Une Approche Coopérative de l'Adaptabilité des Applications Mobiles basées sur MobileJMS

Mejdi Kaddour, Laurent Pautet
Ecole Nationale Supérieure des Télécommunications
Département Informatique et Réseaux
F-75634 Paris CEDEX 13, France
{kaddour,pautet}@inf.enst.fr

RESUME
L'adaptabilité des applications à leur contexte d'exécution s'est imposée comme un critère incontournable dans la conception de systèmes mobiles performants. Dans ce cadre, nous explorons dans ce présent papier les différents facteurs à prendre en compte pour atteindre cet objectif. Nous proposons une approche de l'adaptabilité basée sur la coopération entre les applications et l'intergiciel MobileJMS que nous avons développé précédemment. D'un côté, les applications spécifient leurs contraintes et besoins en terme de qualité de service. De l'autre côté, l'intergiciel doit veiller à ce que ces besoins et contraintes soient respectés. Cette équation est complétée par l'adéquation entre la qualité du service rendu et le contexte dans lequel évoluent les applications.

Mots-clefs
Intergiciel, MOM, JMS, adaptabilité, politique, mobilité, application

ABSTRACT
Application adaptability to context changes has emerged as an absolute necessary criteria in the design of successful mobile systems. In this context, we investigate in this paper various factors to be taken into account to reach this goal. We propose an approach based on the cooperation between applications and the MobileJMS middleware which we developed previously. First, applications specify their requirements and constraints in terms of quality of service. Second, middleware provides primitives which fulfill these requirements and constraints. This equation is completed by the correlation between the quality of the service and the execution context of the applications.

Categories and Subject Descriptors
C.2.4 [**Computer-Communication Networks**]: Distributed Systems – Distributed Applications.

Mobilité & Ubiquité 2004, June 1-3, 2004, Nice, France.

General Terms
Design, Experimentation, Languages

Keywords
Middleware, MOM, JMS, adaptability, policy, mobility, application

1. INTRODUCTION
Les dernières années ont connu des avancées majeures dans le domaine des réseaux sans fil avec l'apparition de plusieurs standards, comme le 802.11x, Bluetooth ou les réseaux 3G. Ces standards arrivent actuellement à maturité et proposent de couvrir de larges types d'usages, aussi bien personnels que professionnels. Par ailleurs, les terminaux mobiles disponibles sur le marché deviennent de plus en plus puissants et dotés de capacités de communication à travers différents types de connexions sans fil.

Bien que le potentiel de ces technologies soit important, le développement des applications mobiles sera, sans doute pour un bon moment au moins, contraint par les limites physiques. En effet, les communications sans fil restent caractérisées par des fluctuations de la bande passante et des déconnexions fréquentes et impromptues. De plus, l'autonomie des terminaux mobiles dépend des limites des batteries actuelles qui doivent aussi tenir compte d'autres facteurs comme la légèreté ou l'ergonomie.

Dans ce contexte, beaucoup de travaux de recherche ont été menés sur le conception d'intergiciels capables de prendre en charge les caractéristiques de l'environnement mobile et de faciliter le développement d'applications mobiles [10][5]. L'une des conclusions importantes de ces travaux est la nécessité de rompre avec le principe de transparence, qui a longtemps guidé le développement des intergiciels dans les environnements fixes. Afin de fournir une qualité de service acceptable, les applications mobiles doivent être capables de s'adapter aux changements qui peuvent se produire dans leur contexte d'exécution, comme les variations de la bande passante, du niveau des ressources sur le terminal ou des services distants accessibles. Cette faculté d'adaptation dépend également de la capacité des applications à « percevoir » ce contexte d'exécution.

Par ailleurs, les intergiciels classiques, basés pour la plupart sur le modèle client/serveur, ont montré leurs limites dans les

environnements mobiles [10]. En effet, ces intergiciels supposent, la plupart du temps, la disponibilité simultanée de tous les participants à la communication, l'utilisation de connexions stables et bénéficiant d'une large bande passante, et la disponibilité de ressources confortables sur les machines des utilisateurs. Ces conditions sont loin d'être réunies dans ce type d'environnements.

Nous avons développé précédemment MobileJMS [8][9], un intergiciel orienté message destiné aux environnements mobiles et basé sur la spécification *Java Message Service* (JMS) [6]. Nous pensons que, compte tenu de ses propriétés d'asynchronisme et de faible couplage entre les composants communicants, ce type d'intergiciel est mieux adapté à la communication dans les environnements mobiles. MobileJMS prend en charge la mobilité à travers un module de communication flexible et à travers la gestion de quelques évènements particuliers comme la déconnexion réseau.

La contribution de ce papier est la proposition et l'élaboration d'un modèle d'adaptation basé sur la coopération entre les applications et MobileJMS. La politique d'adaptation constitue le concept essentiel de ce modèle. Elle définit une relation entre les besoins d'une application et une configuration donnée de l'environnement d'exécution. A chaque application donnée est associé un ensemble de politiques. Une des fonctions introduites dans ce modèle est la possibilité de désigner à tout moment la politique la plus adaptée au contexte d'exécution.

Le reste du papier est organisé de la manière suivante. Nous rappelons dans la section 2 quelques définitions concernant les intergiciels orientés message et JMS. La section 3 présente brièvement MobileJMS. La section 4 décrit les points clés du modèle d'adaptation. Le gestionnaire d'adaptation est détaillé dans la section 5 avec une description des modules qui le composent et leurs différentes interactions. Nous illustrons dans la section 6 le comportement de la plate-forme à l'aide d'une application fournissant un service météorologique. La section 7 présente quelques autres travaux effectués sur ce sujet. Enfin, la section 8 conclue ce papier en mettant l'accent sur les perspectives de notre travail.

2. MOM et JAVA MESSAGE SERVICE

Les intergiciels orientés message (MOMs) trouvent leurs origines dans les modèles « publication/abonnement » et les systèmes de notifications d'évènements [1]. Leur mode de communication est basé sur l'échange asynchrone et fiable de messages. Les MOMs substituent au modèle d'interaction classique (client/serveur) un modèle proche de celui du pair à pair, où chaque entité communicante peut envoyer et recevoir des messages des autres entités.

Actuellement, les MOMs sont généralement utilisés pour l'intégration de différents types d'applications et de données dans les systèmes d'information d'entreprise. Le faible couplage entre les entités communicantes dans un MOM, en terme de disponibilité et de logique fonctionnelle, permet d'incorporer à un système d'information global les applications et les données patrimoniales de l'entreprise, et ouvre la voie à de nouveaux types d'accès comme le Web. L'échange des messages peut se faire dans un contexte interne (pour le EAI) ou externe (pour le B2B ou le B2C) à l'entreprise. Les MOMs sont également dotés de services sophistiqués, comme le routage de messages ou le transformation du contenu, ce qui leur permet de s'adapter à une population hétérogène et variable d'applications et d'utilisateurs.

Nous pensons que ces propriétés permettent aux MOMs de répondre également à certaines problématiques des environnements mobiles. En effet, dans ce mode de communication, des entités différentes peuvent communiquer sans nécessairement se référencer mutuellement, ni se connecter au même moment. Ce modèle de dissémination de l'information se prête assez bien aux environnements caractérisés par la mobilité des entités (changement de l'adresse réseau et de la localisation), par l'interaction d'entités variables et anonymes, et par des connexions réseau intermittentes, comme c'est le cas pour les environnements mobiles en général.

La diversité des MOMs actuels a fait surgir la nécessité d'un standard unifié capable de répondre au besoin d'interopérabilité entre les MOMs eux-mêmes. La spécification JMS propose dans ce sens un modèle uniforme pour les applications communiquant par messages et développées en java. Cette spécification représente un « dénominateur » commun des MOMs actuels.

Un système JMS est organisé autour de *clients* et d'un *fournisseur (provider)*. Le fournisseur reçoit les messages des clients émetteurs et se charge de les transmettre aux clients destinataires. Il est, de ce fait, le nœud où tous les messages doivent transiter. Par ailleurs, JMS permet aux applications de communiquer à travers deux modèles de passage de messages:

1. *Point à point*: chaque message est adressé à une *file (queue)* au niveau du fournisseur. Il n'est adressé qu'à un seul destinataire.
2. *Publication/abonnement*: dans ce modèle, un producteur adresse un message à un *sujet* (*topic*) au niveau du fournisseur. Ce message peut avoir plusieurs consommateurs qui sont anonymes pour le producteur.

Deux concepts sont également importants dans JMS: la *connexion* et la *session*. La connexion encapsule un lien virtuel entre un client et un fournisseur JMS. Elle peut représenter, par exemple, une connexion TCP/IP. Les connexions sont utilisées pour créer une ou plusieurs sessions. La session est un contexte séquentiel pour l'envoi et la réception de messages. Les sessions sont utilisées pour créer les producteurs, les consommateurs de messages et les messages eux-mêmes.

3. MOBILEJMS

MobileJMS est basé sur une implémentation JMS existante (OpenJMS [11]) et propose une prise en charge spécifique de la mobilité. Le principal objectif visé est l'adaptabilité et la configurabilité de la plate-forme afin de mieux prendre en charge les contraintes et la variabilité de l'environnement mobile (connexion intermittente, bande passante variable, ressources limitées des terminaux…). Les applications basées sur MobileJMS sont implémentées en tant que clients JMS.

MobileJMS communique à travers un module adaptable et flexible. Ce module prend la forme d'un graphe de protocoles, construit sur les interfaces du squelette du système de communication *Jonathan* [4]. Les messages échangés entre les applications traversent successivement les couches protocolaires qui forment ce graphe, et cela dans les deux sens. La figure 1 montre un exemple d'un graphe de protocoles.

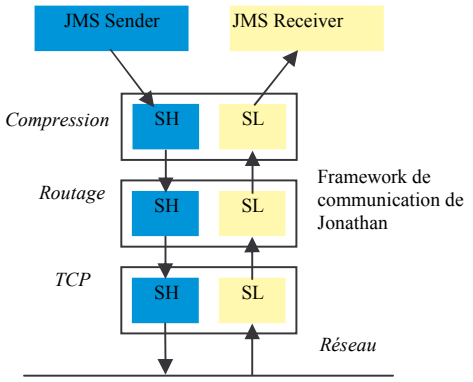

Figure 1. Exemple d'un graphe de protocoles d'un client

Nous avons enrichis les interfaces de Jonathan afin de permettre l'insertion ou la suppression de couches protocolaires durant l'exécution. Ce comportement permet, par exemple, d'insérer dans le graphe une couche de compression des messages dans le cas où la bande passante diminue sous un seuil donné.

Nous ne décrivons pas dans ce papier tous les détails relatifs au fonctionnement l'intergiciel, nous résumons cependant ci-dessous les aspects essentiels.

3.1 Services à valeur ajoutée

Ces services ne font pas partie de la logique JMS elle-même mais visent à optimiser les ressources nécessaires à la communication ou à améliorer le traitement de certains évènements, comme la déconnexion. Ils sont implémentés sous la forme de couches protocolaires et peuvent donc être insérés ou rajoutés, dans n'importe quel ordre, pendant l'exécution. Nous avons implémenté les trois services décris ci-dessous.

3.1.1 Le service stockage et ré-émission (S&F)

Ce service permet aux applications de poursuivre leurs traitements même dans le cas d'une déconnexion ou d'une faible connexion. Les messages sont temporairement stockés dans un cache local au niveau du client JMS avant d'être envoyés lorsque la connexion est de nouveau disponible. Durant les périodes de déconnexion, les messages stockés dans le cache sont classés selon les critères suivants: la priorité JMS, message bloquant ou non, la persistance, la taille et le moment de leur réception par le cache. Un message bloquant doit être envoyé au fournisseur avant que l'application puisse poursuivre son traitement, il est donc plus prioritaire qu'un message non bloquant. C'est le cas d'une requête qui doit être suivie par une réponse. L'ordre d'application de ces critères peut être modifié dynamiquement (voir [8] pour plus de détails).

3.1.2 Le service de compression

Ce service permet de préserver la bande passante en réduisant la taille des messages échangés entre un client et le fournisseur JMS. La compression permet également de réduire l'espace mémoire utilisé si les messages sont stockés dans le cache du service stockage et ré-émission.

Nous avons implémenté pour l'instant deux algorithmes de compression: *gzip* [3], avec un seul taux de compression, et *ZIP* qui permet d'utiliser des taux de compression allant de 1 à 9, qui correspondent respectivement à une compression faible et à une compression très forte des données.

3.1.3 Le service de fragmentation

Ce service permet d'envoyer les messages de grande taille en plusieurs fragments. Il permet en cas de perte de paquets ou de déconnexion de renvoyer uniquement les fragments perdus ou en attente de transmission. Du côté du destinataire, le message initial est reconstitué après la réception de tous les fragments, il est délivré ensuite à la couche supérieure. Ce service prend en charge également les fragments dupliqués ou déséquencés.

3.2 Négociation dynamique

Afin de pouvoir interpréter les messages correctement (compressés ou non, fragmentés ou non,…), le client et le fournisseur JMS doivent entreprendre une phase de négociation avant l'établissement de la connexion JMS. Cette phase est nécessaire pour déterminer le graphe de protocoles qui doit être utilisé de chaque côté. Lors de cette phase, le client envoie une proposition au fournisseur sous la forme d'une représentation symbolique du graphe de protocoles qu'il souhaite utiliser. A la réception de cette proposition, le fournisseur instancie un graphe similaire dans le cas où il décide d'accepter cette proposition. Le fournisseur prend cette décision suivant la disponibilité des protocoles demandés par le client et suivant certains critères de performance également. Ce dernier cas correspond, par exemple, à la saturation du fournisseur par un nombre élevé de clients utilisant la compression.

Une nouvelle phase de négociation peut également s'effectuer durant la durée de vie de la connexion JMS. C'est le cas où le client ou le fournisseur souhaitent modifier le graphe de protocoles en raison de nouvelles conditions dans l'environnement.

3.3 Reconnexion transparente

Dans un système JMS classique, la déconnexion réseau d'un client avec le fournisseur entraîne systématiquement la fermeture de la connexion JMS et la perte des messages non encore transmis par le client. Ce comportement est, à notre avis, inadéquat dans les environnements mobiles qui se caractérisent par des déconnexions plutôt fréquentes. Le mécanisme de reconnexion transparente permet d'occulter aux couches supérieures, la déconnexion qui se produit dans la couche de transport réseau (TCP, par exemple) et de reprendre aussitôt après la reconnexion les tâches restées bloquées. Le changement de l'adresse réseau du client n'influe par sur le déroulement de la procédure; le fournisseur identifie un client par un identificateur unique et globale défini par JMS.

Au niveau du fournisseur, un délai de validité est affecté aux objets JMS (connexions, sessions,…) associés à un client qui s'est déconnecté. Si ce client ne reconnecte pas avant l'expiration de ce délai, ces objets sont définitivement fermés.

4. MODELE D'ADAPTATION

L'adaptabilité d'un système mobile se traduit par sa capacité à réagir aux changements impromptus et fréquents qui peuvent se produire dans le contexte d'exécution de l'utilisateur (bande passante variable, changement de localisation géographique, ressources limitées et variables sur les terminaux mobiles, services disponibles dans le voisinage...). Certains mécanismes doivent être mis en œuvre pour maintenir la même qualité de service lors du changement du contexte d'exécution ou pour continuer d'assurer le service même éventuellement avec une perte de qualité. Par exemple, la diminution de la bande passante réseau, dont bénéficie un utilisateur, peut entraîner une forte dégradation de la qualité de service, si ce service dépend de paramètres temporels. La solution dans ce cas consiste à compresser les données où à les envoyer dans un mode dégradé (par exemple, perte de la couleur de la vidéo transmise).

Nous avons conçu un modèle d'adaptation qui met en œuvre une coopération entre les applications et l'intergiciel. Les applications spécifient, de leur côté, un ensemble de besoins et de contraintes qui leur assurent un certain niveau de qualité service. L'intergiciel, de son coté, veille à ce que ces contraintes soient respectées et envoie une notification à l'application dans le cas où une violation de ces contraintes, se produit à un moment donné. Notons que ce processus d'adaptation s'effectue uniquement au niveau du terminal de l'utilisateur mobile. Afin de mieux appréhender le problème de l'adaptabilité, nous avons défini trois classes de paramètres:

a) **Les besoins de l'application:** cette classe définit les paramètres de qualité de service de l'application (bande passante, latence maximum tolérée…) et les paramètres qui sont liés à la logique fonctionnelle de l'application (adresse du serveur distant, type des messages échangés…).
b) **Le contexte d'exécution:** cette classe représente les paramètres du contexte d'exécution de l'application (type de connexion réseau, bande passante disponible, taux de perte de paquets, niveau de batterie, localisation géographique...).
c) **Le comportement de l'intergiciel :** cette classe représente la configuration qu'adopte l'intergiciel à un moment donné pour répondre aux besoins des applications et pour faire face aux conditions du contexte d'exécution (utilisation de la compression des messages, de la fragmentation ou du service de stockage et ré-émission…).

La politique d'adaptation constitue l'élément central dans ce modèle d'adaptation. Elle consiste en un contrat qui associe un ensemble de besoins d'une application à une certaine configuration du contexte d'exécution. Les politiques d'adaptation sont définies dans le langage XML, la figure 2 représente un exemple d'une politique.

Cette politique spécifie que les messages échangés sont de type image, lorsque la bande passante est supérieure à 200 kbits/s, et que le serveur à contacter est « meteo_fr », lorsque l'utilisateur se trouve dans la zone géographique délimitée par les coordonnées x et y.

Notons que la politique d'adaptation ne spécifie pas le comportement que doit adopter l'intergiciel pour répondre aux besoins, cette décision doit se prendre dynamiquement au moment de l'exécution. Les différents contextes d'exécution, dans lesquels peut évoluer une application, sont représentés par un ensemble de politiques d'adaptation. A chaque contexte d'exécution correspond l'une de ces politiques. Le *gestionnaire d'adaptation* est chargé d'interpréter ces différentes politiques, de choisir la

```
<policyE policyName="HighBandwidth_France">
  <constraints>
    <constraint resName="bandwidth" operator="greaterThan"
                value="200"/>
    <constraint resName="maximum_delay" operator="lessThan"
                value="5"/>
    <constraint resName="coord_x" operator="greaterThan"
                value="10"
    <constraint resName="coord_x" operator="lessThan"
                value="100"
    <constraint resName="coord_y" operator="greaterThan"
                value="10"
    <constraint resName="coord_y" operator="lessThan"
                value="100"
  </constraints>
  <requirements>
    <requirement parName="server" newValue="meteo_fr" />
    <requirement parName="location" newValue="France" />
    <requirement parName="dataType" newValue="image"/>
  </requirements>
</policyE>
```

Figure 2. Exemple d'une politique d'adaptation

politique la plus appropriée au contexte courant et de la mettre en œuvre en interagissant avec l'intergiciel MobileJMS. Nous allons détailler dans la section suivante les mécanismes utilisés pour effectuer ces tâches.

5. LE GESTIONNAIRE D'ADAPTATION

Le gestionnaire d'adaptation est composé des entités suivantes : l'analyseur XML, Le gestionnaire du contexte local et le coordinateur. Nous décrivons ci-dessous le rôle de chaque entité.

5.1 L'arbre de spécialisation

Les politiques d'adaptation associées à une application donnée sont représentées en mémoire sous la forme d'un arbre. Cette structure facilite et optimise la recherche de la meilleure politique. Cet arbre est construit selon le principe de spécialisation; une politique donnée est un descendant d'une autre politique, si elle définit au moins les même besoins et contraintes. Cela implique que si la politique contenue dans le nœud courant, n'est pas adaptée au contexte courant, tous ses descendants ne le sont pas également. La figure 4 représente un exemple d'un arbre de spécialisation.

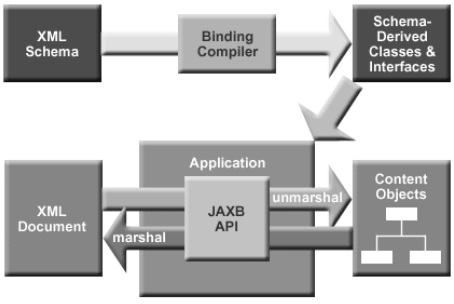

Figure 3. Analyse JAXB (src: java.sun.com/xml/jaxb)

Le module qui gère l'arbre de spécialisation est l'analyseur XML. Il transforme un ensemble de politiques d'adaptation, décrites en XML, en objets java. Ce module est basé sur l'architecture

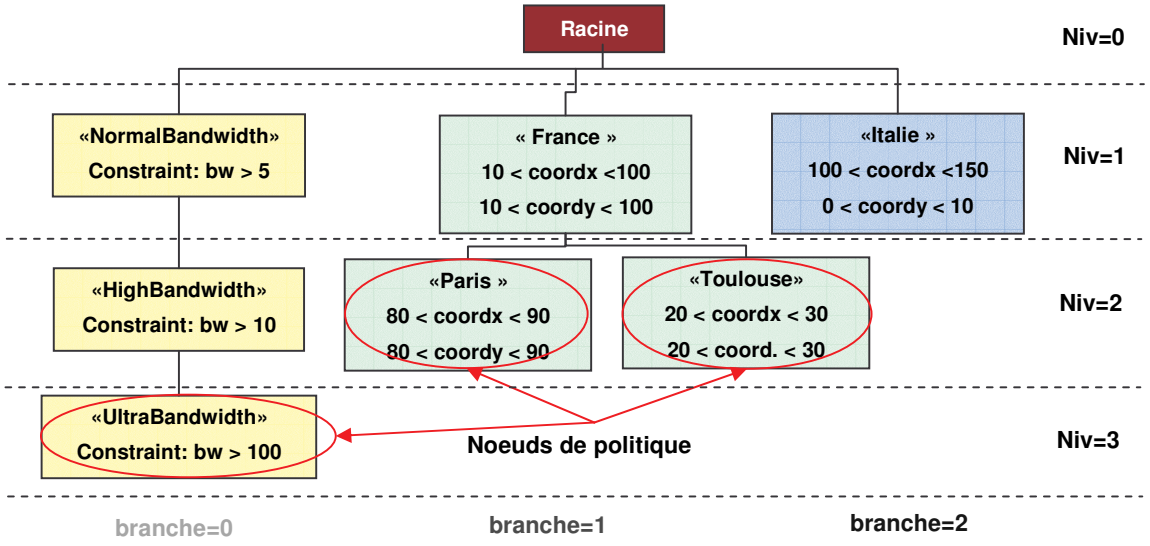

Figure 4. Exemple d'un arbre de spécialisation

JAXB de java [13] (voir figure 3), qui permet d'obtenir automatiquement à partir d'un schéma XML un ensemble de classes et d'interfaces java qui représentent la structure des fichiers XML à analyser.

5.2 Le gestionnaire du contexte local (LCM)

Le LCM maintient et met à jour les paramètres concernant le contexte d'exécution du terminal de l'utilisateur : type du réseau, bande passante, taux de perte de paquets, latence réseau, niveau de batterie, taux d'utilisation du CPU et niveau de l'occupation de la mémoire.

Dans notre implémentation, ces paramètres sont simulés à l'aide d'un fichier XML. La modification de ces paramètres peut se faire à l'aide d'une interface graphique ou à travers la ligne de commande.

5.3 Le coordinateur

5.3.1 Fonction

Il est l'élément central du gestionnaire d'adaptation. Sa principale fonction est de prendre la décision concernant le choix de la politique d'adaptation à appliquer. Il interagit avec les composants suivants:

- L'analyseur XML: en récupérant un pointeur sur l'arbre de spécialisation.
- Le LCM: en lui indiquant les paramètres du contexte d'exécution qui l'intéresse. le LCM le notifie de manière asynchrone à chaque modification de l'un de ces paramètres.
- MobileJMS: en notifiant à celui-ci la politique à appliquer à travers le graphe protocolaire qu'il doit utiliser.

Le coordinateur déclenche le processus du choix de la politique à appliquer à chaque fois qu'il y a un changement significatif du contexte. Il parcours l'arbre de spécialisation et évalue les bilans de chaque politique. Ces bilans sont le résultat de la mise en œuvre d'une politique donnée avec une certaine configuration de MobileJMS.

5.3.2 Bilan et application d'une politique

Les deux critères que nous avons retenus pour l'évaluation du bilan d'une politique, sont la localisation géographique et la bande passante. La politique retenue est celle qui propose le plus grand bilan, sachant qu'une même politique peut proposer plusieurs bilans suivant la manière de l'appliquer. Si deux politiques possèdent un même bilan, la politique retenue est celle qui est la plus spécialisée. Par exemple, la politique qui indique Paris comme zone géographique est plus spécialisée que celle qui indique la France.

Les bilans d'une politique P sont évalués à l'aide du pseudo-code suivant:

```
Si (position_actuelle ∈ P.zone_geographique)
  Si P.BW ≤ Contexte.BW  //BW : bande passante
    Bilan = P.BW
  Sinon
    // la compression est nécessaire
    // N types de compression possibles
    Pour i de 1 à N Faire
      Si P.BW =< Comp(i).BW(
        Bilan(i) = Comp(i).BW / Comp(i).surcoût
      Sinon
        Bilan = 0 // Politique non applicable
Sinon
  Bilan= 0 // Politique non applicable
```

Si la bande passante actuelle du contexte d'exécution est suffisante par rapport à la bande passante requise par une politique

(P.BW), la politique est applicable sans compression et son bilan est égal à cette bande passante. Dans le cas contraire, la compression des messages devient nécessaire et la politique propose plusieurs bilans qui correspondent à chaque algorithme et taux de compression. Chaque bilan est calculé à partir du rapport entre la nouvelle bande passante estimée Comp(i).BW, après la compression i, et le surcoût, en terme d'utilisation des ressources machine, engendré par l'utilisation de cette compression. Notons que si Comp(i).BW reste inférieure à P.BW le bilan est nul.

Comp(i).BW est estimée à l'aide de la table des gains. Cette table définit le ratio de compression des données (rapport entre la taille du message compressé et la taille du message initial) pour chaque type de données (images bmp, données texte, code binaire…) et de type de compression (gzip ou zip avec des taux de 1 à 9). Le ratio de compression permet de calculer la nouvelle bande passante. Notons que la table de gain est construite à partir de moyennes de mesures effectuées sur la compression de différents types de messages.

Par ailleurs, Comp(i).surcoût est estimé à partir du prix actuel qu'il faut « payer » pour l'utilisation de la compression i. Ce prix est calculé à partir du niveau actuel des ressources (CPU, mémoire, batteries). Le principe est que moins ces ressources sont disponibles sur la machine, plus leur prix devient cher. Nous ne n'utilisons pour l'instant que des formules approximatives faisant correspondre le niveau de ces ressources et le type de compression avec le surcoût généré.

5.3.3 Fragmentation

L'usage de la couche de fragmentation est décidé par le coordinateur suivant le taux de perte de paquets observé par le LCM. La fragmentation est utilisée si ce taux dépasse un certain seuil qui peut être configuré dynamiquement. Plus ce taux est élevé, plus la taille des fragments est petite.

5.3.4 Stockage et ré-émission (S&F)

L'activation de ce service est gérée de la manière suivante:

- si le coordinateur obtient une notification de la déconnexion réseau du client et qu'un délai d'attente maximum est spécifié dans la politique actuelle, le S&F est activé et rajouté au graphe protocolaire. Si aucun délai n'est précisé, le coordinateur décide de déconnecter l'application JMS définitivement du fournisseur JMS (fermeture de tous les objets JMS) ;
- si ce délai expire avant la reconnexion du client, l'application JMS est définitivement déconnectée;
- si le client se reconnecte avant l'expiration de ce délai, le coordinateur désactive le S&F après avoir envoyé tous les messages qui étaient en attente.

6. APPLICATION METEO

La figure 5 représente l'architecture d'une application JMS qui permet à un utilisateur mobile de recevoir périodiquement des données météorologiques. Ces données sont fournies par les deux serveurs *meteo_fr* et *meteo_de* qui correspondent respectivement à la météo en France et en Allemagne. Dans la terminologie JMS, ces serveurs sont des producteurs de message alors que l'application est un consommateur de messages. Les différents types de données fournies par ces serveurs sont des images haute résolution, des images basse résolution et du texte.

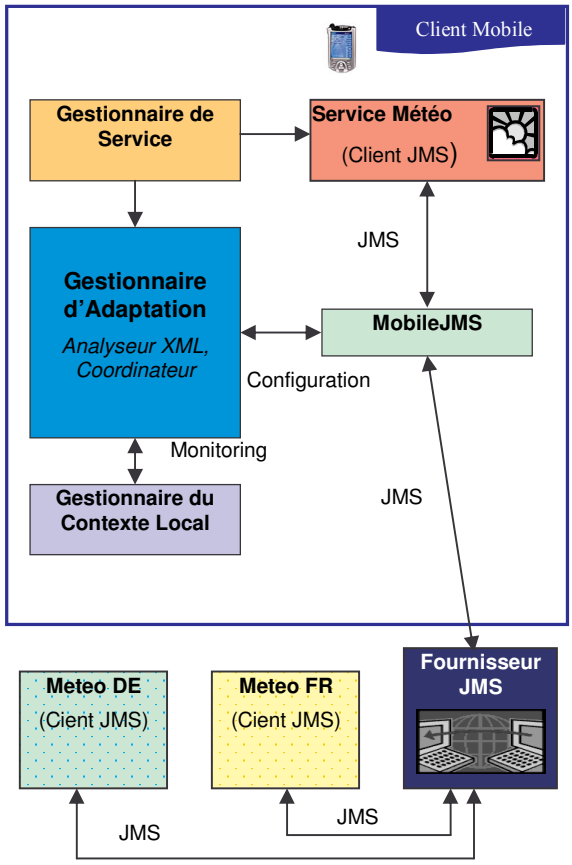

Figure 5. Architecture du service météo

Un ensemble de politiques d'adaptation est associé à cette application. Parmi les paramètres définis par ces politiques:

- **LowBW:** BW>50kbits, la localisation est inconnue, le serveur est meteo_fr et le type de données est "Text".
- **MediumBW:** BW>150kbits, la localisation est inconnue, le serveur est meteo_fr et le type de données est "SmallImage".
- **HighBW:** BW>200kbits, la localisation est inconnue, le serveur est meteo_fr et le type de données est "LargeImage".
- **LowBW_France:** BW>5kbits, la localisation est en France, le serveur est meteo_fr et le type de données est "Text".
- **LowBW_Paris:** BW>5kbits, la localisation est à Paris, le serveur est meteo_fr et le type de données est "Text".
- **MediumBW_Germany:** BW>150kbits la localisation est en Allemagne, le serveur est meteo_de et le type de données est "SmallImage".
- **HighBW_Germany**: BW>200kbits la localisation est en Allemagne, le serveur est meteo_de et le type de données est "LargeImage".

Les paramètres du contexte d'exécution peuvent être modifiés à l'aide du simulateur de ressources (figure 6). Suivant les valeurs de ces paramètres, plusieurs scénarios sont possibles, parmi lesquels:

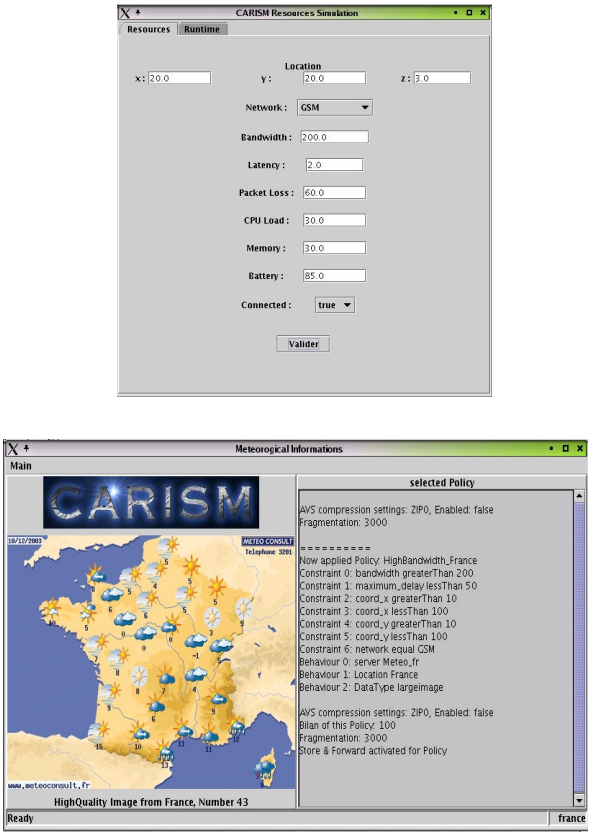

Figure 6. Capture écran du service météo avec le simulateur de ressource en haut, et l'application en bas

- **Bande passante élevée:** la bande passante est de 220kbits/s et la localisation est en France. L'utilisateur reçoit dans ce cas de meteo_fr des images haute définition et non compressées concernant la météo en France.
- **Diminution de la bande passante:** la bande passante diminue à 140kbits/s et la localisation reste en France. L''utilisateur reçoit dans ce cas de meteo_fr des images haute définition compressées avec le taux 7 de zip concernant la météo en France.
- **Bande passante moyenne:** la bande passante est de 100kbits/s et la localisation est à Paris. L'utilisateur reçoit dans ce cas de meteo_fr des images basse définition compressées avec le taux 5 de zip concernant la météo à Paris.
- **Faible bande passante:** la bande passante est de 10kbits/s et la localisation est à Paris. L'utilisateur ne reçoit dans ce cas de meteo_fr que des informations textuelles et non compressées concernant la météo à Paris.
- **Baisse du niveau de batterie:** la bande passante est de 180kbits/s, la batterie restante est de 10% et la localisation est en Allemagne. L'utilisateur reçoit dans ce cas de meteo_de des images basse définition et non compressées concernant la météo en Allemagne. Bien que des images haute définition peuvent être reçues dans ce cas en utilisant la compression, le niveau restant de batterie ne le permettrait pas.

7. TRAVAUX APPARENTES

Plusieurs intergiciels orientés message ont été conçus pour les environnements mobiles. iBus//mobile [12] est une implémentation de JMS destinée aux terminaux mobiles interconnectés à travers un réseau sans fil à grande échelle. Un système iBus//mobile est composé de trois types d'entités: des clients mobiles JMS, une passerelle JMS et un fournisseur JMS. Les clients JMS sont conçus suivant le type du terminal mobile utilisé. Sur les terminaux non programmables, le client JMS communique à l'aide de son mode de communication standard (WAP, SMS…). Sur les terminaux programmables dotés de la plate-forme J2ME, le client JMS communique à travers une implémentation légère de l'API JMS. L'élément clé de l'architecture de iBus//mobile est la passerelle JMS. Sa fonction est quelque peu analogue à une station de base qui connecte au réseau tous les terminaux mobiles se trouvant dans une zone géographique donné. Du point de vue du fournisseur JMS, cette passerelle n'est pas plus qu'un client JMS standard. Du point de vue du client JMS, la passerelle est un nœud où transitent les messages et où s'effectue leur conversion d'un format à un autre.

En comparaison de notre approche, iBus//mobile est plutôt destiné à des terminaux mobiles qui possèdent des ressources faibles (téléphones portables, pagers…). La passerelle permet de faire communiquer des clients très hétérogènes et assure pour le compte des clients quelques fonctions qu'ils ne peuvent pas assurer eux-mêmes (persistance, conversion de messages...).

WebSphere Everyplace [7] est une plate-forme d'IBM qui permet l'accès des utilisateurs mobiles aux données de l'entreprise. Cette plate-forme est conçue comme une extension du serveur d'application WebSphere qui intègre la spécification JMS. Parmi les points intéressants de cette technologie est sa capacité à fournir l'accès à des utilisateurs utilisant des protocoles de communications différents (GRPS, SMS, WAP, Wifi, Bluetooth...). Everyplace utilise pour cela des services pour l'adaptation du contenu suivant les capacités et les formats de données pris en charges par les terminaux mobiles. Un gestionnaire de synchronisation permet également aux utilisateurs mobiles d'opérer en mode non connecté et de synchroniser le résultat de leurs activités avec les bases de données du serveur, une fois que la connexion est rétablie.

Pronto [14] est également un intergiciel JMS destiné aux environnements mobiles. Les messages échangés transitent par une passerelle qui utilise des composants dynamique (*plug-in*) assurant des fonctions comme la compression ou le transcodage sémantique des données. La passerelle peut se situer sur le terminal du client ou sur une autre machine du réseau. Dans ce dernier cas, la communication est basée sur Java/RMI. Afin de gérer les déconnexions et réduire le trafic entre le fournisseur JMS et la passerelle, un mécanisme de cache est mis en œuvre. Si la déconnexion d'un client se produit, la passerelle continue de recevoir les messages. Lors de sa reconnexion, le client se synchronise avec le cache pour récupérer uniquement les dernières mises à jour des données. Pronto propose également un mode décentralisé où les messages sont transmis directement des émetteurs vers les récepteurs (sans la présence d'un fournisseur). Bien que ce mode porte l'opérabilité de cet intergiciel à un environnement

totalement ad-hoc, certaines sémantiques de JMS comme la persistance où la garantie d'envoi des messages risquent de ne pas être parfaitement respectées.

Le projet CARISMA [2] est un intergiciel qui se focalise sur la nécessité que les applications mobiles soient « conscientes » de leur contexte d'exécution. Cette prise en charge du contexte est assurée à travers le mécanisme de réflexivité. Un système réflexif peut modifier son propre comportement à travers l'introspection (le comportement interne du système est exposé) et/ou l'adaptation (le comportement interne du système est modifié dynamiquement). Ce principe est exploité pour adapter le comportement de l'intergiciel au contexte d'exécution interne. Le processus de d'adaptation ou de configuration utilise les méta-données, ces informations représentent le comportement que doit adopter l'intergiciel pour répondre aux requêtes de qualité de service émises par les applications dans des contextes différents. Le comportement de l'intergiciel est décrit par un profil d'application. Ce profil est un ensemble d'associations entre les services que l'intergiciel peut configurer, les politiques qui peuvent être appliquées pour délivrer les services et un contexte d'exécution dans lequel cette politique peut s'appliquer.

Néanmoins, la possibilité offerte aux applications de contrôler le comportement de l'intergiciel, ouvre la voie à des conflits. En effet, des comportements incompatibles peuvent être dictés par une même application ou par plusieurs applications. CARISMA apporte une réponse à ce problème à travers une approche inspirée des techniques micro-économiques. Le système mobile réparti est considéré comme un « capital » que les applications s'approprient et se répartissent pour obtenir la qualité de service dont ils ont besoin.

8. CONCLUSION

Nous avons présenté dans ce papier notre approche de l'adaptabilité dans les environnements mobiles qui se base sur la coopération entre les applications et l'intergiciel MobileJMS. En premier lieu, nous avons distingué les différents concepts et facteurs qui rentrent en jeu. Nous avons également décrit un modèle basé sur les politiques d'adaptation, qui permet d'adapter dynamiquement les applications au contexte d'exécution, suivant leurs besoins et les services proposés par l'intergiciel. Nous avons, par ailleurs, présenté quelques points clés de l'intergiciel MobileJMS et les avantages que procure l'utilisation des solutions basées sur le passage des messages dans les environnements mobiles.

Nous avons l'intention de faire évoluer ce modèle d'adaptabilité afin de prendre en considération quelques autres aspects importants. D'abord, la prise de décision concernant la politique à appliquer devra se faire selon des critères plus fidèles à la réalité. Nous allons multiplier, dans ce but, les expérimentations qui permettront de mieux établir le lien qui existe entre le coût de l'utilisation des services de l'intergiciel et la consommation des ressources de la machine. Nous pensons que l'utilisateur peut être également associé à ce processus de décision pour indiquer ses choix et ses préférences concernant les ressources à préserver ou le type de qualité de service qu'il souhaite obtenir. Par ailleurs, l'autre piste de recherche est la gestion du cas où plusieurs applications solliciteront simultanément les services de l'intergiciel et les ressources de la machine. Le modèle doit être capable de résoudre les conflits qui risquent de s'y produire.

9. REMERCIEMENTS

Ce travail se déroule dans le cadre du projet CARISM qui regroupe l'ENST, l'INT et l'ENST-Bretagne. Nous remercions Guy Bernard et Octavian Foléa (INT) pour leurs propositions et commentaires constructifs. Nous remercions, par ailleurs, Kader Bancole, Samantha Ngapanoun, Christian Müller, Francisco Sánchez et Johannes Zeidler (ENST) pour leur participation à la réalisation de ce travail.

10. REFERENCES

[1] Banavar, G., Chandra, T., Strom, R., and Sturman D. A case for message Oriented middleware. In *LNCS 1693*. Springer Verlag, 1999, pp 1-18.

[2] Capra, L., Emmerich, W., and Mascolo, C. A Micro-economic Approach to Conflict Resolution in Mobile Computing. *In Proeedings. of the 10th ACM SIGS*OFT *foundations of Software Engineering Conference (Charleston, South Carolina, USA)*. pp. 31-40. ACM Press. 2002.

[3] Deutsch, P. Gzip file format specification version. 1996.

[4] Dumant, B., Horn, F., Dang Tran, F., and Stefani, J.B. Jonathan: an Open Distributed Processing environment in java. In *Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing*. The Lake District, U.K., Sept.1998.

[5] Gaddah A., and Kunz, T. A survey of middleware paradigms for mobile computing. Technical Report SCE-03-16. *Carleton University Systems and Computing Engineering.* July 2003

[6] Hapner M., and al. Java Message Service specification. Sun MicroSystems, April 2002.

[7] IBM. M. IBM WebSphere Everyplace Access Version 4.3.0. Oct. 2003.

[8] Kaddour, M., and Pautet, L. Towards an Adaptable Message Oriented Middleware for Mobile Environments. In *Proceedings of ASWN'03*. Bern, Switzerland, July 2003.

[9] Kaddour, M., and Pautet, L. A middleware for supporting disconnections and multi-network access in mobile environments. In *Proceedings of 2nd IEEE International Conference on Pervasive Computing Communications (Percom)*. Orlando, FL, USA, March 2004.

[10] Mascolo, C., Capra, L. and Emmerich, W. Middleware for mobile computing. In *LNCS 2497*. Springer Verlag, 2002.

[11] Openjms.org. OpenJMS project. http://www.openjms.org.

[12] Softwired. iBus//mobile. http://www.softwired-inc.com/products/mobile/mobile.html

[13] Sun MicroSystems. Java The Java™ Architecture for XML Binding (JAXB), Final, V1.0. January 8th, 2003

[14] Yoneki, E., Bacon, J. Pronto: messaging middleware over wireless networks. *In proceedings of ACM/IFIP/USENIX International Middleware Conference (Work in Progress)*. Rio de Janeiro, Brazil, June 2003.