Administration des Services dans le projet Safari

Abdelkrim Hebbar, Bruno Mongazon-Cazavet

Administration des Services dans le projet Safari

Abdelkrim Hebbar
Alcatel Recherche et Innovation
Route de Nozay
91460 Marcoussis, France
abdelkrim.hebbar@alcatel.fr

Bruno Mongazon-Cazavet
Alcatel Recherche et Innovation
Route de Nozay
91460 Marcoussis, France
bruno.mongazon-cazavet@alcatel.fr

RESUME

Le projet Safari [1], supporté par le RNRT [2], a permis l'étude, la réalisation et l'expérimentation d'une architecture de réseau intégrée pour concevoir, déployer et exploiter des services dynamiques dans un réseau IPv6 hybride ad hoc / filaire. Pour illustrer les résultats de cette étude, un premier démonstrateur matériel et logiciel a été installé dans une grande gare SNCF de Paris [3]. Il a permis d'expérimenter l'architecture matérielle et l'environnement logiciel nécessaires à l'utilisation par des voyageurs, de services dynamiques offerts par des fournisseurs de services. Dans cet article nous décrivons plus précisément comment OSGI [4] entre dans la mise en œuvre de la plate-forme logicielle de Safari aussi bien du coté fournisseur que du coté utilisateur de services. Nous verrons comment les services, empaquetés dans le format « bundle », peuvent être administrés et publiés sur les Nœuds Faiblement Mobiles (ou encore NFMs), des machines privilégiées du réseau ad hoc utilisées comme des serveurs, puis mis à disposition des utilisateurs disposant de terminaux nomades clients.

A noter que l'on peut rapprocher l'expérimentation effectuée dans Safari avec celles des projets Muse [10], Jadabs [11] et VHE Middleware [12].

Termes Généraux

Administration, Expérimentation.

Mots Clés

Administration de services, Publication, Osgi.

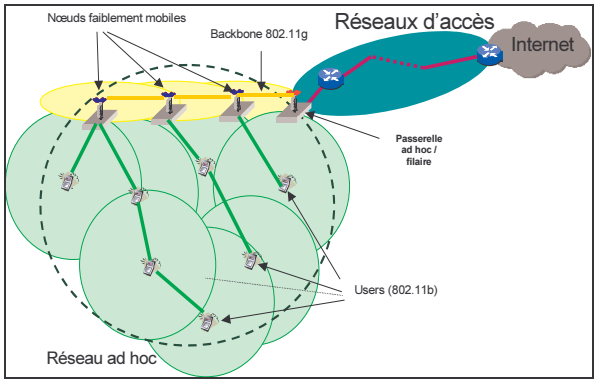

Figure 1 : Réseau Safari de la gare SNCF

1. INTRODUCTION

Les considérations matérielles résultant de l'étude Safari ont amené à la conception d'un réseau matériel hybride Wifi / Filaire indiqué dans Figure 1. Dans ce réseau plusieurs NFMs balisent géographiquement une partie de la gare. Ils sont utilisés comme relais réseau (ou serveur d'applications) mais aussi comme dépôt de services, éventuellement géolocalisés voire contextuels [13], par les fournisseurs de services. Un utilisateur possédant un terminal nomade peut alors se connecter au réseau Safari, découvrir puis exécuter les services se trouvant à proximité sur le NFM le plus proche, ou tous ceux qui répondent à un certain critère de sélection du service recherché comme par exemple une partie du nom, un mot clef de la description, tel fournisseur etc.

Le présent article ne décrit pas les considérations et résultats liés aux spécificités des réseaux sans fil. On se focalisera plutôt sur les techniques mises en œuvre pour la gestion des services. Pour une présentation d'ensemble du projet Safari, on consultera avantageusement [5].

La section 3 décrit plus en détails les briques logicielles basées sur la technologie OSGI qui permettent les actions suivantes :

La section 2 présente un bref rappel des configurations matérielles et logicielles utilisées pour les serveurs de dépôt de services et des terminaux nomades.

- Administrer et publier les services sur les NFMs.
- Découvrir, télécharger et exécuter les services à partir des terminaux nomades.

2. CONFIGURATION MATERIELLE ET LOGICIELLE

2.1 Nœud Faiblement Mobile (NFM)

Le NFM est un PC dont le système d'exploitation est basé sur une distribution Linux Suse (noyau 2.4.26) et qui est dépourvu de ses périphériques habituels : écran, clavier et souris. Il est équipé d'une carte Wifi (de type 802.11b) qui permet et délimite l'accès radio au NFM (bulle Wifi) par les terminaux nomades et d'une autre carte Wifi (de type 802.11g) permettant la connexion des NFMs entre-eux. Pour se connecter au réseau filaire, un NFM privilégié dispose d'une connexion Ethernet.

Le logiciel Safari propre au NFM est écrit en Java et utilise des paquetages Java standard ou issus du domaine public. Le JDK 1.4 de Sun Microsystems est utilisé en tant qu'environnement de développement et d'exécution. Les paquetages utilisés proviennent de Oscar 1.0.1 [6] (implémentation de OSGI) et de JXTA 2.1.1 [7].

On notera que l'ensemble du logiciel utilisé dans le NFM est de type Open-Source. Ceci présente un avantage important en terme de coût et d'efficacité quand à la qualité, l'utilisabilité et la maintenabilité du logiciel utilisé.

2.2 Terminal nomade

Le terminal nomade est un HP iPAQ H5550 dont le système d'exploitation est basé sur la distribution Linux Familiar 0.7.2 (noyau 2.4.19). Il est équipé d'une carte interne Wifi permettant l'accès à une bulle NFM.

Le logiciel Safari propre au terminal nomade est écrit principalement en Java et utilise des paquetages Java adaptés au profil J2ME. La machine virtuelle Java CVM 1.0 de Sun Microsystems est utilisée en tant qu'environnement d'exécution. Elle est basée sur les spécification de CDC 1.0 (JSR 36) [8] (niveau JRE 1.3). Les paquetages Oscar 1.0.1 et JXME 1.0 [9] sont utilisés.

2.3 Réseau

Les différentes bulles Wifi NFM possèdent leur propre adressage sous-réseau et sont interconnectées entre-elles pas le biais du sous-réseau Wifi 802.11g.

Les différents NFMs peuvent ainsi se communiquer les services disponibles dans leur dépôt respectif et en permettre l'accès depuis n'importe quel terminal nomade. Nous verrons plus loin comment le logiciel Safari permet une publicité globale des services qui n'aurait pas été possible par la simple utilisation des paquetages OSGI, JXTA [7] et JXME [9].

La plate-forme Safari est conçue pour fonctionner indifféremment sur les réseaux Ipv4 et Ipv6. Notre expérimentation s'est principalement effectuée sur des réseaux Ipv4 (dû aux limitations de la CVM : les communications Ipv6 ne sont pas prises en charge en JRE 1.3) mais a aussi pu être menée en Ipv6 lorsque les terminaux nomades étaient des PC portables avec cartes Wifi et environnement d'exécution Java basé sur JRE 1.4.

3. OSGI DANS LA PLATE-FORME SAFARI

La plate-forme logicielle Safari se décline en deux parties : une partie dite « serveur » s'exécutant sur chaque NFM et une partie dite « cliente » s'exécutant sur chaque terminal nomade.

Si ces deux parties contiennent des éléments logiciels distinctifs, elles partagent une approche et une architecture communes dans un souci de cohérence, d'homogénéité et de factorisation logicielle.

(Figure 2) montre la décomposition de la plate-forme client-serveur de Safari.

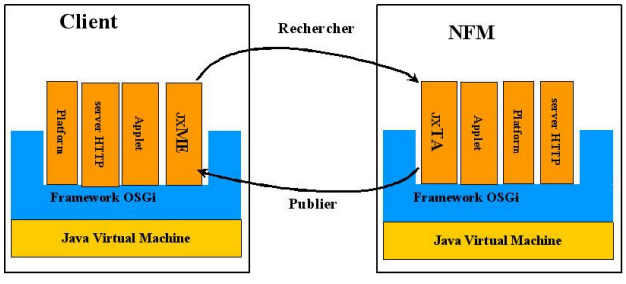

Figure 2 : Plate-forme de Service Safari

3.1 Couplage OSGI-JXTA/JXME

On entend par « service » dans Safari, toute application offerte aux utilisateurs nomades. La notion de « service » est entièrement calquée sur celle de « bundle » dans OSGI. Plus généralement, l'ensemble des principes et mécanismes définis par OSGI en matière de gestion de services a été adopté dans le cadre de Safari. Ceci concerne en particulier :

- la gestion (ajout, retrait, mise à jour) des bundles,
- le téléchargement et l'exécution des bundles,
- la gestion des dépendances entre bundles,
- la construction et le packaging des bundles.

Le choix d'OSGI a été motivé, d'une part, par son couplage naturel avec le langage Java et, d'autre part, par les efforts de définition, standardisation et diffusion qui lui sont attachés.

Si l'approche OSGI permet de répondre aux besoins de Safari dans le domaine général de la gestion de services logiciels, elle n'apporte pas nativement de solution satisfaisante quand à la publicité et la recherche de services dans un environnement nomade ad hoc. Pour satisfaire à ce besoin, nous avons choisi de coupler à OSGI un protocole généraliste de communication peer-to-peer. Notre choix s'est portée sur la technologie JXTA de Sun Microsystems. Cette technologie offre à la fois une grande souplesse d'utilisation et propose deux profils d'utilisation, JXTA et JXME, qui correspondent aux deux profils définis par Safari, NFM et terminal nomade.

JXTA et OSGI présentent néanmoins quelques redondances notamment en ce qui concerne la notion de service (module JXTA). Par contre JXTA offre des capacités non disponibles dans OSGI telles que :

- l'identification et la découverte des peers sur un réseau,
- la publicité (traduction de la notion « advertisement » en langue anglaise) et la recherche de ressources sur un réseau.

Notre approche de couplage a donc retenu la partie gestion de service fournie par OSGI, la partie publication et recherche de service fournie par JXTA/JXME.

3.2 Les Services dans Safari

Les services manipulés dans Safari sont basés sur la définition des « bundle » OSGI. Chaque service ou application est donc un programme java encapsulé dans un « .jar » et contient un fichier « manifest » de méta-datas définissant les attributs du bundle/service. Certains attributs obligatoires sont utilisés directement par les mécanismes OSGI (le nom, la version, l'adresse de chargement, la classe d'activation du service, les dépendances en terme de bibliothèque ou services etc.), d'autres peuvent être utilisés à titre informatif (description etc.).
Les principaux attributs contenus dans le « manifest » OSGI servent à construire la structure de l'objet manipulé (ou encore « *advertisement* ») dans le mécanisme de publication/recherche des services pris en charge par l'infrastructure JXTA/JXME.
Dès qu'un service Safari est mis à disposition des terminaux nomades, un « *advertisement* » est publié sur le réseau Safari en rendant visible, parmi les plus significatifs, les champs suivants :

- **Name**: nom du service
- **Version**: version du service
- **URL**: adresse de chargement du « bundle » associé
- **Dependencies**: les dépendances au sens OSGI

- **Type**: pour l'instant « OSGI » mais peut-être étendu avec par exemple « WebService » etc.
- **Category**: mot-clef pour catégoriser le service Safari
- **Description**: texte libre décrivant les fonctionnalités du service
- **Provider**: identification du fournisseur

Le couplage OSGI/JXTA permet de concevoir une plate-forme dans laquelle les services ont la morphologie « bundle », s'administrent à l'aide de la couche OSGI, se déclarent et se recherchent au niveau utilisateur nomade à l'aide des mécanismes JXTA. Ainsi, la plate-forme permet de :

- Installer un « service/ bundle » et entraîner la publication de l' « advertisement » associé, localement sur un seul nœud JXTA et/ou sur tous les NFMs du réseau Safari. Chacun des champs de l' « advertisement » publié sera indexé pour servir de critère de recherche des services aux utilisateur nomades.
- Désinstaller un « service/bundle » et entraîner la suppression de l' « advertisement » associé sur tous les NFMs du réseau Safari.
- Rechercher un «service/bundle » selon un critère donné.
- Télécharger un «service/bundle» et l'installer dans le dépôt local de bundles.
- Exécuter un «service/bundle» depuis le dépôt local de bundles.
- Arrêter un «service/bundle».
- Supprimer un «service/bundle» du dépôt local de bundles.

Les actions sur les services décrites ci-dessus induisent un modèle de plate-forme «client-serveur» qui sera détaillé dans les sections suivantes.

Le client et le serveur sont bâtis sur un noyau Oscar étendu par des bundles spécialisés. Ainsi, le serveur comporte un service spécifique permettant de l'administrer à distance à partir d'un navigateur Web. Il comporte aussi tous les mécanismes de publication de services JXTA.

3.3 Le serveur de dépôt de services

On entend par « serveur de dépôt de services », la partie serveur de la plate-forme Safari. Il est démarré sur toute machine fixe (passerelle ad-hoc/filaire) ou faiblement mobile (NFM) dotée de suffisamment de capacités pour permettre de faire tourner l'infrastructure JXTA dans l'environnement OSGI/Oscar sur une machine virtuelle java J2SE. Basé sur Oscar, mais sans en avoir utilisé toutes les fonctionnalités disponibles, il peut être lancé indifféremment en mode graphique (Figure 3) ou textuel. Il permet de remplir les fonctions suivantes :

- Administration locale des services: installation à partir d'une URL, désinstallation, mise à jour, activation, arrêt des bundles/services, visualisation de l'état courant des bundles.
- Création d'un nœud JXTA et mise en place de l'infrastructure de communication JXTA/JXME.
- Infrastructure de gestion des publications de services au sens JXTA (publication locale, publication sur nœuds distants, nettoyage des publications obsolètes etc.) à partir de l'installation/désinstallation des bundles OSGI.
- Service d'administration à distance via un navigateur Web (Figure 4) par la création d'un servlet dans un serveur Web embarqué type Jetty (recherche, installation, désinstallation, activation, désactivation etc.).

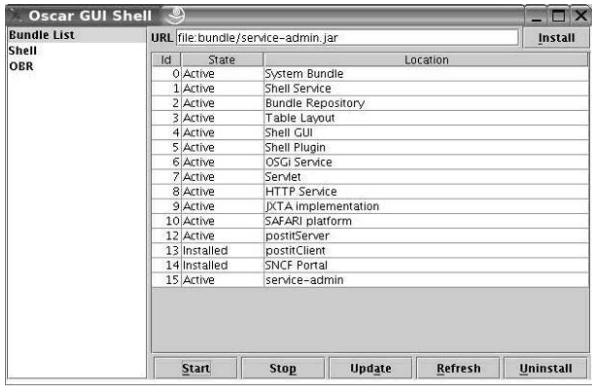

Figure 3 : Serveur Oscar des services Safari

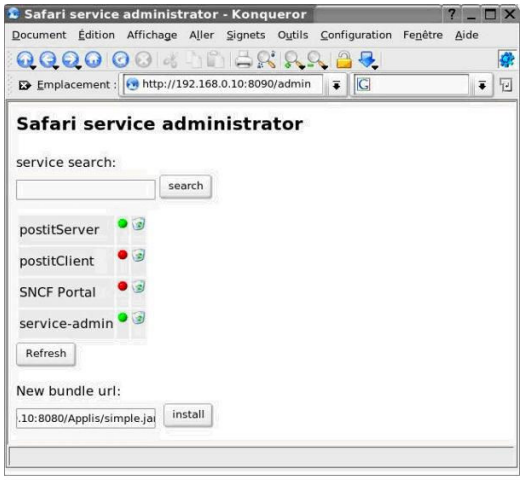

Figure 4 : Administration des services Safari via Web

3.4 Le client Safari

On rassemble sous le terme « client Safari » les éléments logiciels qui équipent le terminal nomade, non compris le système d'exploitation, la CVM et les paquetages OSGI et JXME. Il est à noter que le paquetage JXME est utilisé dans une configuration « proxied » [9].

Le client Safari est entièrement réalisé sous la forme de « bundles » OSGI spécialisés et utilise également des « bundles » standards d'OSCAR. Les « bundles » standards utilisés sont :

- **Bundlerepository**: permet la gestion des bundles pré-installés ou téléchargés sur le terminal nomade.
- **http**: apporte le support du protocole *http* au terminal nomade.
- **osgi-service**: permet la gestion d'entités de service au sein d'OSGI.
- **Servlet**: apporte le support de *servlets* locales au terminal nomade.

Les « bundles » spécialisés définis dans le cadre du client Safari sont les suivants :

- **platform**: apporte la gestion des services (recherche, téléchargement…) de façon générique.
- **jxme-impl** : fournit l'implémentation pour JXME de la plate-forme de services définie par le bundle « platform ». Il

encapsule en particulier, la gestion sous-jacente d'un peer JXME et des communications avec le « proxy » JXTA.

- **network-listener** : détecte les entrées et sorties des bulles Wifi et initialise la plate-forme de services en conséquence. Cette détection est basée sur l'utilisation d'un script shell qui « ping » régulièrement le réseau en mode broadcast et s'attache au NFM dont la réponse est la plus rapide[1].
- **jxme-launcher** : gère la gestion des services sous JXME à la demande du network-listener.
- **bundlestarterservlet** : crée un servlet qui permet de démarrer sur le terminal nomade un bundle particulier en lui fournissant des paramètres. Ce bundle gère de manière transparente la recherche et le téléchargement du bundle à démarrer si celui-ci n'est pas encore connu et/ou disponible.
- **Identification** : point d'entrée du client Safari, il permet à l'utilisateur de s'identifier (les données d'identification sont stockées dans le bundle) et d'accéder aux services de recherche et d'exécution de services (Figure 5).
- **Popup** : permet l'affichage d'un texte dans un popup. Ce bundle peut être démarré à distance sur le terminal nomade par l'intermédiaire du bundle « bundlestarterservlet ».

Outre ces bundles spécialisés, des bundles « applicatifs » offrant des services aux utilisateurs finaux ont été expérimentés dans le cadre de Safari. Parmi les services les plus significatifs, on trouve le « postit virtuel », qui permet à des terminaux nomades de laisser des postits et d'en notifier les destinataires sur les différents NFMs ainsi que portail SNCF personnalisé destiné aux voyageurs qui circulent dans la gare.

De manière générale, l'utilisateur du client Safari accède aux services issus de sa requête de recherche. Les services disponibles sont recherchés sur le réseau JXTA puis présentés à l'utilisateur (Figure 5). Lorsque l'utilisateur clique sur un des services disponibles (Advertised Bundles), celui-ci est téléchargé et installé localement (Local Bundles). Il peut ensuite être activé, désactivé ou désinstallé à l'aide des icônes qui lui sont associées (Figure 5).
Il est à noter que le client Safari a également été expérimenté sur plate-forme Pocket PC 2003 avec la JVM Creme [14].

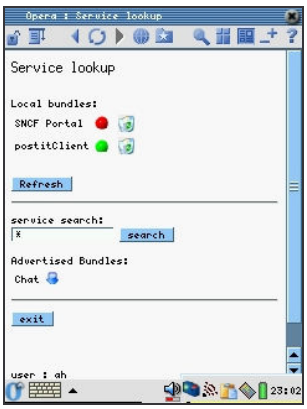

Figure 5 : Interface du Client Safari

[1] Cette heuristique prend en compte la disponibilité globale du NFM et non pas uniquement sa proximité géographique

4. CONCLUSION

Dans le cadre de Safari, la technologie OSGI/Oscar a largement facilité la réalisation d'une véritable plate-forme d'administration de services. De plus, les services étant vus comme des « bundles », cela permet d'enrichir facilement le nombre de services mis à la disposition d'utilisateurs nomades avec tous les « bundles » déjà disponibles sur la toile. Le couplage à la technologie JXTA a contribué à la distribution de ces services dans un environnement IP ad hoc.

5. REMERCIEMENTS

Nous tenons à remercier tous les partenaires du projet Safari qui ont contribué à cette expérimentation et plus particulièrement la SNCF, en la personne de David Sanz, pour nous avoir permis d'effectuer des tests en gare.

6. REFERENCES

[1] SAFARI Annexe technique version finale, février 2003.

[2] Réseau National De Recherche En Télécommunications http://www.telecom.gouv.fr/rnrt/rnrt/projets/res_02_04.htm

[3] SAFARI Sous-Projet 5 : Cahiers fonctionnels du démonstrateur en Gare. L5.03G. Edition 1, mars 2004.

[4] Open Service Gateway initiative: **OSGI,** specifications. http://www.osgi.org, 2002.

[5] Reynaud, L : SAFARI, Services Ad Hoc/Filaire : Architecture de Réseau Intégrée. JS Radiocommunications Haut Débit. ENSTA Paris, 2005

[6] Hall, R.S.: Oscar: Object service container architecture. http://oscar.objectweb.org, 2004

[7] JXTA, http://www.jxta.org

[8] Sun Microsystem. JSR-000036 Connected Device Configuration 1.0b

[9] JXTA-J2ME 1.0 (Proxy Implementation), http://jxme.jxta.org

[10] S. Frenot, Y. Royon : Component deployment using a peer-to-peer overlay. 3rd International Working Conference on Component Deployment, Grenoble, 28 Oct – 02 Nov , 2005.

[11] Jadabs, http://jadabs.berlios.de

[12] Peer to Peer Networks for Virtual Home Environments, http://jerry.c-lab.de/vhelab/r_p2p4vhe.html

[13] J. Coutaz, J.L. Crowley, S. Dobson, D. Garlan : Context is Key. CACM, Vol. 48, N° 3, March 2005.

[14] NSIcom CrE-ME JVM for Windows CE, http://www.nsicom.com