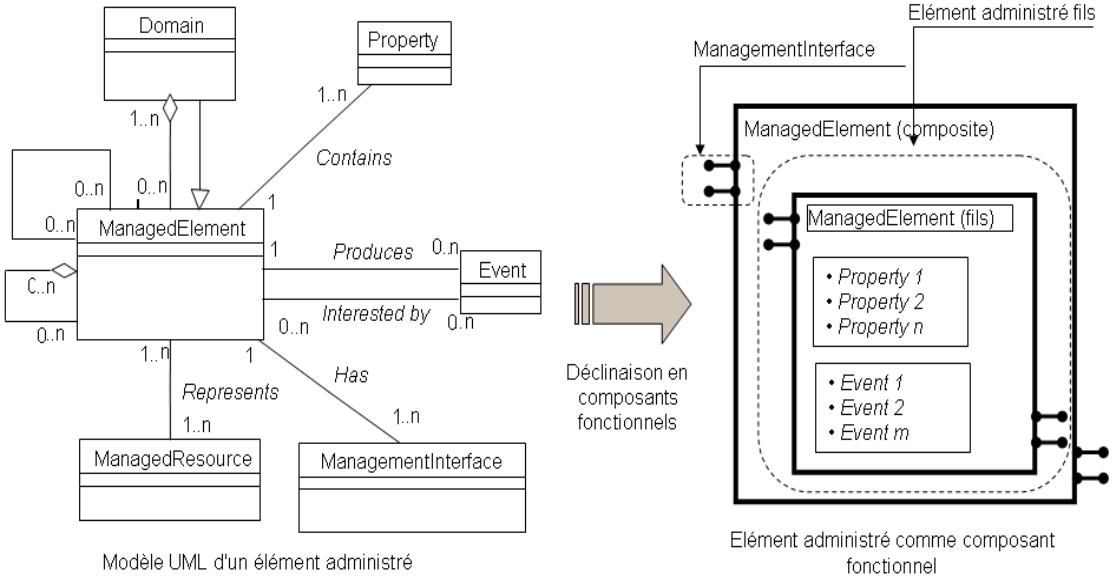


FIG. 6.2 – Modélisation d'un élément administré

Selon le modèle présenté, les éléments administrés peuvent être regroupés en domaines `Domain` qui peuvent être à leur tour des éléments administrés. Un élément administré est obligatoirement assigné à une ou plusieurs ressources réelles à gérer `ManagedRessource`. Chaque élément administré possède une ou plusieurs interfaces d'administration. Ces interfaces sont les uniques leviers de contrôle et d'administration qu'il expose aux applications d'administration. En plus de ces interfaces, un élément administré est caractérisé par un ensemble de propriétés `Property` qui représentent les propriétés qui peuvent être observées lors de l'administration. Un élément administré est également caractérisé par un ensemble d'évènements qu'il produit ou auxquels il peut s'abonner. Ces évènements représentent des notifications concernant les propriétés administrées. Un élément administré peut contenir d'autres éléments administrés. L'élément père est considéré comme élément administré composite, tandis que, l'élément contenu est considéré comme élément administré fils.

6.2.2 Le domaine d'administration

Le domaine d'administration est la pierre angulaire de notre approche de gestion multi-échelles. Ce concept a été défini et introduit dans la section 5.2.3 du chapitre 5 comme un ensemble d'éléments administrés. Toutefois, il peut être à son tour considéré comme élément administré. La figure 6.3 illustre le modèle UML d'un domaine selon notre approche (à gauche) et détaille sa déclinaison en composants fonctionnels (à droite).

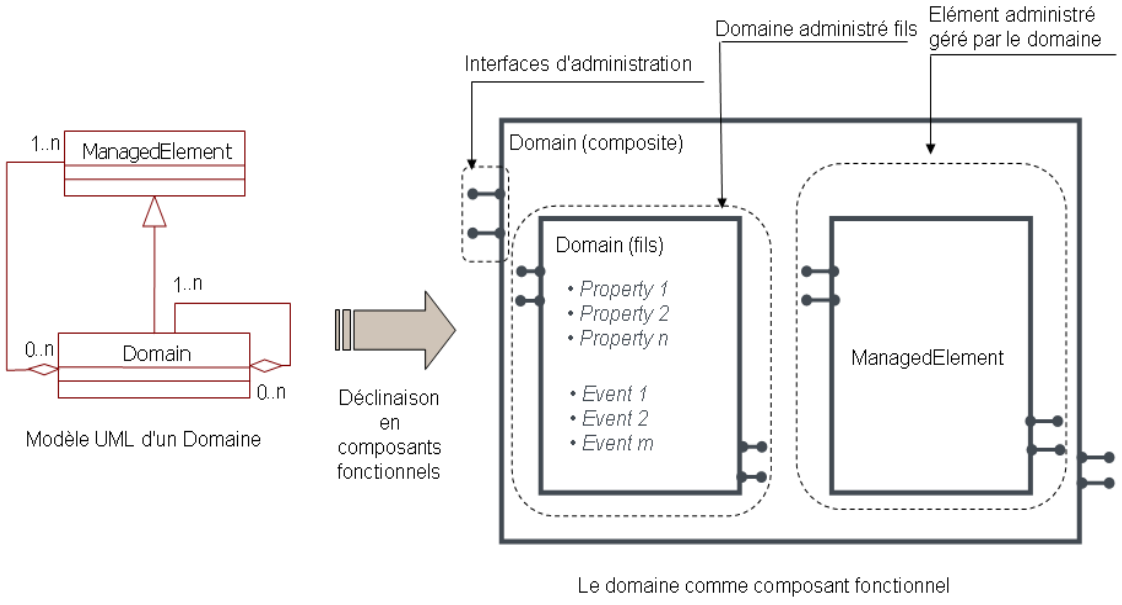


FIG. 6.3 – Modélisation d'un domaine

6.3 Architecture fonctionnelle d'un domaine

Dans cette section, nous nous focalisons sur l'architecture fonctionnelle de la solution d'administration proposée. Il importe de noter que le système d'administration est organisé en domaines d'administrations.

6.3.1 Architecture interne d'un domaine

Le domaine d'administration tel qu'il est considéré dans le cadre de ce travail se décline en trois modules fonctionnels : a) le conteneur des applications d'administration, b) le conteneur des éléments administrés et c) les services communs d'administration. La figure 6.4 illustre ce découpage qui reflète notre volonté de distinguer les modules de communication avec les acteurs externes (le conteneur des applications d'administration et le conteneur des éléments administrés), du noyau des fonctions d'administration (services communs d'administration). Les deux conteneurs gèrent, respectivement, des vues homogènes des systèmes d'administration patrimoniaux et des ressources administrées hétérogènes. Quant aux services communs, ils offrent un ensemble de services aux applications d'administration.

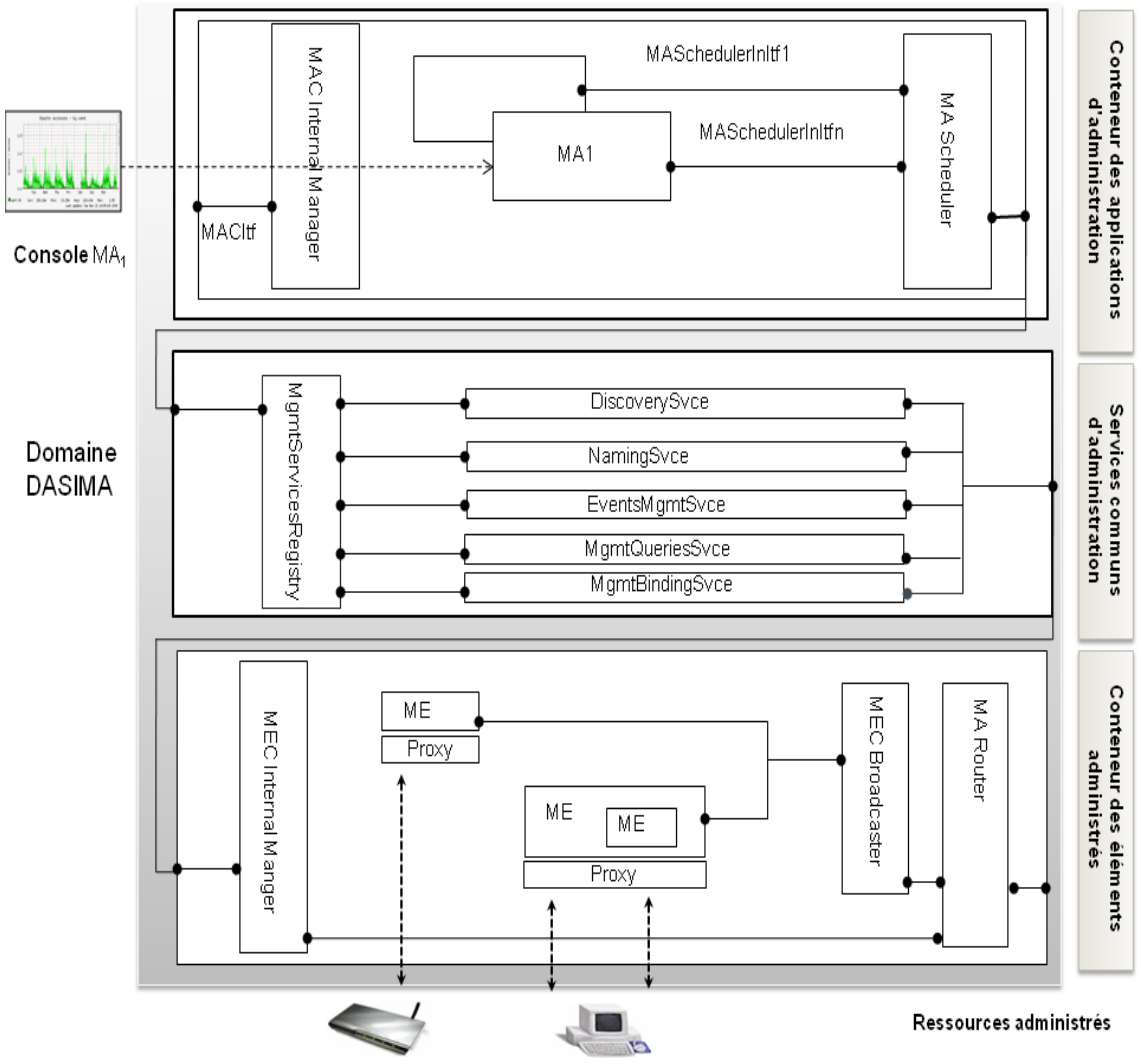


FIG. 6.4 – DASIMA : Architecture fonctionelle d'un domaine

6.3.2 Conteneur des applications d'administration

Le conteneur d'applications est le point d'entrée des applications d'administration pour communiquer avec les ressources administrées. Il permet d'établir le lien entre les applications d'administration et les services communs d'administration fournis par le domaine d'administration. La figure 6.5 illustre les deux fonctions principales du conteneur d'applications d'aministration :

La gestion du cycle de vie des applications d'administration. Cette fonction consiste à injecter ou à retirer des applications d'administration d'un domaine donné. Ces applications sont fournies par l'administrateur du domaine (étapes 1 et 2). Ensuite, ces applications sont injectées dans un domaine DASIMA par l'administrateur du domaine (étapes 3 et 4). C'est le composant `MAC Internal Manager` qui assure ce rôle d'injection dans DASIMA des nouvelles applications d'administration.

Le lien entre MA et annuaire des services communs. Ce lien est assuré par le composant `MA Scheduler` qui permet aux applications d'administration de bénéficier des services communs exposés par le domaine via son annuaire de services (étape 5). Cet annuaire est représenté par le composant MgmtServicesRegistry (voir figure 6.4.

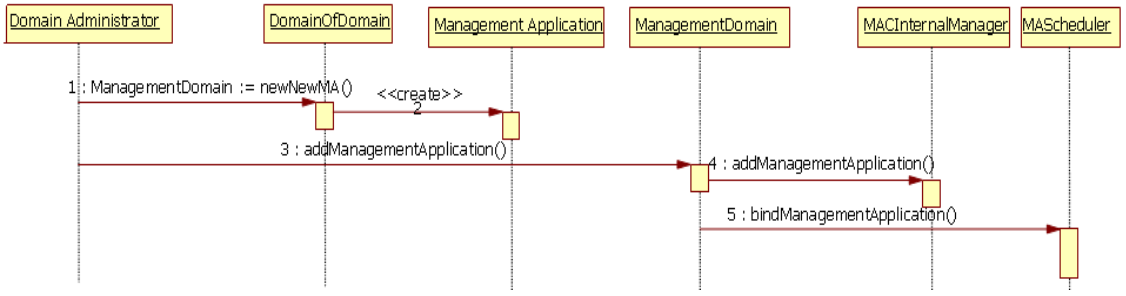


FIG. 6.5 – Processus d'activation des applications d'administration

6.3.3 Conteneur des éléments administrés

Le conteneur des éléments administrés gère les représentations architecturales des ressources administrées. Son rôle englobe principalement :

- La gestion du cycle de vie des éléments administrés : l'instanciation, la mise à jour et la destruction des éléments administrés.
- Le « binding »(ou liaison) des éléments administrés avec les services communs : Un double binding est effectué. Le premier binding permet de lier les services d'administration aux éléments administrés. Ce premier binding est indispensable pour communiquer les commandes d'administration aux éléments administrés. Un second binding est établi entre les éléments administrés et les applications d'administration. Il permet d'acheminer les informations collectées vers les applications d'administration concernées.

L'approche de gestion guidée par l'architecture se base sur une modélisation et une description de la ressource administrée. Cette description permet de guider le système d'administration en lui fournissant une vue architecturale du système administré.

Il existe globalement deux approches pour se procurer cette description architecturale :

- Description architecturale déclarative. Cette approche peut être appliquée par l'administrateur du domaine afin d'administrer des systèmes dont la structure des éléments administrés est connue d'avance. Toutefois elle montre ses limites dès qu'il s'agit d'administrer des systèmes hétérogènes dynamiques où la nature des éléments administrés n'est pas toujours connue à l'avance.
- Description architecturale par introspection. L'intospection concerne cible un ensemble de ressources (un serveur JMX ou un agent SNMP par exemple) et permet d'identifier à la volée la structure des ressources des ressources à administrer. Nous présentons dans la section suivante le service de découverte de ressources de DASIMA qui se charge de cette fonction. Cette approche est plus adaptée à des systèmes dynamiques et imprévisibles de point de vue leur structure. C'est pour cette approche que nous avons optée dans le cadre de ce travail.

6.4 Les services communs d'administration

Les services communs d'administration représentent l'ensemble des services fournis par un domaine d'administration aux applications d'administration avec lesquelles il interagit. Parmis ces services on peut citer le service de découverte des ressources, le service de nommage, le service de gestion de binding, le service de monitoring et le service de gestion des notifications. Cette liste recense les services que nous jugeons indispensables pour tout système d'administration. Elle est loin d'être exhaustive. Ces services peuvent à tout moment être étendus selon les besoins des administrateurs. La figure 6.6 décrit l'architecture globale de la gestion des services communs dans DASIMA qui s'appuie sur le principe des architectures orientées services. En effet, initialement, tous les services sont enregistrés auprès du Management ServiceRegistry. Les applications d'administration vont par la suite chercher les coordonnées du service au niveau de ce registre de services (étape 2 et 3). La dernière étape d'interaction consiste à invoquer les services via leurs interfaces (étape 4). L'ensemble des services communs est détaillé dans la section suivante.

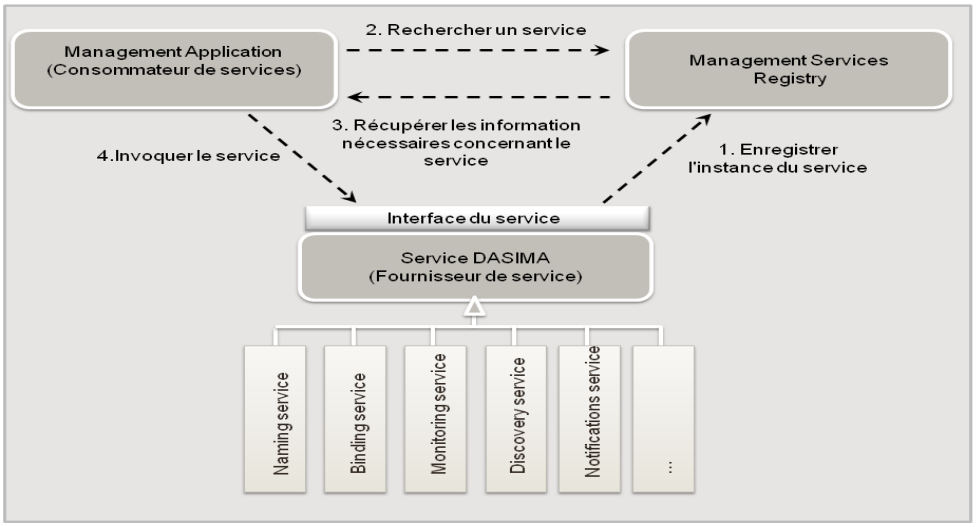


FIG. 6.6 – Services communs d'administration

6.4.1 Le service de découverte des ressources

Le rôle du service de découverte consiste à identifier les ressources susceptibles d'être administrées par les applications d'administration déployées. Ce service est activé à la demande des applications d'administration `ManagementApplication`. Cette demande déclenche le processus de découverte résumé dans le diagramme de la figure 6.7. La première phase de ce processus consiste à effectuer à identifier les ressources à administrer. Cette opération consiste à faire une recherche dans une plage d'adresse réseau (e.g. une plage d'adresses IP)ou sur une URL (étape 1). Une fois que les ressources administrées sont identifiées, le service de découverte extrait certaines méta informations (e.g. nom physique de la ressource administrée, hiérarchie des ressources qui en dépendent, etc.)(étapes 2 et 3). Ces informations seront utilisées pour créer un arbre d'objets à administrer (étape 4) qui sera transmis au service de nommage des ressources administrées (étape 5).

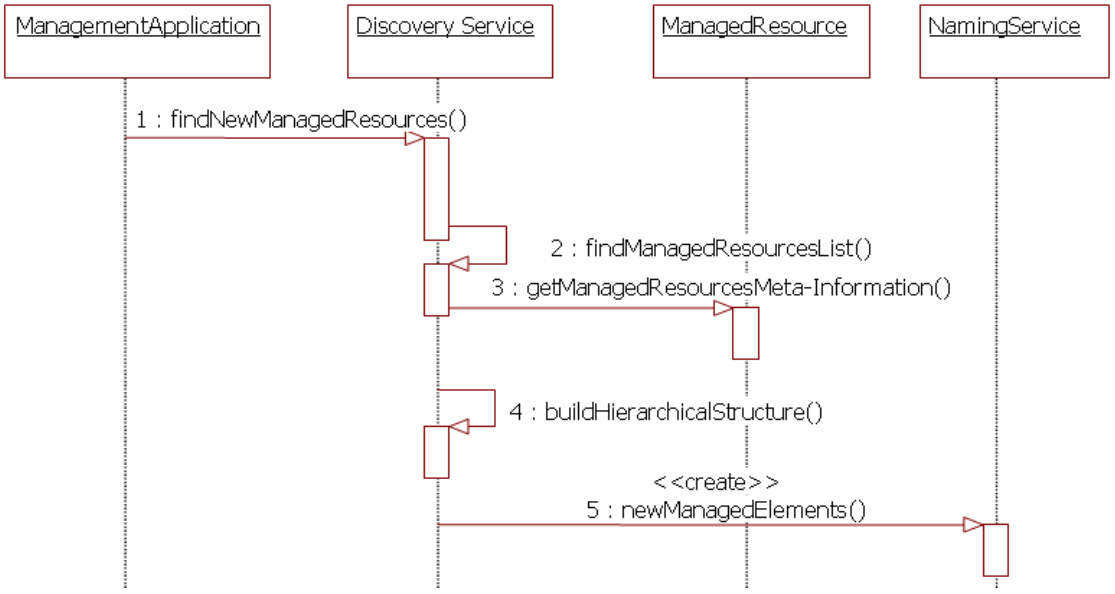


FIG. 6.7 – Services de découverte

6.4.2 Le service de nommage des ressources

Le service de nommage DASIMA est en quelque sorte un annuaire qui joue le rôle d'indexe de ressources administrées. Il permet de gérer l'identification d'une façon unique des ressources administrées par les applications d'aministration.

Ce service prend en entrée un arbre d'objets temporaires contenant des méta-informations sur les ressources administrées. Cet arbre résulte de la phase de découverte (dans le cas où la recherche des ressources a été effectuée par le service découverte de DASIMA) ou créé par les applications d'administration (ajout manuel des ressources administrées). Dés la réception de ces objets temporaires, le service de nommage transforme ces informations dans un format compréhensible par le reste des services DASIMA, ajoute les nouvelles entrées découvertes puis transmet ces informations au contenaire des éléments administrées.

6.4.3 Service de gestion de bindings

Le service de gestion des bindings a pour rôle l'établissement de liens entre les applications d'administration et les éléments administrés. Le processus de gestion de binding, qui est inspiré du modèle global des architectures SOA, est décrit dans la figure 6.8. Chaque `ManagedElement` expose un ensemble d'interfaces d'administration (étape 1). Le service de gestion de binding permet aux applications d'administration de filtrer et de sélectionner ces interfaces (étapes 3 et 4). Une fois que les interfaces sont choisies pour un élément ou un ensemble d'éléments administrés, l'application d'administration sera mise en relation avec ces éléments administrés via les interfaces sélectionnées. La liaison résultant de cette mise en relation s'appelle le `binding`. Une fois que le binding est réalisé, l'application d'administration pourra alors commencer à transmettre les commandes d'administration aux éléments administrés qu'elle gère (étape 5).

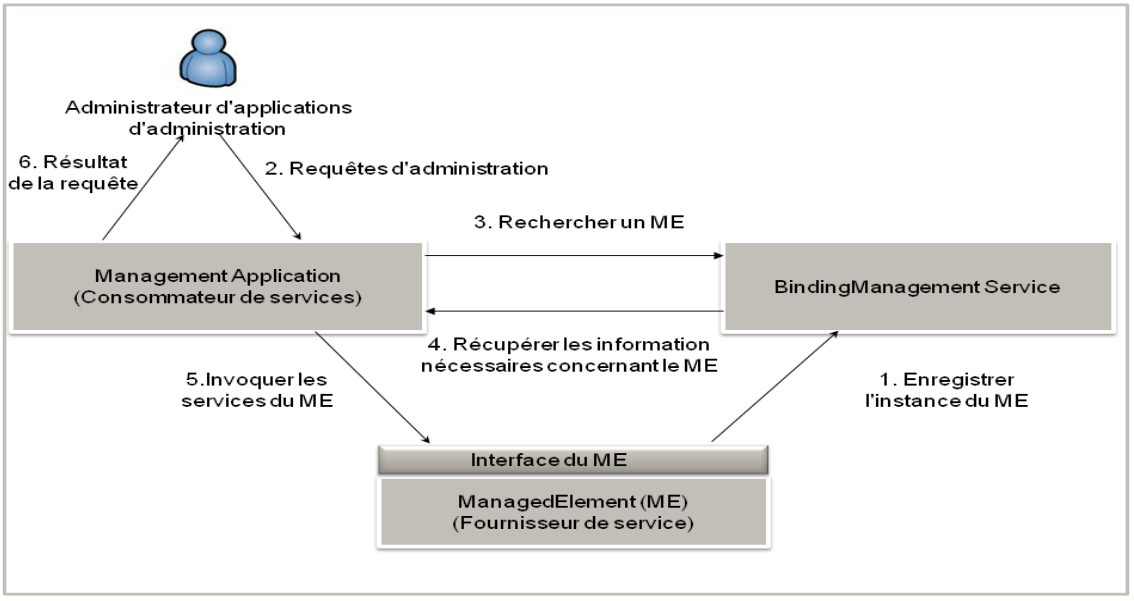


FIG. 6.8 – Service de gestion de binding

6.4.4 Service de monitoring

Le service de monitoring permet aux applications d'administration d'assurer la supervision des ressources administrées. Ce service s'appuie sur un modèle d'administration basée sur l'architecture des systèmes administrés. Il est constitué principalement des 4 composants illustrés et numérotés dans la figure 6.9 :

1. Le `PersistenceScheduler` (composant 1 dans la figure) gère un `timer` d'exécution des requêtes de gestion. Il permet aux applications d'administration de paramétrer la fréquence de rafraichissement des informations de gestion. Ce module est directement lié au module `SnapShotManager`.
2. Le `SnapShotManager` (composant 2 dans la figure) est un composant qui permet d'extraire des captures instantanées de l'état d'un ensemble de ressources administrées.

Ces captures peuvent être effectuées à la demande des applications d'administration via l'interface `SnapshotManagerItf`. Mais, ils peuvent également être activés d'une manière automatisée à une fréquence régulière par le composant `PersistenceScheduler`. Ce composant est lié directement au composant `LocalStorageManager`.

3. Le `LocalStorageManager` (composant 3 dans la figure) joue le rôle d'une base de données de gestion au sein d'un domaine d'administration. Ce composant permet, d'une part, la gestion du stockage physique des données de gestion. Il permet d'autre part d'extraire les informations de gestion en réponses aux requêtes d'administration. Le `LocalStorageManager` est lié au `SnapShotManager` qui lui communique les requêtes de captures instantanées de l'état des éléments administrés. Il est également lié au service de gestion des requêtes qui lui communique les requêtes de gestion exprimées par les applications d'administration.
4. Le `Proxy` (composant 4 dans la figure) joue le rôle d'une passerelle entre l'élément administré et la ressource administrée. Ce composant peut être configuré initialement par le système de découverte lors de l'instanciation de l'élément administré.

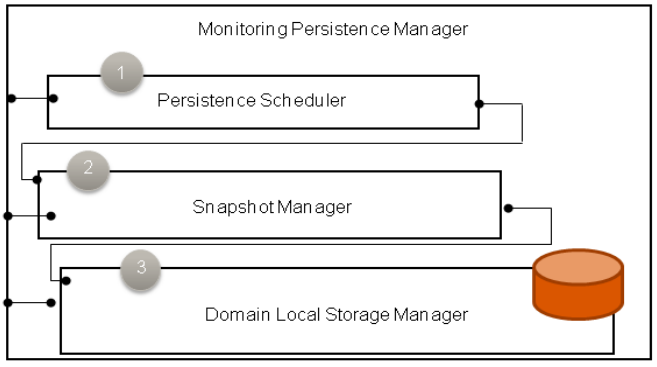


FIG. 6.9 – Architecture fonctionnelle du service de monitoring

Ces 4 composants collaborent pour réaliser deux approches de supervision que nous classons en fonction du niveau de la fraîcheur des résultats souhaités :

- **La supervision en pseudo-temps réel**. Cette approche permet un suivi en pseudo temps réel des ressources administrées. Les informations de supervision sont extraites lors de l'exécution des requêtes des administrations. La figure 6.10 (en partie basse)résume les différentes étapes de supervision en pseudo-temps réel dans le cadre requêtes avec mise à jour. Le processus déclenché par une requête du type `sendQueryWithSnapShot` (étape 10). Cette requête sera transférée au `SnapShotManager` (étape 11) qui active la mise à jour de l'élément administré (étape 12 à 14). Une fois que l'état de l'élément administré est rafraîchi, le `DomainLocalStorageManager` est notifié (étape 15). Il renvoie alors les résultats de la requête à l'application d'administration (étape 16).
- **La supervision en post-mortem**. Cette approche permet un suivi en différé des ressources administrées. L'extraction des informations de gestion est effectuée à des intervalles réguliers, préalablement configurés par les administrateurs des applications d'administration. L'avantage de cette façon de procéder est la rapidité d'exécution vu qu'on exécute une requête sur des données existantes. Toutefois, en procédant de la sorte, on risque d'obtenir des résultats sur des données qui ne reflètent pas l'état

réel des ressources administrées au moment de l'exécution de la requête. La figure 6.10 résume le processus de supervision en différé (partie du milieu).

La partie haute de la figure figure 6.10 (intitulée initialisation) correspond à la phase d'initialisation du service de monitoring. Elle est commune aux deux approches. Elle consiste à initialiser le composant `PersistanceScheduler` qui a pour mission la gestion de la fréquence de sauvegarde des informations de gestion d'un ME donné. Ce composant déclenche l'activité du composant `SnapShotManager` qui active la mise à jour de l'élément administré.

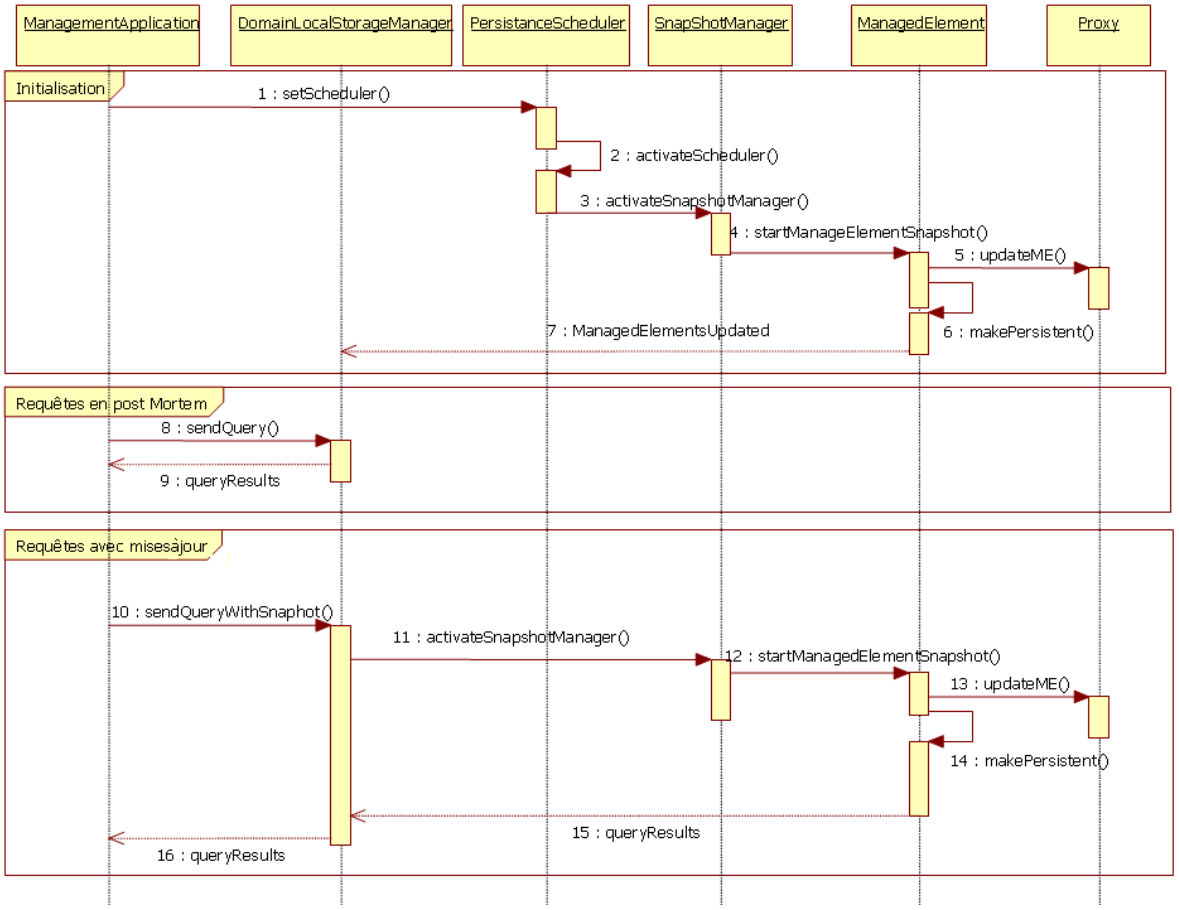


FIG. 6.10 – Les deux processus de supervision de DASIMA

6.4.5 Service de gestion des notifications

Ce service permet de gérer les notifications envoyées par les éléments administrés aux applications d'administration. Chaque domaine gère un ensemble d'évènements produits par les éléments administrés. Ces évènements sont gérés localement au niveau des domaines. Ils peuvent être filtrés, sauvegardés et diffusés vers les applications d'administration intéressées.

Le format des messages échangés entre l'élément administré et les applications d'administration respecte une structure générique que nous résumons dans la figure 6.11. Ce

format comporte deux parties : un entête et un corps du message. L'entête du message est composé de deux champs. Le premier champ indique le nom du message alors que le second indique la date de création du message. La seconde partie du message est le corps de la notification. Cette partie est organisée en un ensemble de champs avec les valeurs correspondantes.

FIG. 6.11 – Format du message

Le processus de gestion des évènements se décline en deux phases (voir figure 6.12) :
- La phase de sélection. Durant cette phase, l'administrateur des applications d'administration commence par récupérer les interfaces de gestion des évènements des éléments administrés (étapes 1 à 3). Ensuite, il sélectionne l'ensemble des évènements jugés pertinents.
- La phase d'abonnement et publication. Cette phase se décline à son tour en trois étapes. Dans la première étape, l'administrateur des applications d'administration s'abonne aux évènements qui l'intéressent (étapes 7 à 8). La seconde étape concerne l'activation du suivi de l'évènement au niveau du Proxy (étape 9). Une fois que l'écouteur de l'évènement est activé, commencera la troisième phase qui concerne la publication des notifications aux entités abonnés (étapes 10 à 13).

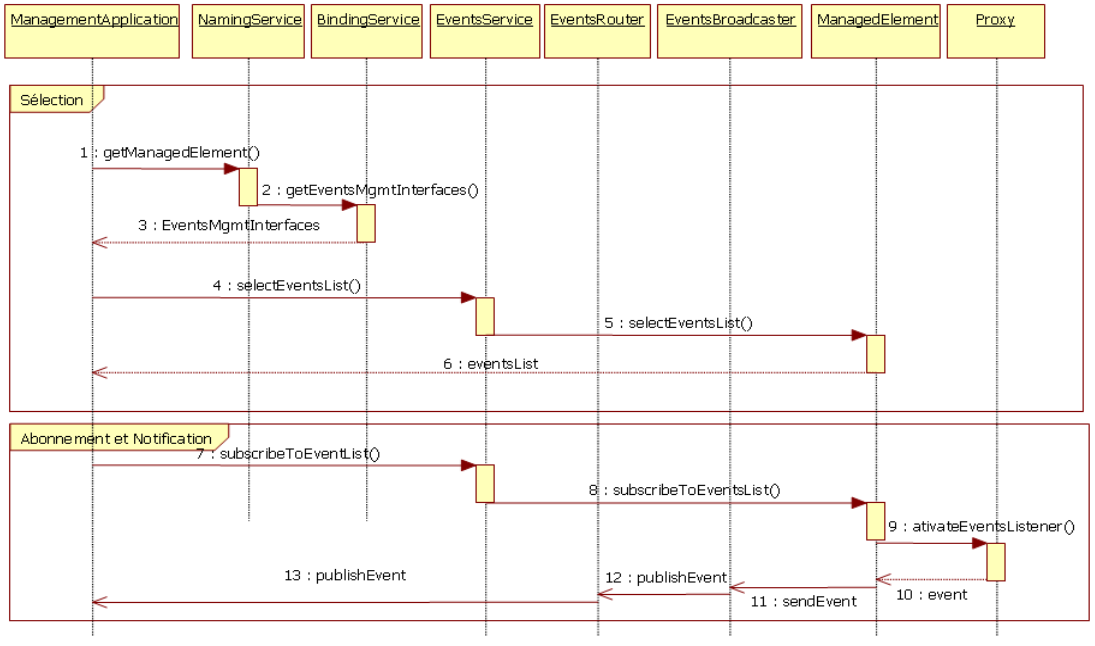


FIG. 6.12 – Service de gestion des notifications

6.5 Le domaine des domaines

Le domaine des domaines est un domaine qui a la particularité de gérer des domaines, c'est a dire que les éléments administrés sont eux mêmes des domaines. En plus des services classiques dont dispose un domaine, le domaine des domaines dispose de deux services supplémentaires : la fabrique des domaines et le service de gestion de paramétrage global.

6.5.1 Fabrique des domaines

La fabrique des domaines est un service spécifique qui permet d'instancier de nouveaux domaines. Ces nouveaux domaines sont considérés par la suite comme des éléments administrés au niveau du domaine des domaines. Le diagramme de séquence illustré dans la figure 6.14 décrit le processus de création de domaines qui est initié par l'administrateur du middleware à travers l'application `ManagementMiddlewareApplication` (étape1). L'étape 2 du diagramme de séquence correspond à la création d'une nouvelle instance de domaine. Ce dernier sera par la suite rajouté au conteneur des éléments administrés du domaine des domaines, en tant que nouvel élément administré (étapes 4, 5 et 6).

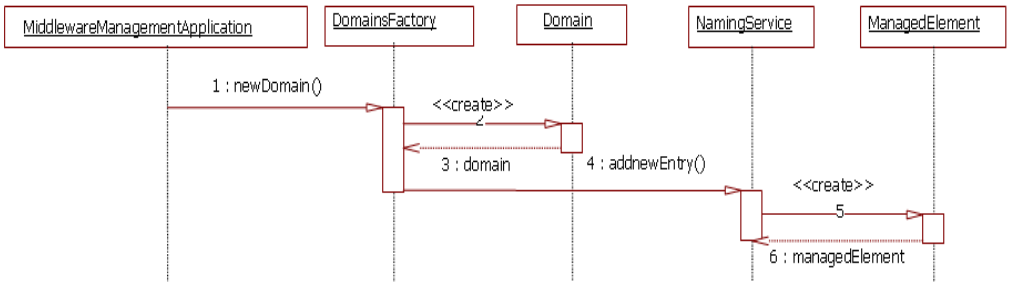


FIG. 6.13 – Fabrique des domaines

6.5.2 Service de paramétrage global

!!! TODO : Ajouter définition maître-escalve + commentaires illustration

Avec la composition des domaines, un élément administré peut appartenir à au moins un domaine. Les éléments administrés peuvent être aussi partagés entre les domaines. Ce partage est traduit selon notre approche par une duplication de l'élément administré entre les domaines concernés. La figure **??** montre un exemple d'interaction possible entre des éléments administrés dupliqués. Deux situations sont possibles :

- **La mise à jour des éléments gérés dupliqués.** L'élément géré initial est considéré comme élément administré maître. Il interagit directement avec les ressources gérées. En cas de mise à jour, un élément administré maître diffuse les informations de mise à jour aux autres duplicats (ou élément administré esclave).
- **Conrôle de la ressource administrée.** Un élément administré maître joue le rôle de vérrou pour accéder à la ressource administrée. Les éléments administrés esclaves ne donnent pas le droit d'accéder directement aux ressources.

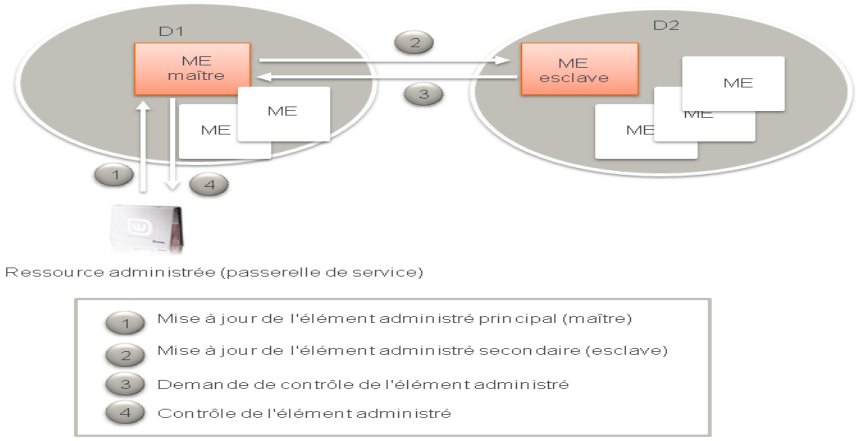


FIG. 6.14 – Gestion des éléments administrés dupliqués

6.6 Synthèse

Dans ce chapitre, nous avons présenté l'architecture fonctionnelle du canevas DASIMA. Nous avons introduit dans un premier temps les briques de base du canevas, à savoir l'élément administré et le domaine d'administration. Nous avons par la suite détaillé l'architecture fonctionnelle d'un domaine. Cette architecture est organisée en un ensemble de services mutualisés pour les applications d'administration hébergées par DASIMA. Nous avons terminé le chapitre avec la description du domaine des domaines, un domaine particulier qui joue le rôle de superviseur pour les autres domaines DASIMA.

Après avoir présenté l'architecture fonctionnelle de DASIMA, nous détaillons dans le chapitre suivant certains détails d'implémentation et nous présentons des exemples de mise en œuvre dans un contexte M2M.

Chapitre 7

Implémentation et mise œuvre

Contents

7.1 Introduction 137
7.2 Implémentation du canevas Dasima 138
7.2.1 Choix du modèle à composants 138
7.2.2 Implémentation du canevas Dasima 140
7.3 Etude de cas : La plateforme Orange M2M 140
7.3.1 La plateforme Orange M2M 141
7.3.2 Modélisation des ressources à administrer 142
7.3.3 Instrumentation applicative avec JMX 143
7.3.4 Instrumentation système avec CompositeProbe 144
7.3.5 Instanciation de la plateforme M2M 145
7.4 Evaluation quantitative 147
7.4.1 Evalution à échelle réduite 147
7.4.2 Evaluation à large échelle 150
7.5 Evaluation qualitative 156
7.5.1 Compatibilité avec les standards 157
7.5.2 Pertinence de l'approche 157
7.6 Synthèse 158

7.1 Introduction

Ce chapitre présente la réalisation du canevas décrit dans le chapitre précédent. Il détaille les différents choix conceptuels et techniques adoptés pour l'implémentation de DASIMA. Il introduit par la suite le cas d'étude M2M considéré dans le cadre nos tests et nos expérimentations. Le chapitre se termine par une présentation et une discussion des différents résultats obtenus.

7.2 Implémentation du canevas Dasima

7.2.1 Choix du modèle à composants

Afin de mettre en œuvre notre canevas d'administration dans un contexte multi-échelles, nous avons opté pour l'approche à composants. Plusieurs avantages découlent de l'adoption des architectures à base de composants pour la conception et réalisation des solutions d'administration des réseaux et des services [VWS99]. Les composants se présentent sous forme de composants techniques qui réalisent les composants fonctionnels identifiés dans le chapitre 6. Afin de concevoir et réaliser notre plateforme d'administration, nous avons choisi le modèle à composants Fractal [BCL+06].

7.2.1.1 Le modèle à composants Fractal

Le moèle à composants Fractal offre des facilités pour concevoir et bâtir des infrastructures logicielles flexibles et configurables à base de composants. Par analogie au concept de cellule en biologie, un composant Fractal est doté d'une membrane et d'un contenu. Cette abstraction permet de distinguer le contenu du composant (aspect fonctionnel) de l'ensemble des mécanismes de contrôle dont disposent les composants gérés au niveau de la membrane. Contrairement aux modèles à composants existants tels que les JavaBeans, les composants OSGi ou les composants COM, un composant Fractal peut contenir à son tour d'autres composants. Cette possibilité d'imbrication hiérarchique permet d'obtenir des composants composites. La 7.1 illustre les différents éléments qui forment un composant Fractal et un exemple d'imbrication des composants Fractal. Un composant Fractal communique avec ses semblables via des interfaces. Les interfaces dans Fractal ont un rôle central. Il existe principalement deux catégories distinctes d'interfaces :

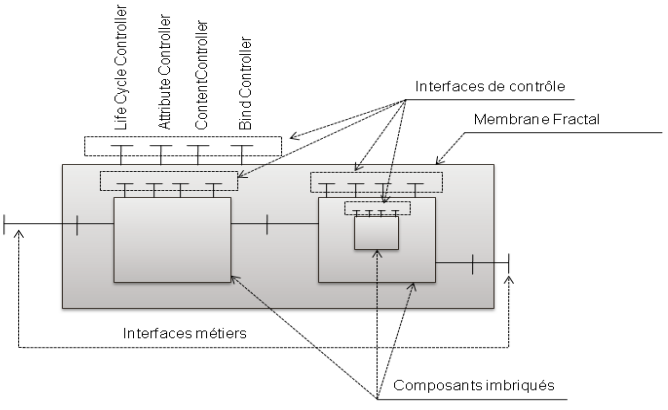


FIG. 7.1 – Le modèle à composants Fractal : exemple de composants imbriqués

– *les interfaces métier* : ces interfaces constituent les points d'accès externes au composant. Une distinction explicite est faite entre les interfaces dites serveurs des interfaces clients. Les interfaces serveurs sont des points d'accès aux services offerts par un composant. Quant aux interfaces clients, elles permettent aux composants de demander des services à d'autres composants. C'est en branchant ces composants les uns aux

autres qu'on parle d'assemblage de composants ou de *binding* .
- *les interfaces de contrôle* (ou contrôleurs) : ces interfaces prennent en charge des propriétés non fonctionnelles du composant. Fractal met à disposition un ensemble d'interface prédéfinies au niveau de la membrane afin d'assurer la gestion du cycle de vie (*Life Cycle Controller*), des liaisons (*Bind Controller*), des attributs (*Attribute Controller*), du contenu (*Content Controller*) pour les composites. Le modèle étant extensible, il est possible de définir ses propres interfaces de contrôle.

7.2.1.2 Disucssion du choix technique

Avant de choisir le modèle à composants Fractal, nous avons examiné les autres travaux et modèles existants en matière de modèles à composants [OSG, DCO, DK06, DK01]. A l'issue de cette étude, nous avons opté pour le modèle à composants Fractal pour les raisons suivantes :

Modèle à composants flexible. L'un des points forts du modèle Fractal réside dans sa souplesse et son extensibilité. Cette flexibilité nous permet d'étendre facilement et à la volée les infrastructures logicielles implémentées avec cette technologie. Des extensions peuvent être étendues, si besoin, au modèle à composants lui-même (rajout de nouveaux contrôleurs par exemple)

Ajustement à la volée. C'est grâce à cette caractéristique que nous nous sommes appuiés pour construire une plateforme d'administration dynamique. Une plateforme fractalisée (conçue et développée en Fractal) peut être étendue à la volée en cours d'exécution. Cette capacité de reconfiguration dynamique peut être très utile dans le cas de la maintenance continue de notre plateforme d'administration que nous souhaitons dynamique.

Une capacité d'introspection. Plusieurs outils sont proposés pour faciliter l'introspection des applications fractalisées. L'introspection est nécessaire pour la supervision des applications en cours d'exécution. Dans les premières applications fractalisées l'introspection a été effectuée via les contrôleurs Fractal. Par la suite ce mécanisme a été enrichi grâce à FractalJMX (une extension du standard JMX pour la supervision des applications Fractalisées) et Fpath et Fscript [PCDC08] (pour la navigation dans les applications Fractal).

Support des composants composites. La composition dans Fractal permet d'avoir une vue uniforme des applications à différents niveaux d'abstraction. C'est grâce à ce mécanisme que nous nous appuyons pour bâtir nos domaines d'administration qui peuvent contenir à leur tour d'autres domaines. C'est ce mécanisme qui nous permet également de gérer des ressources administrées composites (un PC est constituée d'un ensemble de ressources physiques (ses composants matériels) et de logiciels).

Support des composants partagés. Ce mécanisme permet de modéliser les ressources et leur partage, tout en préservant l'encapsulation des composants. Cela peut être utile dans le cas de partage de ressources d'administration ou administrées.

Disponibilité de l'outillage de développement. Il existe différentes implémentations du modèle Fractal en Java (*Julia*), en .NEt (FracNet) en C (*Cecilia, Think*). En plus du support à l'exécution des composants Fractal, d'autres outils tel que FractalADL ont été conçus pour assister les développeurs dans la gestion des descriptions architecturales des composants Fractal connues sous le nom d'ADL (*Architecture Definition Language*). S'ajoutent à ces outils les différents outils mentionnés plus haut

(FractalJMX, FScript et FPath, etc.) pour l'administration et la navigation inter et intra-composants.

7.2.2 Implémentation du canevas Dasima

Le canevas DASIMA est un caneveas d'aministration intégrée dont l'implémentation repose sur le modèle à composant Fractal. Sur le plan technique, DASIMA est basée sur l'implémentation Java de Fractal Julia. Le canevas est constitué d'un ensemble de classes Java dont l'assemblage est effectué grâce à des descripteurs de déploiement couramment appelés ADL et présentés sous forme de fichiers XML. Le code est constitué de 7101 lignes de code Java réparties sur 93 classes et 63 interfaces. Le code de DASIMA contient également 45 fichier fractal ayant un total de 638 lignes. Le tableau 7.1 résume ces information.

Le canevas DASIMA s'appuie sur les bibliothèques suivantes pour l'implémentation de ses services :

- **Julia**[1] : une implémentation Java du modèle à composants Fractal.
- **JORAM**[2] : Une implémentation de JMS (Java Message Service)[3].
- **Fractal-JMX**[4] : Une extension de JMX pour la gestion simplifiée des applications Fractal.
- **Composite-probe**[5] : Une extensions des sondes CLIF pour l'agrégation des sondes d'une manière hiérarchique.
- **Speedo**[6] : Un implémentation de JDO pour la gestion de la presistance des données. Ce logiciel est utilisée comme base de notre service de gestion de persistance.

Tab. 7.1 – Table synthèse des informations concernant le code de DASIMA

Propriétés	Nombre
Classes Java	93
Interfaces Java	63
Fichiers Fractal	45
Lignes de code Java	7101
Lignes de code Fractal	638
Total	7739

7.3 Etude de cas : La plateforme Orange M2M

L'objectif de cette section est de présenter le scénario d'administration adopté afin de valider notre approche. Nous introduisons dans un premier temps la plateforme à administrer. Ensuite, nous présentons le scénario d'administration exécuté.

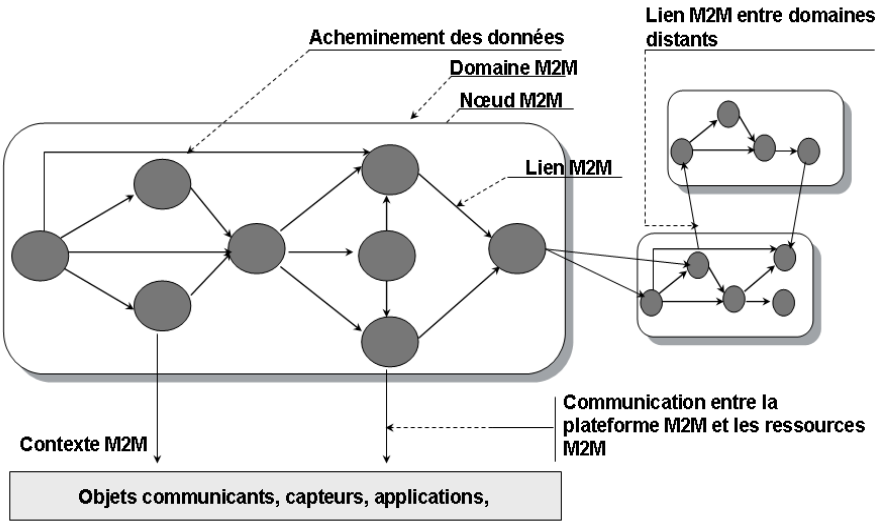


FIG. 7.2 – Vue Globale du workflow applicatif M2M

7.3.1 La plateforme Orange M2M

Les systèmes d'objets communicants, connus sous le nom de systèmes M2M (*Machine to Machine*), sont des systèmes qui ont pour mission l'automatisation des échanges entre machines et objets communicants de toute sorte (ampoules, caméras de surveillance, capteurs RFID, capteurs de température, vêtements intelligents, etc.). Ces sysèmes incluent entre autre diverses solutions applicatives : gestion de flotte de véhicules, connectivité des terminaux sur le réseau, analyse des déplacements, la vidéosurveillance, etc [M2M06].

Ces systèmes représentent un cas d'étude typique des contextes muti-échelles pour plusieurs raisons :

- hétérogénéité des ressources. Les plateformes M2M font intervenir aussi bien des objets communicants divers que des programmes et des applications logicielles.
- forte distribution géographique.
- dynamisme des ressources M2M.
- une ampleur numérique variable (en terme de nombre de ressources potentiellement administrables).

Afin de valider notre approche architecturale d'administration, nous avons évalué la plateforme DASIMA en réalisant des opérations d'administration sur une application réelle utilisée actuellement pour la gestion des communications des services M2M de l'opérateur Orange. Cette application[7] a été conçue sous la forme d'un réseau logique de nœuds M2M. Au sein de ce réseau, l'ensemble des nœuds forme une chaîne de traitement des informations véhiculées entre les machines. Chaque maillon de cette chaîne occupe une fonction précise dans le traitement des informations véhiculées. Certains nœuds occupent

[1] http ://fractal.ow2.org/tutorials/julia/index.html
[2] http ://joram.ow2.org/
[3] http ://java.sun.com/products/jms/
[4] http ://fractal.ow2.org/tutorials/jmx/index
[5] http ://clif.ow2.org/
[6] http ://speedo.ow2.org
[7] Plateforme ODIS : http ://odis.ow2.org

par exemple la fonction de collecte d'information, alors que d'autres nœuds assurent par exemple l'épuration ou la sauvegarde des données collectées.

7.3.2 Modélisation des ressources à administrer

Avant de commencer l'administration de la plateforme M2M, nous avons procédé à sa modélisation afin de cibler les aspects pertinents à administrer. Les deux figures 7.3 et 7.4 décrivent deux modélisations de la plateforme M2M :

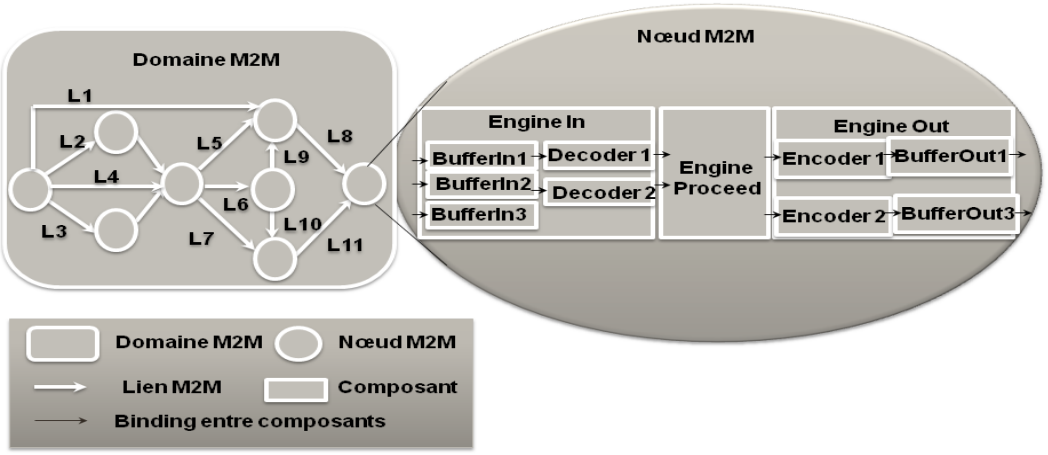


FIG. 7.3 – Une modélisation abstraite des concepts M2M

– **Une modélisation abstraite des concepts M2M.** La figure 7.3 décrit cette modélisation qui met en exergue le concept de *domaine M2M* et de nœuds M2M. Un domaine M2M est un conteneur des *nœuds M2M*. Ces nœuds sont liés par des liens (L1 à L11). Chaque nœud se décline en trois modules : a) un *Engine In* pour l'acquisition des données, b) un *Engine Proceed* pour le traitement des données et c) un *Engine Out* pour la diffusion des données. Chacun des modules d'acquisition de diffusion se compose d'un ou plusieurs sous-modules d'encodage/décodage (ou sérialisation/désérialisation) et d'un espace tampon pour la sauvegarde temporaire des données avant et après le traitement.

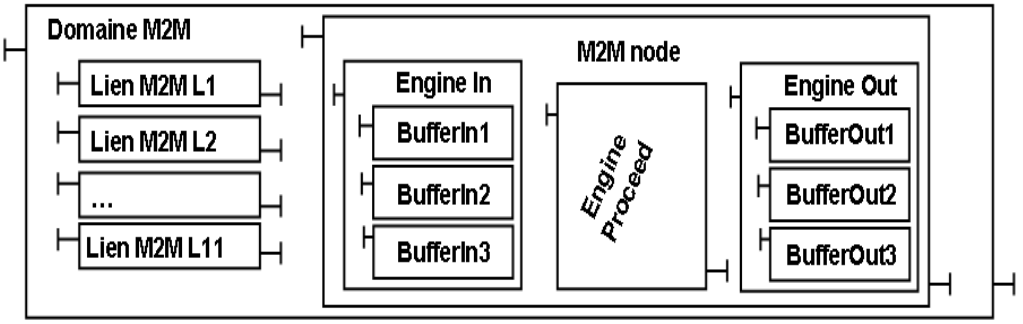


FIG. 7.4 – Une modélisation des concepts M2M en terme de ME DASIMA

– **Une modélisation des concepts M2M en terme de ME DASIMA.** Cette modélisation résumée dans la figure 7.4 se présente sous la forme d'un ensemble de ME DASIMA hiérarchiquement disposés. A la racine de ces ME nous trouvons le ME domaine M2M. Il importe de distinguer cet ME domaine, qui contient un ensemble de ME DASIMA, du domaine DASIMA, qui peut inclure un ensemble de ME domaine.

7.3.3 Instrumentation applicative avec JMX

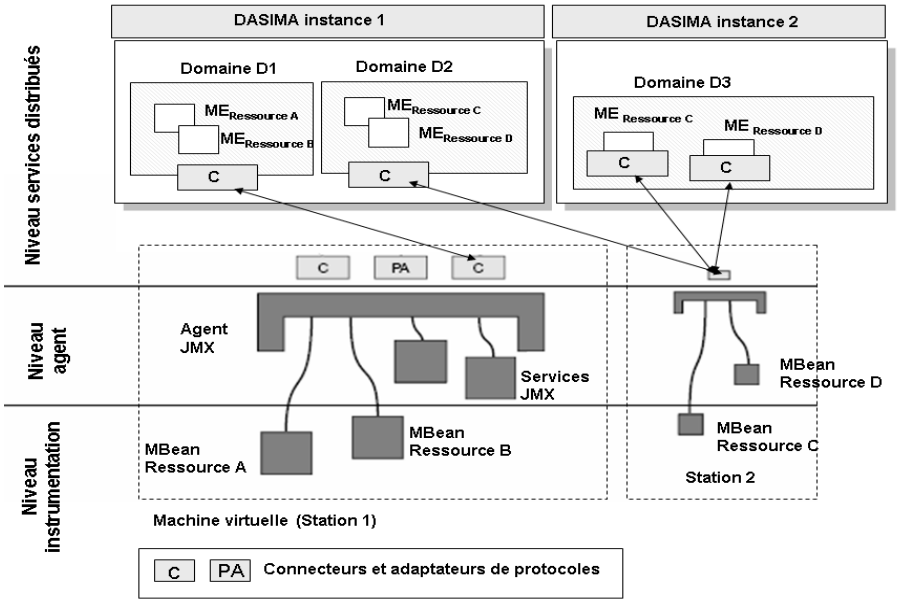


FIG. 7.5 – Intégration de l'instrumentation JMX avec DASIMA

Afin de connecter la plateforme DASIMA à des ressources administrées de type applicatif, nous nous sommes appuyé sur le standard JMX (voir section 2.3.2 pour plus d'informations) pour l'instrumenter les applications considérées.

La figure 7.5 décrit l'architecture globale d'intégration de JMX avec DASIMA. Elle illustre deux exemples de domaines DASIMA à travers les instance 1 et 2 de DASIMA. La première instance représente une instance DASIMA qui gère deux domaines DASIMA D1 et D2. La seconde instance gère quant à elle une seule instance D3 de domaine DASIMA. Les instances DASIMA (couche services distribués) interagissent directement avec les services de gestion exposés par les agents JMX (niveau agent). Deux exemples d'intégration DASIMA-Agent JMX sont illustrés.

Intégration JMX au niveau du domaine. Afin de doter DASIMA d'une capacité d'administration réflexive, nous avons utilisé Fractal-JMX. L'injection d'un Agent Fractl-JMX dans le domaine principal DASIMA permet l'introspection et l'administration de tous les sous-domaines, éléments administrés et services DASIMA. Selon cette approche, le domaine DASIMA est l'interlocuteur unique de l'agent JMX. Le domaine interagit avec l'agent JMX et ensuite une mise à jour des ME est effectuée. L'avantage de cette approche est l'atomicité de l'opération de mise à jour du domaine DASIMA et la forte cohérence des données du domaine. L'inconvénient est la faible marge de manœuvre pour mettre à jour un sous ensemble spécifique sans mettre à

jour l'ensemble des éléments du domaine.

Intégration JMX au niveau des MEs. Chaque ME interagit directement avec l'agent JMX pour mettre à jour son état. L'avantage de cette approche c'est la fine granularité de configuration. On peut en effet, arrêter, accélérer ou mettre en attente l'interaction entre un ME et l'agent JMX.

Il importe de signaler qu'avec JMX nous pouvons mettre en œuvre les trois modèles d'interaction étudiés et proposés dans la section 5.4.2 (pull, push, pooling).

Afin de mettre en place un mécanisme d'instrumentation riche et flexible, nous avons étendu l'instrumentation JMX, déjà existante au niveau du système M2M, avec une instrumentation en mode notification. La figure 7.6 résume les deux cas de figure. La première figure à gauche décrit une instrumentation JMX à base de notifications. Selon cette instrumentation, l'ensemble des nœuds M2M peuvent envoyer des notifications concernant leur état à un nœud d'administration M2M qui transfère à son tour ces notifications aux MEs concernés. La seconde possibilité est illustrée à droite dans la figure b. Elle décrit un mode d'interaction à base de requêtes entre les MEs et les nœuds M2M. Dans ce scénario, chaque ME est considéré comme un MBean. De ce fait, il est mis en relation directe avec le ME sans intermédiaire contrairement à la première technique d'instrumentation. Ces deux approches ont été implémentées et testées avec DASIMA.

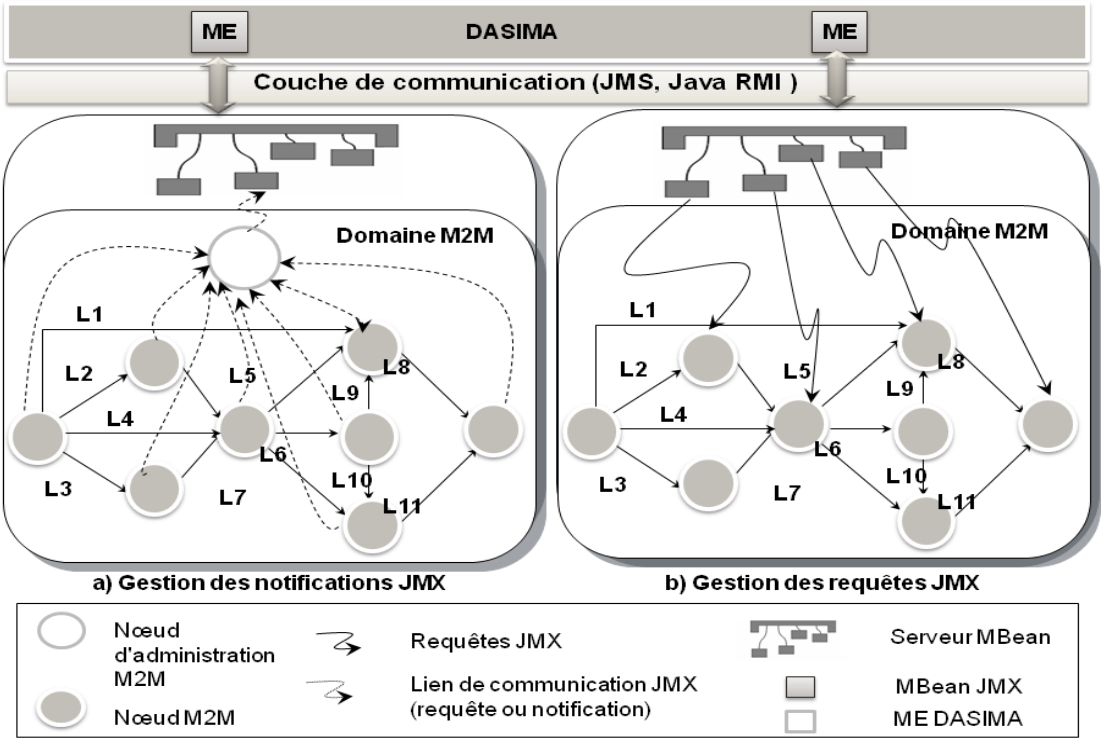


FIG. 7.6 – Instrumentation JMX de la plateforme M2M

7.3.4 Instrumentation système avec CompositeProbe

DASIMA étant une plateforme générique pour l'administration intégrée, nous avons complété l'instrumentation applicative de la section précédente avec une instrumentation

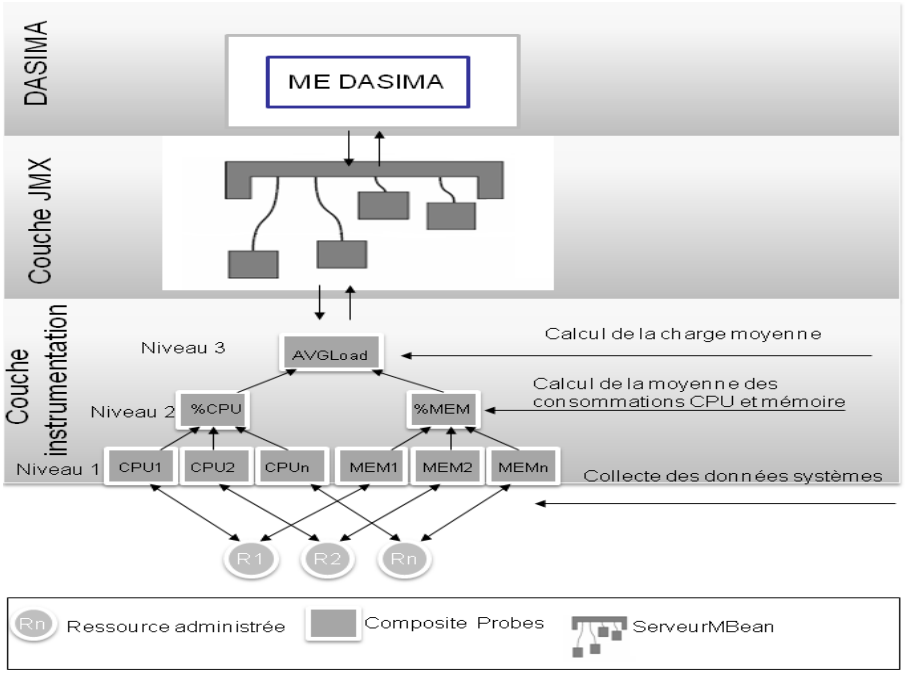


FIG. 7.7 – Administration avec DASIMA et CompositeProbe

de niveau système en intégrant DASIMA avec le canevas *CompositeProbes* [Dia07]. Ce canevas s'appuie sur les sondes Lewys [EC] pour la collecte des informations systèmes (CPU, mémoire, etc.). Du point de vue technique, il est implémenté sous forme de composants Fractal qui exposent leurs services via un serveur MBean. DASIMA considère les sondes *CompositeProbes* comme des ressources administrées accessibles via JMX. La collecte des informations est faite via les interfaces métier Fractal exposées par les MBeans Composites Probes. La figure 7.7 montre un exemple d'intégration de Composite Probes pour la collecte d'informations systèmes avec n ressources d'un cluster de machines.

La couche instrumentation système aurait pu être simplifiée avec des sondes Lewys uniquement ou avec des sondes ad hoc. Toutefois, dans un souci de flexibilité nous avons mis en place ces sondes intelligentes qui, en plus de leur reconfigurabilité, offrent des capacités d'agrégation hiérarchique, dégageant ainsi les ME DASIMA de ces traitements. La figure 7.7 illustre un exemple d'agrégation de CPU et mémoire. Au niveau des deux sondes de second niveau une moyenne des consommations est calculée. Au troisième niveau de hiérarchie des sondes, une moyenne globale de la charge du cluster est calculée.

7.3.5 Instanciation de la plateforme M2M

La mise en œuvre de la plateforme M2M requiert l'écriture d'un fichier XML de configuration. Afin de mener des expérimentations à différentes échelles, nous avons implémenté un outil de synthèse de fichiers de configuration MEM qui permet d'automatiser la génération de ces fichiers. Cet outil nous a été particulièrement utile pour la génération de graphes de nœuds M2M de très grande taille (1000, 100.000 ou 1.000.000 de nœuds). La figure 7.8 décrit le processus que nous avons automatisé. Initialement, l'outil de synthèse nécessite un ensemble de données de configuration (nombre de nœuds, nombre de connexion, nombre

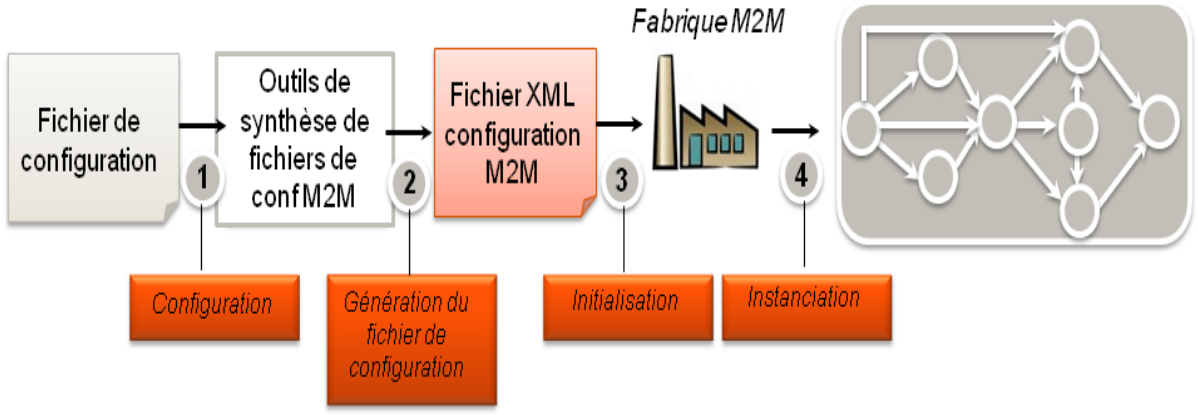


FIG. 7.8 – Processus d'instanciation d'un réseau de nœds M2M

des domaines, type des connexions, etc.) (étape 1). L'outil de synthèse traite ces données et produit un fichier de configuration pour la plateforme M2M (étape 2). Ce fichier est par la suite transmis à la fabrique M2M (étape 3) qui produit en sortie un overlay network de nœuds M2M (étape 4). La figure 7.9 montre à titre d'exemple une visualisation d'un réseau de 100 nœuds M2M dont le fichier de configuration a été créé par notre outil de synthèse.

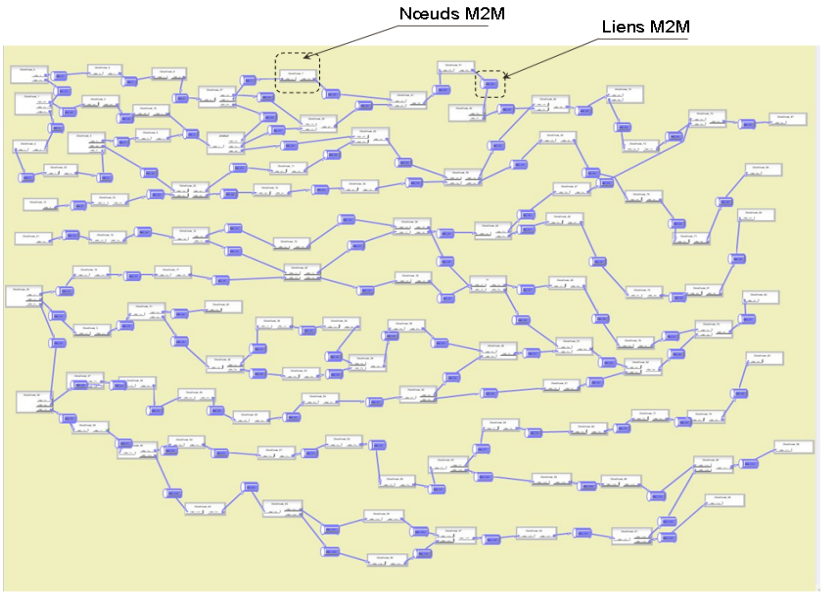


FIG. 7.9 – Visualisation d'un réseau de nœds M2M

7.4 Evaluation quantitative

7.4.1 Evalution à échelle réduite

L'objectif de cette section est d'évaluer la plateforme DASIMA dans un contexte à échelle réduite. Ces résultats ont fait l'objet d'une publication [MkL09].

7.4.1.1 Environnement d'évaluation

Nous avons évalué DASIMA sur un réseau local de 4 PC. L'ordinateur principal qui a hébergé DASIMA dispose de la configuration suivante : 1 processeur de 1.6 GO et 2GO de RAM. Les nœuds M2M ont été déployés sur des machines distantes dotées d'un processeur de 1.6 GO et de 1GO de RAM. La figure 7.10 décrit l'architecture globale de l'environnement de test et d'évaluation que nous avons mise en place. Le nombre de de nœuds M2M est indiqué en bas de chaque machine.

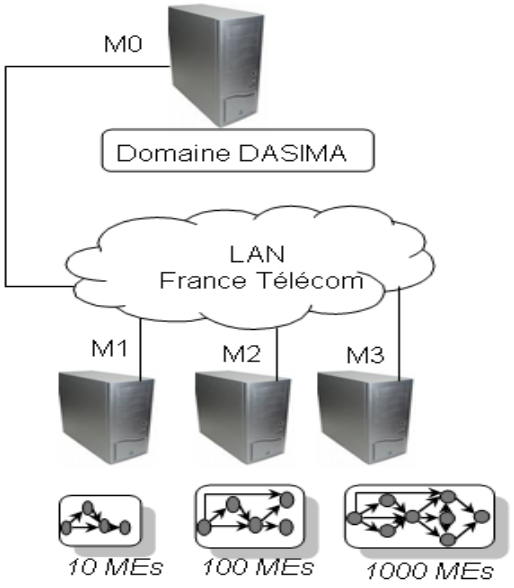


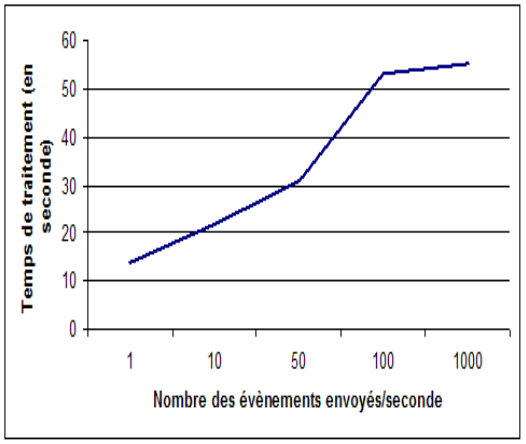
Fig. 7.10 – Platefome d'expérimentation à échelle réduite

7.4.1.2 Evaluation du service gestion des évènements

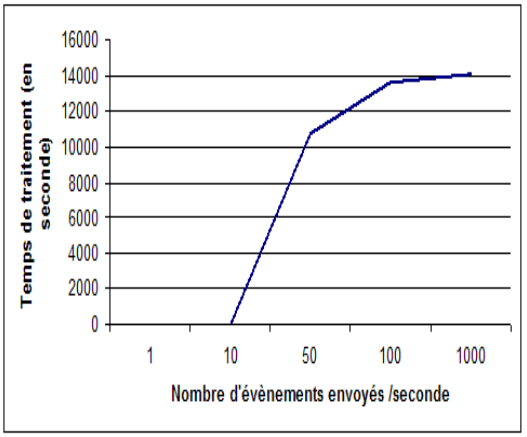
Le scénario que nous avons mis en place consiste à déployer des nœuds M2M sur les trois machines M1, M2, M3. Ces nœuds ont été configurés pour envoyer des notifications concernant leurs états selon une fréquence régulière (toutes les 1, 10, 50, 100, 1000 secondes). Nous avons mesuré par la suite le temps de binding entre le conteneur des ME et l'application d'administration. Nous avons mesuré également le temps de traitement et de mise à jour des MEs.

Les graphiques 7.11(a) et 7.11(b) décrivent les résultats obtenus durant cette campagne d'évaluation. Nous pouvons constater que l'évolution du temps de traitement des évène-

ments n'est pas parfaitement proportionnelle à l'évolution de la fréquence d'émission des évènements. En effet, cette évolution tend à se stabiliser dans les deux graphiques.



(a) Temps de transfert des évènements des MEs vers les MAs



(b) Temps de traitement et mise à jour des MEs

FIG. 7.11 – Evaluation du temps de traiment du service gestion des évènements

Le second constat est que le rythme d'évolution du temps d'acheminement des messages entre MEs et MAs évolue à un rythme inférieur au rythme d'évolution du temps requis pour la mise à jour des MEs suite à la réception d'une notification. Cette évolution trouve son explication dans la complexité des opérations de mise à jour (parcours des MEs hiérarchique, récupération des interfaces de mise à jour, opération de mise à jours) par rapport aux opérations de transfert de messages (simple routage de message en FIFO).

Les figures 7.11(a) et 7.11(b) illustrent l'évolution du temps de traitement des évènements suite à l'évolution de leur fréquence d'émission par les ressources administrées.

Un autre scénario à échelle réduite a été mis en place afin d'évaluer les deux modèles de collecte de données (*pull* et *push*). Nous avons testé cette fois une communication JMS entre les ressources M2M et les MEs. Le scénario s'est déroulé comme suit :

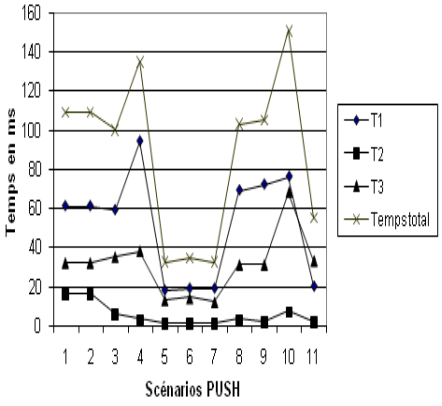
1. Génération de configurations de nœuds M2M selon la table 7.2,
2. Déploiement sur une seule machine M0 des MEs de des nœuds M2M,
3. Tests des deux modes *pull* et *push*,
4. Collecte des mesures du temps de transfert des messages.

Nous avons décomposé le temps de transfert de message afin de voir quelle partie du traitement consomme plus de temps lors du transfert des message entre MR et MEs. Nous nous sommes donc focalisé sur trois Δ temps.

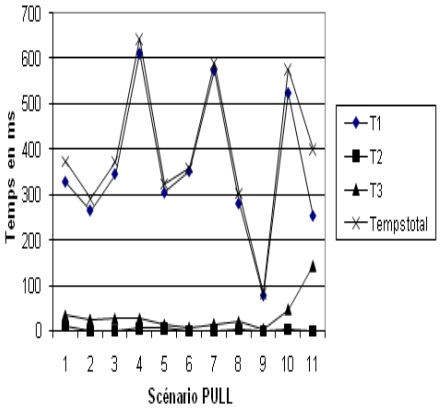
- T1 : Le temps passé entre la ressource administrée et le nœud d'administration M2M,
- T2 : Le temps passé dans le nœud d'administration M2M avant d'envoyer au ME concerné,
- T3 : Le temps de transfert des messages entre le nœud d'administration M2M et le ME DASIMA.
- Le temps total étant la somme des trois Δ (T1+T2+T3).

TAB. 7.2 – Table de paramétrage des tests du service de gestion des évènements

Scénarios	NbrNoeuds	NbrLiens	TailleCnx	Fréq chang état	Fréq pull
1	10	5	1000	1000	1000
2	2	1	1000	25000	1000
3	2	1	1000	1000	1000
4	2	1	1000	250	1000
5	2	1	1000	25	1000
6	2	1	1000	500	1000
7	2	1	1000	500	1000
8	2	1	1000	500	100
9	2	1	1000	500	200
10	2	1	1000	500	2000
11	2	1	1000	500	500



(a) Mesure du temps d'acheminement des évènements en mode push



(b) Mesure du temps d'acheminement des évènements en mode pull

FIG. 7.12 – Mesure des temps en mode push et pull

En analysant ces deux graphiques on peut constater que, dans les 11 scénarios, la portion la plus importante du temps total est passé entre le nœud d'administration et la ressource administrée (le nœud M2M dans notre cas). Nous ne pouvons pas ajuster ce temps pour ne pas perturber le déroulement de l'activité du réseau de nœuds administrés.

Le second constat est que les temps T2 et T3 sont relativement faibles. Cela est dû au paramétrage de l'envoi des notifications et à la fréquence de pull que nous avons adopté. Le scénario 11 illustre le cas où l'on allonge la fréquence du *pulling*. Il en résulte un pic au niveau de la courbe du T2.

Un toisième constat est que dans notre cas, le temps de transfert en mode push est réduit d'un facteur 3 en mode push par rapport au mode pull. En effet, en mode push, on réduit considérablement le temps de transfert de la notification entre le nœud administré et le nœud d'administration, ce qui explique une amélioration considérable du temps de transfert des notifications.

7.4.1.3 Evaluation du service de gestion de binding

Cette évaluation consiste à mesurer le temps nécessaire pour établir la chaîne de liaison entre une application d'administration déployée dans DASIMA et un ME de type M2M géré. L'objectif de cette évaluation est de pouvoir estimer le temps moyen de binding d'une interface ou de toutes les interfaces d'administration exposées par un ME. Le graphique 7.13 décrit le résultat de notre scénario avec 10, 100 et 1000 nœuds. Nous pouvons constater que le temps requis pour le binding de toutes les interfaces n'évolue pas proportionnellement au nombre d'interfaces bindés. Cela est dû à la factorisation du temps d'accès au ME dans le cas où on récupère toutes les interfaces.

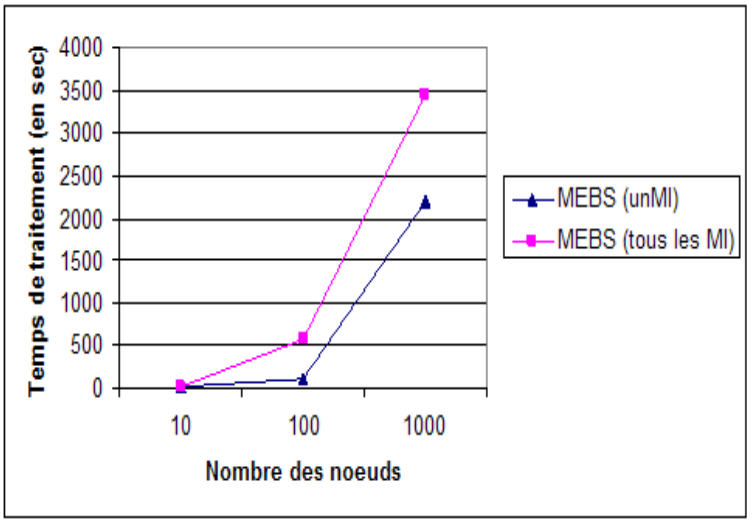


FIG. 7.13 – Evaluation du temps de traitement du service de gestion de binding

7.4.2 Evaluation à large échelle

7.4.2.1 Environnement d'évaluation

Afin d'évaluer dans un contexte large échelle, nous avons déployé DASIMA et l'application M2M sur la grille de calcul expérimentale Grid5000 [8]. Il s'agit d'une grille répartie sur 9 sites géographiques. Elle comporte 1500 nœuds et 3500 cœurs (CPU). La grille dispose d'une connexion réseau 10Gb/s dédiée. La figure 7.14 illustre la ditsribution globale de Grid5000. Elle décrit, l'interconnexion des 9 sites de Grid5000.

7.4.2.2 Variables observées

Durant les scénarios de tests, nous avons observé les informations suivantes au niveau des domaines DASIMA :

– **Consommation des ressources de calcul (CPU)** : Cette information a été collectée à partir du fichier `/proc/stat` à une fréquence d'une seconde.

[8] https ://www.grid5000.fr

FIG. 7.14 – Distribution globale de Grid5000

- **Consommation de la mémoire vive (RAM)** : Cette information concerne la mémoire vive utilisée au sein de la machine virtuelle Java. L'information considérée et collectée grâce à la méthode `public long freeMemory()` de la classe `java.lang.Runtime` Java. Elle est exécutée à chaque rajout d'élément administré au niveau du conteneur des éléments administrés de DASIMA.
- Charge moyenne (*average load*) : La charge moyenne de la machine représente le nombre de processus utilisant, ou en train d'attendre, le processeur. Cette information est disponible via les commandes `top` ou `uptime` sous Unix. Le nombre maximum de processus réellement en cours d'exécution dépend du nombre de processeurs (e.g. 1 pour un monoprocesseur, 2 pour un biprocesseur). Cette métrique permet d'avoir une idée globale sur la charge moyenne d'une machine (e.g. processeur, la mémoire, la vitesse des entrées/sorties sur disques).

7.4.2.3 Scénario1 : Evaluation des performances de DASIMA avec 1000 nœuds M2M

Afin d'évaluer les performances de DASIMA à large échelle, nous avons procédé à un déploiement de DASIMA sur Grid 5000. Ce scénario se décline en deux étapes :

- Le déploiement de DASIMA et de l'application M2M,
- Le déroulement des tests.

Déploiement de DASIMA et l'application M2M. La figure 7.15 décrit le scénario de déploiement d'une instance DASIMA et de 1000 nœuds M2M sur 5 sites géographiques de la grille Grid 5000. Ce déploiement est organisé de la manière suivante :

1. Déploiement d'une instance d'un domaine DASIMA,
2. Déploiement d'une instance d'un domaine intermédiaire DASIMA,
3. Déploiement de 10 instances de 100 de noœuds M2M (total 1000 nœuds),

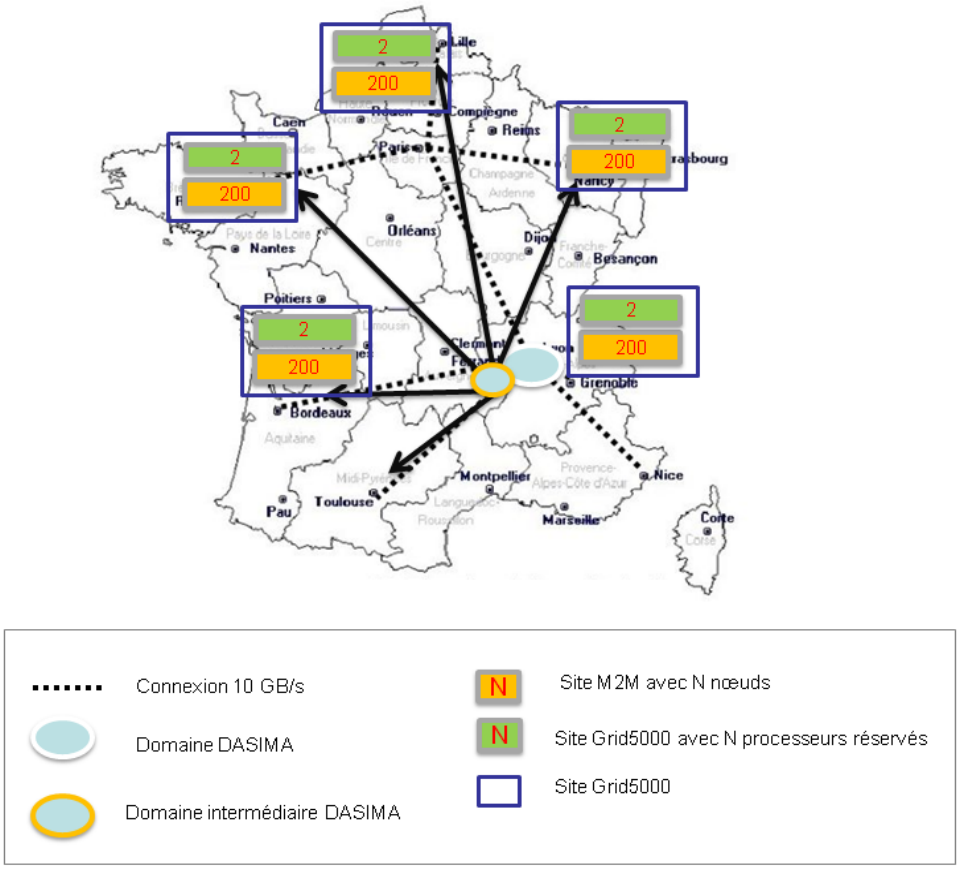


FIG. 7.15 – Architecture de déploiement de 1000 nœuds M2M sur GRID5000

Déroulement des tests. Après le déploiement des noeuds M2M et des instances DASIMA, nous avons procédé à l'exécution des tests comme suit :

- Activation des sondes systèmes au niveau du domaine DASIMA,
- Pramétrage, par l'administrateur, de l'instance du domaine intermédiaire DASIMA avec la liste des sites M2M (groupements de 100 nœuds),
- Affectation des sites M2M découverts, un par un, à l'instance de domaine DASIMA.
- L'instance de domaine DASIMA effectue la découverte des ressources de chaque site M2M découvert,
- L'instance de domaine DASIMA procède à une opération de nommage et de référencement des ressources à administrer,
- L'application d'administration déployée au sein de l'instance du domaine DASIMA effectue des requêtes d'administration.

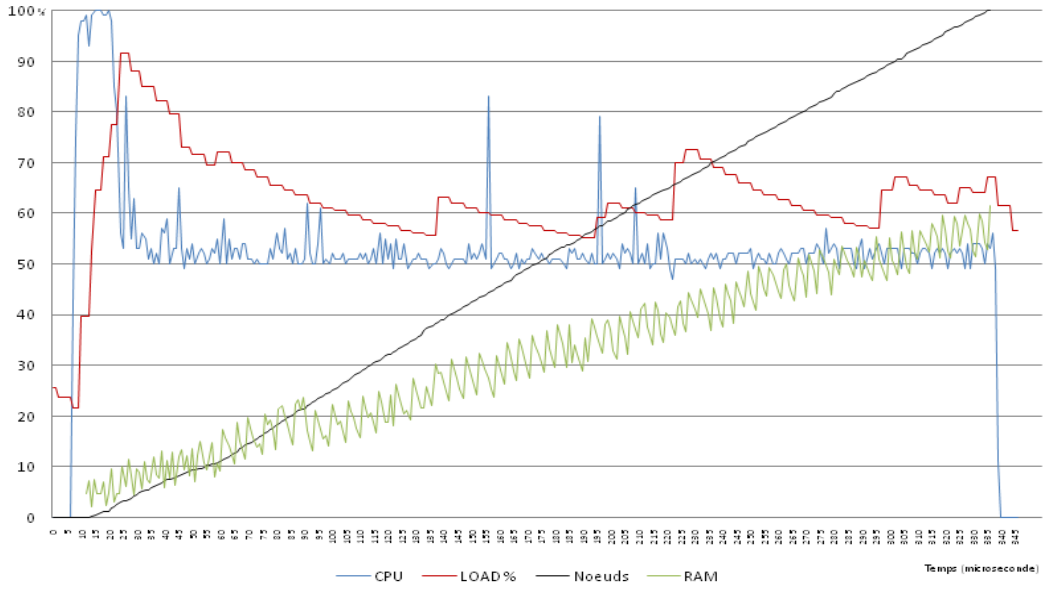


FIG. 7.16 – Courbe de performance du domaine pour 1 à 1000 nœuds M2M

Analyse des résultats. La figure 7.16 illustre différents aspects des performances de DASIMA observés durant le déroulement de ce scénario.

En observant la courbe de consommation de la mémoire vive, nous pouvons constater que l'évolution de la consommation mémoire est proportionnelle à l'évolution du nombre des nœuds administrés. L'aspect dents de scie de la courbe est lié à l'activation volontaire du ramasse miettes de la machine virtuelle durant les tests.

La consommation CPU présente une charge moyenne de 50%. Cette charge est plutôt stable et varie en moyenne entre 50% et 55%. Le pic de consommation CPU correspond au démarrage de DASIMA et au chargement de l'ensemble des services et à la fabrication de l'ensemble de composants de DASIMA.

Quant à la charge moyenne (LOAD), elle oscille, en moyenne, entre 55 % et 73%. Cette charge est acceptable et montre que l'activité de DASIMA durant les opérations d'administration réalisées, ne staure pas les ressources systèmes. Par ailleurs, tout comme la consommation CPU, la charge moyenne connaît un pic à plus de 90% au démarrage de l'application DASIMA.

Afin d'affiner notre analyse nous avons complété la courbe 7.17 avec une courbe indiquant le temps d'activation des services DASIMA pour obtenir la courbe de la figure 7.16. Cette courbe nous permet de comprendre la corrélation entre les variations de la charge système (CPU et LOAD) et l'activation des services DASIMA. Nous pouvons constater, en effet, que les pics de consommation de CPU correspondent aux périodes d'activation des services de nommage et de binding de DASIMA.

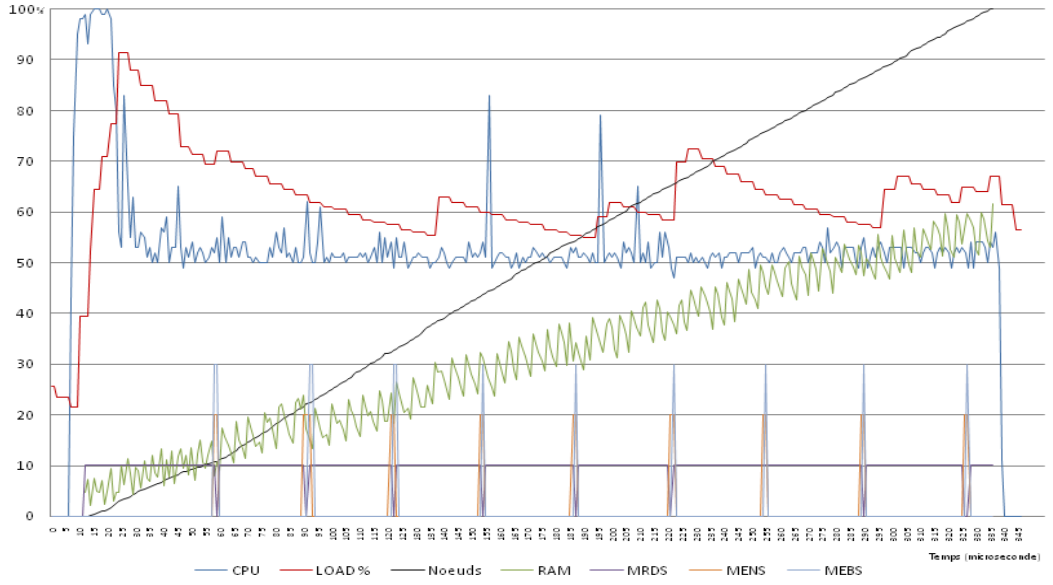


FIG. 7.17 – Performance du domaine pour 1 à 1000 nœuds M2M avec temps services DASIMA

7.4.2.4 Scénario2 : Evaluation des performances de DASIMA avec 1.000.000 nœuds M2M

Ce scénario se décline en deux étapes : a) le déploiement de DASIMA et de l'application M2M et b) le déroulement des tests.

Déploiement de DASIMA et l'application M2M. La figure 7.18 illustre le déploiement de DASIMA et de nœuds M2M sur la plateformce expérimentale Grid 5000. Le déploiement est organisé de la manière suivante :

- Déploiement de 1000 instances de domaines DASIMA,
- Déploiement d'une instance d'un domaine intermédiaire DASIMA,
- Déploiement de 1000 instances de 1000 noœuds M2M (total 1.000.000 nœuds),

Déroulement des tests. La phase de déroulement des tests a été comme suit :

- Activation des sondes systèmes au niveau du domaine intermédiaire DASIMA,
- L'instance de domaine intermédiaire DASIMA découvre grâce à un fichier de configuration la liste des sites (groupement de 1000 nœuds) M2M,
- L'instance de domaine DASIMA demande au domaine intermédiaire DASIMA de lui affecter un site non administré,
- L'instance de domaine DASIMA procède à une opération de découverte des ressources du site M2M,
- L'instance de domaine DASIMA procède à une opération de nommage ainsi qu'une opération des binding des ressources.

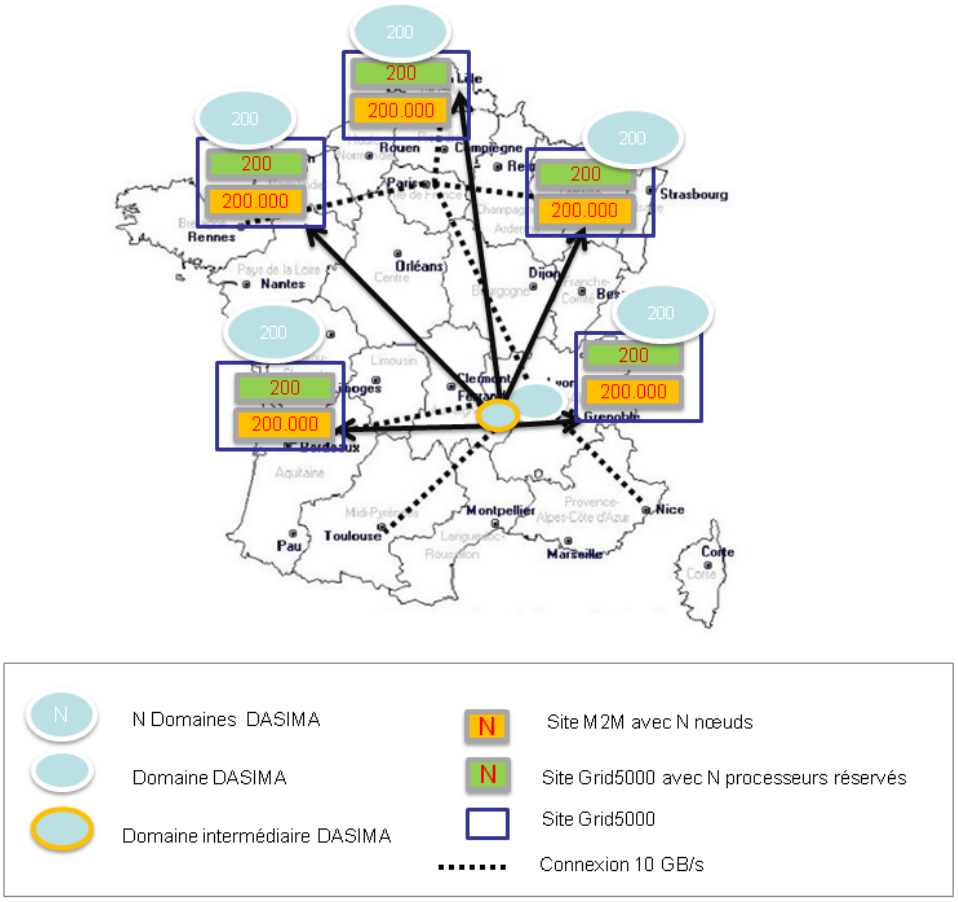


FIG. 7.18 – Architecture de déploiement de 1.000.000 nœuds M2M sur GRID5000

Analyse des résultats. La courbe de la figure 7.19 illustre les performances systèmes du domaine intermédiaire lors de l'affectation de 1000 sites de 1000 nœuds chacun aux domaines DASIMA.

Le domaine intermédiaire gère l'affectation des groupements d'éléments administrés aux domaines DASIMA existants. La courbe d'affectation des nœuds décrit le rythme d'affectation des 1000 sites M2M aux domaines DASIMA.

En observant ce graphique, nous pouvons constater que la charge moyenne du domaine intermédiaire est plus de 20% au lancement et elle est casi-nulle par la suite. Cette charge est liée d'une part au coût de chargement du domaine. Elle inclut également le coût de chargement du fichier d'initialisation du domaine intermédiaire. Nous pouvons constater que l'activité du processeur du domaine intermédiaire est globalement stable et faible. Elle oscille entre 0% et 20% étant donné qu'il est occasionnellement sollicité par les domaines DASIMA. La consommation de la mémoire vive est également stable et ne dépasse pas les 10%. L'allure en dents de scie est liée à l'activation ponctuelle du ramasse-miettes.

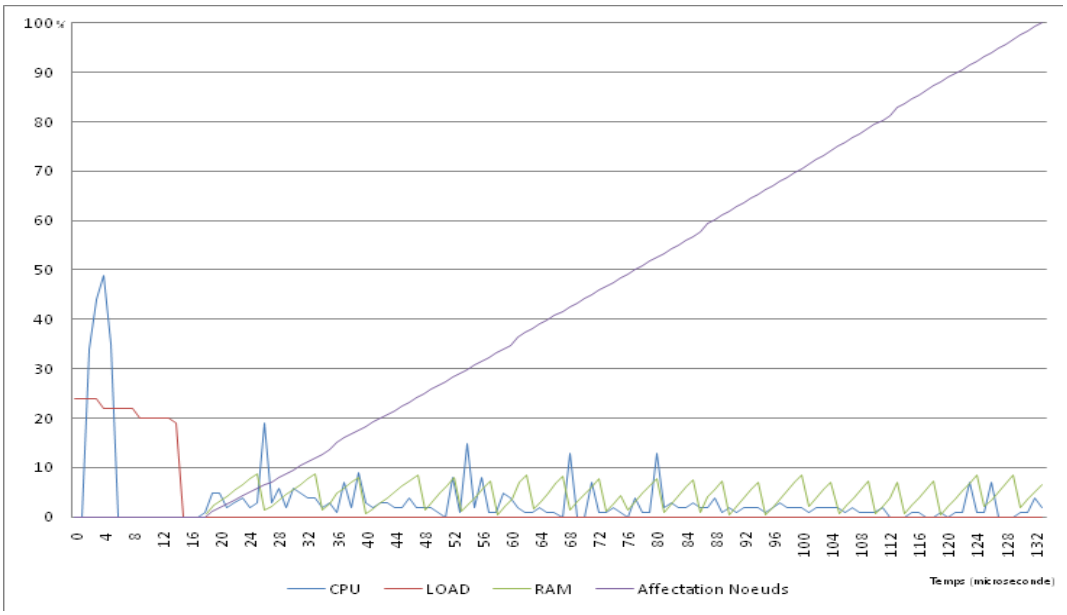


FIG. 7.19 – Courbe des performances du domaine intermédiaire (1.000.000 nœuds) M2M

7.4.2.5 Synthèse

A l'issue de ces expérimentations à large échelle, nous pouvons faire les constats suivants :

- Avec DASIMA, nous pouvons administrer aussi bien un nombre faible de ressources administrées, qu'un nombre potentionnelement important (1.000.000) de nœuds,
- La consommation de la mémoire vive dans DASIMA augmente proportionnellement à l'évolution du nombre des éléments administrés,
- La consommation des ressources de calcul et la charge globale reste acceptable pour une administration de 0 à 1000 nœuds M2M,
- Avec le scénario 2 nous avons constaté que le surcoût (en termes de charge système) d'un domaine intermédiaire est négligeable par rapport aux possibilités de passage à l'échelle qu'il permet,
- Grâce au domaine intermédiaire nous avons pu pallier les limitations des ressources systèmes limités d'un domaine DASIMA.

7.5 Evaluation qualitative

Nous présentons dans cette section une évaluation qualitative de notre travail. Nous discutons les trois points suivants : la compatibilité de DASIMA avec les standards de l'administration, la pertinence de l'approche architecturale et une comparaison de notre travail avec l'état de l'art.

7.5.1 Compatibilité avec les standards

Notre travail est compatible à la fois avec les standards d'administration applicative et de l'administration réseaux. Deux technologies d'instrumentation ont été implémentées : JMX pour l'administration applicative et SNMP pour l'administration des équipements réseaux. Nous avons donc gardé l'instrumentation JMX en conservant les interfaces MBean du serveur. L'application du concept de domaine permet de pallier le problème des structures plates des MBeans JMX dans un contexte multi-échelles. La combinaison de DASIMA et de JMX permet ainsi d'obtenir une hiérarchisation de la vue des ressources administrées. Cette extension présente les avantages suivants :

- une amélioration de la lisibilité de la vue des ressources administrées,
- une gestion par groupes de ressources gérées au niveau d'un même serveur MBean,
- une gestion par groupes de ressources réparties entre plusieurs serveurs MBean.

7.5.2 Pertinence de l'approche

Extensibilité. Le premier point fort de notre approche est la généricité de la plateforme DASIMA qui peut être appliquée aussi bien pour l'administration des réseaux que pour le pilotage de systèmes d'informations dans son intégralité (processus métiers par exemple). En effet, moyennant une description architecturale de la ressource administrée et un adaptateur adéquat, la gestion de la ressource est immédiatement prise en charge par la plateforme d'administration et les applications d'administration correspondantes.

Reconfigurabilité. Le second point fort de notre approche est la possibilité de reconfigurabilité intégrale de la plateforme d'administration. Cette reconfigurabilité peut être opérée sur deux niveaux

- le niveau comportemental : c'est le niveau de configurabilité le plus répandu dans les applications classiques d'administration à travers des paramètres divers tel que la gestion de la fréquence de sauvegarde des données ou de la fréquence de *pooling*.
- le niveau structurel : qui distingue notre application du reste des systèmes d'administration commercialisés. En effet, comme l'intégralité de la plateforme DASIMA est implémentée sous forme de composants, nous avons une très fine granularité de reconfiguration.

Administrabilité. La troisième originalité de notre proposition est la possibilité d'administrabilité de la plateforme d'administration elle-même. Cette caractéristique est matérialisée à travers le concept de domaine de domaines, et est particulièrement intéressante dans un contexte muti-échelles pour plusieurs raisons :

- Gestion proactive des pannes (notamment les pannes liées à la montée en charge),
- Optimisation de l'utilisation des ressources via les leviers d'ajustement en fonction de la charge de la plateforme d'administration,
- Amélioration de la robustesse et la disponibilité des plateformes d'administration à travers les possibilités d'extension dynamique ou la correction dynamique des pannes sans interruption du service.

7.6 Synthèse

Ce chapitre décrit la réalisation du canevas DASIMA et défend les choix conceptuel et techniques adoptés. Ensuite, il introduit l'application M2M considérée dans le cadre de nos expérimentations de validation. Après, il présente une double évaluation de DASIMA est proposée sur le plan qualitatif et quantitatif. Sur le plan quantitatif, nous avons considéré deux aspects : a) la compatibilité avec les standards et b) la pertinence de l'approche d'administration proposée. Alors que sur le plan quantitatif, nous avons procédé à des évaluations à différentes échelles : à une échelle réduite (réseau local) et à grande échelle sur une grille à l'échelle national (Grid 5000). Durant ces évaluations quantitatives, nous avons effectué une observation aussi bien au niveau applicatif (temps de réponse des services) que système (consommation des ressources systèmes). Les scénarios d'évaluation des performances considérés nous ont permis d'évaluer DASIMA avec des applications M2M allant de 100 à 1.000.000 de nœuds. Nous avons ainsi montré l'adaptation de notre approche à différentes échelles de déploiement et avec un nombre de ressources variable et potentiellement important.

Chapitre 8

Conclusion : bilan et perspectives

Contents

8.1 Rappel des objectifs 159
8.1.1 Administration multi-échelles 159
8.1.2 Hétérogénéité des ressources administrées 160
8.1.3 L'administration intégrée 160
8.2 Contributions 160
8.2.1 Une approche d'administration intrégrée multi-échelles 161
8.2.2 Un canevas d'administration à base de composants 161
8.2.3 Implémentation du canevas DASIMA 161
8.2.4 Etude de cas en contexte industriel 162
8.2.5 Evaluation et recommandations dans un contexte large échelle . 162
8.3 Perspectives 162

DANS ce dernier chapitre, nous commençons par rappeler la problématique de la thèse, nous dressons ensuite un bilan des principaux apports de nos travaux et, enfin, nous en présentons les perspectives qui pourront êtres poursuivies.

8.1 Rappel des objectifs

Nous rappelons ici brièvement les objectifs de cette thèse qui s'articulent autour des trois points suivants.

8.1.1 Administration multi-échelles

L'administration multi-échelles consiste à opérer des tâches d'administration via la même infrastructure d'administration à différentes échelles (numériques, organisationnelles et géographiques) [MkL06b]. L'administration des passerelles de services est un exemple concret qui illustre le besoin d'administration multi-échelles. En effet, cette tâche requiert un élargissement du champ d'intervention des opérateurs de télécommunication et des fournisseurs de services. Il s'agit donc d'une administration qui dépasse les frontières du réseau de l'opérateur pour s'étendre aux réseaux domestiques des particuliers. Chacun de

ces deux réseaux représente un contexte différent et requiert donc un système d'administration adapté à ses ressources. Le réseau d'opérateur nécessite, par exemple, des ressources d'administration dotées d'une forte capacité de collecte et de traitement des données. Alors que les réseaux domestiques requièrent des mécanismes d'administration moins gourmands en ressources de calcul ayant un minimum d'impact intrusif sur les ressources administrées. Une solution possible consiste à mettre en place une plateforme d'administration capable d'adapter à la fois sa structure et son «comportement »en fonction de son contexte de déploiement.

8.1.2 Hétérogénéité des ressources administrées

Très souvent, les contextes multi-échelles, tels que les réseaux d'opérateurs, sont caractérisés par une forte hétérogénéité des ressources à administrer. Les services de télécommunication M2M[1] ou "*triple play*" [2] sont des exemples concrets de contextes multi-échelles avec une forte hétérogénéité des ressources. Ces ressources présentent hétérogénéité à différents niveaux (matériel et logiciel). Elles sont réparties et interconnectées par des réseaux. Cette diversité est exacerbée dans contexte multi échelles contenu du nombre et des types des ressources et considérées.

8.1.3 L'administration intégrée

Il y a une décennie, la gestion pour l'entreprise se limitait à la gestion de son infrastructure systèmes et réseaux. Aujourd'hui, le périmètre de l'activité d'administration a radicalement changé. Désormais, l'administration va au delà de l'infrastructure pour couvrir l'ensemble des services et des applications du système d'information. Une administration dite *intégrée* devient alors indispensable afin de permettre une cohérence globale des systèmes et des infrastructures. Selon cette approche, les informations liées à l'infrastructure sont gérées au même titre que les informations liées aux services et aux applications de l'entreprise. C'est ce que nous appelons *administration intégrée*. Un exemple d'administration intégrée peut consister à surveiller et contrôler l'ensemble des couches logicielles et matérielles par lesquelles transitent les services d'un fournisseur de services de télécommunication. Dans un contexte multi-échelle, faire de l'administration intégrée peut s'avérer particulièrement ardu vu l'hétérogénéité des ressources et leur nombre.

En considérant ces objectifs, le résumé et le bilan des contributions de cette thèse est présenté dans la section suivante.

8.2 Contributions

Cette section présente une synthèse des principales contributions théoriques et expérimentales de cette thèse. Nous rappelons, dans un premier temps, notre approche d'administration intrégrée multi-échelles. Ensuite, nous présentons le canevas logiciel DASIMA que nous avons développé afin de mettre en œuvre l'approche proposée. Nous terminons avec les évaluations quantitatives et qualitatives réalisées pour évaluer ce travail.

[1] Machine to Machine

[2] Service de télécommunication combinant à la fois offre de téléphonie fixe, internet et télévision

8.2.1 Une approche d'administration intrégrée multi-échelles

Ce travail présente une approche d'administration intégrée dans un contexte multi-échelles. Cette approche a été introduite dans le chapitre 5. Elle s'articule autour de trois principes :

- **L'administration à base de domaines.** Cette approche permet de disposer d'un système capable de gérer des groupes de ressources d'une manière ensembliste. Nous avons proposé des domaines d'administration comme étant le moyen de grouper des ressources administrées. Nous avons également proposé une classification détaillée des typologies des domaines ainsi que les différentes stratégies d'administration, les organisations possibles des domaines.
- **L'administration orientée par l'architecture des ressources administrées.** L'administration orientée par l'architecture permet de pallier la problématique liée à l'hétérogénéité des ressources en proposant une vue homogène basée sur une abstraction architecturale des ressources administrées.
- **L'administration réflexive.** Cette approche permet de produire un système d'administration « conscient»de son état et qui propose des mécanismes d'auto-administration, facilitant ainsi son adaptation à différentes échelles ou dans différents contextes de déploiement. En injectant un Agent FractlJMX dans le domaine principal DASIMA nous avons pu introspecter et administrer les ressources administrées ainsi que le système d'administration lui même (tous les sous-domaines et services d'administration) rendant DASIMA réflexif.

8.2.2 Un canevas d'administration à base de composants

Nous proposons dans le cadre de ce travail le canevas DASIMA [MkL06a, MkL06b]. Il s'agit d'un canevas à composants pour l'administration intégrée dans un contexte multi-échelles. DASIMA se présente comme un intergiciel d'administration assurant le rôle d'intermédiaire entre des applications d'administration des ressources administrées. Du point de vu fonctionnel, DASIMA offre principalement des services communs d'administration tels que la gestion des notifications, du nommage et de la persistance des informations de gestion. Sur le plan technique, la conception et l'implémentation du canevas s'appuient sur le modèle à composants Fractal.

L'architecture interne de DASIMA est résolument orientée services. Ce choix architectural permet de disposer d'une architecture modulaire et extensible, deux propriétés indispensables dans un contexte multi-échelles. L'ensemble des services DASIMA, ainsi que les domaines d'administration et les représentations architecturales ont été implémentés sous forme de composants Fractal. En combinant composants et services, nous avons obtenu une architecture flexible et faiblement couplée. Dans notre cas, le modèle à composants Fractal nous offre des facilités d'introspection et d'extension à la volée sans interruption de l'activité du système d'administration.

8.2.3 Implémentation du canevas DASIMA

Afin d'implémenter DASIMA, nous nous sommes appuyés sur le langage Java et sur Julia, l'implémentation Java du modèle à composants Fractal. Le canevas est constitué d'un ensemble de classes Java dont l'assemblage est effectué à l'éxécution et à la demande, grâce

à des fichiers ADL (*Architecture Definition Language*) décrivant l'architecture logicielle DASIMA. Le code est constitué de 7101 lignes de code Java réparties sur 93 classes et 63 interfaces. Le code de DASO contient également 45 fichier Fractal ayant un total de 638 lignes et s'appuie sur des briques intergicielles libres tels que JORAM, Fractal-JMX, Composite-Probes et Speedo.

8.2.4 Etude de cas en contexte industriel

Pour la validation de notre approche d'administration, nous avons opté pour une étude de cas issue du monde des télécommunications. Il s'agit d'une application utilisée par l'opérateur France Télécom pour la gestion de la communication entre ressources hétérogènes. La plateforme se présente comme une réseau logique de nœuds pour la gestion des *workflow* des données entre des supports hétérogènes. Cette étude de cas est représentative des contextes multi-échelles et fortement hétérogènes car la taille de ce réseau logique et l'état de ces nœuds peuvent considérablement changer en production.

8.2.5 Evaluation et recommandations dans un contexte large échelle

Afin d'évaluer notre approche, nous avons procédé à des évaluations à deux niveaux, qualitatif et quantitatif [MkL09]. Sur le niveau qualitatif, nous avons discuté la compatibilité de DASIMA avec les standards de l'administration, la pertinence de l'approche architecturale ainsi qu'une comparaison de notre travail avec les travaux de l'état de l'art. Sur le plan quantitatif, nous avons fait dans un premier temps une évaluation sur une échelle réduite. Ensuite, nous avons mené une campagne à l'échelle nationale avec plus d'un million de nœuds M2M déployés sur la plateforme expérimentale Grid5000. Grace à ces évaluations quantitatives, non seulement nous fournissons une boite à outils pour l'administration multi-échelles, mais nous offrons également des données utiles pour son calibrage.

8.3 Perspectives

De nombreux défis restent à relever. En ce qui concerne notre travail, nous avons identifié quelques pistes de recherche. Nous tenons à explorer certaines d'entre elles sur le court et le long terme.

Automatisation du déploiement. L'une des principales limitations de DASIMA est l'absence d'un mécanisme de déploiement automatisé des domaines. Une piste possible pour remédier à cette limitation pourrait être la combinaison des deux technologies Fractal et OSGI [MDD04]. Ce déploiement pourra être élargi pour inclure même le déploiement automatisé des sondes (systèmes et applicatives).

Intégration avec JADE. Nous avons présenté le canevas JADE [DBB+08] dans la section 3.5.1. Bien que notre travail et Jade aient quelques similarités sur le plan technique (choix du modèle à composants par exemple), il existe une nette divergence entre ces choix architecturaux et les problématiques adressées. Jade étant principalement dédiée à la problématique d'administration autonomique pour les clusters

JavaEE, il se focalise sur les règles d'automatisation d'administration ou de déploiement automatisé.

Des discussions entre notre équipe et les équipes de Jade ont été amorcés afin de produire un canevas qui réunisse les deux approches. Trois pistes d'intégration sont envisageables :

- DASIMA comme un ME au niveau de Jade : cette solution permet d'automatiser le déploiement de DASIMA et de l'administrer via Jade.
- Enrichir DASIMA avec les fonctionnalités de Jade : Ces fonctionnalités incluent la base de règles de gestion de Jade, son mécanisme de déploiement ainsi que et sa bibliothèque de sondes.
- Enrichir Jade avec les fonctionnalités de DASIMA : Ces fonctionnalités incluent par exemple la gestion des messages, de la persistance des ME, la gestion par domaine et la découverte automatisée des ME.

Extension pour la gestion des flux de données. La gestion de données de capteurs est devenue un domaine de recherche très actif dans ces derniers temps grâce à la reconnaissance, par les milieux scientifiques et industriels, de l'utilité des capteurs pour divers domaines d'application. Une amélioration possible pour DASIMA pourrait être une extension avec un service de gestion de flux de données afin de prendre en considération les flux d'informations remontées par les agents et les capteurs. Cette extension peut être particulièrement utile pour la gestion du trafic sur le réseau ou pour la supervision en temps réel des flux massifs de données. Une piste possible pourrait être une intégration du service de gestion des flux de proposés dans le canevas SStreaMWare [LGO08]. Pour l'instant, cette piste est en cours d'exploration avec le canevas XSStreamware [GH09].

Bibliographie

[ABKS06] Takoua Abdellatif1, Fabienne Boyer, Jakub Kornas, and Jean-Bernard Stefani. Administration fondée sur l'architecture des serveurs d'applications j2ee patrimoniaux. In *Proceedings of 5ème Conférence Française sur les Systèmes d'Exploitation (CFSE-5)*, Perpignan, France, 2006.

[Ada07] Stadler R. Chunqiang Tang Steinder M. Spreitzer M. Adam, C. A service middleware that scales in system size and applications. In *Proc. IFIP/IEEE International Symposium Integrated Network Management, (IM'09)*, pages 70–79, June 2007.

[BCL+06] E. Bruneton, T. Coupaye, M. Leclercq, V. Quéma, and J.B. Stefani. The fractal component model and its support in java, software practice and experience. *Special issue on Experiences with Auto-adaptive and Reconfigurable Systems*, 36(11) :1257–1284, Septembre 2006.

[BCU+04] G. Blair, G. Coulson, J. Ueyama, K. Lee, and A. Joolia. Opencom v2 : A component model for building systems software. In *Proceedings of IASTED Software Engineering and Applications*, USA, 2004.

[Ber06] Takoua Abdellatif Berrayana. Apport des architectures composants pour l'administration des intergiciels, thèse de doctorat. Technical report, Institut national Polytechnique de Grenoble (INPG), September 2006.

[BJ09] Grampin E. Vidal L. Giachino M. Baliosian J., Visca J. A rule-based distributed system for self-optimization of constrained devices. In *Proc. IFIP/IEEE International Symposium Integrated Network Management, (IM'09)*, pages 41–48, June 2009.

[BM09] Mingardi C. Nunzi G. Brunner M., Dudkowski D. Probabilistic decentralized network management. In *Proc. IFIP/IEEE International Symposium Integrated Network Management, (IM'09)*, pages 25–32, June 2009.

[CA09a] Pavlou G. Chourmouziadis A., Gonzalez Duque O.F. Experiences in using muws for scalable distributed monitoring. In *Proc. IFIP/IEEE International Symposium Integrated Network Management, (IM'09)*, pages 561–568, June 2009.

[CA09b] Del Sol J.A. Nieto G. Cuadra A., Garces F. A generic end-to-end monitoring architecture for multimedia services. In *Proc. IFIP/IEEE International Symposium Integrated Network Management, (IM'09)*, pages 129–132, June 2009.

[CCJ+07] H. Cha, S. Choi, I. Jung, H. Kim, J. Yoo H. Shin, and C. Yoon. Retos : Resilient, expandable, and threaded operating system for wireless sensor networks. In *The Sixth International Conference on Information Processing in Sensor Networks (IPSN'07)*, Massachusetts,USA, April 2007.

[CFK+98] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. A resource management architecture for metacomputing systems. In *Proceeding of Workshop on Job Scheduling Strategies for Parallel Processing*, pages 62–82. Springer-Verlag, 1998.

[CHSS04] S.W. Cheng, A.C. Huang, B. Schmerl, and P. Steenkiste. Rainbow : Architecture-based self-adaptation with reusable infrastructure. *Computer*, 37(8) :46–54, 2004.

[CIM99] Common information model standard (cim) specification v2. Technical report, DMTF, June 1999.

[CL02] T. M. Chen and S. S. Liu. A model and evaluation of distributed network management approaches. *IEEE Journal on selected areas in communications*, 20(4) :850–857, May 2002.

[COM] COM : Component Object Model Technologies, http ://www.microsoft.com/com/.

[Cou07] Geoff Coulson. What is reflective middleware ? *IEEE Distributed Systems Online*, 2007.

[DBB+08] Noel Depalma, Sara Bouchenak, Fabienne Boyer, Daniel Hagimont, Sylvain Sicard, and Christophe Taton. Jade : un environnement d'administration autonome. *Technique et Science Informatiques*, 2008.

[DCO] DCOM (Distributed Component Object Model).http ://www.microsoft.com/com/default.mspx.

[Der95] Luca Deri. Droplets : Breaking monolithics applications apart. Technical Report RZ 2799, IBM, September 1995.

[Der97] Luca Deri. A component-based architecture for open, independently, extensible distributed systems, phd thesis. Technical report, University of Bern, June 1997.

[Dia07] Ada Diaconescu. Composite probes : a monitoring framework for organising data into configurable hierarchies. Technical report, France Telecom Recherche et Developpement, february 2007.

[DK01] Linda DeMichiel and Michael Keith. Corba 3.0 new components chapters. Technical Report OMG TC Document, OMG, 2001.

[DK06] Linda DeMichiel and Michael Keith. Enterprise javabeans, version 3.0. Technical Report Jsr 220, Sun Microsystems, 2006.

[DvdHT02] E. Dashofy, A. van der Hoek, and R. Taylor. Towards architecture-based self-healing systems. In *Proceedings of the 1st Workshop on Self-Healing Systems(WOSS'02)*. ACM Press, 2002.

[EC] Oussama Layaida Vivien Quema Emmanuel Cecchet, Hazem Elmeleegy. In *Proc. OOPSLA 2004 Workshop.*

[GBS+05] V. Gucer, C. Beganskas, F. Salustri, J. Saulman, M.Toure, J. Willis, B. Yamazi, A. Luís Scandolara, G. Shah, S. Wallace, and D. Wouters. Getting started with ibm tivoli monitoring 6.1 on distributed environments. Technical Report IBM Redbooks, IBM, December 2005.

[GCB+04] P. Grace, G. Coulson, G. Blair, L. Mathy, W. Yeung, W. Cai, D. Duce, and C. Cooper. Gridkit : Pluggable overlay networks for grid computing. In *Proceedings of Distributed Objects and Applications (DOA'04)*, October 2004.

[GGL+03] P. Goldsack, J. Guijarro, A. Lain, G. Mecheneau, P. Murray, and P. Toft. Smartfrog : Configuration and automatic ignition of distributed applications. In *Proceedings of 10th Workshop of the HP OpenView University Association (HPOVUA'03)*, June 2003.

[GH09] Levent Gürgen and Shinichi Honiden. An integrated management middleware for networked sensing systems. In *In Pervasive 2009 - Late breaking results and posters.*, May 2009.

[GHVJ95] Quatre Són E. Gamma, R. Helm, J. Vlissides, and I R Johnson. In *Design Patterns : Elements of Reusable Object-Oriented Software*, 1995.

[GJ+05] James Gosling, Bill Joy, , Guy Steele, and Gilad Bracha. The java language specification (third edition). Technical report, Sun Microsystems, Mars 2005.

[Gol96] Germán. Goldszmidt. Distributed management by delegation. ph.d thesis. Technical report, Graduate, School of Arts and Sciences, Columbia University, 1996.

[Gui02] P. Goldsack Guijarro, M. Monjo. Framework for managing large scale component-based distributed applications using jmx. In *Proceedings of 9th International Workshop of the HP OpenView University Association (HPOVUA'02)*, June 2002.

[IEC07] Performance management for next-generation networks. Technical report, The International Engineering Consortium, 2007.

[JC07] Inuk Jung and Hojung Cha. Rmtool : Component-based network management system for wireless sensor networks. In *Proceedings of the 2007 IEEE Consumer Communications and Networking Conference (CCNC'07)*, Las Vegas, USA, 2007.

[JCD90] M. Schoffstall J. Case, M. Fedor and J. Davin. A simple network management protocol (snmp). Technical Report RFC1157, IETF, May 1990.

[JCD04] M. Schoffstall J. Case, M. Fedor and J. Davin. Hp openview network node manager : A guide to scalability and distribution. Technical report, HP, July 2004.

[Jen09] R. Donnelly W. Foley S.N. Lewis D. O'Sullivan D. Strassner J. van der Meer S. Waterford Jennings, B. Brennan. Challenges for federated, autonomic network management in the future internet. In *Proc. IFIP/IEEE International Symposium. Integrated Network Management-Workshops (IM'09)*, pages 87–92, June 2009.

[JLM99] T. Portas J. Luis and T. Ben Meriem. Spécification et implémentation d'un modèle de gestion. In *3ème Colloque Francophone sur la Gestion de Réseaux et de Services (GRES'99)*, pages 290–302, Juin 1999.

[JMX04] Java management extensions (jmx) specification version 1.2. Technical report, Sun Microsystems, Mars 2004.

[JS00] C. Jeong and S. M.M. Shahsavari. Component-based distributed network management. In *Proceedings of the 2000 IEEE, SoutheastCon*, pages 460–466, April 2000.

[KB97] M . Kahani and HWP Beadle. Decentralized approaches for network management. *ACM Computer Communication Review*, 27(3) :36–47, 1997.

[KM09] Mylonas G. Nikoletseas S. Varvarigos E. Rolim J. Kalochristianakis M., Gkamas V. An open and integrated management platform for wireless sensor

networks. In *Proc.International Symposium on Autonomous Decentralized Systems,(ISADS'09)*, pages 1–6, August 2009.

[KYM+00] T. Kawabata, I. Yoda, H. Maeomichi, M. Tago, and K. Yata. Component-oriented network management system development. In *Proceedings of the IEEE/IFIP 2000 Network Operations and Management Symposium (NOMS'00)*, pages 290–302, April 2000.

[LGO08] Cyril Labbé André Bottaro Levent Gürgen, Claudia Roncancio and Vincent Olive. Sstreamware : A service-oriented middleware for heterogeneous sensor data management. In *Proc. the International Conference on Pervasive Services (ICPS'08)*, July 2008.

[LS05] K.-S. Lim and R. Stadler. Real-time views of network, traffic using decentralized management. In *Proceedings 9th International Symposium on Integrated Network Management*, 15-19 May 2005.

[M2M06] Machine to machine (m2m) : enjeux et perspectives. Technical report, Livre Blanc produit par la FING, Syntec Informatique et Orange, mars 2006.

[Mar08] Daniel Bryan Marconett. Autonomic network management : A proposed framework, master of science thesis. Technical report, University of California, Davis, June 2008.

[Mcl68] Malcolm Douglas Mcllroy. Software engineering, report on a conference sponsored by the nato science committee, chapter mass-produced software components. Technical report, Scientific Afairs Division, NATO,Garmisch, October 1968.

[Mcm07] E. Mcmanus. Presentation on the jdk platform and the jmx api. Montbonnot, Juin 2007. Sun Microsystems.

[MDD04] Humberto Cervantes Mikael Desertot and Didier Donsez. Frogi : Fractal components deployment over osgi. In *Proc. 1st French Conference on Software Deployment and (Re) Configuration (DECOR'04)*, October 2004.

[MF02] Jean-Philippe Martin-Flatin, editor. *Web-Based Management of IP Networks and Systems*. Wiley Series in Communications Networking & Distributed Systems, 2002.

[Mic06] Sun Microsystems. Javatm management extensions (jmx), specification, version 1.4 5(final release). Technical Report JSR003, Sun Microsystems, November 2006.

[MkL06a] C. Roncancio T. Coupaye Mehdi kessis, P. Déchamboux and A. Lefebvre. Towards a flexible middleware for autonomous integrated management applications. In *International Multi-Conference on Computing in the Global Information Technology (ICCGI06)*, page 27, 2006.

[MkL06b] C. Roncancio T. Coupaye Mehdi kessis, P. Déchamboux and A. Lefebvre. Un middleware flexible et scalable pour la gestion intégrée des réseaux et des services à large échelle. In *7ème Colloque francophone de Gestion de Réseaux et de Services (GRES'06), thème : L'autonomie dans les réseaux et les services*, 2006.

[MkL09] C. Roncancio Mehdi kessis and A. Lefebvre. Dasima : A flexible management middleware in multi-scale contexts. In *6th International Conference on Information Technology : New Generations, Track Middleware*, 2009.

[MKP99] H.D. Hofmann M. Knahl and A. Phippen. A distributed component framework for integrated network and systems management. *Information management and computer security*, 7(5) :254–260, 1999.

[Nag07] Nagios version 3.x documentation. Technical report, Ethan Galstad, Mars 2007.

[O.08] Negru O. End-to-end qos through integrated management of content, networks and terminals : The enthrone project. In *Proc. IEEE/ACS International Conference on Computer Systems and Applications, (AICCSA'08)*, pages 1019–1020, 2008.

[OM03] Kephart Jeffrey O. and Chess David M. The vision of autonomic computing. In *Computer*, volume 36, pages 41–50. IEEE Computer Society Press, January 2003.

[Ope] HP OpenView Management System, http ://www.france.hp.com/ solutions/ entreprises/ grandesentreprises/ openview/.

[OSG] OSGi, Open Services Gateway Intitative. http ://www.osgi.org/.

[PCDC08] Marc Léger Pierre-Charles David, Thomas Ledoux and Thierry Coupaye. Fpath and fscript : Language support for navigation and reliable reconfiguration fractal architectures. *Annals of Telecommunications, Journal edited by Lavoisier. Special issue on Software Components - The Fractal Initiative*, 2008.

[PVRY05] JB Stefani S. Haridi T. Coupaye A. Reinefield E. Winter P. Van Roy, A. Ghodsi and R. Yap. Self management of large-scale distributed systems by combining structured overlay networks and component. In *In CoreGRID Integration Workshop*, November 2005.

[RK05] V. Robbert and B. Kenneth. The gridkit distributed resource management framework. *Lecture Notes in Computer Science*, 3470/2005 :786–795, 2005.

[RM06] Emre Kiciman Richard Mortier. Autonomic network management : some pragmatic considerations. In *Proc. of the 2006 SIGCOMM workshop on Internet network management*, pages 89–93, 2006.

[RSS] A.J. Oliner M. Gupta J.E. Moreira S. Ma R. Vilalta R.K. Sahoo, I. Rish and A. Sivasubramaniam. Autonomic computing features for large-scale server management and control. In *Proc. 8th joint Conference on Artificial intelligence.*

[SBH05] N. de Palma S. Bouchenak and D. Hagimont. Autonomic administration of clustered j2ee applications. In *Proceedings of 2005, IFIP/IEEE International Workshop on Self-Managed Systems and Services*, May 2005.

[Slo94] Morris Sloman, editor. *Network and Distributed Systems Management.* Addison-Wesley, 1994.

[SM08] Gangopadhyay D. Reddy V. Gupta M. Sethi M., Anand A. An open framework for federating integrated management model of distributed it environment. In *Proc. IEEE Network Operations and Management Symposium, (NOMS'08)*, pages 803–806, 2008.

[Sta04] William Stallings, editor. *SNMP, SNMP V2, SNMP v3 and RMON 1 and 2.* Addison-Wesley, Third Edition, 2004.

[SY07] Jung-kook Hong Komatsu Y. Iwano K. Tutiya S. Katsuyama T. Yoshida H. Satoh Y., Kaneyasu Y. Applying autonomic computing with open standardized resource interface wsdm to managing multi-vendor it systems. In *Proc. IFIP/IEEE International Symposium Integrated Network Management, (IM'09)*, pages 655–66, June 2007.

[Szy97] Clements Szyperski, editor. *Component Software.* Addison-Wesley Professional, 1997.

[Tal05] Vanish Talwar. Scalable management. In *Proc. Second International Conference on Autonomic Computing,(ICAC'05).*, pages 655–66, June 2005.

[Tiv] IBM Tivoli Management System, http ://www-306.ibm.com/software/fr/tivoli/.

[VWS99] C. Malbon T. Richardson L. Sorensen V. Wade, D. Lewis and C. Stathopoulos. Component integration technologies for telecoms management systems. Technical report, TCD-CS. Technical Report, Trinity College Dublin Computer Science Department, 1999.

[wbe] WBEM, DMTF Standard, URL :http ://www.dmtf.org/standards/wbem/.

[Wen08] Chen Wenbo. An integrated management platform for network and server. In *Proc. IEEE International Symposium on IT in Medicine and Education, (ITME'08)*, pages 998–1000, 2008.

[WSD05a] Web services distributed management : Management of web services (mows1.0) part 1. Technical report, OASIS, March 2005.

[WSD05b] Web services distributed management : Management using web services (muws1.0). Technical report, OASIS, March 2005.

[wsm06] Web services for management(ws-management). Technical Report DSP0226, DMTF, April 2006.

[X.792] ITU-T Recommendation X.700. Management framework for open systems interconnection (osi) for ccitt applications. Technical report, International Telecommunication Unit, September 1992.

[YD04] P. Yalagandula and M. Dahlin. A scalable distributed information management system. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications, session Distributed information systems*, pages 379–390. ACM Press, November 2004.

[YpD92] So. Young-pa and E. Durfee. Distributed big brother. In *Proceedings of the 8th IEEE Conference on Artificial Intelligence Applications*, pages 295–301. ACM, March 1992.