Middleware minimal et auto-extensible pour le déploiement de services en environnement pervasif

Amira Ben Hamida
INRIA ARES / CITI Lab. – INSA Lyon
21 avenue Jean Capelle
69621 Villeurbanne cedex Lyon
amira.ben-hamida@insa-lyon.fr

Frédéric Le Mouël
INRIA ARES / CITI Lab. – INSA Lyon
21 avenue Jean Capelle
69621 Villeurbanne cedex Lyon
frederic.le-mouel@insa-lyon.fr

RESUME

Les intergiciels existants sont de plus en plus gourmands en ressources, et donc moins adaptés aux terminaux contraints. Dans cet article, nous proposons la réalisation d'un intergiciel minimal et auto extensible pour le déploiement de services en environnements pervasifs. Notre approche se base sur le découpage de la machine virtuelle et de l'intergiciel de développement en plusieurs services. Un ensemble minimal générique de services constitue le coeur de notre intergiciel et, selon le contexte d'exécution, des services peuvent y être ajoutés ou retirés.

Mots-clés

Intergiciels, JVM, services, environnements pervasifs

ABSTRACT

Existant middlewares are increasingly resource consuming, and so inadapted to constraint devices. In this paper, we propose the realisation of a minimal and self-extensible middleware for services deployment in pervasive environments. Our approach bases on the virtual machine and the middleware slicing into several services. A minimal set of services constitutes the middleware core and according to the execution context, services may be added or withdrawed.

Categories and Subject Descriptors

D.2 [**Software Engineering**]: Interoperability

General Terms

Design, Performance

Keywords

Middleware, JVM, services, pervasive environment

UbiMob '06, September 5-8, Paris, France

1. INTRODUCTION

Face aux capacités restreintes des dispositifs légers, les intergiciels existants ne sont pas bien adaptés; ils n'offrent pas de paradigmes de contexte pour le développement et sont trop gourmands en ressources (mémoire, CPU, etc) pour l'exécution sur ces dispositifs. Des travaux tels que [6] et [7] ont visé la minimisation des plates-formes d'exécution tels que les machines virtuelles Java. D'autres tels que Sun CDC et CLDC, la tinyVM [1], les MVV [2], fournissent des JVM certes minimales en termes de taille, mais très spécifiques à certains types de périphériques légers. D'autre part, d'autres travaux traitent de la prise en compte du contexte pervasif pour la spécialisation d'intergiciels tels que [3], [5], [8], [4], mais fournissent toujours des versions spécifiques (capteurs). Dans cet article, nous proposons la réalisation d'un intergiciel minimal et auto extensible pour le déploiement de services en environnements pervasifs. Notre approche se base sur le découpage de la machine virtuelle et de l'intergiciel de développement en plusieurs services. Nous définissons un service comme étant un objet offrant des fonctionnalités, et pouvant être publié et exploité par des tiers. Un ensemble minimal générique de services constitue le coeur de notre intergiciel et, selon le contexte d'exécution, des services peuvent y être ajoutés ou retirés.

2. PROPRIÉTÉS D'UN INTERGICIEL PERVASIF

Pour réaliser un intergiciel minimal et extensible, nous avons dans un premier temps identifié différentes propriétés que doit respecter notre intergiciel pour le déploiement en environnement pervasif :
(i) Granularité orientée-services : Notre système est orienté services en raison de la possibilité de publication des fonctionnalités qu'offre cette approche. Cette propriété est indispensable pour pouvoir effectuer de la découverte de services dans un environnement dynamique. Différentes techniques de composition et d'intégration sont également supportées par cette approche.
(ii) Minimalité : Un ensemble de services basiques, requis pour l'exécution, constitue le minima de l'intergiciel. L'ajout et le retrait de services s'effectue selon des critères tels que le nombre de services, leur taille, leur occupation en mémoire, etc.
(iii) Adaptabilité au contexte : Notre intergiciel doit être conscient du contexte dans lequel il évolue et doit pouvoir

s'adapter à ses changements, comme la découverte de nouveaux services, la prise en compte d'un profil utilisateur, les reconfigurations matérielles, etc.
(iv) Prise en compte au développement et à l'exécution : nous visons la génération de configuration d'intergiciels pendant la phase de développement et leur évolution par ajout ou retrait de services en cours d'exécution sans que cela interrompe les applications en cours.

3. INTERGICIEL MINIMAL, EXTENSIBLE PAR SERVICES

Aucun des travaux apparentés n'envisage une JVM orientée services. Nous proposons la décomposition en services de la JVM ainsi que de l'intergiciel de développement et d'exécution surjacent (cf. figure 1), où les services pourront être ajoutés ou retranchés selon le besoin. L'intergiciel minimal sera le résultat du choix de services pouvant appartenir à la JVM ou à l'intergiciel surjacent. Une des difficultés réside dans l'identification des services ainsi que les dépendances entre ces services. Après avoir évalué un graphe de dépendances sur des paquetages Java, nous avons constaté une dépendance presque totale entre tous ces éléments. Nous proposons donc de découper ce graphe en brisant certaines dépendances, d'extraire le graphe connexe minimal le plus important et d'ajouter/retirer dynamiquement les services ayant des dépendances non fortes.

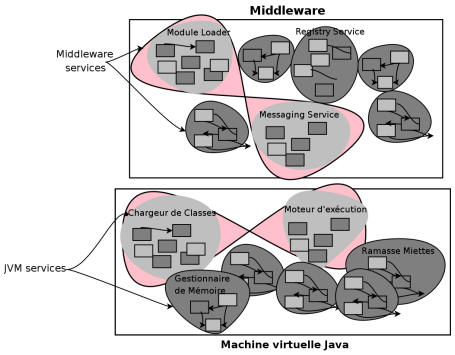

Figure 1: Décomposition de la JVM et du middleware

La figure 2 illustre une usine de fabrication d'intergiciels (middleware factory), responsable de créer divers types d'intergiciels. En effet, nous nous basons sur un modèle générique que nous instancions selon les préférences des développeurs et le contexte pervasif. La structure basique de notre intergiciel étant sous la forme d'un coeur sur lequel viendrait se superposer des services. Étant donnée que nous considérons l'approche orientée services aussi bien pour la décomposition de la JVM que pour l'intergiciel, tous les éléments sur lesquels nous allons agir sont des services. Ainsi, pour adapter notre JVM sur un PDA, nous allégeons le déploiement, en le restreignant juste aux services nécessaires. De même étant donnée qu'un téléphone mobile est plus contraint en termes de ressources, nous allégeons encore plus cette structure et déployons dynamiquement au besoin les services (auto-extensibilté).

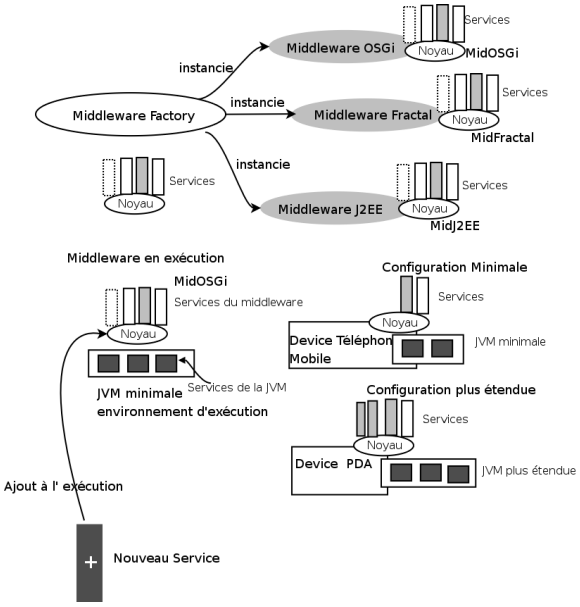

Figure 2: Middleware minimal auto extensible

4. CONCLUSION ET TRAVAUX FUTURS

Nous proposons un intergiciel minimal, auto-extensible. Il se base sur un découpage en services de la JVM et des intergiciels surjacents et les choix du sous-ensemble minimal de ces services le plus adapté au contexte pervasif (matériel, services, utilisateur, etc.). Nous nous basons sur une structure en noyau au dessus de laquelle nous superposons les services de l'intergiciel souhaité. Nous prévoyons d'étudier les différentes typologies des intergiciels existants pour dégager les éléments basiques et d'autres additionnelles de chaque infrastructure. Nous visons à travers cela l'adaptation dynamique d'intergiciels selon le contexte d'exécution par ajout et retrait de services.

5. REFERENCES

[1] Tinyvm. http://tinyvm.sourceforge.net/.
[2] B. Folliot. The virtual virtual machine project. *HPCA, Brésil*, 2000.
[3] A. Frei and G. Alonso. A Dynamic Lightweight Platform for Ad-hoc Infrastructures. *IEEE PerCom*, 2005.
[4] T. Gu, H. Pung, and D. Zhang. A middleware for building context-aware mobile services, 2004.
[5] D. Janakiram, R. Venkateswarlu, and S. Nitin. COMiS: Component Oriented Middleware for Sensor Networks. *14th IEEE (LANMAN)*, 2005.
[6] P. Levis and D. Culler. Maté: A tiny virtual machine for sensor networks. *ASPLOS XII, USA*, 2002.
[7] P. Stanley-Marbell and L. Iftode. Scylla: a smart virtual machine for mobile embedded systems. *IEEE WMCSA*, 2000.
[8] T.Liu and M.Martonosi. Impala: A middleware system for managing autonomic, parallel sensor systems. *PPoPP*, 2003.