UNIVERSITÉ DU QUÉBEC

MÉMOIRE

PRÉSENTÉ À

L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI

COMME PARTIELLE EXIGENCE

POUR LA MAÎTRISE EN INFORMATIQUE

PAR

ABDELALI GOUNDAFI

DÉVELOPPEMENT D'UN MODÈLE DE GESTION D'OBJETS GÉO-LOCALISABLES CENTRALISÉ UTILISANT DIFFERENTS MOYENS DE COMMUNICATION DANS UN ENVIRONNEMENT OSGI

4 décembre 2010

Résumé

Pendant la dernière décennie, la vente des GPS a connue une explosion à l'échelle planétaire. Les applications logicielles et les améliorations matérielles liées à cette industrie, ne cessent de proliférer et ce à une vitesse exponentielle.

Désormais, les systèmes de géo-localisation, et plus précisément le GPS, sont utilisés dans plusieurs domaines. Plusieurs compagnies de transport l'utilisent pour contrôler l'itinéraire de leurs conducteurs afin d'optimiser l'utilisation du réseau routier, d'économiser la consommation d'essence, le temps de transit, etc. Les soldats l'utilisent lors des conflits pour se localiser dans les endroits les plus difficiles d'accès, ils utilisent le GPS aussi pour le guidage précis de leurs bombes et missiles afin de minimiser les dégâts collatéraux. Dans le domaine de la santé, le GPS permet d'augmenter les chances de survie des personnes atteintes d'Alzheimer en les localisant très vite. Un dispositif appelé VPS peut, via un cardiogramme intégré, déceler l'imminence d'une crise cardiaque et envoyer la position du patient aux services de santé afin d'intervenir à temps et par conséquent sauver le patient en question.

Afin de pouvoir gérer tous les services de localisation mentionnés précédemment de manière automatique, nous avons besoin d'un système qui puisse interconnecter tous ces services de façon transparente sans que le code soit un frein pour le développement ni que l'ajout de nouveaux services alourdisse le système et rende la maintenance une tâche très difficile au programmeur.

Le deuxième point de la problématique concerne la communication, les données peuvent être envoyées à distance, dans un système de géo-localisation classique, via un réseau cellulaire. L'usage du réseau cellulaire est coûteux et nous avons besoin d'autres alternatives pour assurer une communication sans-fil à distance à moindre coût.

Le troisième point à résoudre est la précision des dispositifs de géo-localisation. Cette précision n'est pas toujours assurée avec l'usage classique du GPS. Des erreurs dans le système GPS peuvent augmenter considérablement l'imprécision de l'erreur sur le positionnement, pour palier ce problème nous devons disposer de solutions qui permettent d'augmenter la précision et de déterminer si le relevé de la position calculée par le GPS est fiable pour le service de géo-localisation.

En mettant tous les services de géo-localisation dans un même système, nous parviendrons à créer un environnement coopératifs où chaque service pourra utiliser un ou plusieurs autres services pour optimiser le temps de réponse et d'intervention ce qui constitue un quatrième point de notre problématique. La vision globale des services coexistant dans le même système permettra aux décideurs de haut niveau de trouver la logistique optimale pour que les services gérés soient dans leurs pleines capacités opérationnelles.

Les approches de localisation existantes sont basées sur une architecture client serveur où on utilise une connexion GPRS qui est très coûteuse pour envoyer les relevés de positionnement à distance et de manière périodique au serveur. Au niveau du serveur on dispose d'une application pour un service de géo-localisation souvent spécifique à un seul domaine d'application. Toute évolutivité du système devient difficile et onéreuse dépendamment de la complexité de l'application.

Dans ce mémoire, j'ai conçu un modèle pour la gestion des objets géo-localisables qui nous permet de gérer plusieurs services de géo-localisation dans une même plateforme. Cette solution apportera des améliorations par rapport aux modèles standards au niveau du système de communication, la fiabilité des données de localisation, la programmation des composantes du système, de leur intégration et de leur maintenance. Nous avons implémenté ce modèle (dans un outil de gestion et de contrôle d'objets géo-localisables) pour répondre aux quatre problématiques mentionnées précédemment.

Remerciements

Je tiens tout d'abord à remercier mon directeur Hamid Mcheick qui m'a offert toute son aide et soutien ce qui m'a permis de mener à bien mon projet de maîtrise. Il a été minutieux quant au contenu et contenant de mon travail, très disponible à répondre à mes questionnements et d'une grande aide à me remettre dans la bonne direction à mes moments d'égarement.

Je remercie aussi, mon père Omar, ma mère Majda et ma sœur Nour de m'avoir toujours soutenu et encouragé. Mon père m'a offert une aide précieuse en corrigeant mes documents lorsque le temps ne me le permettait pas. Il m'a aussi aidé à adapter mon style d'écriture, tout comme mon directeur, afin de rendre les idées véhiculées dans mon mémoire plus intelligibles et compréhensibles pour un quelconque lecteur.

Je n'oublierai point l'influence positive de mes amis qui m'ont tout le temps soutenus et encouragés quand je perdais mon inspiration et que ma tâche me semblait colossale et décourageante. Tout comme mes chers parents et ma cher sœur, ils m'ont réconforté en mes moments de doute et m'on rassuré en croyant fort à mon travail et en m'encourageant à aller vers l'avant. Merci Carlo, Émilie, Faustine, Jérôme, Jimmy, Mélanie et tous les autres.

Je n'ai pu accéder à cette maîtrise sans l'aide de mon Oncle Moustapha et ma tante Raymonde qui m'ont offert leur soutien. En plus de m'offrir un toit et subvenir à mes besoins ils m'ont offert la chaleur familiale qui m'a aidée à m'épanouir, à m'éviter plusieurs soucis que connaissent plusieurs étudiants internationaux vivants seuls. À vous je serais toujours reconnaissant et ne vous remercierai jamais assez pour toute votre générosité

et support. Merci du fond du cœur à toute personne, de prés ou de loin, m'ayant soutenue et cru en moi.

Table des matières

Liste des figures

Liste des tableaux

I

INTRODUCTION

Depuis plusieurs années, les systèmes de localisation et d'identification des objets géo-localisables (personnes, voitures ou autres) deviennent de plus en plus accessibles au large public et ne cessent de s'améliorer. Nous devons cette amélioration aux progrès technologiques comme la mise en œuvre de nouveaux satellites avec des signaux plus précis pour le calcul des positions [D.Fontana & Al, 2001]. Les équipements géo-localisables permettent de suivre les déplacements d'un objet géo-localisable à n'importe quel moment et dans n'importe quel lieu. Les systèmes de localisation deviennent de plus en plus indispensables pour améliorer notre mode de vie d'où la nécessité de la mise en place des services de suivi ou de contrôle d'objets localisables opérant à distance [M.Russel & Al, 2002] [C.Pellerin, 2006].

Le travail présenté dans ce mémoire est motivé par des situations critiques que l'on vit en société, où un système de localisation efficace permet d'apporter une aide précieuse, voir même sauver des vies. D'après la Société Alzheimer de Montréal : « une personne qui souffre de cette maladie a une chance sur deux de se blesser ou même de mourir si elle n'est pas retrouvée dans les douze heures suivant sa disparition» [D.Rousseau, 2009]. Selon les statistiques de la société « Alzheimer rive-sud », 119 000 Québécois âgés de plus de 65 ans sont atteints d'Alzheimer [Alzheimer-rive, 2009].

Plusieurs personnes doivent aussi leur vie au VPS (Vital Positioning System). Muni d'un GPS et d'un mini-cardiogramme, l'équipement VPS permet d'alerter les services de santé quand un patient risque une crise cardiaque jusqu'à huit minutes avant qu'elle ne se produise [J.Faure, 2003].

Plusieurs publications ont aussi mentionné le fait qu'un système de localisation permet de sauver des vies et d'éviter des complications liées à des troubles psychiques ou de santé physique.

Les systèmes de localisations classiques utilisent une architecture client/serveur. Cette architecture permet de disposer d'appareils mobiles distants qui déterminent la position des clients. Ces appareils utilisent le réseau téléphonique sans fil pour envoyer les données vers un serveur distant. Ce dernier centralise les données de provenance de tous les appareils mobiles distants du système et offre une interface graphique pour visualiser ces données, sur une carte par exemple.

1.1 Énoncé de la problématique

Dans un premier temps, un système de géo-localisation pour des services tels que la santé publique, la sécurité routière, les services de police, etc. doit tenir compte de la précision de la position. En effet cette précision est critique pour un rendement efficace et une intervention rapide. Dans ce cas de figure nous devons disposer d'un système de positionnement qui : i) offre une marge d'erreur de positionnement très minime, et ii) offre le moyen de vérifier l'intégrité des données de positionnement pour éviter de se fier à de fausses informations. Hélas, l'équipement de positionnement tel que le GPS n'est pas capable, à lui seul, de garantir l'exactitude de l'information de géo-localisation [WAAS, 2010].

Deuxièmement, les dispositifs distants doivent envoyer à fréquence régulière leurs données de positionnement à un ou plusieurs serveurs qui centralisent et traitent ces

informations. Les deux moyens de communication classiques sont le GPRS ou le SMS (voir chapitre 2). Ces derniers utilisent un réseau de téléphonie cellulaire privé, ceci engendre un coût d'utilisation facturable par l'opérateur téléphonique et ce pour chaque équipement communiquant avec les serveurs centraux.

Le troisième point de la problématique posée concerne l'implémentation d'un système regroupant plusieurs services. Ce regroupement a pour but d'interconnecter plusieurs services tels que les services de santé avec les services de police et de sécurité routière. Cette interconnexion nous permet d'avoir une vision globale du système afin d'optimiser la coordination entre les services mentionnés précédemment et gagner en efficacité d'intervention et de disposer d'une meilleure logistique sur le terrain. Ce regroupement de services n'est pas sans contraintes techniques. D'ailleurs, l'ajout de nouveaux services peut être problématique surtout si nous confions cette tâche à une autre équipe de développeurs qui n'avait pas contribuée au développement initial du système. Par conséquent, ils doivent le réétudier pour comprendre le fonctionnement et ainsi pouvoir intégrer des fonctionnalités nouvelles, mettre à jour le système, etc. Cette contrainte engendre un plus grand coût de développement qui peut être évité quand la programmation de nouveaux services ou la mise à niveau de ceux déjà existants se faisait de manière indépendante du code et transparente au programmeur.

1.2 Principe de solution

Pour le premier point soulevé dans la problématique concernant la précision, j'ai introduit un modèle basé sur des services de correction des données de positionnement

(voir A-GPS au chapitre 2). En ce qui concerne le deuxième point dans notre problématique, j'ai proposé un modèle réseau pour optimiser le coût de communication ainsi qu'un algorithme de communication qui permet de choisir le moyen de communication le plus adéquat afin d'assurer l'envoi des données vers notre serveur central. Pour ce qui est du troisième point de notre problématique concernant la programmation, j'ai proposé l'usage d'un intergiciel qui implémente OSGI (voir chapitre 3) afin de faciliter l'intégration des nouveaux services et la mise à niveau des services déjà existant.

Il est à noter que certains développements en cours visent à créer un réseau de lecteurs RFID en utilisant différentes techniques tels que RSS ou TOA [A.Assad, 2007] [U.Bischoff & al., 2006]. Peu importe la technique utilisée, la mise en place de tels systèmes est complexe, que se soit par rapports aux algorithmes employés ou les limitations imposées par l'environnement tel que la gestion du « Multipath ». Le but de ce mémoire n'est pas de réinventer la roue mais bien de trouver le moyen le plus simple et le plus efficace en combinant différentes technologies pour aboutir à une localisation raisonnablement précise et à faible coût de développement et de déploiement.

1.3 Contribution

Le présent mémoire a pour but de décrire les principaux systèmes de géo-localisation et de communication afin de comprendre les avantages ainsi que les limites de chaque système. Cette analyse nous permet de prendre une décision réfléchie quant au choix des implémentations possibles d'un système de géo-localisation. Ce mémoire a aussi pour

objectif de démontrer la faisabilité du système proposé et d'offrir une preuve de conception matérialisée par le développement d'une application de gestion d'objets géo-localisables. L'usage d'OSGI n'est pas très répondu comme plateforme cible des développeurs mais son potentiel a été démontré dans ce mémoire via la description de ses atouts dans le chapitre 3 et l'application développée puis exposée au chapitre 5. Pour résumer, le système proposé dans ce mémoire répondra aux points suivants :

- Développer un modèle de surveillance systématique pour identifier un objet géo-localisable. Cet objet peut être une personne, un téléphone, une voiture, etc.
- Faciliter le processus de détection et d'intervention rapide (en cas de dépassements de vitesse pour les voitures, urgence médicale liée aux troubles psychiques ou de santé physique comme pour les personnes cardiaques ou atteinte d'Alzheimer, etc.)
- Généraliser ce modèle pour une utilisation dans différents domaines nécessitant une détection précise et un délai d'intervention rapide. En particulier, ce modèle permet au service de police de réagir rapidement dans les cas de voitures en infraction ou d'envoyer une équipe d'intervention médicale pour secourir à temps les personnes égarées, atteintes des troubles psychiques ou souffrant de maladies à risque de mort imminente tel qu'un infarctus.
- Développer un outil pour implémenter ce modèle. Cet outil permet de localiser un ensemble d'appareils GPS distants de façon centralisée en utilisant plusieurs types de connexions alternatives (GPRS, Internet, radio sans-fil, etc.). Quand l'une d'elles échoue, la suivante qui est disponible prend la relève, en commençant par la moins coûteuse.

- Proposer un modèle standard qui consiste en un middleware (OSGI-knopflerfish) pour faciliter et minimiser le coût et le temps du déploiement, de la mise à niveau et de l'intégration de nouveaux services.
- Mettre en œuvre un algorithme permettant aux différents objets géo-localisables de coopérer entre eux afin de relayer leurs données vers notre serveur central.

Le présent mémoire comprend six chapitres. Le présent chapitre pose la problématique et définit les points qu'on doit solutionner par la suite. Le chapitre 2 présente l'état de l'art en ce qui concerne i) les systèmes et techniques de géo-localisation, et ii) les moyens de communication sans fil. Le chapitre 3 présente l'outil de développement via l'intergiciel OSGI permettant la portabilité des données, la réutilisation, la facilité d'intégration et de maintenance du code via l'approche de programmation par composantes. Dans le chapitre 4, je propose un modèle de solution pour ce système de gestion d'objets géo-localisables. Quand au chapitre 5, il porte sur l'implémentation concrète de ce modèle à travers des captures d'écrans de l'exécution du programme et les échantillons du code de programmation au niveau des objets mobiles et du serveur central ainsi que le coût estimé pour l'implémentation du système. Notons que le code détaillé est donné en annexe à la fin de ce mémoire. Le chapitre 6 comporte la conclusion et les suggestions proposées pour des améliorations futures.

II

REVUE DE LITTÉRATURE

Comme mentionné à la fin du chapitre 1, ce chapitre présente l'état de l'art quant aux techniques de localisation (par satellites et par réseaux de détecteurs) et les moyens de communication permettant le transfert des données sans fil à distance. Le but de ce chapitre est aussi de décrire les éléments nécessaires pour produire la solution présentée plus tard au chapitre 4.

2.1 *Les techniques de positionnement*

Il existe différentes techniques pour déterminer la position d'un objet géo-localisable. Dans ce chapitre, j'ai classé ces techniques en deux grandes catégories selon leur nature : i) techniques de localisation via des satellites, et ii) techniques permettant la localisation à travers un réseau de détecteurs au sol.

2.1.1 Techniques de positionnement par satellites

Ces systèmes utilisent une constellation de satellites en orbite autour de la terre pour envoyer des ondes radio au sol. Ces ondes sont prises par un capteur au sol qui reçoit le signal de plusieurs de ces satellites simultanément. Le récepteur procède à la détermination de la position quand il y a suffisamment de satellites visibles pour effectuer une trilatération ou la triangulation [Triangulation, 2006]. Les sous-sections suivantes vont décrire les systèmes de positionnement par satellites existants : GPS, GALILEO, EGNOS, MEOSAR et GLONASS.

2.1.1.1 Système GPS

Le système GPS est le plus populaire des systèmes de positionnement par satellites. Créé par le département de défense américain (DoD), le système GPS a pour mission de couvrir toute la terre grâce à une constellation comptant 24 satellites [USAirForce, 2010]. Les satellites GPS sont répartis sur six orbites, chaque orbite compte quatre satellites GPS. Ce système satellitaire fournit aux récepteurs GPS des informations en continue. Le récepteur mesure le temps de propagation de l'onde reçue et calcule ainsi la distance qui le sépare du satellite. Il répète cette opération pour chaque satellite en vue. Une fois que les distances entre le récepteur et les satellites sont connues, il devient possible de calculer la position par triangulation (voir la figure 2.1 concernant la triangulation). Nous avons besoin de trois satellites pour déterminer un point et quatre satellites pour avoir une position en 3D (incluant la profondeur). Nous pouvons déduire que plus il y a de satellites GPS visibles au récepteur, plus la position calculée est exacte.

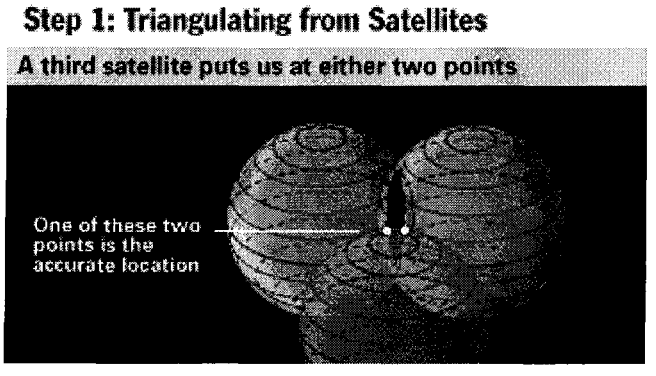

Figure 2.1 : Triangulation via 3 sphères relatives à la couverture de 3 satellites [TriangTrimble, 2010]

Dans la figure 2.1, Nous constatons que deux sphères se croisent et forment une zone d'intersection mais qui reste large à l'échelle géographique. L'ajout de la troisième sphère de couverture d'un troisième satellite permet de créer une autre intersection en deux points précis. Pour déterminer quel point choisir, Nous pouvons faire une déduction logique dans la plupart des cas. L'un des deux points peut se trouver dans un endroit très loin de la terre ou bien il peut avoir une vitesse démesurément grande. Avec l'élimination du point illogique, Nous déduisons que le deuxième point qui reste est la position qui tend le plus vers la position exacte. Si un quatrième satellite est en vue, nous n'avons pas besoin de faire de déduction logique puisque sa nouvelle sphère d'intersection va coïncider exactement avec le vrai point représentant la position du récepteur. Plus en aura de satellites visibles par le récepteur, l'erreur sur le positionnement sera minime étant donné qu'une autre intersection délimitera d'avantage le point de localisation [TriangTrimble, 2010].

L'exactitude de la position dépend aussi du récepteur, la majorité des récepteurs ont une précision de dix mètres qui varie selon la qualité de leur conception (performance de la puce intégrée, type d'antenne, etc.). Les facteurs d'erreur sur la position les plus importants sont décrits comme suit :

i) Pour déterminer la position avec triangulation, nous devons connaître la distance entre le récepteur et chaque satellite. Chaque satellite émet un signal avec sa date d'émission. En recevant le signal, un récepteur fait la différence entre la date d'émission et de réception pour déterminer le temps de propagation. En multipliant

le temps de propagation du signal reçu par la vitesse de la lumière, nous arrivons à connaitre la distance qui sépare le récepteur du satellite. Sauf que la réalité diffère de ces faits. La vitesse de la lumière n'est constante que dans le vide. Les différentes couches de l'atmosphère retardent la propagation de l'onde [ErrTrimble, 2010] comme la figure 2.2 le montre.

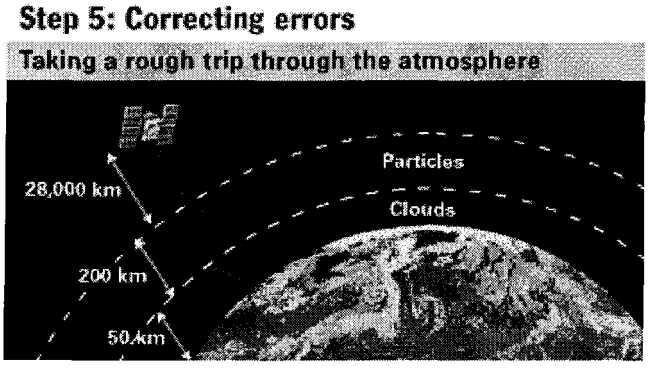

Figure 2.2 : Les différentes couches atmosphériques déviant et freinant le signal du satellite [ErrTrimble, 2010]

Pour éviter le problème d'erreurs causées par les couches atmosphériques, on utilise une technique plus avancée qui consiste en l'utilisation de deux signaux différents L1 : 1575.42 MHz et L2 : 1227.60 MHz. Les lois de la physique disent que les signaux basse-fréquence sont plus réfractés ou retardés que les signaux haute-fréquence en traversant un milieu chargé d'ion comme l'ionosphère ou chargé de particules d'eau (nuages) comme dans la troposphère [ErrTrimble, 2010]. La mesure simultanée des deux signaux donne une idée sur les variations que subie l'onde. Le récepteur équipé de ce système de mesure peut alors ajuster les paramètres et déduire une distance plus proche de la réalité.

ii) Le multi trajet est un autre problème à gérer plus spécialement dans un milieu urbain. Les signaux des satellites, avant d'atteindre le récepteur, vont rencontrer des obstacles comme les grands bâtiments (effet canyon). Ces obstacles vont réfléchir le signal dans plusieurs directions, selon la disposition de leur surface et du matériau de construction. Le récepteur recevant un signal réfléchi, calculera une distance plus longue que la distance qui le sépare réellement du satellite, car une ligne directe est toujours plus courte qu'un chemin qui change de direction. La figure 2.3 montre un signal réfléchi par des édifices.

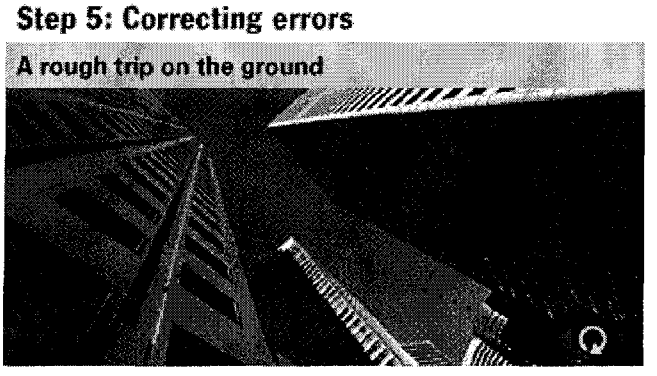

Figure 2.3 : Réflexion du signal sur les hauts édifices [ErrTrimble, 2010]

En réalité, un récepteur GPS peut recevoir un signal direct en premier et un signal réfléchi en retard. Pour éviter de prendre en compte le signal réfléchi, certains GPS utilisent des techniques de rejet de signal qui détectent le signal réfléchi et l'ignorent (signal rejection) [ErrTrimble, 2010].

iii) Nous avons vu que pour la triangulation, le satellite émet un signal avec une date d'émission qui permet au récepteur de calculer le temps de propagation du signal.

En effet, chaque satellite dispose d'une horloge atomique très précise synchronisée via des stations de contrôle terrestre. Toutefois, il arrive que cette horloge désynchronise au moment de l'envoi d'un signal. Cette désynchronisation affecte la date d'envoi inscrite dans le signal envoyé vers les récepteurs au sol. Le récepteur calcule la distance avec un faux temps d'émission ce qui résulte en une fausse distance calculée. La désynchronisation touche plus encore les récepteurs eux mêmes car ils sont munis d'horloges internes à base de quartz beaucoup moins précises que les horloges atomiques du satellite. Les récepteurs doivent constamment synchroniser leurs horloges internes avec celles des satellites. Ce type d'erreurs peut être corrigé par des techniques tel que le DGPS qu'on verra plus tard dans les sous sections qui suivent.

iv) L'indice PDOP (Position Dilution Of Precision) dépend de la disposition géométrique des satellites. Plus les satellites sont rapprochés et plus cet indice est élevé. Par conséquent, l'erreur sur le positionnement augmente puisque la zone d'intersection des sphères de couvertures des satellites est plus large comme le montre la figure 2.4.

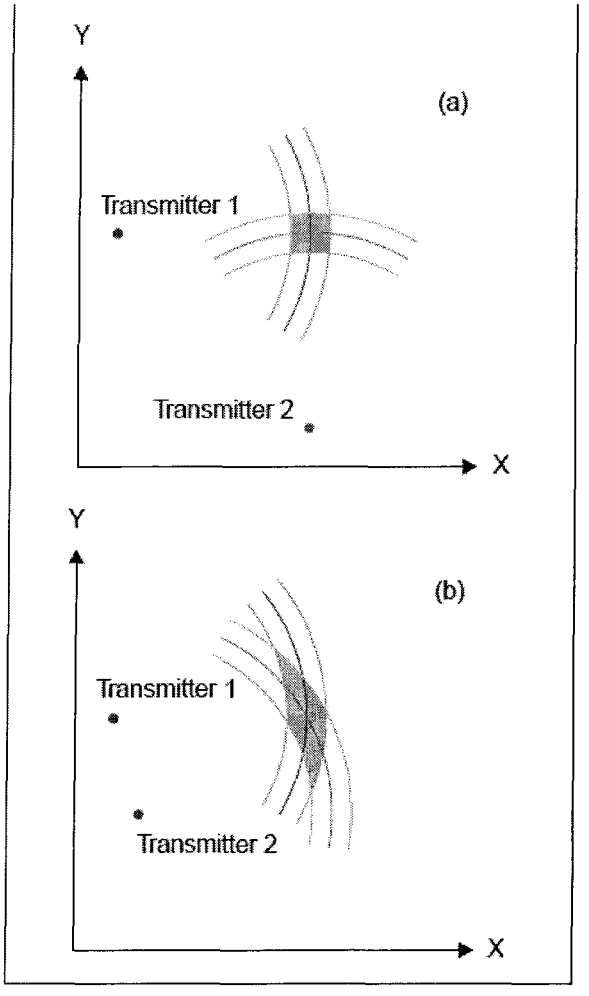

Figure 2.4 : *Comparaison de deux cas de figures influant sur le DOP (Dilution Of Position)*

Dans la figure 2.4, deux cas de figures sont considérés (a) et (b), qui représentent chacun deux signaux radios émis d'un transmetteur 1 en rouge et un transmetteur 2 en vert. Les cercles d'intersection des rayons de couverture de chaque émetteur constituent une intersection en rouge et vert. Dans le cas de figure (a), l'intersection est plus petite puisque les angles de diffusion de chaque émetteur sont très éloignés. Ceci implique une petite incertitude quand à la position. Dans le deuxième cas de figure (b), la zone d'intersection

est plus grande ce qui résulte en une plus grande incertitude sur la position. Dans le cas de figure (b) nous constatons que les deux transmetteurs émettent d'un presque même angle [B.Langley, 1999][R.Thompson & Al, 2009].

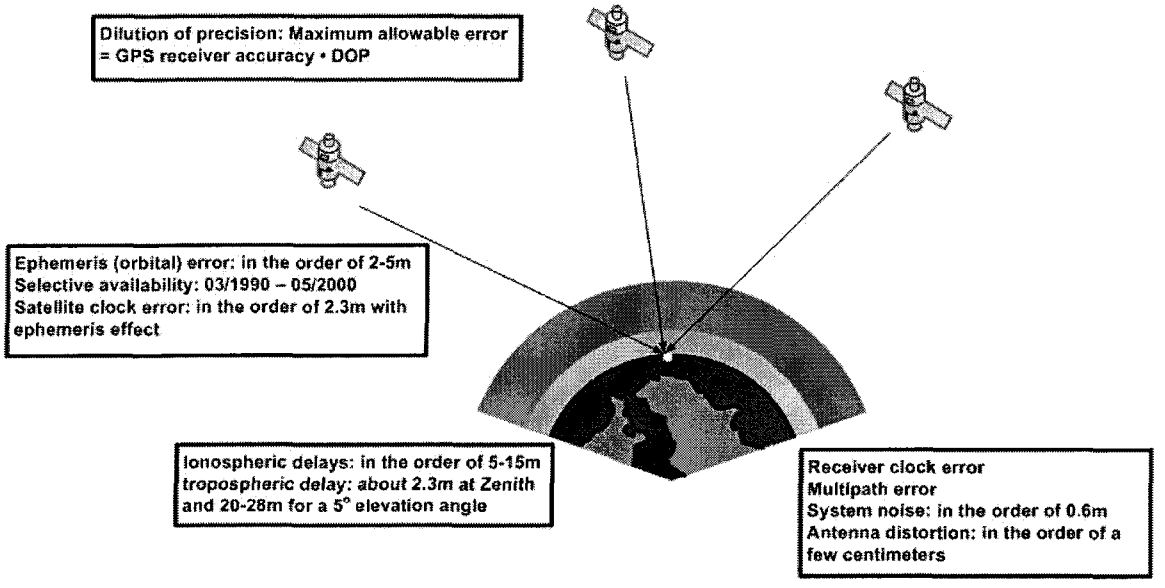

Figure 2.5 : Facteurs d'erreurs affectant la précision du GPS

La figure 2.5 illustre les facteurs d'erreurs les plus fréquents [H.Dana, 94] [PNT, 2008] affectant la précision du GPS. Les facteurs d'erreurs les plus fréquent sont:

- Effets ionosphériques ± 5 mètres
- Les changements dans les orbites des satellites ± 2.5 mètres
- Erreur d'horloges des satellites ± 2 mètres
- Effets multi trajets ou canyon ± 1 mètre

- Effets Troposphériques ± 0,5 mètre
- Erreur d'arrondi de calculs ± 1 mètre

La somme moyenne des erreurs est généralement de ± 15 mètres. Ceci implique une grande marge d'erreurs pour que la donnée de positionnement soit fiable pour notre application. Il est donc nécessaire de corriger les erreurs via les services DGPS tels que WAAS, EGNOS et A-GPS. Les erreurs corrigées avec ces techniques sont des erreurs dont l'équipement GPS n'est pas responsable. Il s'agit de l'effet ionosphérique, des changements dans les orbites des satellites, des erreurs d'horloge des satellites, etc. Ces erreurs sont corrigibles par les stations terrestres qui vérifient ces erreurs et peuvent ainsi communiquer les corrections aux GPS capables de lire ces informations de correction. La correction des erreurs via les techniques DGPS ramènent l'erreur à plus au moins 3 à 5 mètres [kowoma, 2009] [PNT, 2008].

Pour ce qui est du calcul de la vitesse de déplacement d'un équipement GPS, il existe deux méthodes de calcul :

i) Vitesse moyenne : Cette méthode consiste à journaliser les positions relevées par le GPS, ensuite on calcule une vitesse moyenne sur un intervalle de temps. Cette méthode n'est pas fiable avec une précision de l'ordre du km/h, étant donné que chaque position comporte une erreur, variant en fonction de la précision des relevées de points de positionnement. Quand nous connaissons la position exacte en T1 et en T2, nous pouvons déduire la vitesse exacte. Mais si une erreur trop

importante advient en déterminant les pseudos-distances pour T1 ou/et T2, nous aurons des points de positionnement différents des points réels. Par conséquent, le calcul de vitesse est erroné [T.Chalko, 2007].

ii) Vitesse instantanée par effet Doppler : L'effet Doppler [E W.Weisstein, 2007] est le fait que la fréquence captée par le GPS diffère de la fréquence générée par le satellite dans un intervalle de temps dû au déplacement de l'émetteur vis-à-vis du récepteur. Pour simplifier, admettons que nous avons une source sonore fixe et que nous écoutons en se déplaçant sur l'angle d'émission de cette source, nous constatons que le son change d'ampleur. En mesurant ces variations, nous obtenons la vitesse du récepteur selon la formule : $v_r = v_s \cdot \cos\theta$ où vr est la vitesse du récepteur et θ l'angle de la ligne de visée vis-à-vis de l'émetteur.

La méthode de mesure de vitesse par effet doppler est plus précise que la méthode de calcul de la vitesse moyenne en réduisant l'erreur à moins de 5 cm/s. La méthode de mesure de vitesse par effet doppler est aussi insensible aux variations atmosphériques [T.Chalko, 2009]. Cependant cette précision est variable, elle dépend du nombre de satellites utilisés et leur emplacement à l'horizon. Pour palier cette imperfection, des méthodes de calcul utilisant l'effet doppler ont été mises en place sauf que ces méthodes restent propriétaires [T.Chalko, 2007].

Pour ce qui est de la précision concernant la position, il existe plusieurs techniques et services qui peuvent diminuer la marge d'erreur sur la position tels que les décrivent les sous sections suivantes.

2.1.1.1.1 GPS différentiel (DGPS)

Le DGPS (Differential GPS) permet l'utilisation des stations de références terrestres fixes (voir GCC dans la figure 2.7) qui calculent les erreurs de positionnement des satellites. Les données de correction sont transmises aux équipements GPS à la portée des antennes de ces stations de références [L.Denis, 2000]. Les deux principales causes d'erreurs sont le SA (selective availability) et la dégradation du signal dans la couche ionosphérique. Le SA est une dégradation volontaire du signal faite par le DoD pour mener l'erreur jusqu'à 100 m pour que l'ennemi ne puisse se servir correctement du signal [PNT-SA, 2010]. Actuellement, nous disposons de plusieurs systèmes d'amélioration du GPS se dérivant du DGPS qui ont fait que le SA n'a plus aucun effet sur la dégradation volontaire du signal GPS. La politique américaine concernant ce sujet a été changé en désactivant la SA en mai 2000 [PNT-SA, 2010] pour un usage public du signal GPS. La figure 2.6 compare la précision des technologies comme le radar avec le DGPS.

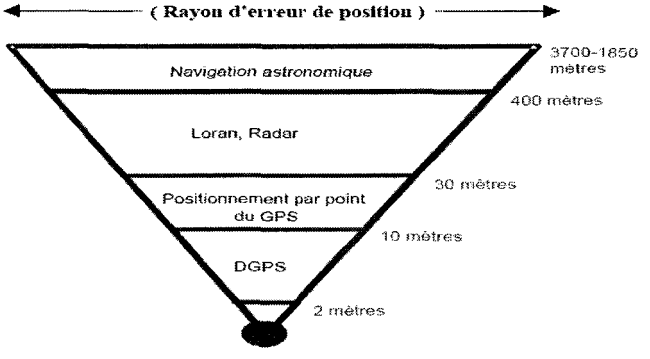

Figure 2.6 : Précision des systèmes de détection de la position

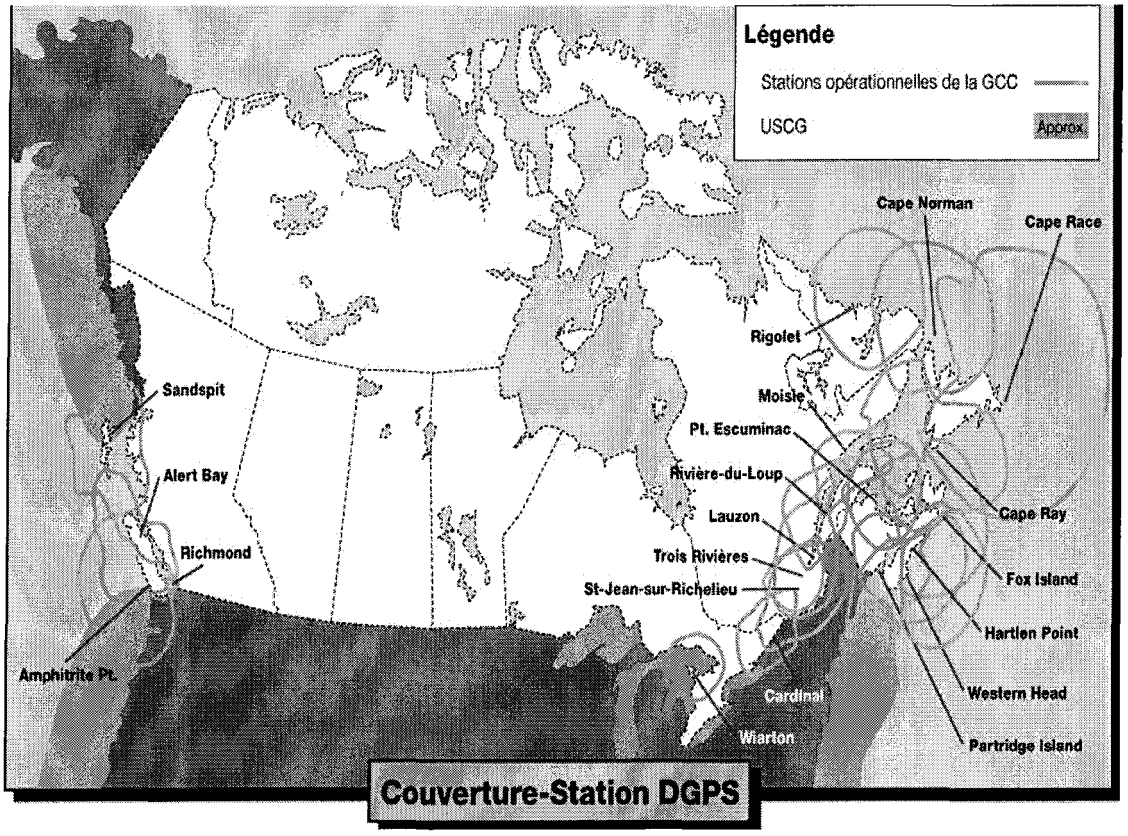

Figure 2.7 : Couverture territoriale du systeme DGPS au Canada [L.Denis, 2000]

Légende :

GCC : Canadian Coastal Guard (Garde-côte Canada)

USCG: US Coast Guard (Garde-côte EU)

2.1.1.1.2 WAAS (wide Area Augmentation System)

À l'instar du DGPS, le WAAS est une technique différentielle. Il consiste en trois satellites géostationnaires et vingt-cinq stations au sol (WRS : Wide area Reference Stations), il a la capacité de ramener la précision à 3 mètres en horizontale et verticale [WAAS, 2010].

Les stations WRS collectent les données recueillies en analysant les satellites GPS. Les données traitées sont envoyées à deux stations principales WMS (Wide area Master Stations) situées sur les côtes Ouest et Est aux États Unis. Les WMS, à leur tour, calculent les corrections d'horloge des satellites GPS ainsi que l'intégrité des informations recueillies. Une fois les erreurs des satellites calculées, les WMS envoient les données de correction aux satellites géostationnaires WAAS. Ces derniers diffusent les données de correction depuis le ciel en couvrant une large zone territoriale ce qui est un grand avantage quant aux stations terrestres DGPS qui ont une portée limitée. Ceci dit, les récepteurs GPS compatibles WAAS peuvent faire les corrections requises pour une plus nette précision. Si l'information sur l'intégrité des données s'avère au dessous du seuil toléré, le WAAS est désactivé pour que le signal soit traité seulement avec le signal GPS de base [WAAS, 2010].

L'avantage de cette technique est qu'elle ne nécessite pas de matériel supplémentaire, mais seulement un équipement pouvant lire le signal L1 (1575,42 MHz) émis depuis les trois satellites géostationnaires WAAS. Malheureusement, le signal WAAS n'est utilisable qu'en Amérique du nord, là où les stations existent pour le moment, car même si le signal est capté ailleurs, il n'y a pas de stations de référence qui transmettent des données relatives à cette région. Cependant, il a été pensé pour l'Europe de mettre un signal DGPS via le système EGNOS (voir section 2.2.1.3) qui sera entièrement opérationnel avec le système satellitaire GALILEO (voir section 2.2.1.2).

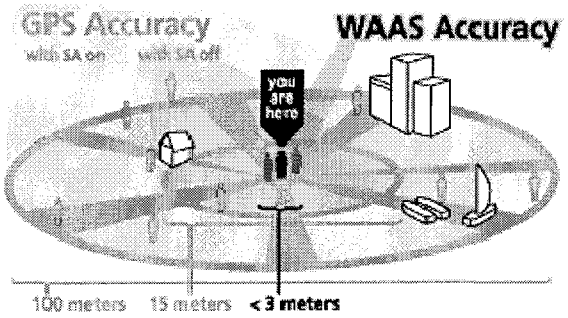

Figure 2.8 : Précision du WAAS contre le GPS seul [WAAS, 2010]

La figure 2.8 montre la précision du système GPS avec SA activé qui est volontairement étendue a 100 m. Elle donne aussi une idée sur le signal GPS sans SA activée donnant la précision théorique du GPS qui est de 15m et, finalement, une précision de moins de 3 mètres pour le WAAS qui bénéficie des corrections recueillies par les stations terrestres fixes retransmises par les satellites géostationnaires WAAS.

2.1.1.1.3 AGPS (Assisted GPS)

Cette technique sert de support pour les récepteurs GPS. Les équipements GPS standard, ne peuvent donner la position exacte dans un délai normal lorsque le signal est affaibli. Le système A-GPS consiste en un module complémentaire greffé à l'appareil mobile. Ce module va se connecter sur un serveur A-GPS pour i) récupérer les données de correction des erreurs comme énoncé dans les points i, ii et iii dans la section 2.2.1.1 et ii) assister l'équipement GPS pour se fixer rapidement sur un satellite en réduisant le TTFF (Time To First Fix) en cas de faiblesse du signal du satellite GPS, comme expliqué plus tard dans la présente section. Une puce A-GPS est souvent intégrée à l'équipement localisable, tels que des téléphones cellulaires qui offrent la connectivité requise (réseaux

cellulaires GPRS) pour avoir les données d'assistance et de correction [A-GPS, 2010] [GPSBase, 2010]. Le service A-GPS est offert en mode connecté ou non connecté.

Le mode connecté :

En se basant sur une connectivité GSM-GPRS, une connexion est établie avec la cellule où se trouve le cellulaire muni d'une puce A-GPS pour acheminer la requête d'assistance au serveur A-GPS et recevoir ses données de correction. Le serveur AGPS dispose déjà des informations sur l'état des satellites et a déjà calculé les facteurs d'erreurs et créé les corrections nécessaires en temps réel prêtes à être délivrées aux appareils mobiles distants [AGPS-intro, 2008]. Cette solution engendre des frais de connexion et de téléchargement de données puisqu'elle se fait par le biais du réseau cellulaire. Mais l'avantage est d'avoir une information fraiche et un court temps d'acquisition des satellites qui est de l'ordre de plusieurs minutes si le signal est faible à quelques secondes prés puisque l'orbite des satellites sur lesquels devrait se fixer le GPS est connu ou plutôt donné d'où le terme assistance. Des compagnies comme Gtop offrent des solutions en mode non connecté pour aider les GPS à se localiser. Gtop offre des données qui expirent dans 7 jours pour garder une précision optimale sur les orbites. Au cas où le GPS est allumé chaque jour, les données A-GPS peuvent être téléchargées directement des satellites (un GPS allumé depuis longtemps s'est déjà fixé sur des satellites et n'a donc pas besoin de l'AGPS pour se fixer mais les données AGPS pourraient être utiles ultérieurement quand il change de position et donc de zone couverte)

Le mode non connecté :

Ce mode consiste en une solution moins onéreuse mais offre des données moins fraiches que pour le mode connecté. La connexion Internet sur le serveur A-GPS est établie à travers un réseau sans-fil wifi ou via le réseau TCP/IP, etc. pour mettre à jour la base de données AGPS de l'équipement GPS. Pour une utilisation extérieure où l'on ne dispose pas d'une connexion à internet par câble ou par réseau wifi, les données de correction deviennent obsolètes faute de mise à jours, et par conséquent inutilisables car elles peuvent augmenter l'erreur de positionnement [AGPS-intro, 2008].

Pour résumer les avantages de l'A-GPS, celui-ci permet : i) de réduire le temps d'acquisition des données de positionnement qu'on appelle aussi TTFF (Time To First Fix) en rendant l'équipement GPS opérationnel même dans les zones à faible signal GPS, ii) de corriger les erreurs de positionnement en fournissant à l'équipement GPS des données de correction, iii) il est aussi possible que le serveur A-GPS calcule la position de l'équipement géo-localisé (voir figure 2.9), ceci permet aux équipements tels que des téléphones cellulaires ne disposant pas d'une puce aux fonctionnalités GPS complètes, de trouver leurs positions. La possibilité du calcul de la position à distance par le serveur A-GPS permet la miniaturisation et la réduction du coût de fabrication de tels dispositifs.

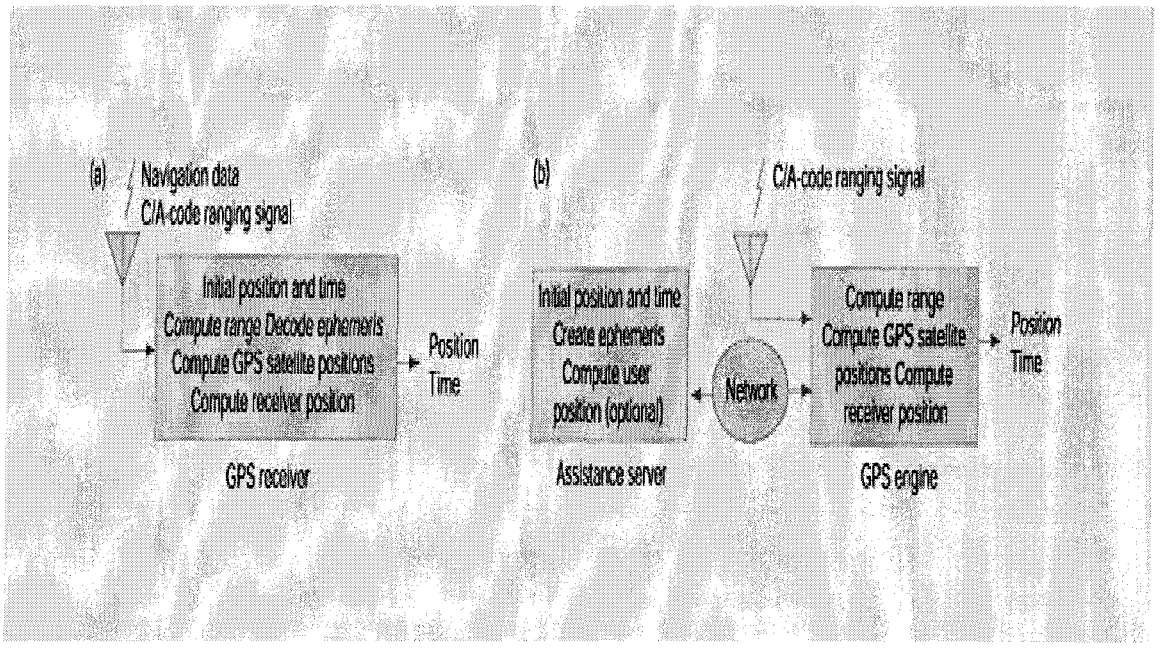

Figure 2.9 : *Mécanisme d'utilisation du service A-GPS*

Comparé au système WAAS, l'AGPS offre plus d'avantages. La comparaison faite dans la figure 2.9 montre le rôle important de l'A-GPS en réduisant le TTFF et en offrant des traitements complémentaires. Sur l'équipement (a) muni seulement d'une puce GPS, cette dernière fait tous les calculs pour déterminer la position alors que dans l'équipement (b) qui utilise le service A-GPS, le serveur A-GPS a déjà calculé le temps initial pour être synchrone avec l'horloge ultra précise du satellite et a créé l'éphéméride qui représente la position exacte de chaque satellite dans la constellation GPS. Le serveur A-GPS peut même dispenser le dispositif (b) du calcul de la position en exécutant cette tâche. Un autre avantage de l'A-GPS par rapport aux autres techniques DGPS est qu'il peut être disponible en tout temps alors que les satellites WAAS peuvent être non accessibles à cause des

conditions météorologiques peu favorables. Le désavantage de cette solution est qu'elle est payante. En mode connecté, nous payons pour les données envoyées en mode GPRS. En mode non connecté, nous payons pour un abonnement afin de pouvoir télécharger via internet les données de correction depuis le site du fournisseur du service A-GPS.

2.1.1.1.4 eGPS (enhanced GPS)

L'eGPS ou GPS étendu est une technologie issue de la coopération entre Motorola et le CSR (Cambridge Positioning Systems). Elle est destinée à servir sur des téléphones mobiles afin d'accélérer le processus d'acquisition de satellites, de réduire le coût d'implémentation et le temps de calcul requis [CSR, 2010]. La puce eGPS utilise les données du réseau cellulaire (cell ID) pour accélérer le traitement des positions dans des endroits à faible signal voir même zéro signal GPS. Le CSR espère offrir un coût d'intégration d'eGPS à moins de 1$ par unité [CSR-MOTOROLA, 2008].

2.1.1.2 Système GALILEO

À l'inverse du GPS américain et du GLONASS russe, le système GALILEO est développé pour un usage non militaire par l'union européenne pour une couverture globale grâce à 30 satellites (27 opérationnels et 3 en réserve). Ce système est très compétitif au GPS puisqu'il se base sur la même technologie que le GPS mais offre une bien meilleure précision qui va jusqu'à 1 mètre pour le service commercial qui peut être ramené au centimètre en utilisant les corrections des stations de base terrestres avec le système EGNOS déjà déployé [GALILEO, 2009]. Ce système est non seulement plus précis que les

autres systèmes existants mais il est aussi très fiable. Il garanti la validité des données de localisation en intégrant un signal d'intégrité du message. Ce signal d'intégrité permet d'informer l'utilisateur d'une quelconque erreur dans les données reçues.

GALILEO peut aussi être un complément pour le GPS et GLONASS, puisqu'il utilise le même type de signal BOC (Binary Offset Carrier) et les mêmes bandes de fréquence L1, L2, etc. (E1, E2, E5, etc. pour GALILEO). Cet avantage permet d'utiliser un même équipement matériel et que seules les données transportées diffèrent d'un système à un autre. L'équipement pouvant lire les deux types de signaux (GPS et GALILEO), peut alors traiter chaque donnée à part et bénéficier d'une double précision [GALILEO-GPS, 2009] [GALILEO-ESA, 2010]. Cette compatibilité permet à un même équipement de basculer d'un système à un autre suivant la couverture. Par exemple, une zone en Europe moins couverte par le GPS amènera l'équipement de géo-localisation à utiliser exclusivement GALILEO. Le système est toujours en phase de test mais l'infrastructure satellitaire est en cours de déploiement et les tests sont menés avec le système EGNOS, le prédécesseur de GALILEO [GALILEO-ESA, 2010].

Déjà deux satellites expérimentaux sont en orbite, GIOVE-A et GIOVE-B. Ces satellites ont réussi à vérifier les technologies critiques au système GALILEO [GALILEO-ESA, 2010]. Ce dernier sera complètement opérationnel en l'an 2014 [GALILEO-EC, 2010].

2.1.1.3 Système EGNOS (European Geostationary Navigation Overlay Service)

C'est le système prés-GALILEO et, à l'instar du WAAS (voir section 2.2.1.1.2), le service EGNOS se base sur des stations terrestres, déployées dans plusieurs pays européens. Ces stations permettent de calculer les erreurs de positionnement et d'apporter des corrections via des satellites géostationnaires EGNOS [EGNOS, 2010]. Ce système de correction utilise les deux systèmes déjà existants, GPS et GLONASS, afin d'aboutir à une meilleure précision et d'offrir plus de disponibilité à l'utilisateur. Il permet une localisation de moins de 5 mètres et fournit des informations sur l'intégrité des données reçues pour assurer la fiabilité des données de positionnement [EGNOS, 2010].

Le système a commencé la phase de test dans le premier semestre de 2005. Il est devenu complètement opérationnel en fin 2006 mais ce n'est qu'en 2007 qu'on a pu apporter des mises à niveau pour garantir les services SoL (Safety-of Life) pour une couverture continue et fiable en précision. Le signal diffusé par les trois satellites géostationnaires a la même structure que le signal GPS. Il est émis sur la bande L1 (1575.42 Mhz) résultant en une possibilité d'interopérabilité entre GPS, GALILEO et GLONASS [EGNOS-FAQ, 2006].

2.1.1.4 Système MEOSAR (Medium Earth Orbit Search And Rescue)

Le système MEOSAR est conçu pour les opérations de recherche et de sauvetage grâce à des équipements (radiobalises) au sol et à des capteurs qui constituent un second chargement aux satellites qu'on appelle SAR. Ceux-ci se composent d'un récepteur d'ondes à 406Mhz et d'un processeur pour démoduler le signal d'alerte. Les équipements

au sol envoient un signal d'alerte sur la bande standard de 406Mhz (accord entre États-Unis, Union Européenne et Russie) vers des satellites munis de récepteurs SAR. Les satellites relayent le signal de détresse géo-localisé vers des stations terrestres LUT (Local User Terminal). Les données reçues par les LUT sont transférées vers des centres de traitement qu'on appelle MCC (Mission Control Centers) et, finalement, les données traitées sont aiguillées vers des centres de coordination de sauvetage RCC (Rescue Coordination Centers) qui alertent les services de sauvetage et leur fournissent la position de la balise de détresse activée [Cospas-Sarsat, 2009].

Le système GALILEO sera équipé de récepteurs SAR pour permettre ce genre d'opérations dans les quatre coins du monde. La couverture estimée pour GALILEO est de l'ordre de 95-97% de la surface terrestre [EGNOS-FAQ, 2006] et il est estimé avec 90% de probabilité qu'au moins 4 satellites seront visibles pour n'importe quel utilisateur à n'importe quel endroit. Pour ce qui est du GPS et du GLONASS, la mise en place de ce nouveau système dépend de la mise à niveau de leur système satellitaire.

Le traitement des signaux se fera comme pour la navigation en GPS et autres, en triangulant le signal reçu de plusieurs satellites par l'équipement au sol. Dans la deuxième étape, l'équipement utilisateur au sol utilise un transmetteur pour envoyer le signal de détresse vers les satellites dans le sens ascendant [Cospas-Sarsat, 2009].

2.1.1.5 Système GLONASS (GLObal'naya Navigatsionnaya Sputnikovaya Sistema)

GLONASS (Système GLObal de Navigation par Satellite) a été conçu par la Russie lors de la guerre froide pour concurrencer le GPS américain. Il se base sur le système géodésique (calcul des distances spatiales) russe PZ-90.02 conforme au système géodésique mondial WGS84 utilisé par le GPS après une adaptation en septembre 2007 permettant une interopérabilité avec le système GPS. À l'inverse du GPS, les satellites diffusent le même code mais avec des fréquences différentes de chaque satellite. Les satellites utilisent 25 canaux séparés avec un intervalle de 0.5625 MHz en deux plages de fréquence, 1602.5625-1615.5 MHz et 1240-1260 MHz [Spaceandtech, 2001].

Le système compte 23 satellites en 2010, sauf que la durée de vie des satellites est plus limitée par rapport aux satellites GPS [GLONASS, 2010]. En décembre 2010, le système pourra atteindre une couverture globale de 99.9% avec 24 satellites opérationnels. La modernisation en cours des satellites permettra l'interopérabilité avec GALILEO et GPS, ainsi que l'augmentation de la précision avec la nouvelle génération des satellites GLONASS-K[V.Glotov, 2009]. Le programme GLONASS qui s'étendra jusqu'en 2020, vise à atteindre des performances comparables au GPS et au GALILEO en 2011 [V.Glotov, 2009].

2.1.2 Techniques de positionnement par réseaux de détecteurs WSN

À l'inverse des systèmes de position par satellites, les WSN (Wireless Sensor Networks) utilisent un réseau de détecteurs pour localiser un objet à leur portée. Cette

approche de localisation d'objets se divise en deux catégories en terme de précision dite aussi granularité. La catégorie moins précise Coarse-grained et la plus précise Fine-grained [R.Tesoriero & al,. 2008].

i) **Coarse-grained** : Dans cette catégorie, nous trouvons les RFID (Radio Fréquence Identification) qui utilisent des étiquettes à faible portée (seulement quelques mètres). Pour couvrir une zone de seulement une centaine de mètres de diamètre, nous devons installer une dizaine de lecteurs RFID et une centaine pour couvrir une petite ville. Cette technique permet la localisation en détectant la proximité de l'étiquette vis-à-vis d'un lecteur RFID. La précision dans ce cas est de l'ordre de la largeur de la zone de couverture du lecteur [Wilson & al., 2007].

ii) **Fine-grained** : Les systèmes appartenant à cette catégorie sont plus précis que dans la catégorie précédente. On y utilise plusieurs techniques, en particuliers : a) RSS (Radio Signal Strenght) qui fonctionne comme un radar. Plus la force du signal est forte, plus l'émetteur est à proximité [Wilson & al., 2007], et b) TDoA (Time Difference on Arrival) qui est une technique encore plus précise et qui consiste à envoyer notre signal radio en même temps qu'une onde ultrason. En comparant le temps d'arrivée des deux ondes nous pouvons déterminer la distance de l'émetteur (objet à localiser) [Bischoff & al., 2006]. Comme pour la première catégorie, le nombre des antennes à utiliser est proportionnel à la zone à couvrir.

L'approche WSN requiert la mise en place d'un grand réseau de détecteurs à portée réduite. Ceci pour assurer une couverture étendue dans un grand espace comme, le cas

d'une ville, étant donné que les objets mobiles utilisés dans cette approche ne font qu'émettre un signal pour être localisés par les détecteurs. Par conséquent, tous les calculs de position se font dans l'infrastructure du réseau de détecteurs (par exemple un serveur central pour la collecte des données et le calcul des positions de tous les objets mobiles du réseau WSN) et non au niveau des équipements mobiles. Cette charge de travail peut vite devenir un goulot d'étranglement pour le système quand il doit gérer plusieurs objets simultanément. Par contre, le GPS est matérialisé par une infrastructure publique accessible n'importe où dans le monde. Le calcul des positions concernant le GPS se fait sur les objets mobiles eux-mêmes. Ainsi, le système GPS n'a pas les limites de calcul des positions simultanées ni un problème de couverture comme pour le WSN.

2.2 Les moyens de communication

Dans la section 2.1, nous avons vu les différentes techniques et approches pour localiser des objets. Comme pour le GPS, l'équipement de géo-localisation calcule sa position en local. Cependant, cette information, dans le cas d'application de suivi ou de gestion d'objets mobiles, a besoin d'être envoyée vers un gestionnaire ou un serveur qui pourra la sauvegarder ou aider d'autres usagers à retrouver la position d'une personne malade, d'une voiture ou d'un cargo de marchandises de façon distante et en temps réel. Pour arriver à cette fin, nous devons utiliser un moyen de communication sans-fil (puisque les objets géo-localisables sont mobiles) pour transmettre les données de localisation à distance. Les sous-sections suivantes décrivent les moyens actuels de communication sans-fil.

2.2.1 GSM/GPRS

Le GPRS (General Packet Radio Service) est une norme pour la téléphonie mobile basée sur le GSM (Global System for Mobile communication) qui permet d'utiliser ce réseau sans fil pour transmettre des données en mode paquet. Le débit de transmission de données se situe dans la plage de 20-50Kb/s [GPRS, 2004], selon la disponibilité des ressources. Cette technologie bénéficie de la grande disponibilité planétaire du réseau GSM et de sa grande couverture de larges zones urbaines et leurs périphéries aussi bien que dans le milieu rural. Cependant, le réseau GSM reste privé et appartient aux compagnies de télécommunication et donc l'usage est facturable à la quantité des données envoyées et reçues.

2.2.2 Radio sans fil

La transmission radio fut mise en œuvre au départ pour assurer une communication point à point sur de longues distances (faisceaux hertziens, liaisons satellites géostationnaires) entre éléments fixes du réseau.

Comme les appareils mobiles se trouvent souvent dans un environnement urbain, les bâtiments obstruent la vue des antennes des équipements radio qui se situent au niveau du sol. Le principe de mobilité introduit des techniques pour palier le problème de la non visibilité de l'équipement radio-mobile par la station de base terrestre [X.Lagrange, 2000]. Les ondes ne vont plus se propager en visibilité seulement mais l'on prendra en

considération les ondes se réfléchissant sur tous types d'obstacles (immeubles, toits des maisons, arbres ...).

La communication dans un milieu urbain via des ondes radio est véhiculée par des signaux radio multi-trajets. Les ondes les plus utilisées appartiennent à la bande de fréquence UHF (300 MHz-3GHz) pour assurer une communication mobile en milieu urbain puisque ce type d'ondes permet de traverser des obstacles avec une perte de puissance du signal tolérable, dépendamment du matériau traversé (pertes selon le matériau utilisé : Bois 4dB, Béton 10dB)[COS, 99].

Pour réaliser une communication radio, nous avons besoin de modems radio et d'antennes radio pour augmenter le gain du signal affaibli par le trajet. Il existe plusieurs types d'antennes, mais celles qui nous intéressent dans cette recherche, sont les antennes omnidirectionnelles de type Whip (antennes qui émettent dans toutes les directions et qu'on n'a pas besoin d'orienter). Nous retrouvons ces antennes dans les téléphones cellulaires. Ces antennes permettent un gain de 2dBi et la longueur d'antenne est de 6.35 à 12.7 cm [mes799, 2007]. La taille de l'antenne lui permet d'être intégrée dans des appareils mobiles de petites tailles faciles à porter par des personnes (PDA, téléphones cellulaires, GPS personnels, etc.).

Les antennes colinéaires [mes799, 2007], à l'instar des Whip, sont aussi omnidirectionnelles mais permettent un plus haut gain (4-10dBi). Ce type d'antennes est constitué d'un empilement de plusieurs antennes, plus le nombre d'antennes est grand plus

le gain est élevé. À cause du principe d'empilement, ces antennes sont de plus grandes tailles que les Whip et sont plus adaptées pour les véhicules (Voitures de police, ambulances, etc.).

2.2.3 Technologie RFID

Le RFID (Radio Frequency IDentification) est une technologie permettant l'échange de données à distance via des ondes radio. Les deux composantes du système sont :

a) RFID tag (étiquette) : c'est un transpondeur de très petite taille contenant une antenne pour recevoir et émettre des radio fréquences du lecteur RFID. Ces étiquettes se dérivent en deux types : i) Les RFID passifs, n'ont pas besoin de source d'alimentation pour fonctionner et utilisent l'énergie de l'onde émise du lecteur RFID à proximité pour s'alimenter et communiquer avec ce dernier. L'avantage est qu'ils n'utilisent pas de sources d'alimentation telles que des piles ou des batteries. Grâce à cet avantage nous pouvons atteindre des degrés de miniaturisation et d'intégration très importants. Le désavantage de ce type d'étiquettes est que la puissance d'émission est très réduite et donc ces lecteurs ne peuvent bénéficier d'une grande portée pour la transmission des données. ii) Le second type est dit actif, ce sont des étiquettes qui utilisent une source d'alimentation telles que des batteries pour s'alimenter. À la différence des étiquettes passives, ils ont assez de puissance pour émettre sur un plus grand rayon.

b) Lecteur RFID : le lecteur RFID est un dispositif qui permet d'interroger à distance les étiquettes RFID. Il émet des ondes radio à une fréquence déterminée et les

étiquettes, répondant à la même fréquence, s'activent pour renvoyer le signal vers le lecteur avec les données qu'elles ont en mémoire.

La technologie se base sur deux approches pour transférer l'énergie du lecteur vers l'étiquette [R.Want, 2006], l'induction magnétique et la capture d'ondes électromagnétiques (EM). Les deux modèles se basent sur les propriétés RF (radio fréquence) des antennes, « near field » et « far field ». Les antennes « near field » sont utilisées dans les étiquettes passives et fonctionnent avec des fréquences de moins de 100MHz et sont limitées en terme de portée. Les antennes « far field » concernent les RFID actifs et opèrent dans les bandes UHF (Ultra High Frequency) comme la bande 2.4 GHz. Non seulement ces antennes peuvent émettre plus d'énergie et donc ont plus de portée, mais le débit de données est plus élevé que pour les « near field » qui s'alimentent par inductance [R.Want, 2006].

La portée des étiquettes RFID passives peut atteindre 2 mètres [GAO, 2010], celle des actives atteint 35 pieds dans un milieu qui comporte beaucoup d'obstacles tels que des murs. Cette portée est plus élevée quand l'étiquette est en visibilité directe avec le lecteur RFID (100 pieds) [MSSI, 2005]. Les RFID actives peuvent être programmées pour émettre en permanence, ce qui permet de les localiser en tout temps.

Les RTLS (Real Time Location System) sont des systèmes électroniques utilisés pour localiser les petits équipements électroniques tels que des appareils portatifs. La plupart d'entre eux opèrent sur une plage de fréquences publiques 300-433.92 MHz et 2.45 GHz [GAO, 2010].

Le « Sapphire DART » de la compagnie « MSSI » fonctionne sur une bande radio ultra-large 5.925-7.250 GHz. Il peut atteindre une portée de 200 mètres en visibilité directe et 50 mètres en milieu dense en obstacles. L'étiquette RFID mesure 0.5 x 1.0x 0.25 pouces. La batterie dure 4.5 ans et la précision pour la localisation est de 10 à 30 cm [MSSI, 2005].

2.2.4 Wifi et WiMAX

Le Wi-Fi (Wireless Fidelity) est une technologie qui permet d'interconnecter un ensemble d'équipements sans fil régi par l'ensemble de normes 802.11 de l'IEEE. Grâce au Wi-Fi, nous sommes capables de créer des réseaux locaux (WLAN) à hauts débits qui interconnectent des ordinateurs de bureaux, des équipements sans fil, des PDA etc. Le système Wi-Fi offre une plateforme sécuritaire munie de techniques de cryptage telles que les clés WEP et WPA. Seuls les équipements autorisés peuvent accéder et échanger des données avec les réseaux d'équipement Wi-Fi [WiFiorg, 2010].

Cette technologie offre des débits très intéressants pour un LAN sans fil, allant de 11 Mbits/s pour la norme 802.11b et 54 Mbits/s pour la 802.11a et 802.11g pour atteindre un maximum théorique de 600 Mbits/s pour la norme 802.11n. La couverture en milieu fermé est d'une dizaine de mètres (20-50 mètres) et de centaines de mètres en endroits ouverts. Avec l'utilisation d'antennes directionnelles dans un espace ouvert, nous pouvons atteindre plusieurs centaines de mètres de couverture [IEEE-std, 2008] [WiFiorg, 2010]. Le Wi-Fi permet de fonctionner en deux modes :

a) Mode infrastructure : les périphériques sans fil se connectent tous sur un point central qu'on appel point d'accès ou AP (Access Point). L'AP fait office de routeur et de point de connexion vers d'autres réseaux (LAN, WAN). Cette configuration est idéale pour des périphériques qui seront toujours à la portée de l'AP (ex : ordinateurs fixes, matériels au sein d'un complexe industriel etc.) et qui ne font pas objet de mobilité permanente. La configuration et la gestion des ressources et des canaux radio alloués se font de façon *automatique* dans ce mode via l'AP.

b) Mode ad hoc : dans ce mode, chaque équipement configuré en ad hoc constitue une maille (nœud d'un réseau maillé) et permet de communiquer d'égal à égal avec les autres équipements en mode ad hoc, sans passer par un point d'accès. Ceci implique que l'on doit configurer manuellement chaque équipement pour lui attribuer un canal radio et le même nom de réseau (SSID) avec clé de cryptage. Comme ce mode constitue un réseau maillé, l'avantage est que l'on peut avoir un système mobile, vu que l'on ne se soucie plus d'être sous la couverture d'un point d'accès mais tout simplement être à la portée du signal de l'un des équipements configurés en ad hoc, faisant déjà partie du réseau (chaque équipement en mode ad hoc fait office de routeur et de répéteur).

Le Wi-Fi étant conçu au départ pour des réseaux locaux [normesIEEE, 2008], il ne peut couvrir une zone de grande envergure telle qu'une ville. Une nouvelle norme 802.16 définissant le WiMAX vient étendre la portée du Wi-Fi [EklundMarks, 2002]. Cette

technologie se base sur les fonctionnalités du Wi-Fi et se veut complémentaire à cette technologie. Le WiMAX peut transmettre des données à haut débit via des ondes hertziennes avec une portée qui peut atteindre une dizaine de kilomètres, ceci nous permet de couvrir une grande partie d'une ville ou une ville toute entière, avec seulement quelques antennes comme points d'accès [normesIEEE, 2008]. La dernière norme 802.16m peut atteindre un débit de 1 Gbits/s en mode stationnaire et 100 Mbits/s en très grande vitesse de déplacement.

Cette section avait pour but de présenter les différentes techniques de positionnement ainsi que les différents moyens de communications sans fils susceptibles de transporter les données de positionnement. Le chapitre suivant portera sur l'architecture logicielle qui appuie notre système en utilisant la technologie d'OSGI pour une meilleure intégration, maintenance et une modification facilitée du code de l'application de gestion des objets géo-localisables.

III

OUTIL DE DEVELOPPEMENT OSGI

3.1 Introduction

Actuellement, le développement des applications devient plus coûteux pour les entreprises étant donné la complexité qu'atteignent les logiciels. En effet, nous exigeons de nos applications de gérer plusieurs aspects, d'offrir plusieurs services et fonctionnalités. L'application résultante est donc très complexe en terme de codage, les librairies utilisées dépendent aussi d'autres librairies qui ne sont pas utilisables en intégralité ce qui rend l'application encore plus lourde [OSGI, 2010].

Les applications sont en perpétuel changement de fonctionnalités et de comportements. Leur cycle de vie est de plus en plus court étant donné la grande évolution des marchés, les nouveaux domaines d'application, le changement constant des exigences des clients, etc. Les programmeurs sont maintenant confrontés au problème de la réutilisation du code. Le développement logiciel consiste en l'adaptation de plusieurs fonctionnalités existantes dans un nouvel environnement. Étant donné que plusieurs fonctionnalités de base se répètent, qu'on ai besoin de mettre à jour quelques parties du code sans pour autant changer toutes les parties interdépendantes, il est impératif que les développeurs s'orientent vers la programmation par composants [OSGI, 2010].

Afin de rendre le développement de notre application plus aisé et offrir un environnement très évolutif pour de futurs développements, j'ai opté pour OSGI. Le présent chapitre décrit en détail les éléments essentiels de cet intergiciel.

3.2 L'approche OSGI

OSGI (Open Services Gateway Initiative) est un intergiciel universel basé sur l'usage de composants orientés services, en permettant à chaque composant de masquer son implémentation pour ne communiquer qu'à travers des services [OSGI, 2010]. Il définit les spécifications pour une plateforme s'exécutant dans un environnement Java ce qui lui donne un aspect de portabilité étant donné que Java est très répandue dans plusieurs types de plateformes matérielles (PDA, téléphones cellulaires, serveurs, ordinateurs personnels, etc.). La technologie OSGI permet de développer des applications orientées composants. La plateforme fournit des primitives standardisées pour construire des applications avec de petits composants, réutilisables et collaboratifs [OSGI, 2010]. Cette approche de programmation par composants permet de réutiliser des aspects récurrents avec moins d'effort, que ce soit pour l'implémentation, le test (possibilité de tester module par module) ou le déploiement.

Une plateforme OSGI offre des fonctions qui permettent de changer dynamiquement les périphériques connectés sur plusieurs types de réseaux sans redémarrage. L'OSGI fournit aussi les services nécessaires au couplage de ces périphériques. Il leur permet aussi de se découvrir mutuellement au cas où ces derniers veulent utiliser les fonctionnalités d'un autre périphérique [OSGI, 2010]. Par exemple le périphérique GPS va pouvoir utiliser le service du périphérique radio, s'il ne le trouve pas il utilisera le service du périphérique GSM-GPRS pour l'envoie de sa position, ce qui assure une collaboration entre différents composants du système.

Pour faciliter encore la tâche aux programmeurs, l'alliance OSGI a développée plusieurs interfaces de composants standards. Il s'agit des fonctions communes telles que les serveurs HTTP, configuration, authentification, sécurité, administration des utilisateurs, XML et autres. Les différents composants peuvent être greffés facilement sur le système sous forme de « plug-in ». Les composants peuvent provenir de différents vendeurs. Nous n'aurons qu'à choisir le produit qui nous intéresse, selon les fonctionnalités et les prix proposés par les vendeurs. Ces composants spécifiques à l'environnement OSGI se nomment *Bundles*.

3.2.1 Présentation du Framework OSGI

L'adoption de l'approche OSGI permet de réduire le temps nécessaire au développement et à la commercialisation. Ceci s'explique par l'utilisation de modules préconstruits et pré-testés, facilement intégrés à l'application basée sur ce Framework. Le temps de maintenance est aussi réduit après la commercialisation du produit, la mise à niveau moins complexe. La figure 3.1 montre l'architecture de la plateforme OSGI qui se compose de plusieurs couches :

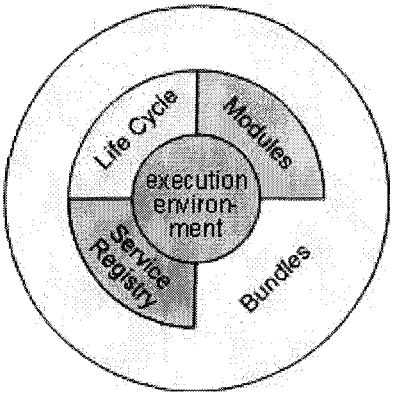

Figure 3.1 : Les différentes couches du Framework OSGI [OSGI, 2010]

Le cœur du Framework OSGI qui permet de gérer les composants ou bundles se compose de 3 couches :

- Couche 0 (Execution environment) : Environnement d'exécution du Framework, conforme aux spécifications de l'environnement d'exécution Java compatible avec les configurations et profils Java tels que J2SE, CDC, CLDC, MIDP, etc. Le Framework OSGI est donc capable d'être déployé sur n'importe quelle plateforme offrant un environnement d'exécution Java, allant des équipements portatifs tels que ceux utilisés dans mon modèle, jusqu'aux plus grands équipements tels que des serveurs (voir chapitre 3).
- Couche 1 (Modules) : Couches des modules qui définissent les politiques de chargement des classes. En Java il y a un seul chemin d'exécution (classpath) qui contient toutes les classes et les ressources. La plateforme OSGI permet d'ajouter la « modularisation » via la couche « Modules » en ajoutant des classes privées pour que chaque module puisse utiliser des ressources distinctes d'un autre module. Cette couche « Modules » permet aussi de gérer les connexions entre les différents modules suivant une architecture de sécurité complètement intégrée. Ce modèle offre des options pour déployer des systèmes fermés ou protégés, avec une gestion choisie par le développeur.
- Couche 2 (Life cycle) : le cycle de vie du bundle (composant) est géré par cette couche. Bien que ce soit la couche des modules qui s'occupe du chargement des

classes d'un bundle, la couche cycle de vie permet d'ajouter des fonctionnalités de façon automatique via une API. Cette API permet donc d'installer un bundle via la commande INSTALL plus le chemin du bundle. La fonction START plus l'ID du bundle installé, pour le lancer. L'arrêt de l'exécution se fait via la fonction STOP plus l'ID du bundle. La désinstallation du bundle se fait via la commande UNINSTALL plus le chemin d'accès du bundle. Cette couche offre un robuste système de gestion des dépendances pour assurer une parfaite exécution des bundles. L'architecture de sécurité utilisée, rend virtuellement impossible de compromettre les opérations du cycle de vie à travers les attaques de virus informatiques.

- Couche 3 (Service registry) : la couche registre de services fournit un modèle coopératif prenant en charge la dynamique des bundles (nouveaux services, services qui n'existent plus). Plusieurs événements sont définis pour gérer les services qui entrent et deviennent disponibles et les services qui sortent et qu'il faut dé-enregistrer. Les services sont de simples objets Java qui peuvent représenter un serveur (ex : http), un téléphone, un appareil Bluetooth, etc. Un modèle de sécurité de services permet de sécuriser la communication entre les bundles.

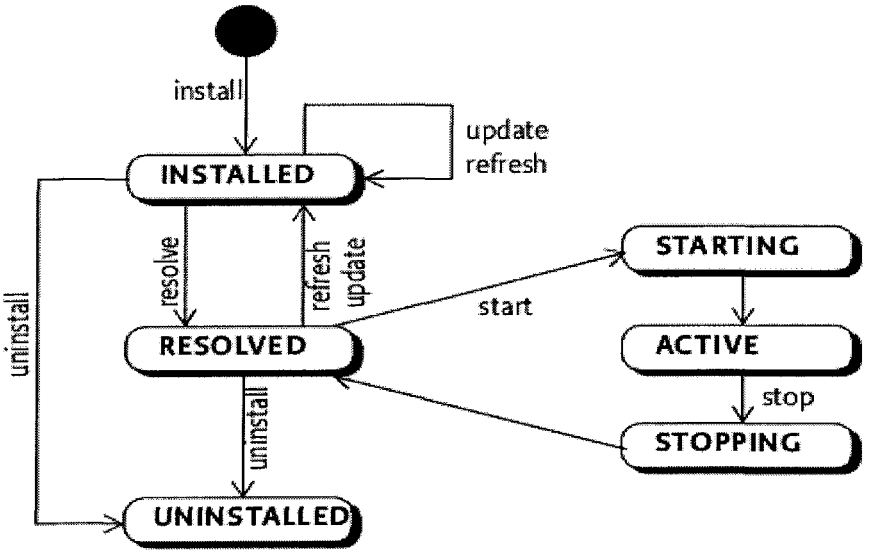

Figure 3.2 : : Cycle de vie d'un bundle [OSGI, 2010]

La figure 3.2 illustre les différents états de gestion des *bundles* par le conteneur ainsi que leurs enchaînements possibles. La plateforme OSGI, tel que knopflerfish utilisé dans le présent travail (voir chapitre 5) offre une API standard afin de gérer le cycle de vie d'un bundle. Le tableau 3.1 décrit les états possibles d'un bundle :

Etat	Descriptif
Installé (*installed*)	État dans lequel se trouve un *bundle* juste après avoir été installé, la résolution des dépendances n'ayant pas encore été réalisée.
Résolu (*resolved*)	État dans lequel se trouve un *bundle* après avoir été installé, la résolution des dépendances ayant juste été réalisée.

En train de démarrer (*starting*)	État dans lequel se trouve un *bundle* lorsqu'il est en train d'être démarré. Cet état correspond à un état transitoire entre les événements *Résolu* et *Actif.*
Actif (*active*)	État dans lequel se trouve un *bundle* lorsqu'il a été démarré avec succès. Le *bundle* ainsi que les services qu'il expose sont disponibles pour les autres *bundles.*
En train de s'arrêter (*stopping*)	État dans lequel se trouve un *bundle* lorsqu'il est en train d'être arrêté. Cet état correspond à un état transitoire entre les événements *Actif* et *Résolu.*
Désinstallé (*uninstalled*)	État dans lequel se trouve un *bundle* une fois qu'il a été désinstallé.

TABLEAU 3.1 : Les différents états dans le cycle de vie d'un bundle OSGI

Félix [FelixApache, 2010] qui est aussi une implémentation des spécifications OSGI par la communauté Apache, offre les commandes suivantes (Tableau 3.2) pour gérer le cycle de vie des bundles :

Commande	**Descriptif**
install	Installation de *bundle(s).*
ps	Affichage de la liste des *bundles* installés.
refresh	Rafraichissement des *packages* des *bundles.*
resolve	Tentative de résolution des *bundles* spécifiés.
services	Affichage de la liste des services enregistrés ou utilisés.
start	Démarrage de *bundle(s).*

stop	Arrêt de *bundle(s)*.
uninstall	Désinstallation de *bundle(s)*.
update	Mise à jour d'un *bundle*.

TABLEAU 3.2 : Les différentes commandes pour gérer les bundles OSGI sous Felix

3.2.2 Bundles et services

Les composants dans l'environnement OSGI sont appelés bundles. Les bundles sont stockés dans des fichiers .jar prêts à être installés via la commande Install. Les informations de déploiement sont spécifiées dans le fichier Manifest.MF dans le répertoire META-INF. Le tableau 3.3 décrit les entêtes du fichier « Manifest.MF » :

En-tête	Descriptif
Bundle-ManifestVersion	Correspond à la version de fichier MANIFEST du *bundle*
Bundle-SymbolicName	Spécifie l'identifiant symbolique du *bundle*
Bundle-Name	Spécifie le nom du *bundle*
Bundle-Version	Spécifie la version du *bundle*
Bundle-DocURL	Permet de préciser l'adresse de la documentation du *bundle*.
Bundle-Category	Spécifie la catégorie du *bundle*.
Import-Package	Spécifie les noms et les versions des *packages* utilisés par le *bundle*.

Export-Package	Spécifie les noms et les versions des *packages* mis à disposition par le *bundle*.
DynamicImport-Package	Spécifie les noms et les versions des *packages* utilisés par le *bundle*. Cet en-tête se différencie de *Import-Package* par le fait qu'il ne soit pas nécessaire que les dépendances soient présentes au démarrage du *bundle*. Il suffit qu'elles le soient au moment de l'exécution.
Bundle-NativeCode	Spécifie la liste des bibliothèques natives présentes dans le *bundle*.
Require-Bundle	Spécifie les identifiants symboliques des *bundles* nécessaires au bon fonctionnement du *bundle*.
Bundle-Activator	Spécifie le nom de la classe dont les traitements sont exécutés lors du démarrage et de l'arrêt du bundle. La classe Activator doit être présente dans le bundle et implémenter l'interface BundleActivator.
Bundle-Classpath	Spécifie le classpath du *bundle*. Par défaut, la valeur implicite correspond aux classes utilisées par ce bundle. et il faut veiller à ne pas l'oublier lorsque des bibliothèques sont spécifiées explicitement.

TABLEAU 3.3 : Description du fichier « Manifest.MF» d'un bundle OSGI

Par exemple, le contenu d'un fichier « Manifest.MF », pour la description du composant «Simple Bundle », peut être décrit comme suit :

Manifest-Version = 1

Bundle-Activator = ca.uqac.osgi.SimpleActivator

Bundle-SymbolicName = simple-bundle

Bundle-Name = Simple Bundle

Bundle-Description = Simple Bundle.

Import-Package = org.osgi.framework;version=1.3

Export-Package = ca.uqac.osgi.services

Bundle-Version = 1.0.1

Bundle-License = http://www.apache.org/licenses/LICENSE-2.0.txt

Dans le fichier « Manifest.MF » qui décrit les paramètres du bundle «Simple Bundle » comme expliqué dans le tableau 3.3, le paramètre « Import-Package » permet de décrire les services dont le bundle a besoin pour pouvoir s'exécuter. Un bundle peut aussi offrir des services à son tour, c'est via le paramètre « Export-Package » qu'un bundle montre à l'intergiciel OSGI quels services il rend accessibles pour d'autres bundles. Nous verrons dans le chapitre 5 les bundles développés pour notre application et leur configuration.

Le chapitre suivant décrit le modèle pour la gestion des objets géo-localisables qui repose sur la plateforme OSGI.

IV

MODÉLE DE GESTION D'OBJETS GÉO-LOCALISABLES

4.1 Introduction

Comme cité précédemment, le présent chapitre traite des aspects devant conduire à la conception d'un modèle de système de positionnement permettant de réaliser les points suivants :

i) d'accroitre la précision des dispositifs GPS via A-GPS (données de correction de la position disponibles sur les serveurs A-GPS privés tel que expliqué dans la section 2.1.1.1.3) en évitant d'utiliser le réseau GSM pour le téléchargement des données A-GPS (voir figure 4.2).

ii) de centraliser les données de positionnement de plusieurs types de dispositifs GPS. Chaque dispositif est utilisé dans un domaine particulier : santé, sécurité routière, surveillance, etc. Ceci permet d'avoir une vision globale du système pour gérer efficacement les différents objets géo-localisables. Par exemple, la connaissance des positions exactes des ambulances va permettre l'envoi rapide d'une unité à proximité d'une personne signalée perdue ou qui risque de se perdre lorsque celle-ci est munie d'un système de localisation portable (voir figure 4.4).

iii) La coopération entre les différents objets géo-localisables. Les unités mobiles à faible portée, ne pouvant atteindre l'antenne radio de notre serveur, peuvent transmettre leurs données de localisation vers un autre objet mobile disposant d'un modem radio longue portée. Cette coopération permet de créer des relais pour acheminer les données vers notre serveur (voir figure 4.3).

Nous avons vu dans le chapitre 2 qu'il existe plusieurs techniques pour permettre la localisation d'un objet. Les techniques coarse-grained comme le RFID, sont moins précises. Les techniques fine-grained sont capables de donner une meilleure précision. Leur inconvénient résulte dans la complexité de leur mise en œuvre et le coût de l'installation d'un réseau de détecteurs. Finalement, le GPS peut offrir une bonne précision, sans pour autant exiger une infrastructure supplémentaire pour l'utilisateur final. L'infrastructure utilisée existe déjà et reste ouverte au public : il s'agit du réseau satellitaire GPS qui couvre tout le globe. Le GPS peut aussi être amélioré via WAAS, EGNOS et A-GPS pour atteindre une meilleure précision. Dans un usage classique, pour acquérir les données GPS à distance, il faudra utiliser une connexion GPRS via le réseau cellulaire GSM. Cette connexion est facturée au volume de données échangées. Ce chapitre va décrire la solution, permettant par le biais d'un modèle réseau et logiciel, de résoudre les problèmes de précision et de coût de communication.

4.2 Modèle de gestion d'objets géo-localisables

Notre solution est d'utiliser le GPS pour les avantages cités précédemment. Pour envoyer les données de localisation dans le sens ascendant (vers mon serveur, voir figure 3.3), nous évitons, dans la mesure du possible, de passer par le réseau GSM. Dans le sens descendant (vers les objets mobiles géo-localisables, voir la même figure 3.3), les données de correction de la position sont diffusées via l'antenne radio longue portée connectée au serveur. Les sous-sections suivantes vont décrire les différentes parties de la solution.

4.2.1 Architecture matérielle du système de localisation

Cette section présente les solutions au niveau de l'infrastructure matérielle (GPS, radio, équipements, etc.) pour résoudre le problème de précision et de communication à moindre coût par rapport au système classique GPS/GPRS/A-GPS.

4.2.1.1 Les types d'objets géo-localisables

Tout d'abord, nous divisons les objets géo-localisables en deux catégories. Elles sont présentées ci-dessous :

a) **Les appareils portatifs** : C'est un ensemble d'équipements portatifs tel que : les téléphones cellulaires, les GPS portatifs et tout autre appareil portable de petite taille comprenant une puce GPS. Étant donné la petite taille de ces dispositifs et pour garder l'aspect portabilité, nous devons les coupler à des modems radio de petite taille pour pouvoir communiquer leurs données de positionnement à distance. L'inconvénient de la taille se traduit par une faible portée du signal radio (800m à 2km).

b) **Les objets de grande taille** : tels que les véhicules, les conteneurs de marchandise, etc. Ces objets nous offrent plus de marge de manœuvre (plus d'espace disponible) quant à l'installation de plus grands modems et antennes radio. Sur un véhicule, nous pouvons installer un modem radio de taille plus conséquente et une antenne à grande sensibilité (ex. sur le capot d'une voiture). La portée devient plus grande et peut aussi atteindre plusieurs dizaines de kilomètres, comme avec le modem radio du serveur.

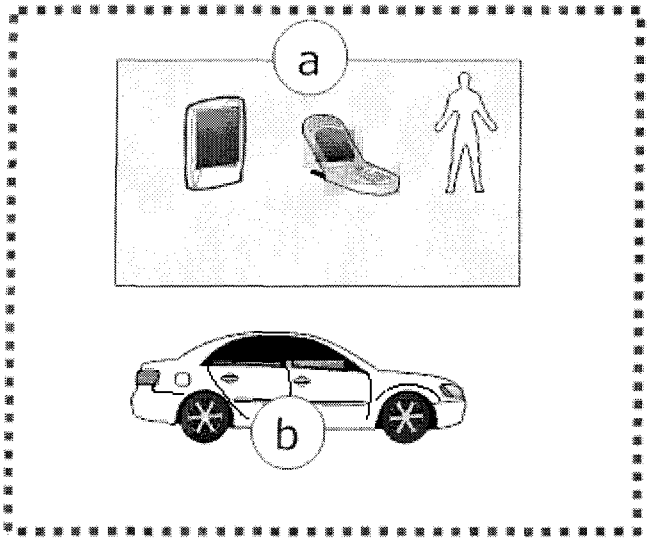

Figure 4.1 : Types d'objets mobiles géo-localisables

4.2.1.2 Téléchargement des données de correction A-GPS

L'utilisation classique de l'A-GPS via GPRS pour acquérir les données de correction n'est pas rentable. Pour pallier le coût de communication, nous allons utiliser un autre canal de transmission des données. Il s'agit d'utiliser des modems et antennes radio pour créer le lien entre le serveur A-GPS et les équipements mobiles qui utilisent les données de correction.

La caractéristique de diffusion des antennes radio va permettre de diffuser à partir d'un seul point (serveur A-GPS) vers plusieurs points (objets mobiles de type « a » et type « b », figure 4.1).

Les serveurs A-GPS sont des structures appartenant à d'autres organismes (gouvernement, compagnie de téléphonie, etc.). Par conséquent, nous ne pouvons pas installer notre antenne radio directement sur ces serveurs de données de correction.

Je propose le modèle suivant qui consiste en l'utilisation d'un intermédiaire entre le serveur A-GPS et les appareils de localisation. Notre serveur central est connecté sur internet et communique avec le serveur A-GPS pour télécharger les données de correction. Une fois les données téléchargées sur le serveur central, ce dernier se connecte sur le port d'un modem radio pour lui envoyer les données de localisation. Le modem radio module les données en ondes radio et diffuse l'information avec une antenne radio à longue portée.

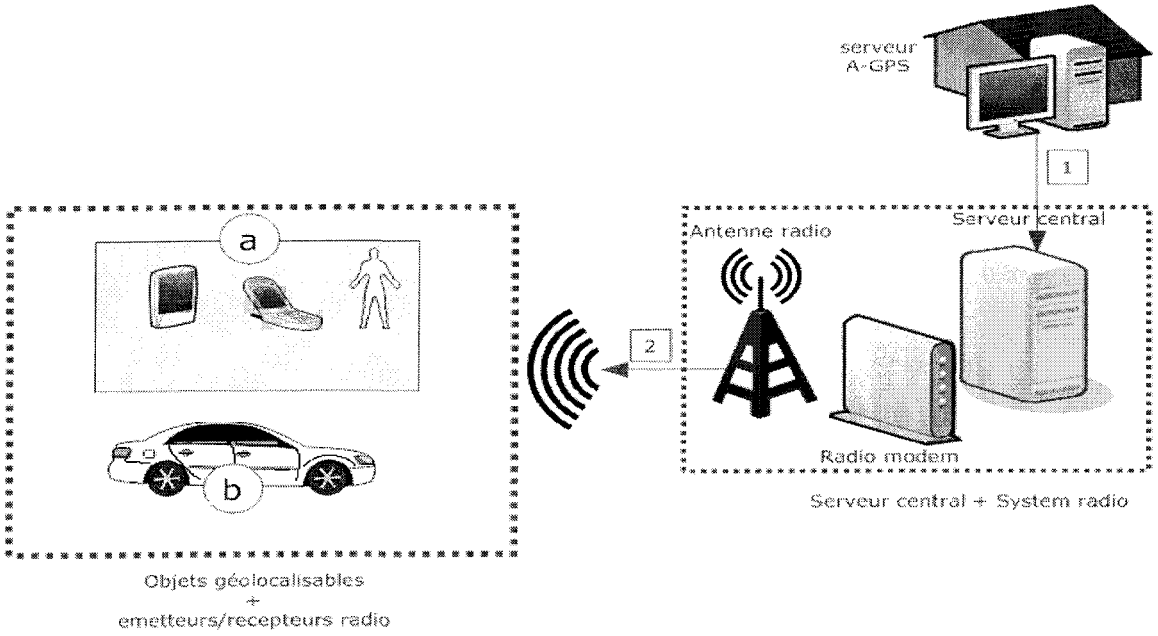

Figure 4.2 : Modèle d'utilisation du service A-GPS via une connexion radio

Dans la figure 4.2, les objets mobiles « a » et « b », utilisent des modules GPS capables de lire les données de correction (ex. format RINEX) pour le post-traitement des positions.

Les données de correction vont être téléchargées via un serveur connecté sur une station de correction terrestre (serveur A-GPS) à travers une connexion Internet. Une antenne sera utilisée via un modem radio connecté sur notre serveur central et pourra diffuser les données de correction sur une dizaine de kilomètres : le modem que j'ai choisi peut atteindre un rayon de transmission de 50 km en terrain ouvert.

4.2.1.3 Modèle coopératif pour transmettre les données vers le serveur central

Nous avons vu dans la section 4.2.1.1 qu'il y a deux catégories d'objets géo-localisables dans notre système. Le cas des grands objets localisables (catégorie « b »), ne pose pas un problème de la portée pour transmettre les coordonnées de localisation ou de recevoir les données de correction depuis notre serveur. Les plus petits objets géo-localisables (catégorie « a ») sont souvent loin de l'antenne de notre serveur. Si le groupe d'objets « a » ne peut atteindre le serveur « c » avec une communication radio directe, il demande à l'objet « b » ayant une plus grande portée de relayer l'information vers « c ». La figure 4.3 décrit ce processus.

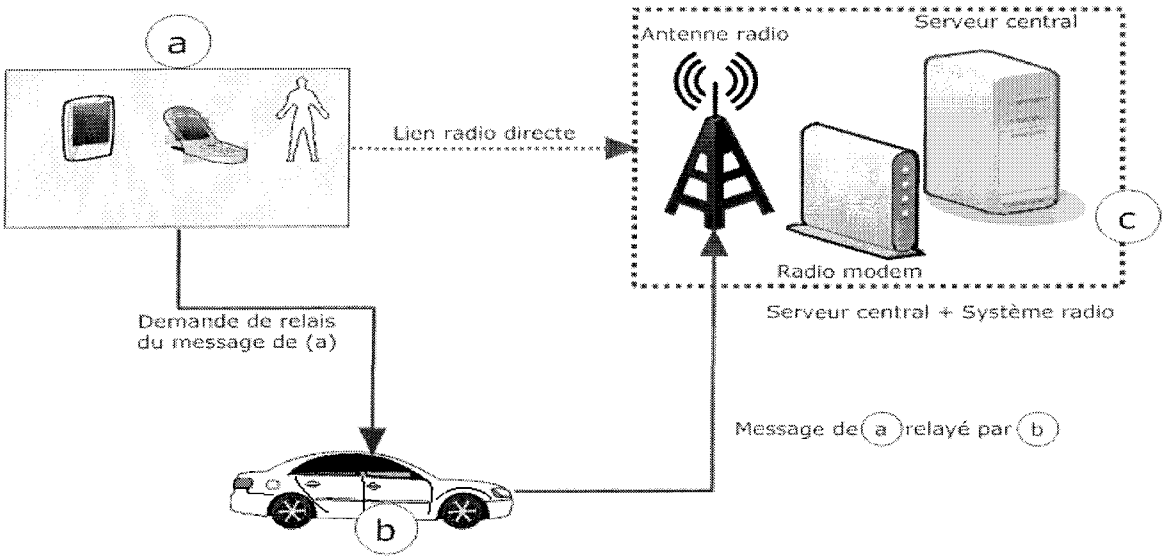

Figure 4.3 : Modèle de coopération entre les objets mobiles via des liens radio

4.2.1.4 Modèle global de l'infrastructure matérielle du système de positionnement

Cette section décrit le modèle global regroupant les solutions citées précédemment. Je propose un modèle logiciel qui permet aux appareils portatifs de coopérer pour relayer les messages radio vers l'antenne connectée au serveur central. Les grands objets géo-localisables disposant de plus de portée peuvent relayer les messages des appareils à faible portée se trouvant à leur proximité.

Dans le cas où un modem à faible portée ne peut atteindre l'antenne du serveur pour une connexion directe et qu'un relayeur n'est pas disponible, nous devons nous assurer que les données de localisation soient transmises par un autre moyen. À l'expiration du délai d'attente de transmission radio directe et indirecte (relais), le système bascule vers le mode GSM-GPRS (voir les points A2 de la figure 4.4). La figure 4.4 montre le modèle global de l'infrastructure matérielle. Elle regroupe les objets mobiles (points 2 et 3), le serveur (point

6), la station A-GPS (point 4) et les modes de connexions (points A1, A2, B1, B2, C1, D, E, F, G) pour assurer l'interconnexion de ces éléments.

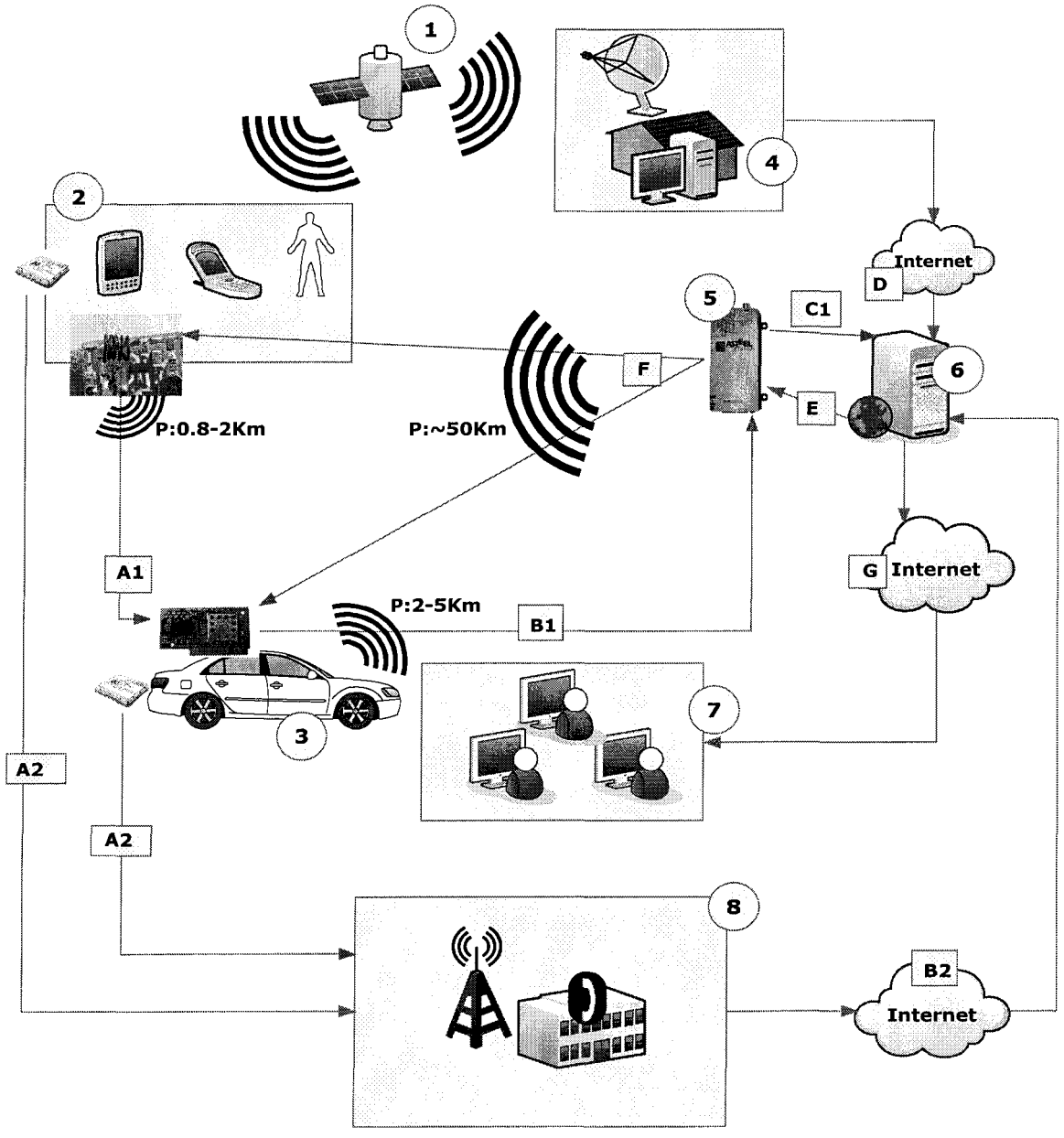

Figure 4.4 : Topologie réseau de notre système pour la communication Radio et GPRS

La topologie réseau de la figure 4.4 est constituée des huit composants et des liens de communication decrits comme suit :

1- Le satellite GPS envoie le signal vers les points 2, 3 et 4 qui calculeront leurs positions respectives en analysant le signal GPS.

2- Un ensemble d'appareils portatifs qui peuvent communiquer en mode radio A1 pour envoyer leurs positions vers notre serveur se trouvant au point 6. Au cas où la communication radio échouerait, le système bascule en mode GSM-GPRS en A2 qui envoie les données via internet en B2 vers notre serveur en 6 .

3- Les véhicules sont équipés de plus grands modems radio avec une plus grande portée ce qui leur permet de relayer les données de localisation du groupe 2 en B1. Les voitures peuvent aussi être équipées de modules GSM-GPRS complémentaires et en faire usage comme pour les appareils portatifs en 2.

4- Une station A-GPS (4) qui permet de calculer l'erreur de positionnement du GPS dans sa zone. Ces données sont téléchargées par notre serveur via une connexion internet D. Les données de correction GPS sont diffusées en F sur une large zone via un modem longue portée en 5. Ceci permet à nos equippements reseaux mobiles (2) et (3) d'utilser les données de correction de positionnement calculées en (4) pour ajuster leurs positions respectives.

5- Un modem radio longue portée pouvant atteindre 50 km. Il reçoit les données de localisation de notre serveur avec une connexion serie (USB, RS232, etc.) via E,

transforme les données en paquets radio et les diffuse à nos équipements de localisation distants en 2 et 3 via des ondes radio en F.

6- Notre serveur a pour mission de récolter les données de localisation via B1 et B2, de formater les données et de les stocker sur une base de données. Il permet aussi de consulter régulièrement le serveur A-GPS en 4 pour télécharger les données de correction de la position et de les diffuser via le modem radio en 5. Il a aussi pour mission de fournir une interface d'accès à une application (GoogleMap) qui permettra de visualiser les données de localisation via une carte sur des terminaux (ordinateurs, PDA, etc.) dans 7 à travers une connexion internet en G.

7- Un ensemble d'utilisateurs qui disposent des droits d'accès à notre serveur en 6.

8- Une infrastructure appartenant à l'operateur telephonique qui nous fournis le service GPRS.

4.2.2 Architecture logicielle du système de positionnement

Cette section présente les solutions proposées au niveau logiciel que j'ai adapté pour OSGI afin de gérer l'infrastructure matérielle décrite dans la section précédente. La gestion de la communication (radio, GPRS, Internet) se fera à travers des bundles installés dans un environnement OSGI implémenté avec l'intergiciel Knopflerfish pour faciliter le déploiement, la maintenance et la réutilisation du code (voir OSGI chapitre 3).

4.2.2.1 Architecture logicielle de gestion des objets mobiles géo-localisables

Notre application est installée sous forme de bundles (voir le modèle OSGI dans le point A). Chaque bundle définit un service spécifique (voir OSGI chapitre 3). Les modes de connexion paraissent en tant que services de communication.

Un service de localisation permet de lire les positions en temps réel depuis le module GPS qui lui est connecté via un port de communication COM (USB, RS232, etc.). Ce service consulte la liste des services de communication disponibles (Radio, GSM) et en choisit un pour l'envoi des données de localisation. Le service de communication choisi tente de transmettre les données directement vers notre serveur central quand une connexion directe est possible. Dans le cas où la connexion directe échoue, le service de localisation demande de relayer les données à transmettre à un objet à proximité disposant d'une plus grande portée (voir point « 3 » de la figure 4.4). Si aucune unité n'est disponible, le service de localisation demande au service de communication GPRS d'envoyer le message vers le serveur central (voir l'algorithme de communication dans le point B).

A) Modèle OSGI pour Objets mobiles

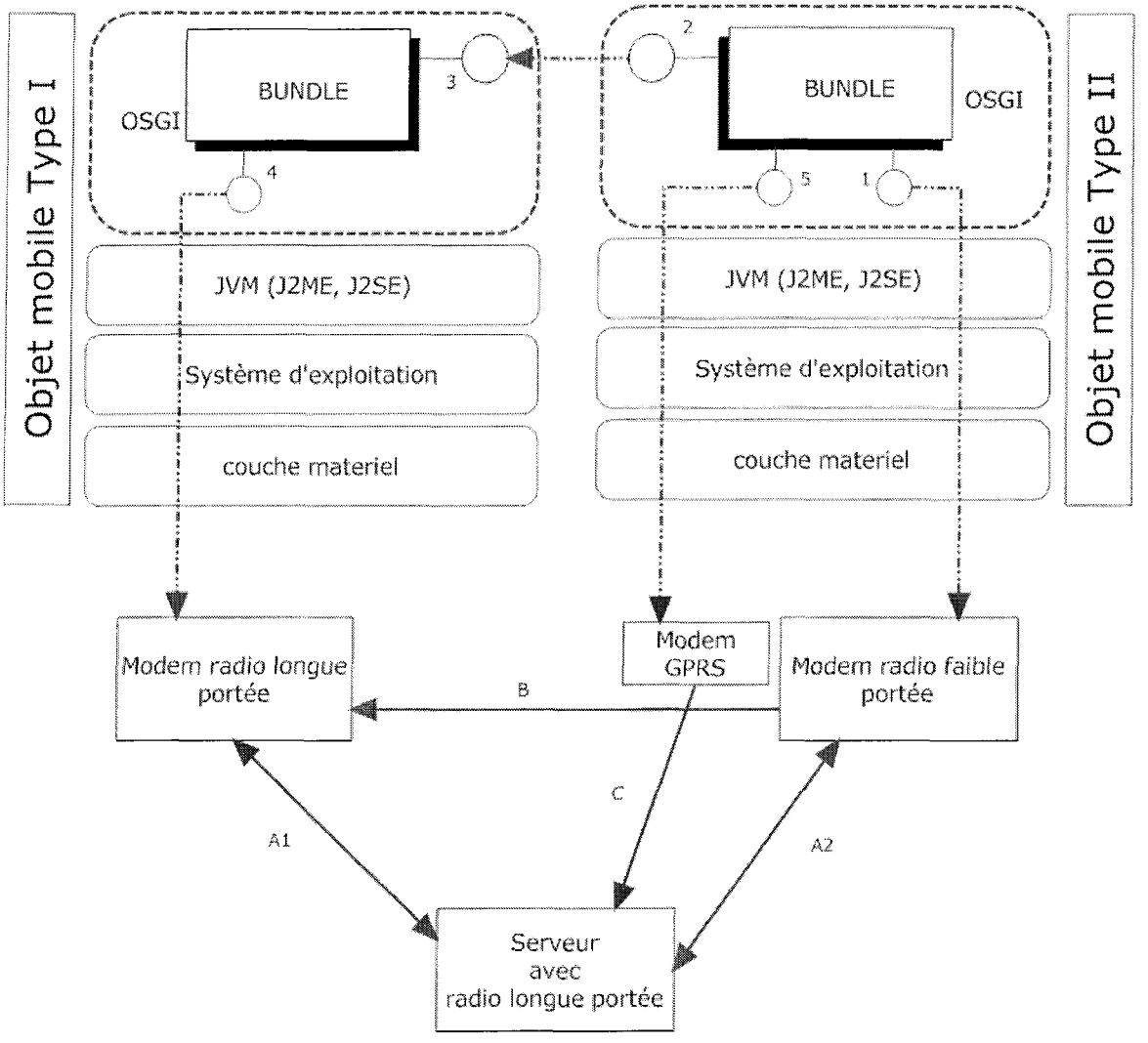

Figure 4.5: Schéma descriptif de la structure OSGI dans les appareils mobiles

La figure 4.5 décrit le fonctionnement d'OSGI à travers les points suivants :

i) Les appareils portatifs (type II) avec un petit GPS et un petit modem radio à faible portée.

ii) Les plus grands objets tels que des voitures (Type I) munies de plus grands modems radio. Les points mentionnés dans la figure 4.5 sont expliqués ci-dessous :

1- Le service au point 1, appartenant à l'objet mobile de type II, envoi des données de localisation via le modem radio faible portée, couplé à l'équipement mobile portatif via le lien radio du point A2.

2- En cas d'échec d'envoi par le service 1, le service au point 2 essaye de demander à un objet mobile de type I de relayer son message via un périphérique radio longue portée (voir mécanisme en résumé après la figure 4.6).

3- Le service au point 3, appartenant à l'objet mobile de type (I), prend les données de localisation communiquées par le service 2 de l'objet portatif type (II) via la connexion radio dans le point B. Il stocke en mémoire ces positions qui sont ensuite relayées, via le service dans le point 4 vers le serveur, avec une communication radio longue portée dans le point A1.

4- Le service au point 4 utilise la connexion radio longue portée en A1, pour envoyer ses propres données de localisation ainsi que celles en attente d'être relayées appartenant aux objets mobiles de types II.

5- Si le service du point 2 échoue (pas d'acquittement du service au point 3 pour les données à relayer), le service au point 5 prend la relève et envoie les données de localisation via le modem GPRS intégré à l'appareil mobile via le lien C vers notre serveur.

B) Algorithme pour la gestion de la communication

Le processus de gestion de la transmission des données décrit en début de cette section est présenté à travers l'algorithme de communication suivant :

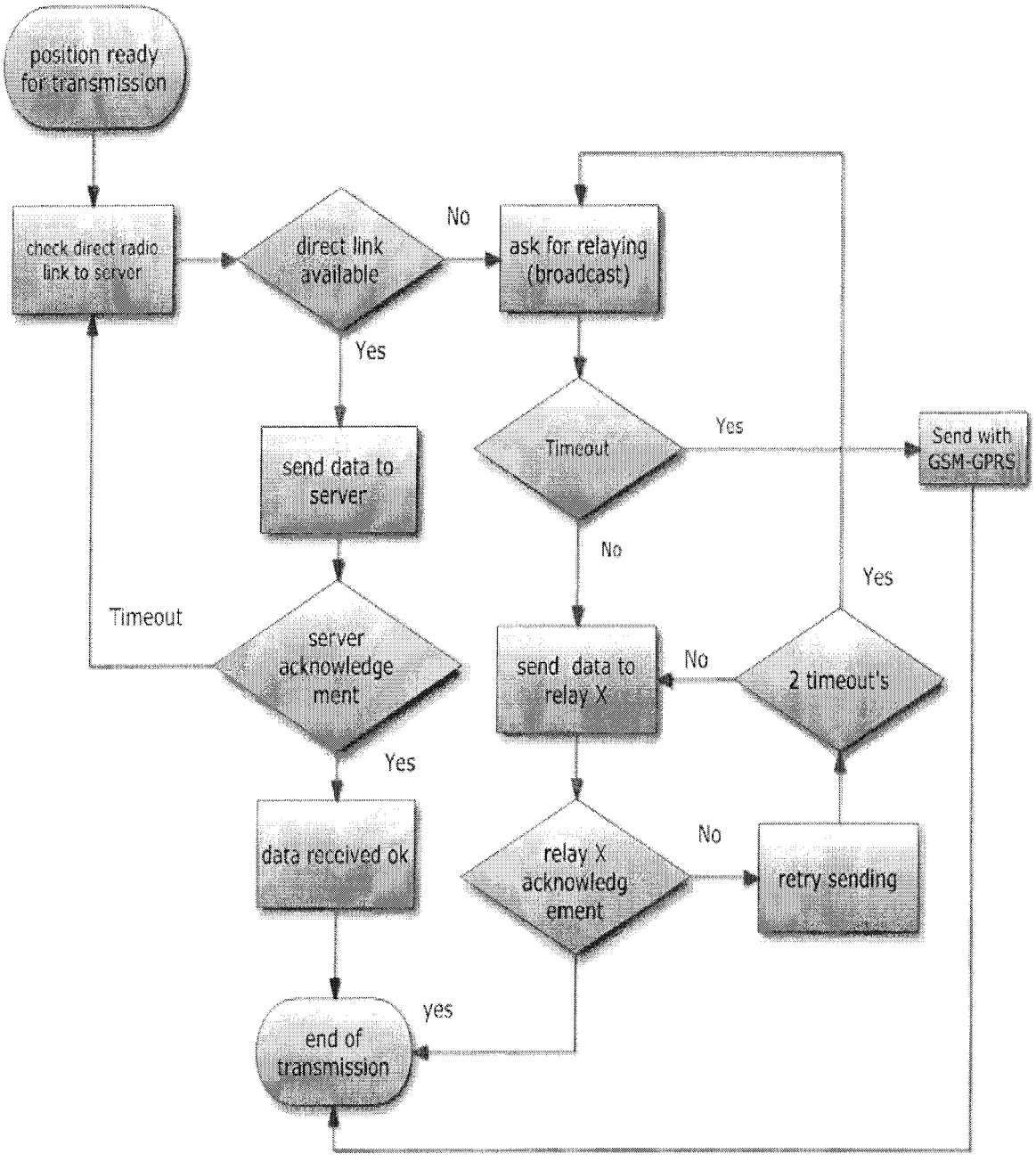

Figure 4.6: Algorithme de communication pour les objets mobiles géo-localisables

Pour résumer le schéma décrivant l'algorithme de communication, un nœud en attente de transmission de données de localisation, tente une transmission directe vers notre serveur. Si le serveur acquitte positivement la demande de transmission, le nœud peut transmettre ses données vers le serveur. Dans le cas où le nœud du réseau ne reçoit pas une réponse (acquittement) positive, ce qui explique que le serveur et hors de sa portée de transmission, le nœud actionne son mécanisme de découverte de nœuds relayeurs. Le mécanisme de découverte consiste à diffuser une requête vers les relais potentiels de façon périodique. Si un nœud relais se trouve dans la portée du nœud demandeur, le nœud relais renvois un acquittement positif avec l'ID du nœud demandeur et de son propre ID. Le nœud demandeur ayant intercepté un acquittement positif comportant son ID et l'ID du nœud relais, comprend qu'il est autoriser à envoyer son message vers le relayeur identifié par son ID. En ce qui concerne le relayeur, celui-ci dispose d'une mémoire interne pour sauvegarder les messages collectés des nœuds du réseau avec leurs ID respectif. En parallèle du processus de collecte des messages de positionnement à relayer, un nœud relais vérifie périodiquement sur un autre canal radio la proximité et la disponibilité du serveur afin de lui transmettre les messages collectés.

4.2.2.2 Architecture logicielle de gestion au niveau du serveur

Notre serveur comporte plusieurs services décrits comme suit:

i) Un service pour la collecte des données reçues de nos objets distants géo-localisables et le tri de ces informations selon le type de l'objet (voiture de police, voiture civile, patient, ambulance, etc.) afin de fournir une interface spécifique pour

chaque type d'application (service de police pour la localisation des véhicules en infraction et les véhicules de police disponibles à leurs proximités, service de santé pour la localisation des patients, etc.).

ii) Un service interne de base de données qui sauvegarde des données de localisation triées dans la base de données.

iii) Un autre service interne de base de données pour extraire les données nécessaires pour l'authentification et l'affichage des données utilisateur.

iv) Un ensemble d'interfaces utilisateurs fournies selon le profil de l'utilisateur authentifié où seulement les données qui lui sont accessibles seront affichées via un service HTTP.

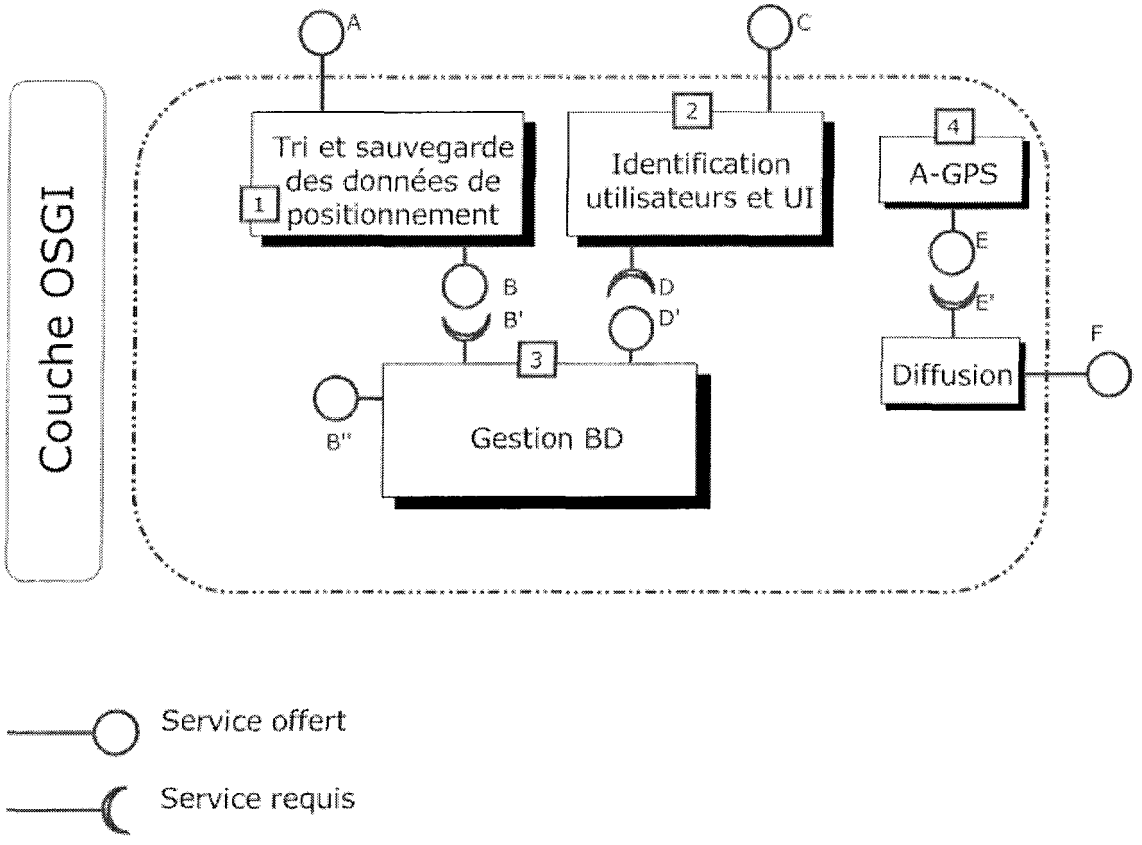

Figure 4.7: Schéma descriptif de la structure OSGI sur notre serveur

La figure 4.7 illustre les composants de la couche de l'intergiciel OSGI installé sur le serveur. Cette couche se compose de plusieurs bundles qui offrent différents services décrits via les points suivants :

Les bundles :

1. Bundle regroupant les fonctionnalités nécessaires pour le traitement des données de localisation à savoir :

i) le tri des informations par origine (patient, véhicule de police, ambulance, etc.).

ii) Extraction des données de localisation (position, vitesse, ID du dispositif).

iii) Sauvegarde des données dans la base de données dans les tables spécifiques à partir du résultat du tri (si le processus de tri détermine que les données sont reçues depuis un véhicule de police, le service de sauvegarde met ces informations dans la table Police avec ID du véhicule et coordonnées).

2. Bundle regroupant les fonctionnalités d'authentification permettant :

i) La gestion et l'authentification des utilisateurs depuis les profils utilisateurs fournis dans la base de données.

ii) La Fourniture des interfaces graphiques pour visualiser les données de localisation selon le domaine d'application (Police, santé, etc.) et selon les droits d'accès de l'utilisateur authentifié.

3. Bundle regroupant les fonctionnalités de base de données permettant les opérations d'entrée-sortie sur notre base de données, accessible seulement via les services des bundles cités précédemment.

4. Bundle permettant de se connecter sur un ou plusieurs serveurs A-GPS pour télécharger les données de correction de positionnement.

Les services :

A. Service appartenant au bundle 1. Il permet d'intercepter les messages en écoutant sur le port internet ou le port de l'antenne radio.

B. Service appartenant au bundle 1. À la demande du service A, il permet de traiter les informations récoltées par A. il trie les informations et demande au service de stockage B'' de mettre les informations dans des tables spécifiques au type d'objet mobile qui a envoyé ces données, comme expliqué dans le point 1.

B′. Ceci n'est pas un service, ce point montre que le service B est requis par le bundle 3 pour pouvoir offrir le service de stockage des données B″.

B″. Service appartenant au bundle 3. Il permet d'utiliser les informations extraites par le Service B afin de les mettre dans les champs adéquats dans la base de données.

C. Service appartenant au bundle 2. Il permet d'authentifier les utilisateurs en demandant au service D' de lui fournir, depuis la base de données, les informations sur l'utilisateur et ses droits d'accès à l'application demandée. Une fois l'utilisateur authentifié, le service C s'occupe de fournir à l'utilisateur les données de localisation via l'UI en demandant une deuxième fois au service D' de fournir les champs demandés depuis la base de données.

D′. Service appartenant au bundle 3. Il permet de fournir les champs de données demandés par d'autres services internes à l'application tel que le service C.

D. Comme pour B', ceci n'est pas un service, mais montre que le bundle 2 requiert le service D' pour offrir le service C.

E. Ce service appartient au bundle 4. Il offre les données de correction de la position exploitables par le bundle 4.

E′. Le service F requiert les données de correction du service E via le point E′.

F. Service appartenant au bundle 4, il permet de prendre les données de correction collectées par le service E. Il se connecte via une connexion série (USB, RS232, etc.) sur le modem radio pour diffuser les données de correction vers les unités mobiles pour corriger leurs données de positionnement.

4.3 Étude du coût

	Solution classique GPRS		Solution modem radio		
Coût communication GPRS[1]	65 $ / Mois (Rogers)		15$ / Mois (réserve données GPRS en cas d'échec de la communication radio)	-	-
			Petit équipement mobile	Grand équipement mobile	Équipement serveur
Coût d'achat du matériel par unité[2]	250 $ (GPS+GPRS+A-GPS)	1000 $ (serveur de données)	35 $ (GPS+A-GPS) + 75 $ (CM) + 90 $ (RF 1-2km) + 75 $ (GPRS) = 275 $	35 $ (GPS+A-GPS) + 75 $ (CM) + 200 $ (RF 10-24 Km) = 310 $	500 $ (RF 40-50 Km) 1 000 $ (serveur de données) = 1 500 $
Coût sur une période de 5 ans	100 unités (75 objets mobiles + 25 vehicules) ((65*12*5*100)+(25		75 unités (petits objets mobiles) ((15*12*5*100)+(35+75+90+75)*75) =	25 unités (véhicules) ((35+75+90+75)*25) =	2 unités (1 relais + 1 central)

[1] Coûts basés sur la moyenne des forfaits et frais de services GPRS des principales compagnies de téléphonie cellulaire au Québec tels que ROGERS, FIDO et BELL.

[2] Coûts basés sur les tarifs des revendeurs des équipements GPS et de communication sans fil : www.sparkfun.com, www.semiconductorstore.com, www.data-linc.com, www.electronicdiscountsales.com.

	0*100)+1000) = 416 000 $	111 525 $	7 750 $	((500*2)+(1000*2))= 3 000 $
Coût total pour 5 ans	416 000 $	122 275 $		
Gain avec notre système sur une durée de 5 ans	293 725 $ (2.4 fois le coût de notre système)			

TABLEAU 4.1 : *Comparatif des coûts entre notre solution de communication radio et la solution classique via GPRS*

NB : La plus grande partie du coût est liée au coût de communication via GPRS.

Notre modèle va nous permettre d'économiser sur le coût de la communication en utilisant des communications radio sans fil. La présente section donne une idée sur les économies réalisables en comparant notre solution à la solution classique qui consiste en l'utilisation d'une communication GPRS payante. Le tableau ci-dessus montre cette comparaison.

Si l'on prend le cas de notre modèle, les équipements mobiles ont une faible portée est peuvent se trouver loin de l'antenne de notre serveur pour une connexion directe ou loin des voitures qui peuvent relayer leurs informations vers notre serveur central. La ville de Chicoutimi au Québec dispose d'une superficie de 157 Km2, que nous pouvons grossièrement représenter par un carré de 12-13 Km de coté.

Afin de maximiser la disponibilité des données de localisation en provenance des équipements mobiles à faible portée, nous devons déployer les véhicules qui servent de

relais à ces petits équipements de manières à couvrir toute la superficie du carré. Les véhicules disposent de modems radio pouvant couvrir en moyenne 24 km en terrain ouvert avec quelques obstacles selon les spécifications des constructeurs.

Dans un milieu urbain très dense en structures, j'estime que cette portée peut être réduite à 15 Km. La portée des véhicules dépasse la longueur du coté du carré qui est de 12-13 Km et par conséquent, les voitures sont souvent à la portée de l'antenne de notre serveur. Lorsque l'on se trouve dans une superficie qui n'est pas carrée mais plus rectangulaire, comme par exemple 3 Km x 23 Km de cotés, le plus long coté est de 23 Km. Si l'on suppose que notre antenne du serveur central se trouve au centre de la ville, elle pourra donc être à potée de n'importe quel véhicule dans la ville étant donné que la distance de l'antenne jusqu'aux bords de la ville est inferieure ou égale à 11,5 Km (23 Km / 2) ce qui est suffisant pour un véhicule d'une portée de 15 Km afin de lui permettre de contacter notre serveur central. Pour augmenter la portée, j'utilise dans mon modèle un serveur auxiliaire avec une autre antenne longue portée similaire aux caractéristiques du serveur central. La portée nécessaire pour être à la porté des antennes du serveur devient ainsi 5,75 Km (11,5 Km / 2).

La problématique se pose plus au niveau de la portée des petits objets mobiles géo-localisables. Ces derniers ont souvent une portée maximale de 1,5-2 Km. Pour palier ce problème, nous devons déployer nos véhicules, qui serviront de relais pour les équipements mobiles à faible portée, à une distance de 1,5 Km de ces derniers. Nous pouvons donc diviser la superficie de la ville en petites cellules de 2,25 Km2 et dans

chaque cellule nous devons disposer d'un relayeur, soit les véhicules doivent être présents dans 70 cellules (157 Km2 / 2,25 Km2 = 69,778). Si nous disposons de 70 véhicules stationnaires nous pouvons couvrir tous les petits équipements de la ville de Chicoutimi. Pour parcourir 13 km, un véhicule peut faire le parcours en 21 minutes (estimation GoogleMap) soit 0,619 Km / minute.

Dans notre tableau, nous avons estimé le coût pour 25 véhicules. Pour rester dans le même nombre de véhicules et en divisant les 70 cellules sur les 25 véhicules nous obtenons 2.8 qu'on peut arrondir pour 3, ceci veut dire que nos véhicules couvrent seulement 1/3 des cellules en mode stationnaire. Pour parcourir les 2/3 cellules non couvertes, chaque véhicule doit parcourir 1,5 Km x 2 soit 3 Km. Avec la vitesse moyenne de 0,619 Km, les deux cellules restantes vont être parcourues dans 3Km / (0,619 Km / minute) soit 4,85 minutes qu'on peut arrondir à 5 minutes.

Dans notre modèle proposé, au pire un relevé de positionnement via le modem radio peut prendre 5-6 minutes en comptant le temps de transferts de l'équipement mobile vers le véhicule relais et du véhicule relais vers notre serveur. Si chaque objets géo-localisable fait un relevé de données chaque seconde, en 5 minutes nous aurons 5 x 60 soit 300 relevés. Chaque message de positionnement relevé contient 32 caractères, chaque caractère est codé en ASCII 8 bits soit 1 octet. Dans 5 minutes de relevés nous avons 300 messages de positionnement de 32 octets chacun. Le poids total des 300 messages est de 300 x 32 octets

soit 9600 octets ou 9,6 Ko. Le temps de transferts avec un débit de transfert radio standard3 de 9600 bits/s nous permet de transférer les 300 messages en 8 secondes étant donné que 1 octet = 8 bits. Ce temps de transferts est très satisfaisant pour une cellule qui est normalement parcourue en 4,8 minutes / 2 = 2,4 minutes soit 144 secondes. Dans ce cas, le modem radio peut recevoir, dans une même cellule et sur un même canal radio 144s / 8s soit 18 lots de messages de 300 messages chacun. En d'autres mots, au passage du véhicule de relais dans une cellule, nous pouvons récolter les messages de 18 objets géo-localisables.

D'habitude, les équipements radio de grande taille comme ceux de nos véhicules relais, disposent de plusieurs canaux radio et d'une plus grande bande passante ce qui permet de décupler le nombre de 18 objets calculé. En augmentant d'un cran le nombre de véhicules relais, nous parviendrons à diminuer le temps de parcours des cellules et par conséquent nous pouvons augmenter le nombre d'objets géo-localisables par cellule.

En résumé, notre solution nous permet de gérer un grand nombre d'objets géo-localisables avec des communications radio gratuites et de réaliser des économies colossales quand nous comparons cette approche avec celle utilisant exclusivement la connexion GPRS comme le montre le tableau de comparaison des coûts.

4.4 Récapitulatif

Dans ce chapitre, nous avons proposé un modèle réseau basé sur des équipements radio qui nous permettent d'établir des communications sans fil gratuite. Ce modèle réseau nous

[3] Basé sur les spécifications du constructeur : http://www.adhocelectronics.com/Products/XCite-OEM-RF-Modules-900-MHz-4mW.

permet, d'après l'étude de coût établie, de minimiser le coût quant à notre solution de gestion d'objets géo-localisables en comparaison avec la solution classique utilisant GPRS. Le coût peut encore être minimisé si l'on dispose de plus de véhicules vont relayer les messages de positionnement des petits objets géo-localisables en couvrant le plus de cellules possibles dans une zone urbaine. Si l'on parvient à une couverture totale nous pouvons ne pas utiliser de communication GPRS de réserve incluse dans l'étude de coût de 15$ / Mois. Ceci revient à dire que pour les 75 petits objets géo-localisables à faible portée nous réalisons une économie sur le coût de communication supplémentaire sur 5 ans de 15 x 75 x 12 x 5 = 67 500$.

Notre modèle se base aussi sur une solution logicielle qui consiste en l'utilisation du middleware OSGI. Cette solution nous permet de développer des modules indépendants facilement déployés. Le coût lié au déploiement, à la mise à jour et à la maintenance de notre application est aussi réduit vu le gain en temps et en effort requis par l'équipe de programmation et de déploiement.

Afin de concrétiser notre modèle, nous allons voir dans la section suivante son implémentation dans un environnement OSGI. L'outil prototypé que nous allons présenter constitue une preuve de conception permettant de mieux comprendre notre solution et servir de base pour des développements et des améliorations futures. L'implémentation encouragera aussi les programmeurs à utiliser les implémentations OSGI comme plateforme cible pour leurs applications en prenant conscience des atouts qu'offre cette nouvelle approche.

V

RÉALISATION DU MODELE DE GESTION D'OBJETS GÉO-LOCALISABLES

5.1 Introduction

Dans cette section, je décris la mise en œuvre du modèle du système présenté dans le chapitre 4. J'ai choisi Java comme langage de programmation pour sa portabilité et sa popularité. Java est utilisé pour développer les bundles (composants) OSGI et l'interface de communication entre les différents composants du système. Les avantages d'OSGI sont: la réutilisation, la sécurité, la facilité d'intégration et de maintenance (voir chapitre 3). Les ports de communication COMx sont utilisés pour communiquer avec des périphériques en local via la librairie javax.comm. Le suffixe x dans le terme COMx est le numéro du port. Par exemple, le GPS est connecté sur le port physique COM3 sur lequel nous lirons les données de localisation. Le modem radio est connecté sur le port physique COM10 pour envoyer les données à distance, des ports logiques COMx peuvent être créés pour partager un port physique entre plusieurs services en même temps sans avoir à le liberer. Pour l'interface web, j'ai codé avec PHP et utilisé l'API Ajax GoogleMap permettant d'incorporer des cartes géographiques. Cette interface web sur le serveur central permet d'afficher à travers une page web les données de positionnement sur une carte GoogleMap (position, vitesse, date, icônes d'objets localisés, etc.). Le langage SQL est utilisé pour les opérations sur ma base de données MySQL à travers les pages développées en PHP.

Les bundles OSGI sont utilisés pour i) établir la communication entre les composants du système (lien direct objet mobile vers objets mobiles, objet mobile vers serveur, serveur vers les objets mobiles), et ii) trier et stocker les informations dans la base de données.

Ce chapitre comprend deux parties. La première concerne le programme utilisé avec les objets mobiles (GPS). La deuxième partie concerne le développement du côté serveur, à savoir la centralisation des données et les interfaces nécessaires pour afficher les données collectées. Pour les deux parties, nous allons montrer l'usage de l'OSGI implémenté par l'intergiciel Knopflerfish. Des échantillons de code sont présentés, appuyés par des captures d'écrans d'exécutions pour mieux comprendre le fonctionnement du système.

5.2 Bundles de gestion d'objets mobiles

Il s'agit dans cette section de montrer les différents bundles d'OSGI utilisés par l'intergiciel Knopflerfish. Le modèle OSGI représentant ces bundles est donné dans la section 4.2.2.1. Du point de vue de la programmation, les bundles qui sont installés sur les appareils mobiles sont les suivants :

i) Bundle de service de communication GPS « gpsservice » : Ce bundle sert pour l'extraction des données de positionnement. Ce bundle offre aussi un service pour utiliser ses fonctionnalités par un autre bundle local (voir section 5.2.1).

ii) Bundle de communication radio distante « gpsserviceuser »: Il permet d'utiliser le service de communication du bundle précédent. Il intercepte les données GPS offertes par le service de communication. Ce bundle envoie les données reçues sur le port COM10 du modem de communication radio. Le modem radio est couplé au dispositif de localisation mobile. Un lien radio est créé sur le port COM10 du dispositif de localisation local vers le port radio distant (COM12) du serveur. Le lien est assuré par des ondes radio (voir section 5.2.2).

iii) Bundle GUI locale « guipda » : Il permet d'afficher l'interface utilisateur sur le dispositif local (PDA, terminal d'une voiture, PC, etc.). À l'instar du bundle (ii), il utilise le service du bundle de communication (i) pour extraire les données de positionnement du GPS. Il crée ensuite une interface graphique utilisateur pour afficher les données de positionnement sur l'écran (voir section 5.2.3).

Ces trois bundles sont détaillés dans les sous-sections suivantes.

5.2.1 Bundle de service de communication GPS

La figure 5.1 montre l'organisation des packages du bundle « gpsservice » sous Eclipse.

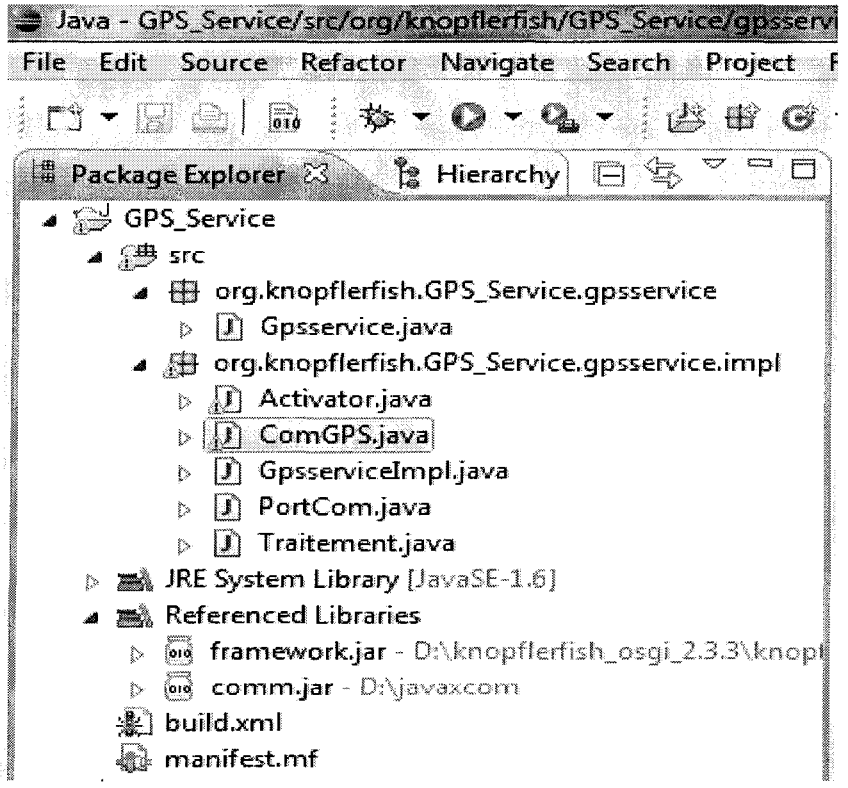

Figure 5.1 : Bundle « gpsservice » sous Eclipse

Gpsservice.java : Il s'agit de l'interface du service de communication GPS. Il permet d'afficher les fonctionnalités utilisables par d'autres bundles :

```
public interface Gpsservice {
   public String LirePosition();
   public void envoyerMSG(String msg, String port);
   public void Arret();
}
```

PortCom.java : Classe java qui permet d'ouvrir un port de communication et de créer des flux d'entrées et de sorties, pour lire et écrire les données sur un port COM choisi :

```
public class PortCom {
   /** Flux d'entrée*/
     protected BufferedReader is;
     /** Flux de sortie */
     protected PrintStream os;
     /** Port de communication */
     CommPort port;
     /** vitesse de transmission des données sur le port de
     communication */
     int BAUD=115200;
      //...
}
```

Le constructeur PortCom permet de saisir un nom de port en paramètre, par exemple COM3. La création de l'objet PortCom permet de créer un flux de lecture sur le port choisi en paramètre à travers la variable « is » (input stream) et un flux d'écriture sur le même port avec la variable « os » (output stream). « is » permet de lire des données telles que la position du GPS et « os » permet d'envoyer un message sur le port. Par exemple, os.println(« >QVP< ») va envoyer le message QVP (Query Velocity Position) qui est une commande du protocole TAIP, sur le port du GPS. Ce dernier va recevoir la commande en format ASCII et retournera une réponse RVPxxx…(Entête ResponseVelocityPosition + les données de positionnement xxx) qui contient les données de localisation. Cette réponse est

lisible sur le port via la variable « is » en écrivant is.readLine(). Ces variables sont utilisables par d'autres classes via les fonctions Lire(commande TAIP) et Ecrire(message à envoyer) comme décrites ci-dessous :

```
/* suite de la classe PortCom */
public  PortCom(String nomport) throws Exception {
   ...
   is = new BufferedReader(new
   InputStreamReader(port.getInputStream()));
   os = new PrintStream(port.getOutputStream(), true);
    ...
}

public String Lire(String cmd){
   ...
   os.println(cmd);
   return is.readLine();
   ...
}
public void Ecrire(String msg){
   os.println(msg);
}
```

ComGPS.java : Classe Java qui utilise la classe PortCOM.java précédente. Cette classe crée un objet de type PortCOM pour initialiser le port de communication GPS (par défaut COM3) et traiter les messages lus depuis le GPS :

```
public class ComGPS {
  PortCom PC = null;
  public ComGPS(String nomport) {
        PC = new PortCom(nomport);
  }
```

La fonction Affiche() utilise la fonction Lire() de la classe PortCOM.java pour lire le message GPS dans la variable « msg ». La classe Traitement.java permet de construire l'objet « x » pour traiter le message « msg » via la fonction TraitementMsg(). La chaine de caractères « GPS_DATA » permet de stocker les différents paramètres de positionnement

(latitude, longitude, vitesse, etc.). Cette chaine de caractères sera retournée à l'invocation de la fonction Affiche() décrite ci-dessous :

```
public String Affiche(){

String msg=PC.Lire(">QPV<");
Traitement x = new Traitement(">QPV<");
x.TraitementMsg(msg);

...

GPS_DATA=(x.latitude()+";"+x.longitude()+";"+x.signal_qualite+";"+x.ve
locity()+";"+PDOP_niv+";"+x.precision_source+";"+x.precision_niveau);
return GPS_DATA;

}
...
```

Traitement.java : Cette classe permet de formater une chaine de caractères contenant les données de localisation lues par la classe précédente ComGPS.java. Les données à traiter sont sous format texte conforme au protocole TAIP (">RPVxxxx...<") :

```
class Traitement {
...
/* la phrase TAIP change de structure selon la requête employée pour
questionner le GPS. Dans notre cas nous utilisons la réponse TAIP de
type RVP... en réponse à la requête QVP (Query Velocity Position) */

   Traitement(String TaipCmd) {
         MsgType = TaipCmd;
  }


  static void TraitementMsg(String InMsg) {
  MSG = InMsg;
  ...
        /* ici nous testons le type de message TAIP pour choisir le
        traitement approprié*/
        if(MsgType.equals(">QPV<"))
       {
  ...
```

```
//les fonctions suivantes vont transformer les valeurs de données
//de localisation dans le type correspondant (Int, Double, etc.)

float latitude() {
      if(!available) return 0;
      /* conversion du format de la latitude de xxxx.yyyy à
      xx.yyyyy*/
      return Float.valueOf(latitude).floatValue()/100000 * north;
}

...

Double velocity() {
      if(!available) return 0.0;
      // conversion de la vitesse du Mph vers Km/h
      return Double.valueOf(Mph).doubleValue() * 1.6093;
}

...
```

GpsserviceImpl.java : Cette classe i) importe les classes précédentes pour se connecter sur le port COM3 du GPS, ii) lit les messages TAIP, iii) les traite pour extraire les informations de localisation (latitude, longitude, etc.). Cette classe permet de concaténer ces informations dans une chaine de caractère, séparées avec le caractère « ; » dans le format (latitude;longitude;vitesse; etc.). Cette chaîne de caractères est ensuite mise à disposition d'autres bundles à travers le service défini dans la classe Activator.java. Cette classe offre aussi une fonction envoyerMSG(message, port) qui permet d'envoyer un message ou une commande sur le port choisi. Par exemple, cette fonction envoie les données de localisation sur le port radio COM10 vers notre serveur personnel.

```
public class GpsserviceImpl implements Gpsservice {
      String port="";
      private ComGPS GPS=null;

      public GpsserviceImpl(String x){
            port=x;
```

```
            GPS= new ComGPS(port);
        }
        public String LirePosition() {
            return GPS.Affiche();
        }

        public void envoyerMSG(String msg, String port){
            ...
            port2 = new PortCom(port);
            ...
            port2.Ecrire(msg);
        }
    }
```

Activator.java : La classe Activator.java est par défaut la classe OSGI qui permet d'enregistrer les services du bundle ou d'en invoquer d'autres. Elle permet de fournir les informations nécessaires à l'intergiciel Knopflerfish ou autres pour démarrer le bundle.

```
public class Activator implements BundleActivator {
  public static BundleContext bc = null;
  private GpsserviceImpl serv=null;

  public void start(BundleContext bc) throws Exception {
        ...
        Activator.bc = bc;
        serv = new GpsserviceImpl("COM3");
        ServiceRegistration registration =
        bc.registerService(Gpsservice.class.getName(), serv, new
        Hashtable());

        ...
  }
  public void stop(BundleContext bc) throws Exception {
        ...
        serv.Arret();
        Activator.bc = null;
  }
}
```

manifest.mf : Le fichier manifest.mf comprend la configuration de ce bundle pour que l'intergiciel Knopflerfish connaisse les points suivants :

i) les services offerts via « Export-Package: »

ii) les bundles requis via « Import-Package: »

iii) Le nom symbolique du bundle pour invoquer ses services par un autre bundle via « Bundle-SymbolicName: »

iv) La classe Activator.java que l'intergiciel consulte pour lancer et arrêter le bundle via « Bundle-Activator: »

Les points énumérés ci dessus sont décrits dans un fichier « manifest » comme suit :

```
Manifest-Version: 2.0
Bundle-Name: gpsservice
Bundle-SymbolicName: org.knopflerfish.GPS_Service.gpsservice
Bundle-Version: 1.0.0
Bundle-Description: service communication GPS
Bundle-Vendor: A.Goundafi
Bundle-Activator:
org.knopflerfish.GPS_Service.gpsservice.impl.Activator
Bundle-Category: service
Import-Package: org.osgi.framework, javax.comm
Export-Package: org.knopflerfish.GPS_Service.gpsservice
```

build.xml : Le fichier build.xml est un fichier ANT qui permet de générer le bundle en lui fournissant les chemins d'accès aux fichiers .jar utilisés. Dans ce cas, nous utilisons framwork.jar qui contient les classes nécessaires pour construire un bundle conforme à l'OSGI. Le fichier comm.jar est la librairie de communication javax.comm qui permet d'ouvrir les ports de communication et d'échanger les données si dessus.

```
<?xml version="1.0"?>
<project name="gpsservice" default="all">
<property name="kf.dir" location="D:\knopflerfish_osgi_2.3.3"/>
<property name="javacomm.dir" location="D:\javaxcom"/>
<property name="framework.jar"
location="${kf.dir}/knopflerfish.org/osgi/framework.jar"/>
<property name="comm.jar"
...
<pathelement location="${framework.jar}"/>
```

```
<pathelement location="${comm.jar}"/>
<target name="jar">
...
manifest = "manifest.mf"/>
...
</project>
```

5.2.2 Bundle de communication radio distante

Ce bundle « gpsserviceuser » permet d'utiliser le service du bundle de communication GPS précédent « gpsservice ». Ce dernier lui fournit les données GPS. Une fois que ce bundle dispose des données GPS, il les envoie via le modem radio vers le serveur personnel. L'envoi du message vers le modem radio se fait vers le port COM10. La figure 5.2 montre l'organisation du package de ce bundle :

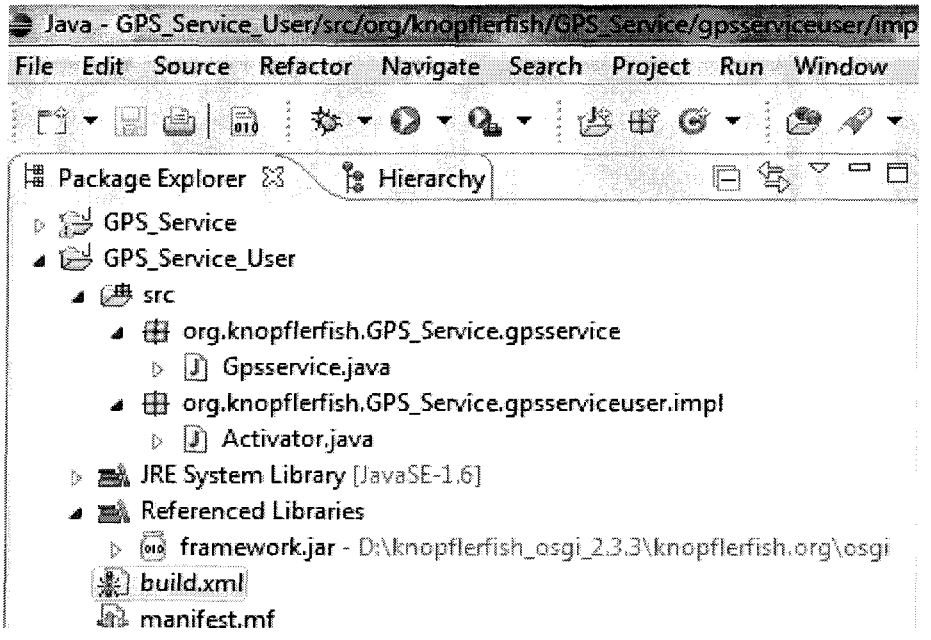

Figure 5.2 : Bundle « gpsserviceuser » sous Eclipse

Activator.java : La classe Activator.java joue le même rôle que dans le bundle précédent. La seule différence c'est qu'on y invoque le service du bundle « gpsservice » qui permet de lire les données GPS (envoyerMSG(msg+"\n", "COM10") et d'envoyer un message radio sur le port COM10 du modem radio.

```
import org.knopflerfish.GPS_Service.gpsservice.Gpsservice;
...

public class Activator implements BundleActivator {
   ...
   Gpsservice serv=null;
   public void start(BundleContext bc) throws Exception {
         ...
         Activator.bc = bc;
         ServiceReference reference =
         bc.getServiceReference(Gpsservice.class.getName());
          serv = (Gpsservice)bc.getService(reference);
         ...
         String msg="";

         /*utilise le service gpsserice pour lire les données de
         positionnement dans « msg » */

         msg=serv.LirePosition();
         ...
         /* utilise le service gpsserice pour envoyer les données de
         positionnement vers le serveur personnel */

         serv.envoyerMSG(msg+"\n", "COM10");
         ...
   }

   public void stop(BundleContext bc) throws Exception {
         ...
         serv.Arret();
         Activator.bc = null;
   }
}
```

Gpsservice.java : La même classe Java de l'interface du service de communication GPS « gpsservice ». Nous l'incluons dans ce bundle pour pouvoir utiliser les fonctionnalités du service.

build.xml : Mêmes fonctionnalités que celles du bundle précédent.

```
<?xml version="1.0"?>
<project name="gpsserviceuser" default="all">
<property name="kf.dir" location="D:\knopflerfish_osgi_2.3.3"/>
<property name="framework.jar"
location="${kf.dir}/knopflerfish.org/osgi/framework.jar"/>
```

manifest.mf : Mêmes fonctionnalités que celles du bundle précédent.

```
Manifest-Version: 2.0
Bundle-Name: gpsserviceuser
Bundle-SymbolicName: org.knopflerfish.GPS_Service.gpsserviceuser
Bundle-Version: 1.0.0
Bundle-Description: utilisateur du service de communication GPS
Bundle-Vendor: A.Goundafi
Bundle-Activator:
org.knopflerfish.GPS_Service.gpsserviceuser.impl.Activator
Bundle-Category: bundle
Import-Package: org.osgi.framework,
org.knopflerfish.GPS_Service.gpsservice
```

5.2.3 Bundle GUI locale

Le bundle « guipda » permet de lire les données de positionnement en utilisant le service du bundle « gpsservice » et de créer ensuite une interface graphique pour visualiser ces données. La figure 5.3 suivante montre l'organisation du bundle « guipda » :

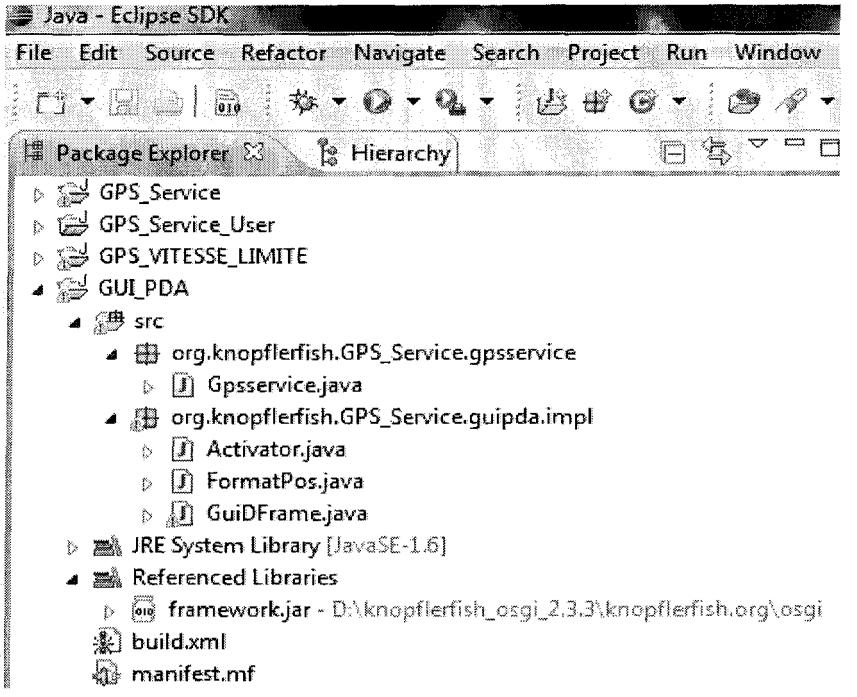

Figure 5.3 : Bundle « guipda » sous Eclipse

FormatPos.java : Cette classe permet de lire une chaine de caractères comme celle fournie par le bundle « gpsservice » contenant les données de positionnement.

```
public class FormatPos{

  private float latitude; // ddmm.mmmm
  private float longitude; // ddmm.mmmm
  private String signal_qualite;
...

/* cette fonction prend une chaine de caractères, elle divise les mots
compris entre les caractères de séparation « ; » et met chaque mot
dans une case du tableau « Tab » */

  public FormatPos(String msg){
            message=msg;
            ...
            Tab=message.split(";");
      }

/* cette fonction permet de retourner une chaine de caractères
contenue dans le tableau « Tab » en utilisant son numéro de position
« pos » dans ce dernier */
```

```
private String Lire(int pos){
            ...
}

...
/* Les méthodes suivantes permettent de convertir les valeurs de
données de localisation dans le type de données approprié */
public float Latitude(){
            latitude=Float.valueOf(Lire(0)).floatValue();
            return latitude;
}

public float Longitude(){
            ...
}

public String Qualite(){
      ...
}

...
public int PDOP(){

      PDOP=(Lire(4)=="1") ? 1 : 0;
      return PDOP;
}

public String Source(){
      ...
}

public int Precision(){
      ...
}
...
```

GuiDFrame.java : Cette classe permet de créer une interface graphique qui contient les informations de positionnement du GPS local. La classe « FormatPos » précédente lui fournit les données de positionnement déjà formatées.

```
public class GuiDFrame extends JFrame
{

/*on initialise les variables pour enregistrer les données de
positionnement (latitude, longitude, vitesse, etc.) et les données sur
la qualité des données (PDOP, âge du message, etc.) */

  private String lattxt="";
```

```
    ...
    private Color code=Color.red;

    JLabel lab_lo=null;
    ...
    JTextField lo;
    ...

/* La fonction refresh(...) permet de mettre à jour les informations
affichées dans notre interface graphique */

public void refresh(float a,float b,String c,double d, Color coul,
String source){

    la.setText(""+a);
    ...
    signa.setBackground(coul);
    ...



    }

/* Le constructeur GuiDFrame permet de construire et d'initialiser
l'objet de l'interface graphique avec quelques données de
positionnement  */

 public GuiDFrame(float lat,float lon,String signal,double vit) {

        ...
}
```

5.3 Bundles du côté serveur

Dans cette section, nous allons voir le bundle et les pages web dans notre serveur personnel (voir figure 4.7). Il est à noter que les middlewares actuels, utilisant l'implémentation 4.2 d'OSGI courante, n'ont pas encore implémenté des bundles JDBC pour permettre d'enregistrer un service de pilotes de bases de données. Ayant rencontré des erreurs à l'invocation de la connexion à ma base de données à cause de cette lacune, j'ai cédé cette tâche à des pages web PHP. J'ai par conséquent développé, du côté serveur, un seul bundle dont toutes les fonctionnalités sont actives sous Knopflerfish sauf les appels à

la base de données. Par contre, ces appels fonctionnent parfaitement en environnement PHP. Les éléments du côté serveur central (seulement le premier bundle est développé, les autres ont été substitués par des pages web et des fichiers java standards) sont présentés comme suit :

i) Bundle de communication radio distante « serveurbundlelecture » : qui va intercepter les données envoyées à distance par les objets mobiles géo-localisables. Dépendamment du canal de transmission, ces données sont lues de deux façons : i) sur le port COM du modem radio connecté au serveur personnel (COM12), ii) à travers des sockets dans le cas d'une transmission GPRS (le mode GPRS permet de transmettre les données sur internet en utilisant le réseau cellulaire mobile). Ce bundle offre un service qui va être utilisé par le bundle de gestion de la base de données. Ce service permet d'obtenir les données de positionnement qui vont être stockées dans la base de données du serveur personnel. Le bundle ajoute aux données de positionnement reçues, l'identificateur de l'objet mobile permettant de connaitre de quels types d'objets il s'agit : appareils mobiles types I, type II, voitures de police, ambulances, etc.

ii) Bundle de gestion de base de données : Ce bundle offre différentes fonctionnalités :

   a) Utiliser le service offert par le bundle précédent pour acquérir les données de positionnement avec leurs Device_ID . Dans notre base de données nous avons préalablement une table de correspondances entre le

Device_ID, le type d'objet (type I, typeII, etc.), domaine d'application (santé, police, usage personnel, etc.). Cette correspondance permet de savoir à quelles applications un appareil géo-localisable appartient et qui seront les usagers qui peuvent visualiser les données spécifiques de ces objets.

b) Offrir un service pour le bundle d'identification d'utilisateurs et une interface utilisateur. Ce service offre les méthodes nécessaires pour lire et extraire les informations concernant les utilisateurs des applications sur notre serveur personnel. Par exemple, une interface utilisateur du bundle d'identification d'utilisateurs et UI permettra de saisir le login, mot de passe et le type d'application souhaitée (police, santé, etc.). Une fois les données validées, l'interface utilisateur demande au service de la base de données de vérifier la correspondance entre le login et le mot de passe saisis, et le login et le mot de passe stockés dans une table Login. La table Login, en plus des champs mot de passe et login, contient aussi un champs avec les applications accessibles pour l'utilisateur. Par exemple une ligne dans cette base de données correspondant dans l'ordre aux champs login, mot de passe, application sera comme suit : « user1 », « user1_pass », « police; santé; ». Ceci signifie que l'utilisateur « user1 » qui sera authentifié avec son mot de passe « user1_pass » aura droit d'utiliser l'interface utilisateur pour la police de même que

l'interface graphique pour les services de santé afin de repérer les patients (cardiaques, Alzheimer, etc.).

iii) Bundle d'authentification des utilisateurs et des interfaces utilisateurs : Ce bundle permet d'utiliser le service du bundle de base de données précédent pour :

a) Authentifier l'utilisateur et connaitre les applications auxquelles il a droit.

b) Offrir une interface graphique pour l'application demandée par l'utilisateur authentifié.

iv) Un bundle A-GPS qui va se connecter sur un serveur A-GPS via internet pour télécharger les données de correction GPS. Ce bundle offre un service au bundle de diffusion suivant les données de correction récoltées.

v) Un bundle de diffusion des données de correction GPS. Ce bundle utilise le service offert par le bundle A-GPS précédent pour acquérir les données de correction GPS et les diffuser ensuite via le modem radio connecté à notre serveur personnel vers les appareils de localisation distants (objets type I et type II, voitures de police, etc.). La diffusion des données radio se fera sur le même port radio COM12 qui a servi au bundle (i) d'intercepter les données radio des objets géo-localisables.

Les fonctionnalités des bundles i et ii sont substitués par un code PHP. Cependant, ils ne sont pas implémentés pour OSGI à cause de l'absence du bundle fournissant un pilote de

base de données comme expliqué précédemment. Les bundles iv et v ont aussi été annulés, car j'ai besoin d'un GPS qui peut lire les données de correction téléchargées d'un serveur A-GPS (par exemple les données de correction en format RINEX). L'envoi de ces données vers le GPS se fait de la même manière que pour l'envoi d'une commande TAIP à notre modem pour recevoir un message de réponse contenant les données de positionnement. Pour ce faire, nous devons utiliser une communication sur le port COM du GPS à travers la variable os (output stream) comme utilisée dans la classe PortCom.java. La section 5.3.1 décrit un bundle qui gère : l'interception des messages radio, formatage des données et stockage des données formatées en base de données. La section 5.3.2 décrit les interfaces web utilisées pour authentifier un utilisateur en prenant en considération son choix d'application (police, santé, véhicules). L'interface Web utilisateur permet d'ouvrir l'application souhaitée après authentification, en affichant une carte GoogleMap avec les points de positionnement.

5.3.1 Bundle de communication radio distante

Ce bundle permet d'intercepter les données envoyées depuis les unités mobiles et sauvegarde les informations de géo-localisation dans la base de données locale de notre serveur central. La figure 5.4 montre l'organisation de ce bundle sous Eclipse.

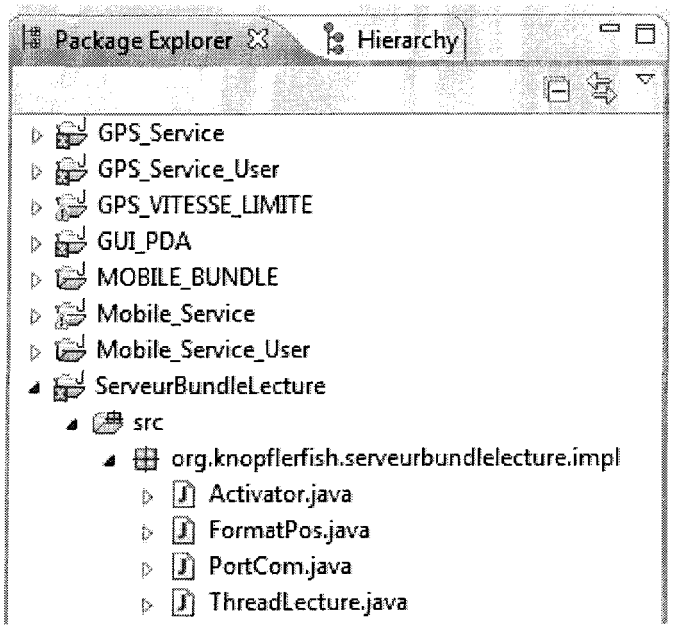

Figure 5.4 : Bundle « serveurbundlelecture » sous Eclipse

Activator.java : Cette classe active notre bundle sous OSGI. Elle permet d'invoquer la classe ThreadLecture.java pour i) ouvrir un port de communication radio COM12 et lire les messages radio reçus des équipements géo-localisables distants. ii) formater les messages reçus pour en extraire les données de localisation. iii) invoquer une méthode de connexion de base de données pour sauvegarder les données de géo-localisation extraites.

```
public class Activator implements BundleActivator {
      ...
}
      public void stop(BundleContext bc) throws Exception {
            ...
      thread.stopThread();
      thread.join();
      }
}
```

PortCom.java : Cette classe permet d'ouvrir un port de communication radio COM12.

```
public class PortCom {
...
      public  PortCom(String nomport) throws Exception {

                  /* portId reçoit l'identificateur du port choisi */
            CommPortIdentifier portId =
            CommPortIdentifier.getPortIdentifier(nomport);

            /* Ouverture port */
            port = portId.open("Connexion au GPS", 10000);
            …
            SerialPort monPort = (SerialPort) port;

            // configuration du port série
            monPort.setSerialPortParams(BAUD, SerialPort.DATABITS_8,
            SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);

            is = new BufferedReader(new
            InputStreamReader(port.getInputStream()));
            ...
            os = new PrintStream(port.getOutputStream(), true);
      }


      /* lecture du flux d'entrée  */
      public String Lire(){
            ...
            if (is.ready()==true){
            in=is.readLine();
            return in;
            ...
            }
      }

      /* Fermeture du port et fin de la communication radio sur le port
      COM1  */
      public void Fermercnx() throws IOException{
            ...
            os.close();
            is.close();
            port.close();
            }
}
```

FormatPos.java : Cette classe permet de prendre un message de géo-localisation et d'extraire chaque donnée de localisation qu'il contient (longitude, latitude, vitesse, précision de la position, etc.).

```
public class FormatPos{
```

```
...
public FormatPos(String msg){
      message=msg;
      /* extraction des informations de géo-localisation séparés par des
      ";" */
      Tab=message.split(";");
      }

      /*Lire(pos) permet de se positionner sur le numéro de la position
      dans le tableau Tab[] pour extraire les données qui y sont
      sauvegardées */
      private String Lire(int pos){
      ...
      return position;
}

public float Latitude(){
      ...
      return latitude;
}
/*Transformation des données de localisation dans leurs formats
respectifs depuis les chaines de caractères stockées dans Tab[]*/
public float Longitude(){
...
}
public String Qualite(){
...
}
public double Vitesse(){
...
}
public int PDOP(){
...
}
...
}
```

ThreadLecture.java : Cette classe utilise les classes précédentes pour invoquer les méthodes : de lecture des messages radio, d'extraction des données de localisation et sauvegarde de ces données dans une base de données sur notre serveur central.

```
public class ThreadLecture extends Thread {
...

      private void enregistrer(float latitude, float longitude, double
      vitesse, String source, String ID) {
            ...
```

```
            /* Enregistrement du pilote JDBC pour MySQL */
            Class.forName("com.mysql.jdbc.Driver").newInstance();
            ...
            Connection con = DriverManager.getConnection (...)
            ...
            id=  stmt.execute("SELECT id from position where
            point='"+ch+"'");
            /*Si on trouve une position semblable dans notre base dans la
            variable id précédente, on met à jour la ligne correspondante
            dans notre BD avec les nouvelles données de positionnement */
            if (id)  {
                  stmt.executeUpdate("UPDATE position " +"set
                  `point`='"+ch+"', `vitesse`='"+vitesse+"',
                  `precision`='"+source+"', `GPS_ID`='"+ID+
                  "' WHERE `id`="+id);
                  con.close();
            }
            // Sinon, il s'agit d'une nouvelle position que nous ajoutons
            à notre //base de données
            Else
            {
                  stmt.execute(
                  "INSERT INTO position (`point`, `vitesse`, `precision`,
                  `GPS_ID`)VALUES ('"+ch+"', '"+vitesse+"', '"+source+"',
                  '"+ID+"')");
                  con.close();
            }
      }
      ...
}

      /* constructeur du thread de notre classe */
      public ThreadLecture() {
            port=new PortCom("COM12");
            ...
            }

            /* la méthode run() est exécutée à l'invocation du thread */
            public void run() {
            ...
            while (running) {
            msg = port.Lire();
            /* S'il y a des données à lire dans la variable msg on cree
            une variable message qui contient les données de localisation
            traitées par la classe FormatPos */
            if (msg!="RAS"&&port.is.ready()==true){
                  ...
                  message=new FormatPos(msg);
            /*Si la précision est bonne et que l'indice PDOP est faible
            (==1) on enregistre la position dans notre base de données */
                  if (message.Precision()>3  && message.PDOP()==1) {
```

```
                    enregistrer(message.Latitude(),
                    message.Longitude(), message.Vitesse(),
                    message.Source(), message.Utilisateur());
                    ...}
            ...
            /* Arrêt de la boucle while du thread pendant 5 secondes pour
            recommencer le processus de lecture de nouvelles données */

            Thread.sleep(5000);
            ...
            }
            ...
}
```

5.3.2 Interface utilisateur

Comme on n'a pas développé de bundles spécifiques à OSGI pour fournir des interfaces utilisateurs, nous allons voir des interfaces développées avec des pages Web dynamiques via PHP-MySQL.

Index.html : il s'agit d'une page HTML qui contient un formulaire pour l'authentification avec un nom d'usager, un mot de passe et l'application à laquelle nous souhaitons accéder.

```
...
<form action="login.php" method='post'>/*on crée les champs du formulaire qui sont envoyés vers la page login.php qui va les traiter*/
...
  <td>Utilisateur :</td>
  <td><input type="text" name="login" maxlength="250"></td>/*champs avec nom d'utilisateur*/
...
  <td>Mot de passe </td>
  <td><input type="password" name="pass" maxlength="10"></td>/*champs avec mot de passe*/
 </tr>
...
/*bouton "acceder" pour envoyer les données du formulaire à la page login.php*/
  <td colspan="2" align="center"><input type="submit" value="acceder"></td>
```

```
  </tr>
</table>
</form>
```

Login.php: il s'agit d'une page en PHP qui va vérifier la concordance du nom d'utilisateur et son mot de passe, reçus du formulaire précédent, avec ceux stockés dans notre base de données. Si ces informations concordent, nous vérifions si l'utilisateur a droit d'accéder à l'application choisie. Si l'une de ces étapes de vérification échoue, nous affichons la première page d'authentification, sinon nous activons une variable `$_SESSION` qui garde l'utilisateur authentifié dans la page de l'application.

```
<?
/* Ouverture de la connexion à notre base de donnée*/
$link = mysql_connect("localhost", "agoundaf", "gpsuqac") or die("Could not connect: " .
mysql_error());
mysql_selectdb("agoundaf",$link) or die ("Ne peut utiliser dbmapserver : " .
mysql_error());
/*si tous les champs du formulaire sont remplis on continue le traitement*/
if(isset($_POST) && !empty($_POST['login']) && !empty($_POST['pass'])) {
 extract($_POST);
 /* on récupère le mot de passe de la table qui correspond au login du visiteur*/
 $sql = "select pass from authentification where login='".$login."'";
 $req = mysql_query($sql) or die('Erreur SQL !<br>'.$sql.'<br>'.mysql_error());
 $data = mysql_fetch_assoc($req);
//si le mot de passe ne concorde pas on remet le formulaire d'authentification
 if($data['pass'] != $pass)
        {
        ...
        include('index.html'); // On inclut le formulaire d'identification
         exit;
         }
 else /*si le mot de passe concorde, on vérifie si l'utilisateur a un droit d'accès à l'application
choisie*/
        {
        $sql = "select id from applications where login='".$login."' and app='".$select."'";
        $req = mysql_query($sql) or die('Erreur SQL !<br>'.$sql.'<br>'.mysql_error());
        $data = mysql_fetch_assoc($req);
```

```
        if ($data)
                {
                ...
                include('carte.html'); /* On inclut la page de l'application, dans cet exemple
                on inclus directement la carte des positions d'objets géo-localisables stockés
                dans notre base de données du serveur central et on ferme la page courante*/
                ...
                }
        else
                {
                include('index.html'); /* On inclut le formulaire d'identification si l'utilisateur
                n'est pas autorisé à accéder à l'application*/
                }
        }
}
else {/*si l'un des champs du formulaire est vide, on remet la page d'identification*/
        ...
        include('index.html'); // On inclut le formulaire d'identification
        ...
}
?>
```

carte.html: il s'agit d'une page en PHP et HTML qui va invoquer l'API GoogleMap pour afficher une carte avec les points correspondants aux positions relevées par nos équipements GPS. Les positions sont extraites via des requêtes SQL sur la base de données de notre serveur central. Des fonctions JavaScripts vont donner pour chaque point une couleur.

La classe « pointInPolygon » (voir pointInPolygon.php en annexe), permet de tester si un point de positionnement (longitude, latitude) se trouve dans une route ou une zone, définie dans notre base de données sous forme de polygone. Si nous trouvons un point inclut dans une zone, nous comparons sa vitesse à la vitesse limite correspondante à la zone et qui est définie dans la base de données. Si la vitesse du GPS dépasse la vitesse limite, nous utilisons la fonction de marquage pour créer un point en rouge. Si la vitesse est

inferieure à la vitesse limite, nous créons un point vert. Si le point se trouve dans une zone inconnue (que nous n'avons pas encore définie dans notre base de données), nous mettons le point en gris.

Les points affichés sont cliquables. Un clic permet d'afficher une info bulle sur le point pour afficher les informations complémentaires telles que : la vitesse, la latitude, la longitude, la qualité du relevé (donnée fiable si le PDOP =1 et la source est en 3D-GPS ou 3D-DGPS), la date de l'acquisition de l'information, l'identificateur du GPS qui a envoyé la donnée, etc.

```
...
<html>
...
/*invocation de l'API GoogleMap pour pouvoir créer une carte : il faut enregistrer le nom de domaine auprès de Google pour obtenir une clé afin d'avoir accès à l'affichage des cartes GoogleMap sur notre site web*/
<script
src="http://maps.google.com/maps?file=api&v=2&key=ABQIAAAAiIMtvyf4fUjlRHOz4c
oiGhR_oxCzrNtMB78bo91aW47W28DBtBSx3MqJ1q941AUWFCov1nzJV16yGw"
type="text/javascript"></script>
...
<body>
/*ici on met un formulaire avec un bouton de déconnexion pour finir la session utilisateur*/
<form  method="post" action="logoff.php">
...
</form>
...
/*ici on crée un cadre pour notre carte, la carte a une résolution choisie de 1360x768*/
<div align="left" id="map" style="width: 1360px; height: 768px">
  <script type="text/javascript">
//<![CDATA[
/* ici on crée des icônes de couleurs rouge, vert et gris*/
var iconR = new GIcon();// icône rouge
iconR.image = "http://labs.google.com/ridefinder/images/mm_20_red.png";
iconR.shadow = "http://labs.google.com/ridefinder/images/mm_20_shadow.png";
iconR.iconSize = new GSize(12, 20);
```

```
iconR.shadowSize = new GSize(22, 20);
iconR.iconAnchor = new GPoint(6, 20);
iconR.infoWindowAnchor = new GPoint(5, 1);

var iconV = new GIcon();// icône verte
...

var iconG = new GIcon();// icône grise
...

/* Fonction de création de points et utilisation des icones précédentes*/
function Marker_rouge(point) {
var marker = new GMarker(point, iconR);
return marker;
}

function Marker_vert(point) {
...
}

function Marker_gris(point) {
...
}
/*creation de la carte dans la variable "map" et ajout des controles qui permettent de
deplacer, zoomer et changer le type d'affichage pour la carte*/
var map = new GMap2(document.getElementById("map"));
map.addControl(new GLargeMapControl());
map.addControl(new GMapTypeControl());
map.addControl(new GScaleControl());

/* Les fonctions suivantes créent des points de couleurs rouge, vert et gris cliquables */
function createMarkerR(point, text)
{
var marker1 = new Marker_rouge(point);
/* Ajoute un texte à l'infobulle qui s'affiche en cliquant sur le point */
var html = text;
GEvent.addListener(marker1, "click", function() {marker1.openInfoWindowHtml(html);});
return marker1;
};

function createMarkerV(point, text)
{
...
};
```

```
function createMarkerG(point, text)
{
...
};

<?php
 ...
/*Création de lien à notre base de données*/
$lnk = mysql_connect("localhost", "agoundaf", "gpsuqac") or die("Could not connect: " .
mysql_error());
mysql_selectdb("agoundaf",$link) or die (...);
/*on sélectionne toutes les données de géo-localisation de tous les objets géo-localisables*/
$rs1 = mysql_query("SELECT * FROM position",$link);
...
/* ajout de la classe pointLocation incluse dans le fichier PointInPolygon.php qui permet de
tester si un point est inclut dans un polygone comme une route par exemple*/
include_once("pointInPolygon.php");
/*extraction de point par point jusqu'à la fin de la table position*/
while($row1 = mysql_fetch_array($rs1))
{
 $qzone="SELECT * FROM  zone";

 $rs2 = mysql_query($qzone,$link);
 $trouve=false;

/*extraction de zone par zone jusqu'à la fin de la table zone en testant pour chaque point de
la table position s'il est inclut dans une zone de la table zone.*/
 while($row2 = mysql_fetch_array($rs2))
  {
...
/*Teste si le point $point est inclut dans la zone $polygon */
  $inclus=$pointLocation->pointInPolygon($point, $polygon);
  ...


  /* Si le point est inclut dans une zone, on compare la vitesse relevée à la vitesse limite de
la zone*/
  if (($inclus==1) and ($row1['vitesse']>$row2['limite']))
   {
/* On génère le code pour afficher un point en rouge quand la vitesse de l'objet géo-
localisable dépasse celle de la vitesse limite de la zone où il se trouve  */
 echo "var point = new GLatLng(" . $coord['x'] . "," . $coord['y'] . ");\n";
```

```
 echo "var marker = createMarkerR(point, 'vitesse:" . addslashes($row1['vitesse']) ." <br>
Source:". addslashes($row1['precision']) . " <br> date:". addslashes($row1['dat']) ."<br>
_____ <cite><em>Copyright A.Goundafi</em></cite>');\n";
 echo "map.addOverlay(marker);\n";
 echo "\n";
 $trouve=true;
 break;
  }

 /* Si la vitesse est inferieure à la vitesse limite on met le point en vert*/
 else if($inclus==1 and $row1['vitesse']<=$row2['limite'])
  {
...
echo "var marker = createMarkerV(point, 'vitesse:" . addslashes($row1['vitesse']) ."
...
  }
 }//fin while zones

/*Si le point se trouve dans une zone qui ne figure pas dans notre base de données on met le
point en gris*/
 if($trouve==false)
  {
...
echo "var marker = createMarkerG(point, 'vitesse:" . addslashes($row1['vitesse']) ." <br>
...
  }
}//fin while points
...
?>
/*Apres avoir créé tous les points de positionnement, on affiche une carte centrée sur les
coordonnées que nous voulons*/
map.setCenter(new GLatLng(48.39555, -71.08789), 11, G_HYBRID_MAP);
//]]>
 </script>
...
</html>
```

5.4 Scenarios d'exécution des bundles et interfaces web

Dans cette section, nous allons voir l'exécution et les interactions entre les différents bundles de notre système ainsi que l'application web pour la visualisation des données

récoltées depuis les dispositifs géo-localisables. La figure 5.5 montre l'intergiciel knopflerfish avec les deux bundles côté objets géo-localisables, « gpsservice » et « gpsserviceuser ». Du coté serveur, le bundle « serveurbundlelecture ».

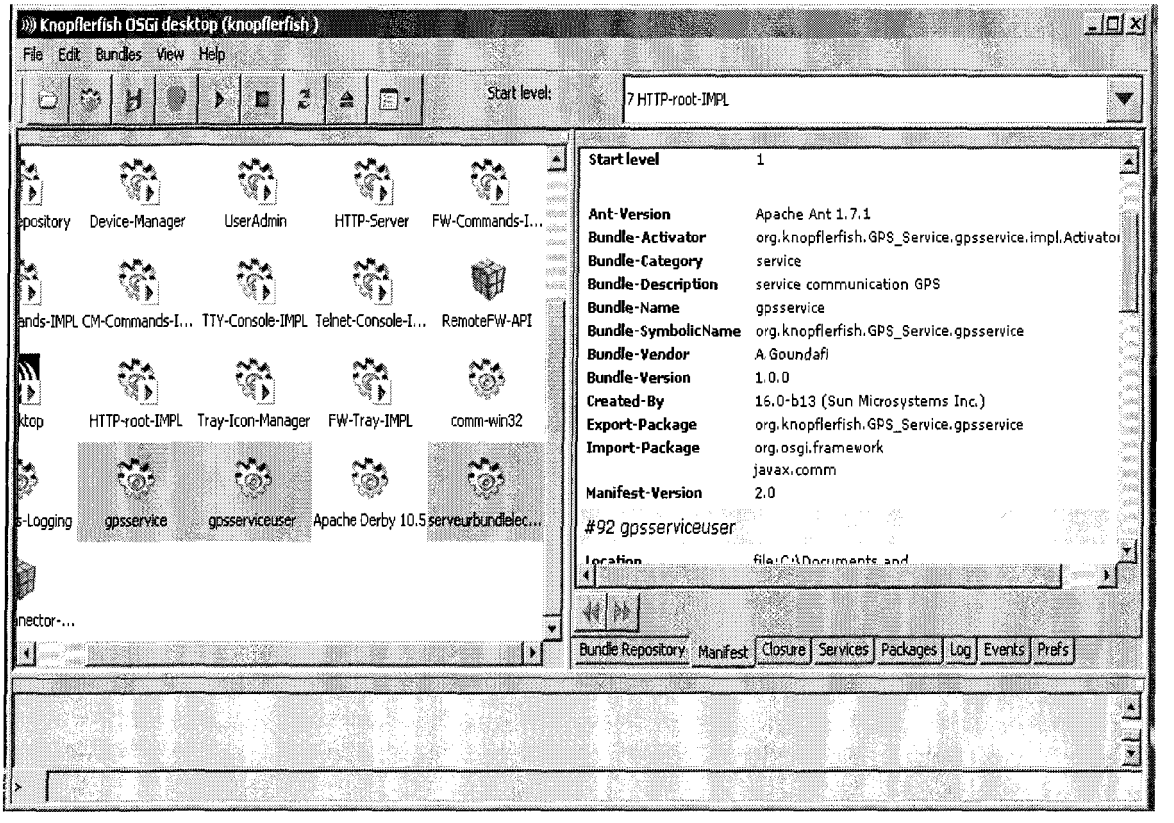

Figure 5.5 : Environnement d'exécution des bundles Knopflerfish

5.4.1 Bundles côté dispositifs géo-localisables

Il s'agit ici de l'exécution des bundles dans un environnement OSGI via l'intergiciel knopflerfish. Ces bundles peuvent s'exécuter sur n'importe quel dispositif mobile où JME (Java Micro Edition) est présente pour supporter l'intergiciel. Ces bundles sont les suivants :

gpsservice: Ce bundle enregistre un service utilisable par d'autres bundles pour se connecter sur un dispositif GPS et en extraire les données de positionnement.

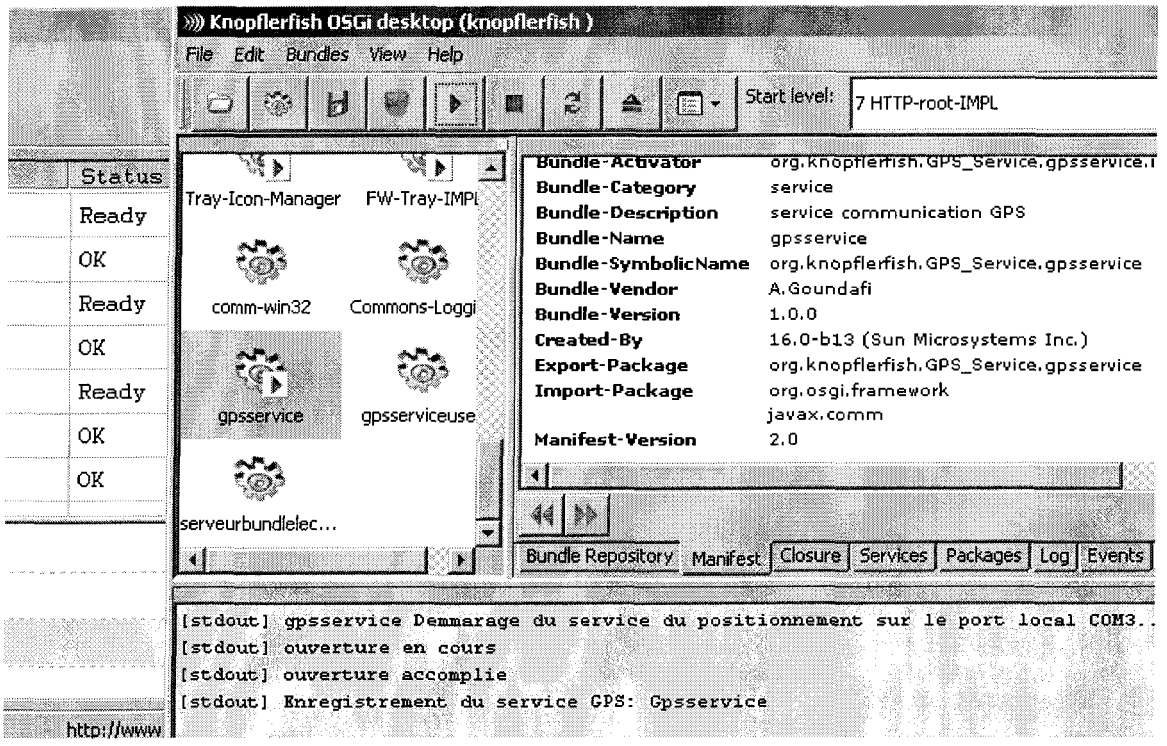

Figure 5.6 : Bundle « gpsservice » sous Knopflerfish

Il est à noter que ce bundle importe les librairies de communication javax.comm fournies par le bundle qui apparait dans la figure 5.6 sous le nom de comm-win32. Ce bundle de communication est actuellement disponible en version 32 bits seulement.

gpsserviceuser: Ce bundle enregistre et utilise le service du bundle « gpsservice » pour se connecter sur le GPS au port de communication COM3 et extraire les données de positionnement. Une fois les données extraites et formatées par ce bundle, il utilise encore

une fois le service « gpsservice » pour envoyer les données de positionnement vers notre serveur central en utilisant le port radio COM10. La figure 5.7 montre l'exécution de ce bundle.

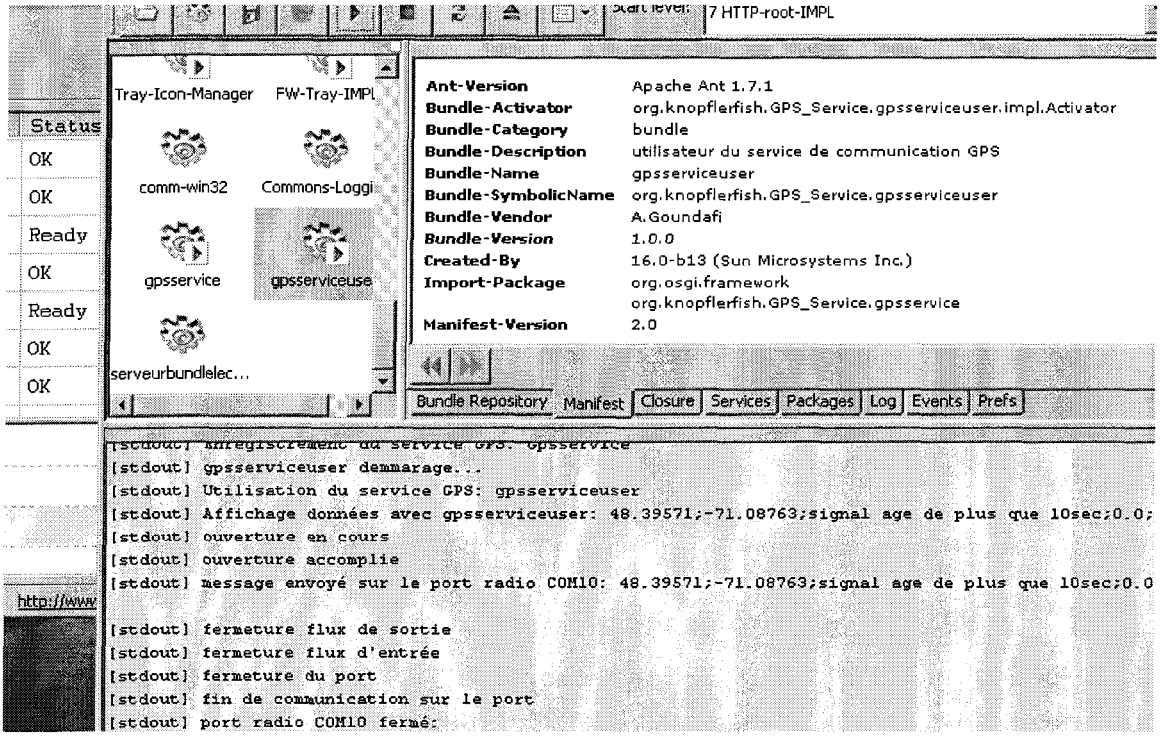

Figure 5.7 : Bundle «gpsserviceuser» sous Knopflerfish

guipda: Ce bundle permet d'utiliser le service « gpsservice » pour acquérir les données de localisation et les afficher dans une fenêtre graphique consultable sur l'appareil mobile.

La figure 5.8 montre le démarrage de l'interface graphique du bundle avec tous les champs de données initialisés.

La figure 5.9 montre un signal relevé qui est âgé de plus de 10 secondes. Le code couleur est mis en jaune pour signaler que même si la source de données est bonne et que le

PDOP est faible, les données sont moins fiables puisqu'elles ne reflètent pas la position actuelle vu l'âge des données.

La figure 5.10 montre un signal fraichement relevé avec une bonne source de données en 3D-GPS, ce qui implique que plus de trois satellites sont utilisés dans le calcul de la position. Cependant, ces données ne sont pas très fiables car le PDOP est élevé, possiblement à cause de la position des satellites très rapprochés.

La figure 5.11 montre des conditions parfaites pour sauvegarder les données de positionnement sur la base de données de notre serveur central. Le code couleur en vert se traduit par la coexistence d'une bonne source de données (3D-GPS ou 3D-DGPS), d'un PDOP faible et d'un relevé de signal à la seconde même. Dans la figure 5.12, c'est un cas de figure où le signal est fraichement relevé mais le PDOP est élevé et la source de données est en 2D-GPS, ceci implique que seulement 3 satellites sont visibles au GPS et que ces satellites sont trop rapprochés.

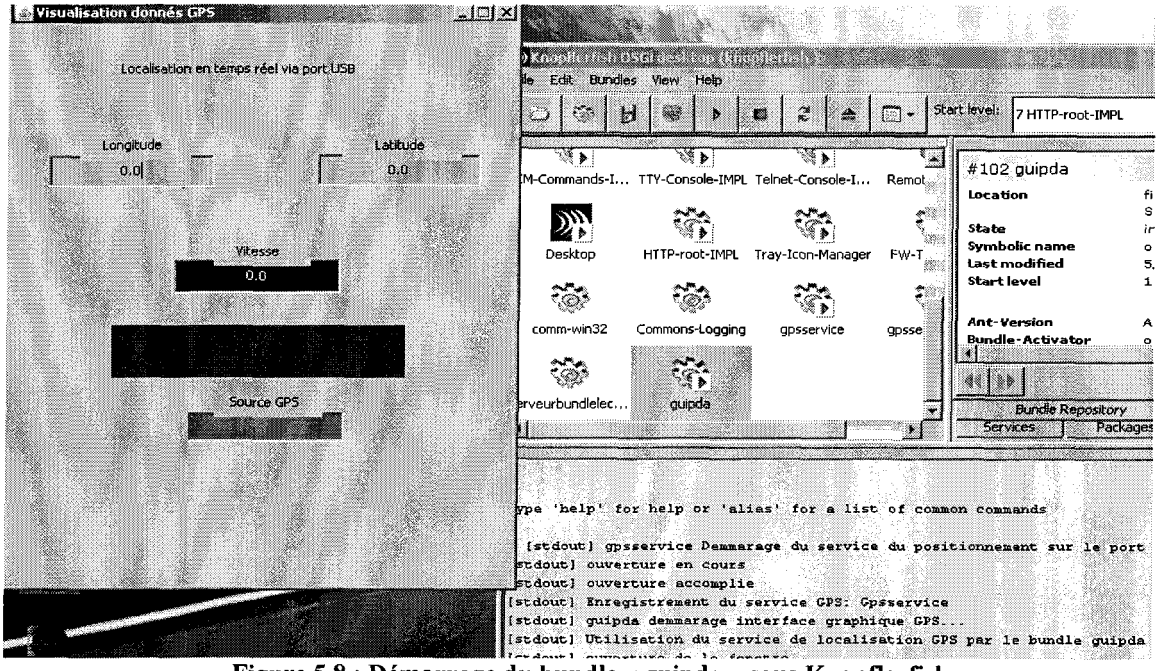

Figure 5.8 : Démarrage du bundle « guipda » sous Knopflerfish

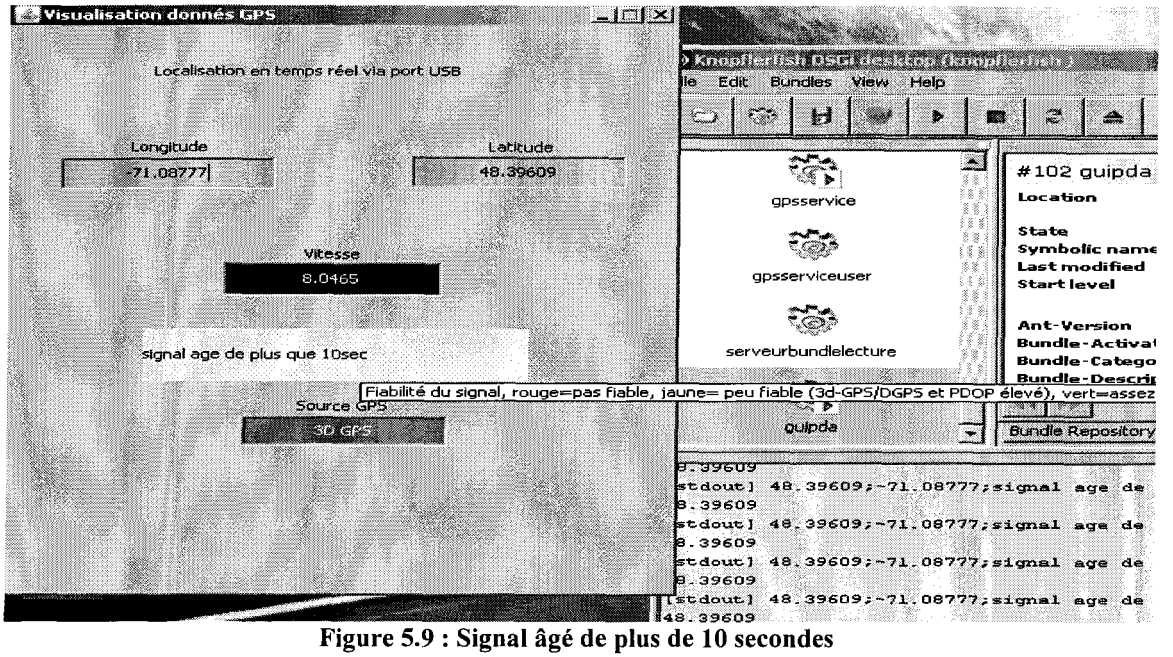

Figure 5.9 : Signal âgé de plus de 10 secondes

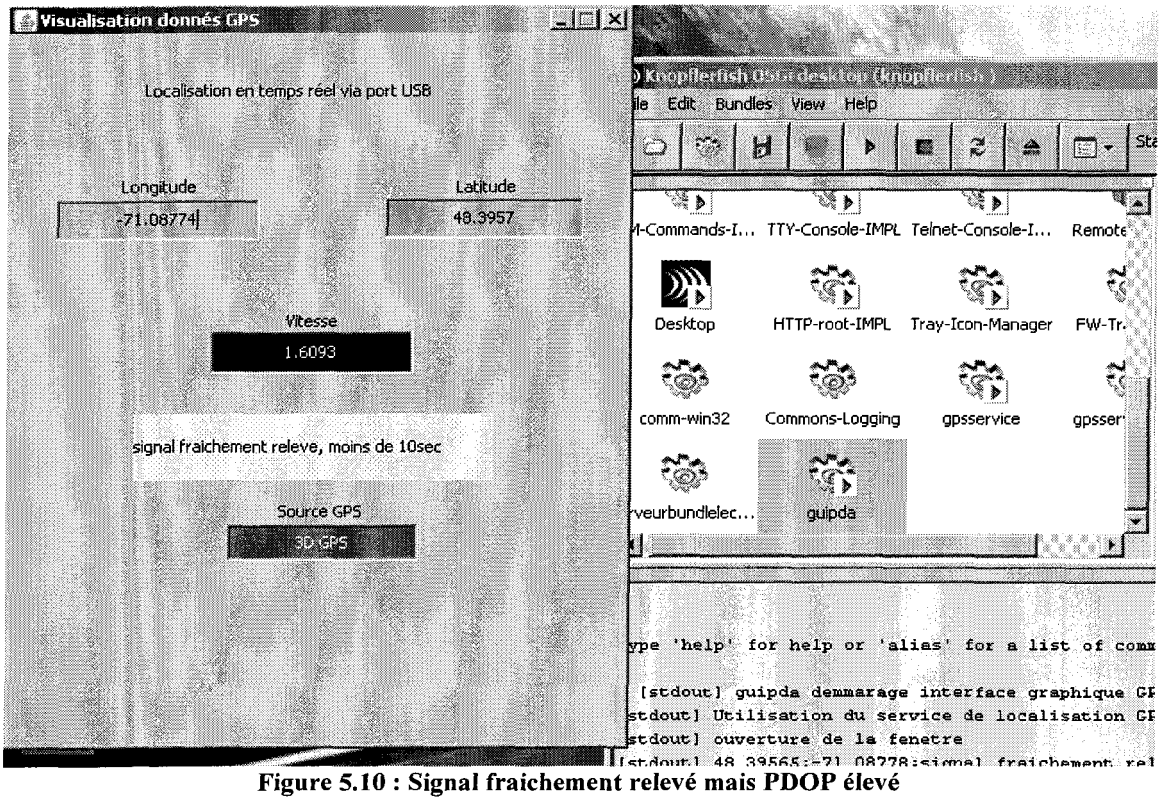

Figure 5.10 : Signal fraichement relevé mais PDOP élevé

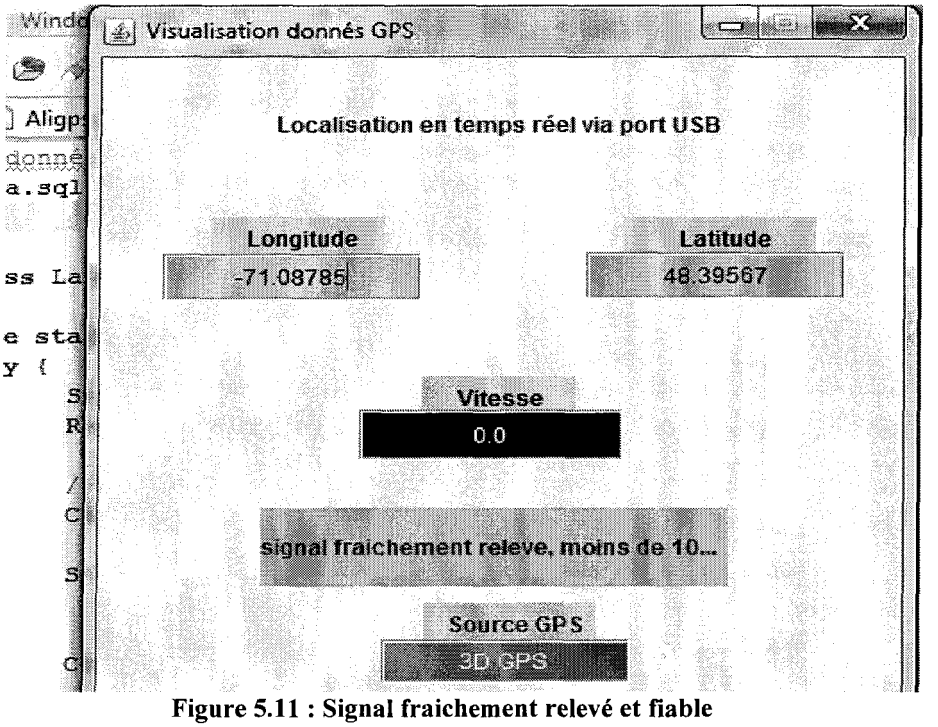

Figure 5.11 : Signal fraichement relevé et fiable

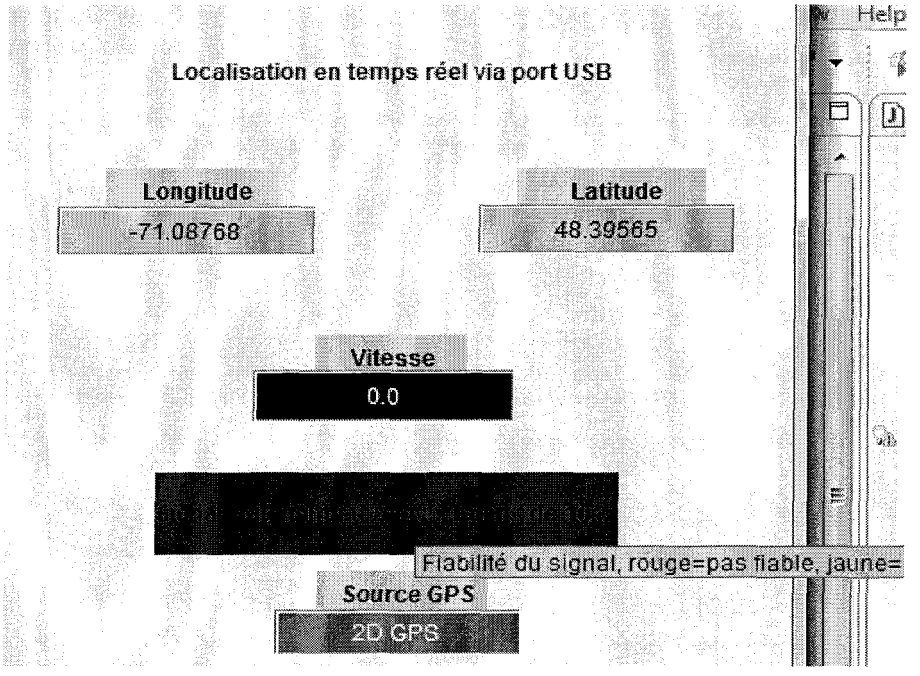

Figure 5.12 : Signal fraichement source faible et PDOP élevé

5.4.2 Bundle côté serveur

Dans cette section je présente le bundle « serveurbundlelecture » installé sur l'intergiciel OSGI de notre serveur central. Ce bundle se connecte sur le port radio COM12 pour intercepter les messages radio contenant les données de géo-localisation. Ces données de géo-localisation sont formatées et stockées dans notre base de données dans la table contenant les positions de nos objets géo-localisables.

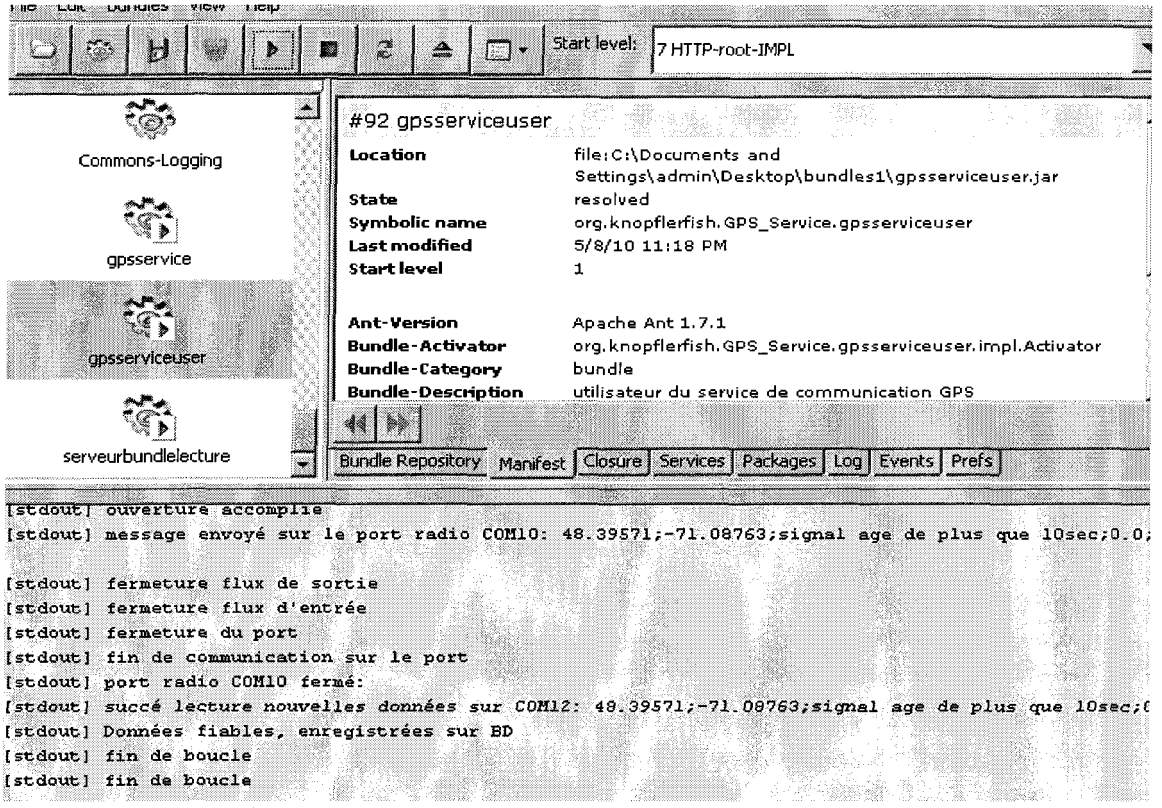

Figure 5.13 : Bundle «serveurbundlelecture» sous Knopflerfish

5.4.3 Application Web côté serveur

Dans cette section je présente les captures d'écran de l'interface web pour l'authentification et l'affichage de la carte avec les positions et les informations relatives pour chaque point affiché. L'interface web est accessible via le lien http://sunensweb.uqac.ca/~agoundaf/.

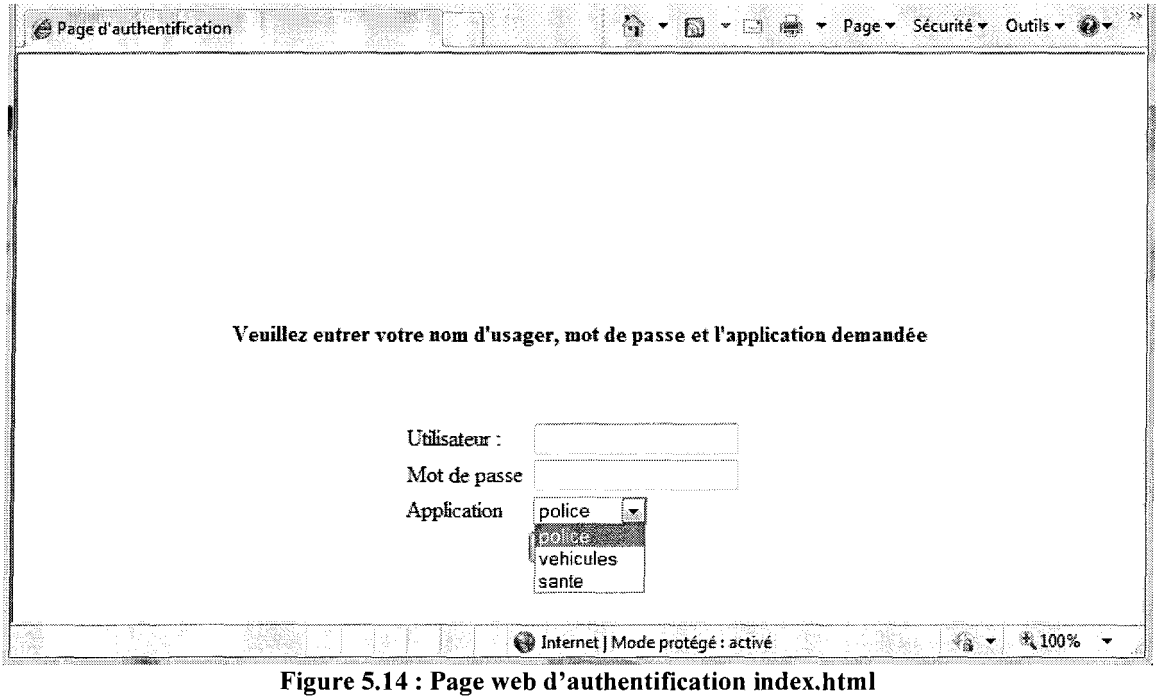

Figure 5.14 : Page web d'authentification index.html

La figure 5.13 montre l'interface web d'authentification où l'utilisateur doit entrer son nom d'usager, son mot de passe et l'application à laquelle il souhaite accéder. Une fois les données entrées, il doit cliquer sur le bouton « acceder » pour être redirigé vers l'application choisie. Si l'utilisateur oublie de rentrer l'un des champs d'authentification, il sera notifié par un message lui rappelant de rentrer tous les champs comme le montre la figure 5.14. Si le login ou/et le mot de passe sont incorrects, il sera notifié que le nom de l'usager ou le mot de passe sont incorrects comme montré dans la figure 5.15. Si le login et le mot de passe sont corrects et que l'utilisateur n'est pas autorisé à utiliser l'application choisie, il en sera notifié comme le montre la figure 5.16. Si toutes les informations d'authentification sont correctes et que l'utilisateur a droit d'accéder à l'application demandée, il sera dirigé vers son application comme le montre la figure 5.17. Une variable

session sera activée dans toutes les pages auxquelles l'utilisateur a accès (voir login.php en annexe).

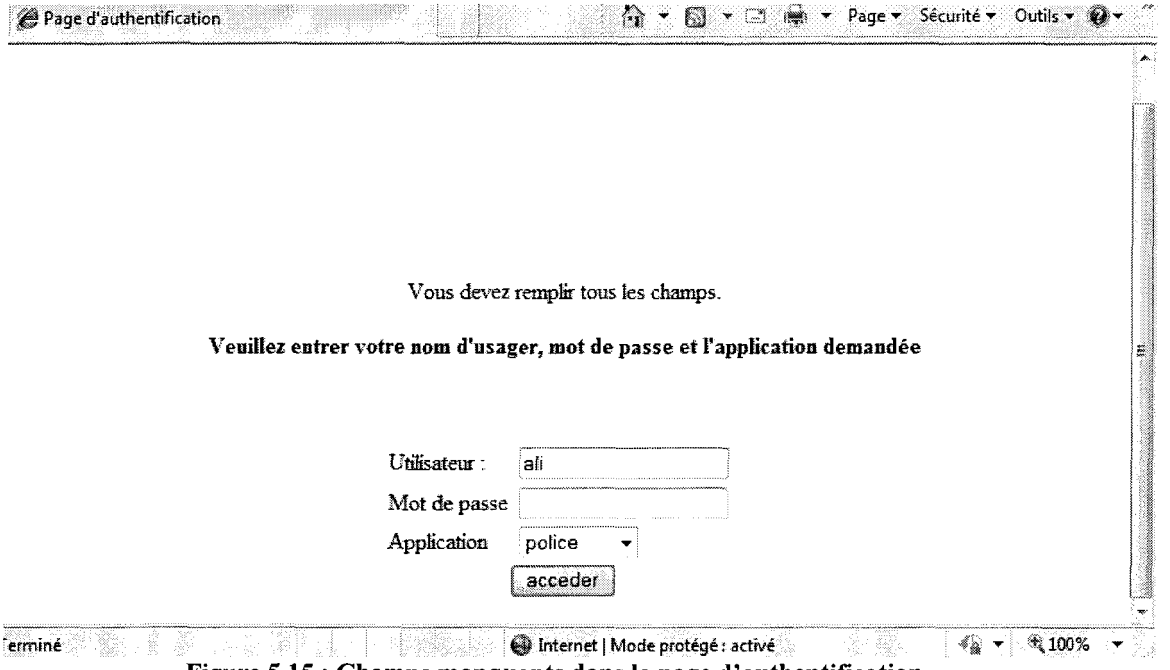

Figure 5.15 : Champs manquants dans la page d'authentification

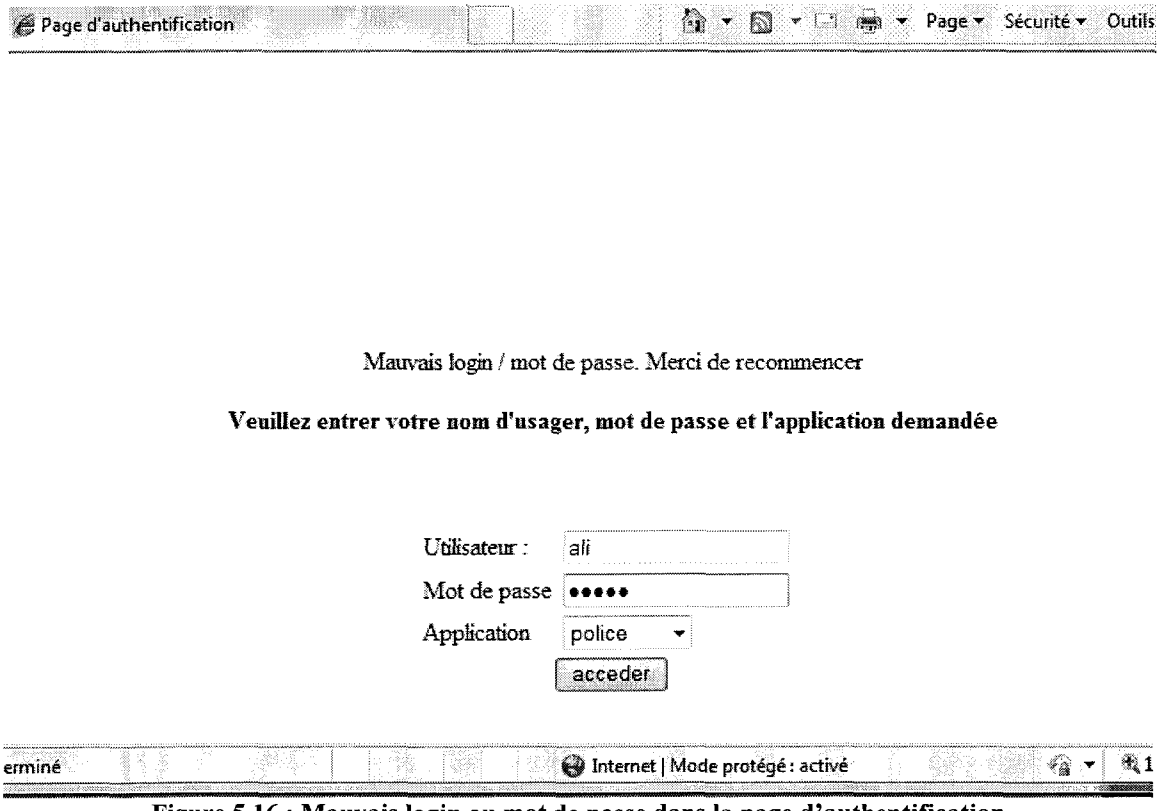

Figure 5.16 : Mauvais login ou mot de passe dans la page d'authentification

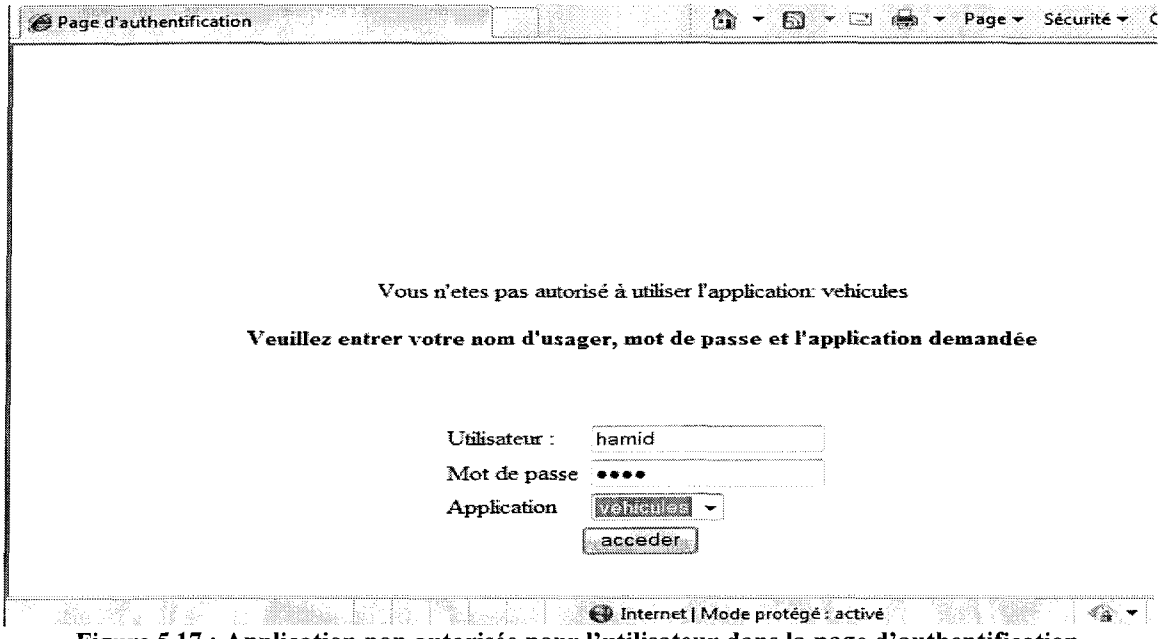

Figure 5.17 : Application non autorisée pour l'utilisateur dans la page d'authentification

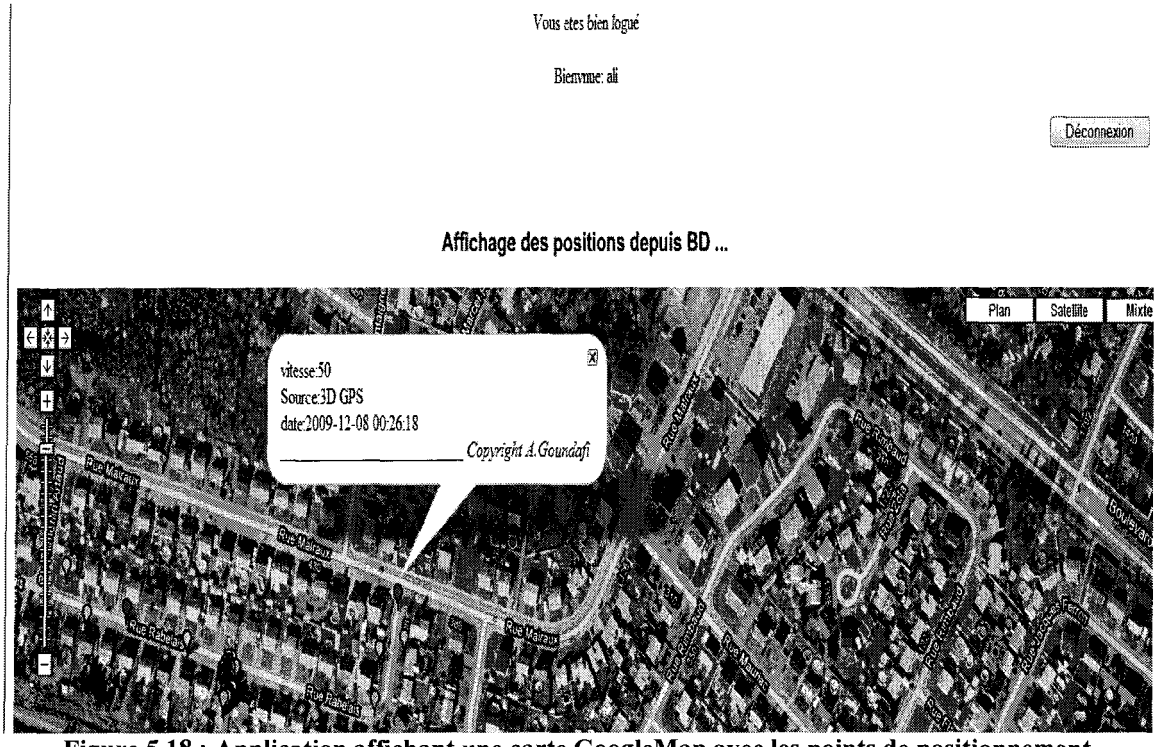

Figure 5.18 : Application affichant une carte GoogleMap avec les points de positionnement

Pour finir sa session, l'utilisateur doit cliquer sur le bouton déconnexion dans le coin supérieur droit (voir figure 5.18). La déconnexion permet de détruire la variable session de l'utilisateur pour éviter qu'une autre personne non authentifiée, puisse utiliser l'application. La déconnexion remet la première page d'authentification (figure 5.14) à l'écran.

5.4.4 Outil de simulation des ports radio

Pour pouvoir simuler une communication radio, nous verrons dans cette section un logiciel qui permet de créer des ports virtuels que l'on peut configurer (débit des données, parité, etc.) comme nous le faisons pour de vrais ports de communication série d'un modem radio. Le logiciel en question est VSPE « Virtual Serial Port Emulator» de la compagnie « eterlogic ». Ce logiciel permet aussi de créer des ports TCP/IP avec un numéro de port

pour diriger le flux de données vers un autre port virtuel sur un autre hôte sans aucune programmation de notre part. Cette fonctionnalité nous permet de simuler des envois à distance via nos ports virtuels. Je vous propose ici un bref tutoriel pour utiliser cet outil.

5.4.4.1 Création d'un port série

La création d'un port série virtuel (connector sous VSPE) permet au système de voir ce port comme un port physique. Les figures 5.19 et 5.20 montrent les étapes pour créer ce port au sein de VSPE.

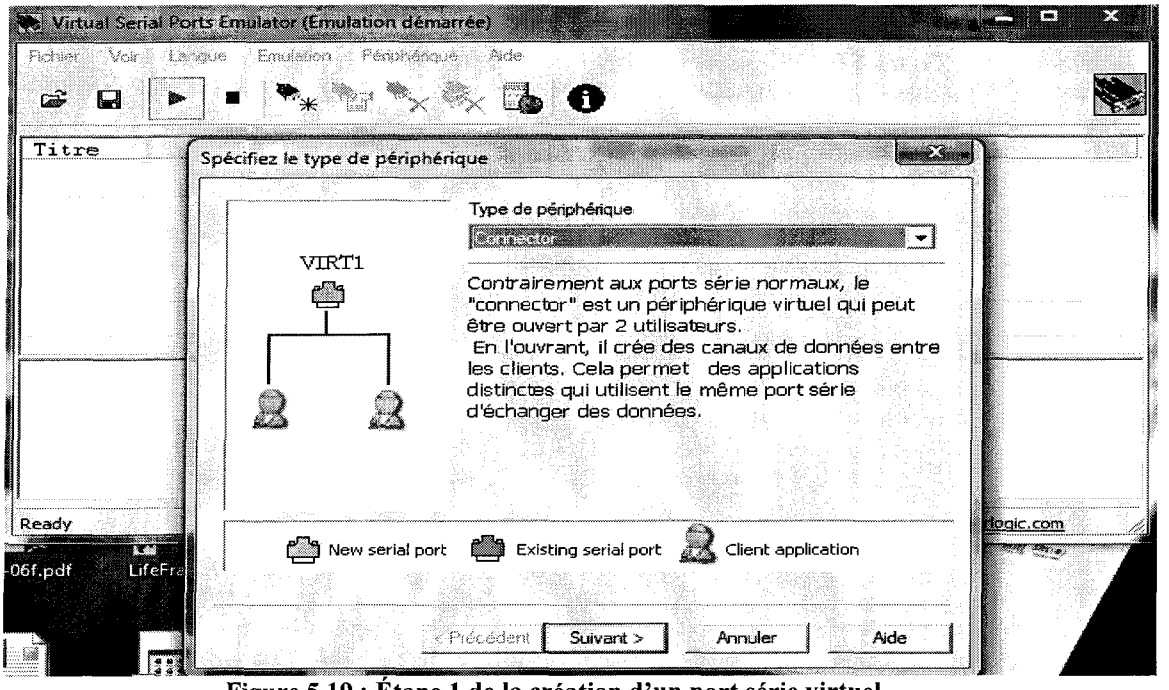

Figure 5.19 : Étape 1 de la création d'un port série virtuel

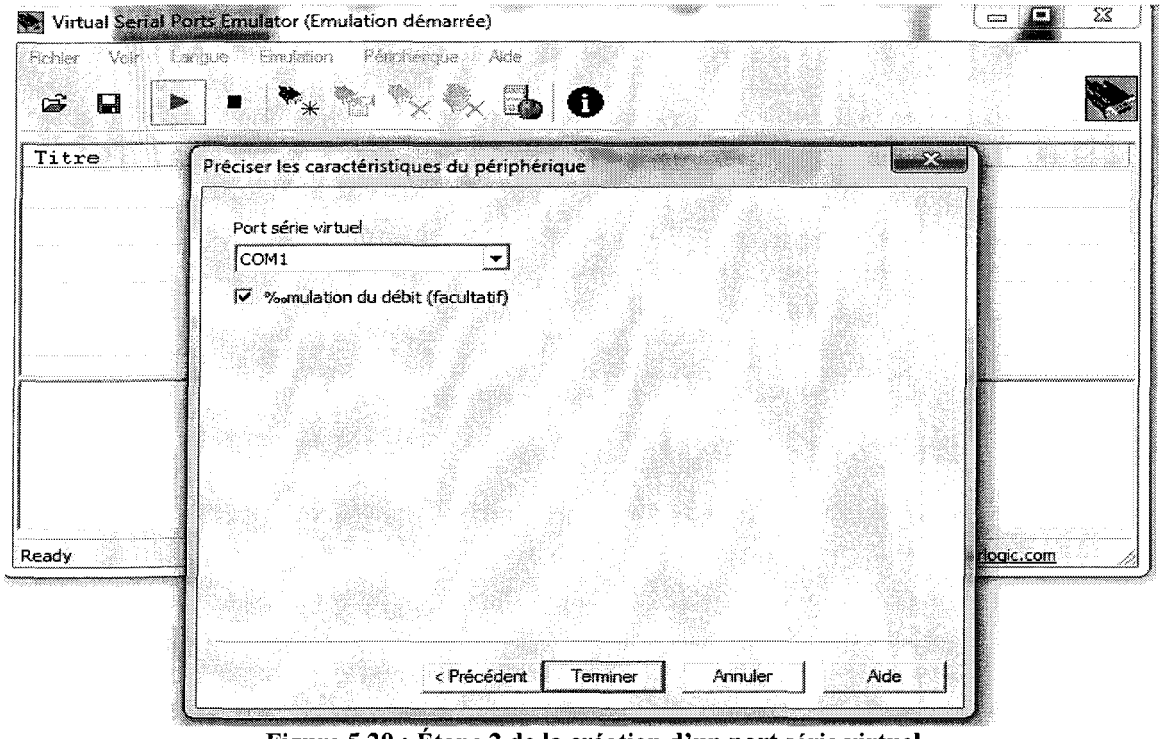

Figure 5.20 : Étape 2 de la création d'un port série virtuel

Dans la figure 5.20 nous pouvons choisir n'importe quel port non utilisé dans le système, dans ce cas le port COM1 est inutilisé par le système et nous pouvons donc créer un port virtuel COM1. En cochant la case « simulation du débit », les échanges de données via ce port vont se faire avec le débit spécifié par l'application exprimé en Baud.

5.4.4.2 Création d'un « splitter »

Un « splitter » est un port virtuel qui permet de partager un même port série entre plusieurs applications en même temps. Il est utile en cas de test d'applications qui partagent le même port. Si une application bloque il est plus facile de libérer le port connecté avec un splitter que de libérer un port directement lié à l'application. Les figurent 5.21 et 5.22

montrent les étapes pour créer un splitter en se basant sur un port physique ou virtuel déjà existants.

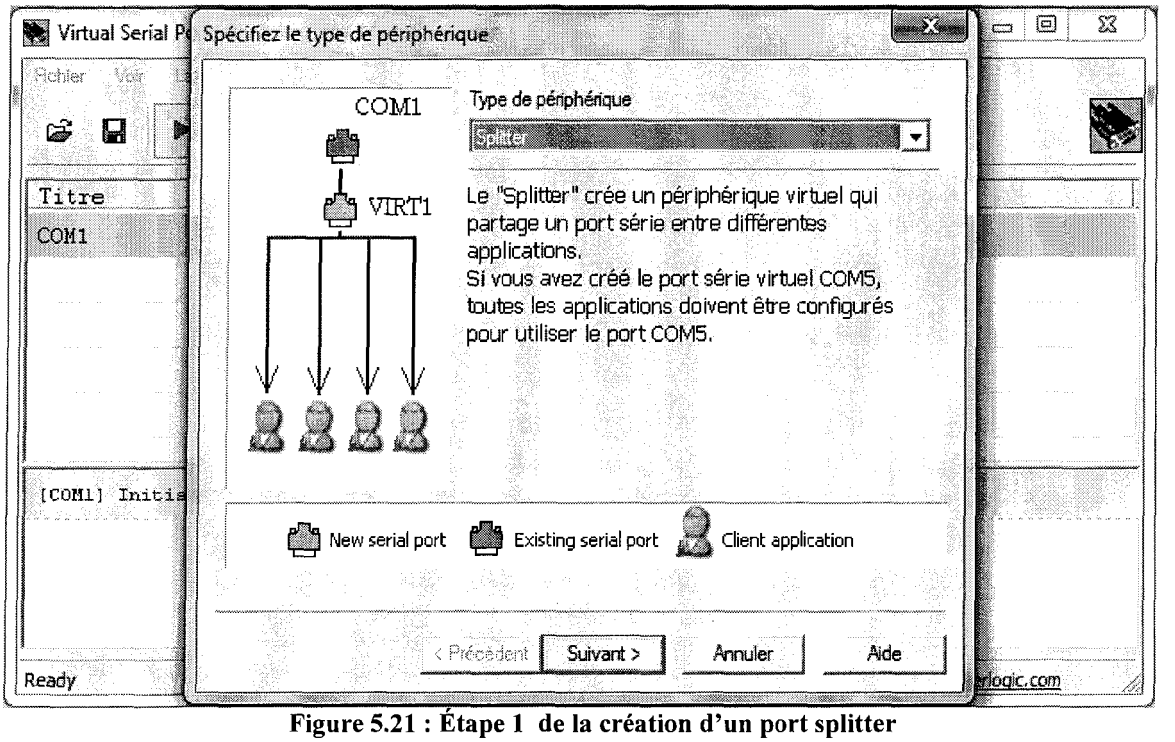

Figure 5.21 : Étape 1 de la création d'un port splitter

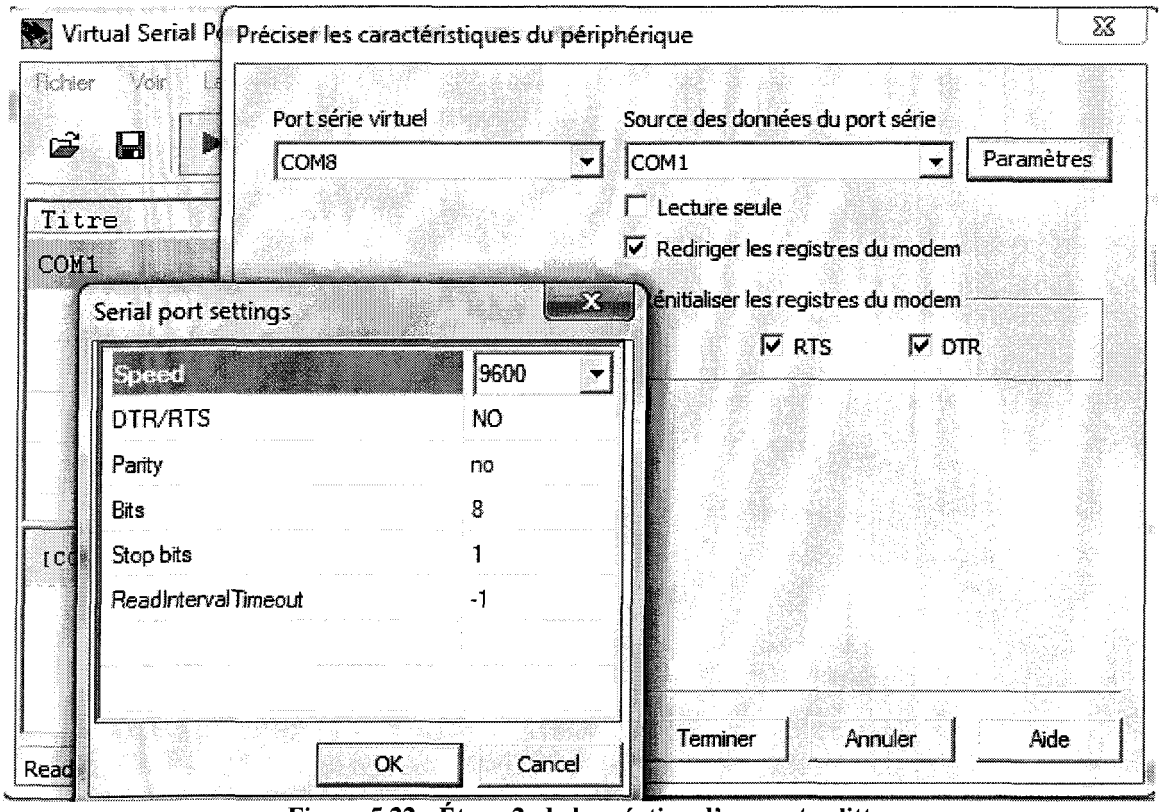

Figure 5.22 : Étape 2 de la création d'un port splitter

Dans la figure 5.22, nous choisissons le port que nous voulons partager dans la liste déroulante « source de données de port série ». Nous choisissons le port COM1 créé précédemment et nous lui associons le port partagé COM8 dans la liste déroulante « port série virtuel ». En cliquant sur le bouton paramètres, la fenêtre « serial port settings » s'ouvre pour définir la vitesse de transmission des données, la parité, le nombre de bits de données, etc.

5.4.4.3 Port « TcpClient »

Le port « TcpClient » permet de créer un port internet pour envoyer les données vers un serveur distant moyennant son adresse IP. Les figures 5.23 et 5.24 montrent les étapes de la création de ce port.

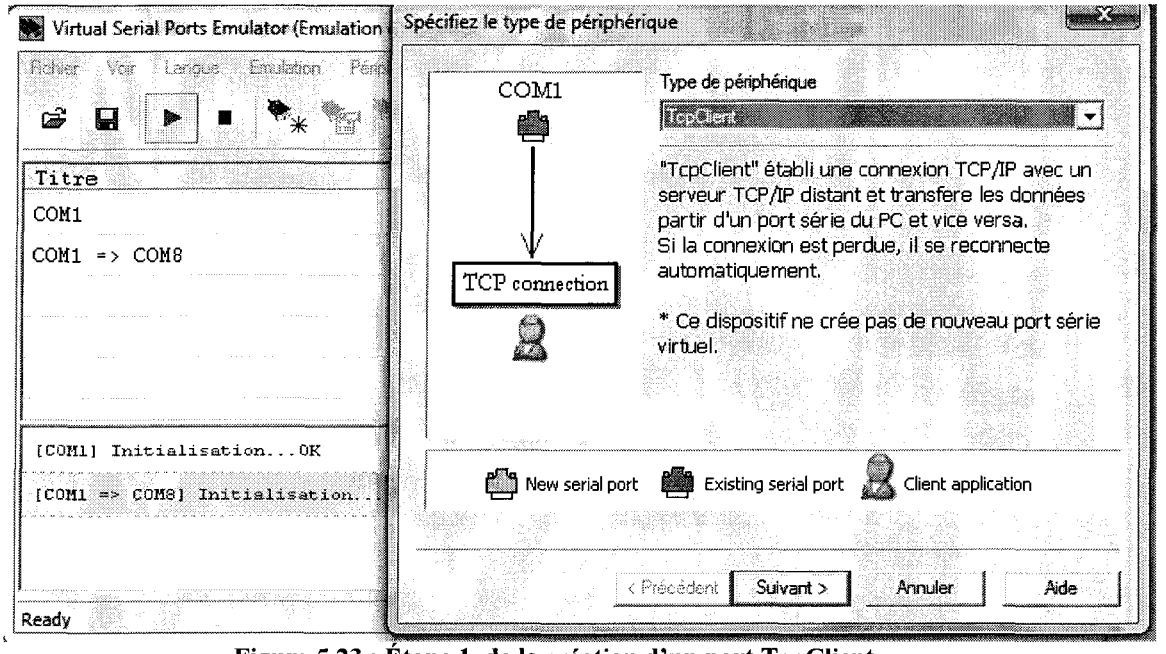

Figure 5.23 : Étape 1 de la création d'un port TcpClient

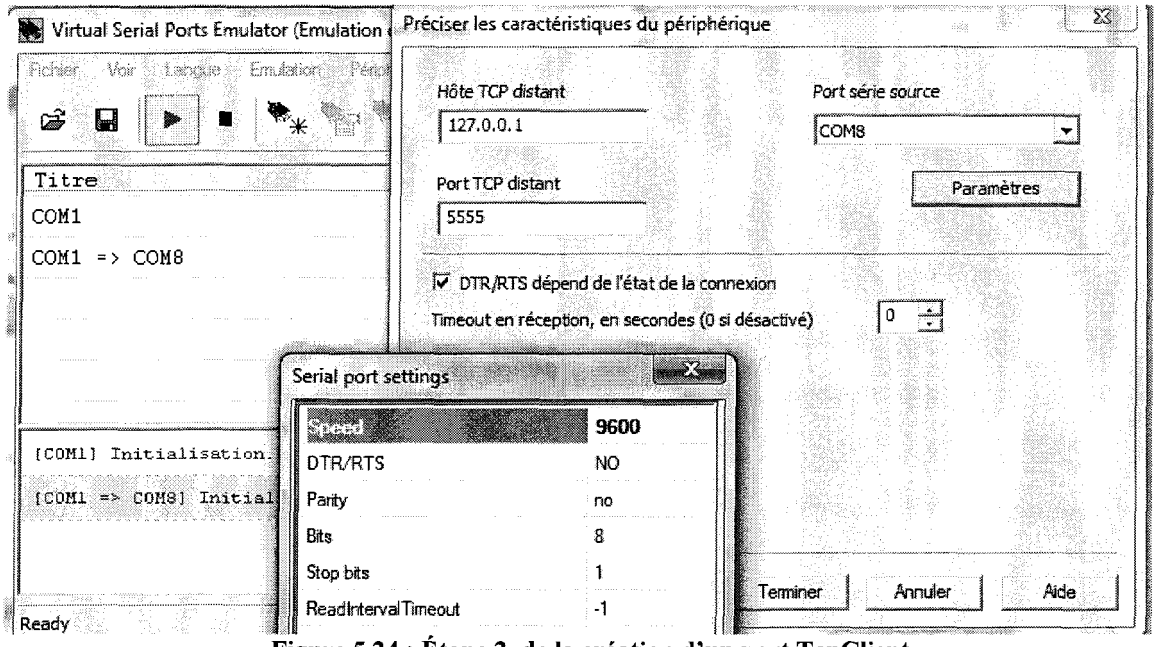

Figure 5.24 : Étape 2 de la création d'un port TcpClient

Dans la figure 5.24, nous prenons comme source de données à rediriger vers le serveur distant le port COM8. Toutes les données écrites sur le port COM8 sont transmises vers l'adresse IP spécifiée dans le champ « hôte TCP distant ». Le port distant sur lequel nous devons transmettre ces données est spécifié dans le champ « port TCP distant ». Les autres paramètres du port comme la vitesse sont spécifiés à l'aide du bouton « paramètres ».

5.4.4.4 Port « TcpServer »

Le port « TcpServer » permet à l'application côté serveur d'écouter sur un port TCP, par exemple sur le port 5555 spécifié précédemment dans le port TcpClient COM8. Les étapes pour la création de ce port sont montrées dans les figures 5.25 et 5.26.

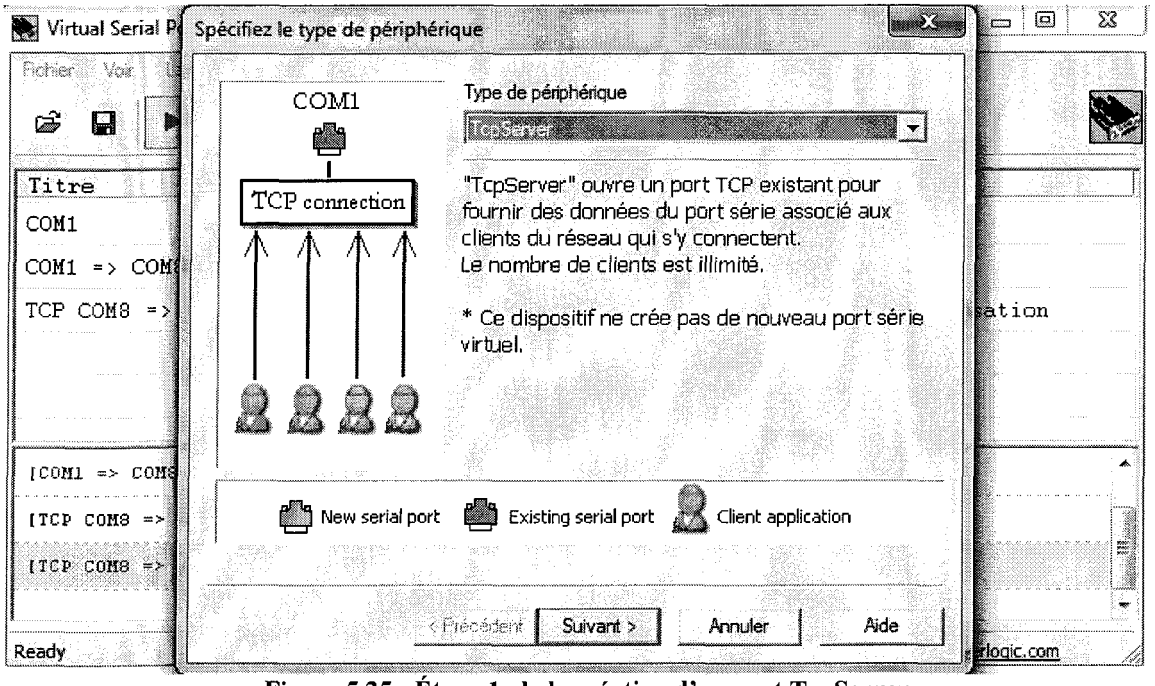

Figure 5.25 : Étape 1 de la création d'un port TcpServer

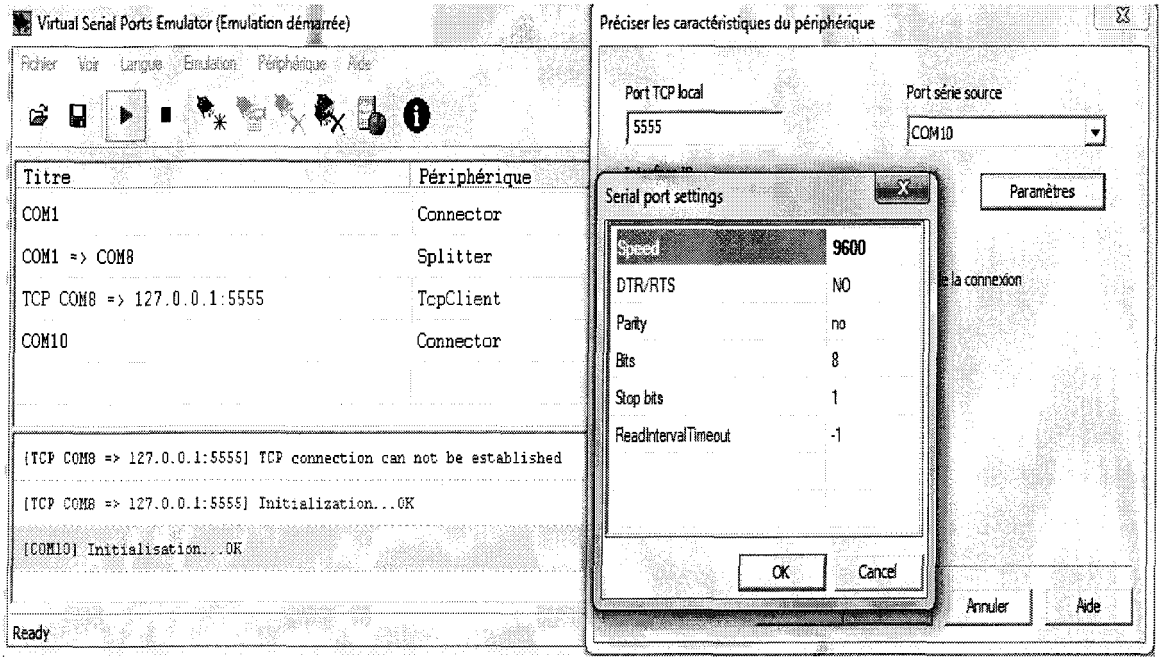

Figure 5.26 : Étape 2 de la création d'un port TcpServer

Dans la figure 5.26, nous avons préalablement un port série de lecture COM10 (virtuel ou physique) auquel nous associons un port TCP 5555 dans le champ « port TCP local ». Le port COM10, sur lequel nous lisons les données distantes envoyées via le port COM8 du client, est spécifié dans le champ « port série source ». Les paramètres du port sont personnalisés via le bouton « paramètres ».

5.4.4.5 Configurations des ports pour l'application GPS centralisée

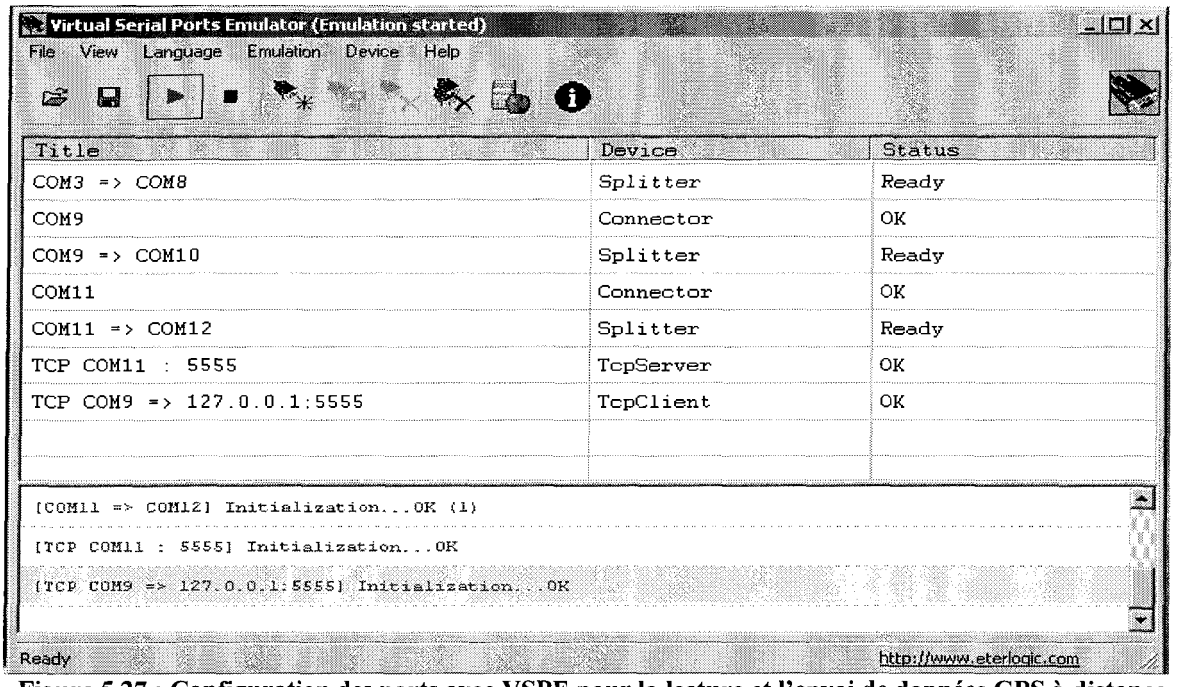

Figure 5.27 : Configuration des ports avec VSPE pour la lecture et l'envoi de données GPS à distance

La figure 5.27 montre les ports COM séries virtuels et physiques et les ports TCP, utilisés via le logiciel VSPE. Le port physique sur lequel est connecté le dispositif GPS est le COM3. Ce port GPS est partagé via le port splitter COM8. Le port COM9 est un port TCP/IP qui envoie les données qu'il reçoit vers une adresse IP (dans ce cas c'est l'adresse locale 127.0.0.1) en spécifiant le numéro de port TCP distant (5555) que le serveur utilise pour lire ces données. Le port COM11 est le port distant qui se trouve sur le serveur pour intercepter les données GPS envoyées depuis le client via le port COM9. Le port COM11 écoute sur le port TCP 5555. Le Port COM12 permet de partager le port COM11.

En résumé, le bundle « gpsservice » lit les données GPS sur le port COM8 qui est en réalité le port COM3 physique du GPS. Le bundle « gpsserviceuser », utilise les données

lues via « gpsservice » et les envoi via le port COM10 du modem radio, qui est dans notre application le port COM9 TCP client. Le port COM12 est le port radio du coté serveur, il est représenté dans notre application par le port TCP serveur COM11.

VI

CONCLUSION

Le présent mémoire vient d'apporter une solution quant à la mise en œuvre d'un système de géo-localisation englobant plusieurs services indispensables dans nôtre société tels que :

a. La gestion du trafic routier en assurant un suivi des véhicules et un contrôle de la vitesse de façon dynamique,

b. La gestion des patients atteints d'Alzheimer et qui risquent de se perdre et par conséquent mettent leurs vies en danger, ou les patients atteints d'une maladie cardiaque et qui risquent un arrêt cardiaque à tout moment,

c. La gestion d'un groupe de véhicules permettant une intervention rapide comme c'est le cas d'urgences médicales (comme les cas mentionnés au point b) où le système va repérer les unités médicales à proximité du lieu de l'incident (par exemple des ambulances) et les guider vers ce lieu, etc. Le système ayant une vision globale de toutes les composantes des services gérés, peut orchestrer les opérations d'intervention de façon automatisée et rapide.

Pour pouvoir réaliser cette application, il a fallu faire un tour d'horizon concernant ce qui se fait actuellement en matière de géo-localisation, en mettant en lumière les différents systèmes, que ce soit les systèmes basés sur des constellations de satellites ou des WSN basés sur des réseaux de détecteurs. L'étude n'est pas exhaustive concernant les systèmes étudiés mais j'ai mis l'accent sur les plus fréquents, les plus utilisés et viables. Par exemple, le système de géo-localisation chinois Beidou n'a pas été mentionné car il est seulement destiné à un usage privé sur le sol chinois, le système est coûteux et sa précision n'est pas concurrentielle au GPS, à GALILEO ou même à GLONASS et ne bénéficie pas de

l'interopérabilité qu'offrent les trois. J'ai aussi décrit certains services auxiliaires permettant d'améliorer la précision des systèmes étudiés tel que WAAS, EGNOS et A-GPS.

En voulant améliorer la précision et l'envoi des données à distance, nous avons été confrontés au coût de la communication. Notre solution concernant ce point est d'utiliser le moyen de communication le moins coûteux car un usage classique des services de localisation à distance se faisait à travers une connexion GPRS payante. Tout d'abord, il nous fallût étudier les moyens de communication sans fil existants afin de comprendre leur fonctionnement, leur force et leur faiblesse, pour en retenir ceux qui offrent une solution optimale quant à la portée, la facilité d'intégration et le coût. Le résultat de cette étude des systèmes de communication nous a amené au fait que la solution la moins coûteuse et qui offre une portée optimale est l'utilisation de modems radio. Cependant, cette solution souffre d'une portée limité quand il s'agit d'utiliser des modems de taille réduite afin de les intégrer dans des systèmes portatifs tels que ceux que portent des personnes (PDA, téléphones cellulaires, etc.): les modems radio sont la seule solution étudiée qui offre la meilleure portée en utilisant les plus petits émetteur/récepteurs possibles, le WiMAX malheureusement n'offre pas le degré de miniaturisation requis et le Wi-Fi n'a pas assez de portée.

Notre solution est de combiner entre les meilleurs systèmes possibles pour établir une communication distante en optimisant le coût de la communication. Cette optimisation est rendue possible via l'algorithme de communication proposé dans le chapitre 4. Cet

algorithme cherche à établir une connexion radio sans fil au serveur, s'il n'est pas capable de s'y connecter, il demande à un autre périphérique radio, disposant d'une meilleure portée, de relayer ses données vers le serveur. Si toutes les tentatives d'envoi de données via le modem radio échouent, l'algorithme fait basculer le système de communication au mode GPRS pour passer à travers le réseau téléphonique payant afin d'assurer l'envois des données de localisation et maintenir une traçabilité complète de nos dispositifs mobiles distants. L'étude de coût réalisée au chapitre 4 constitue aussi une preuve d'appui à notre solution. Nous avons pu constater que l'utilisation de communications radio comme substituts au GPRS, réduisent considérablement le coût globale de notre application.

Pour faciliter la maintenance, l'ajout de nouvelles fonctionnalités au système ou la mise à niveaux de celles déjà existantes sans compromettre l'exécution du système, il a fallu trouver une meilleure approche de programmation. Pour ce faire, nous devons d'abord nous assurer que le codage ne sera pas un frein pour le développement futur par d'autres programmeurs qui n'ont pas travaillés auparavant sur la conception initiale du système. Il faudra que le code soit transparent pour le développeur et que les fonctionnalités du programme soient séparées en modules indépendants pour un meilleur contrôle en s'inspirant du proverbe « divide et impera » (diviser pour régner). Le choix d'OSGI est motivé par le fait que les intergiciels qui l'implémentent, offrent un environnement portable que l'on peut déployer sur n'importe quelle machine supportant la machine virtuelle Java. Par conséquent, OSGI peut s'exécuter sur des équipements allant d'un simple PDA limité en ressources jusqu'à un serveur avec une grande puissance de calcul et disposant d'amplement de ressources. Comme l'application est déployée sous forme de composants

appelés bundles, il est très facile de manipuler ces composants séparément comme démontré dans le chapitre 5. Étant donné que le code d'un bundle est transparent aux autres bundles, ceci permet d'avoir un environnement sécuritaire, en plus, la conception même de l'intergiciel se base sur un modèle de sécurité avancé pour protéger d'avantage l'environnement d'exécution des applications. Nous avons montré dans ce mémoire cette conception modulaire, la facilité et les fonctionnalités qu'offrent cette nouvelle façon de programmer que j'encourage beaucoup d'adeptes de programmation classique de découvrir. Ceci leur permettra de gagner beaucoup de temps sur la maintenance et la mise à niveau de leurs applications en réduisant aussi le coût de déploiement. Cependant, il existe certaines limites pour les implémentations d'OSGI mais ceci n'est qu'une question de temps pour combler certaines lacunes. OSGI étant une approche assez jeune, les développeurs sont en cours d'intégration des services qui permettront d'élargir les horizons de cette plateforme en termes de possibilités de conceptions d'application de tous types. Il existe actuellement certains services de base tels que des serveurs HTTP, des services d'authentification et de sécurité et bientôt des services de base de données et bien plus.

Points réalisés :

Par rapport aux points à réaliser mentionnés dans le premier chapitre, j'ai pu réaliser les points suivants :

- Conception du modèle et développement de l'outil qui implémente une partie de ce modèle de géo-localisation.

- Ce modèle convient à n'importe quelle application de géo-localisation où l'on peut facilement ajouter de nouveaux services.

- Le processus de détection et d'intervention rapide est assuré grâce à l'automatisation de notre système, il suffit d'ajouter d'autres services de géo-localisation à mon modèle et de créer les connecteurs entres ses services pour qu'ils puissent communiquer entre eux en suivant le modèle de service des bundles montrés au chapitre 5. Par exemple, ajouter un service de notification des services de police dans un bundle des services de santé.

Points non réalisés et limites de notre approche

- Je n'ai pas intégré l'algorithme de communication permettant de choisir la communication radio adéquate dans mes bundles. Ce fut plus une limitation technique vu que je ne disposais pas de tout le matériel radio et GPRS pour intégrer et tester cet algorithme.

- Je n'ai pas ajouté le service A-GPS pour la correction d'erreurs de positionnement, étant donné que mon équipement GPS ne pouvait fonctionner qu'avec WAAS et EGNOS. Mais il est assez facile de diffuser les données de correction avec notre système de communication radio aux équipements géo-localisables de notre réseau mobile. Les équipements GPS doivent être capables de post-traiter les données A-GPS (plusieurs puces GPS sur le marché supportent cette fonctionnalité).

- Je n'ai pas développé les bundles qui gèrent la base de données de notre serveur central. Ceci est aussi une limitation technologique mais provisoire puisque le fournisseur de l'intergiciel knopflerfish va intégrer le support aux bases de données ainsi que d'autres services dans les prochaines versions.

- Le déploiement de notre application nécessite la présence d'un intergiciel OSGI qui lui est basé sur la JVM. Ceci peut être une limitation sur les dispositifs ne supportant pas Java.

- Une panne au niveau du serveur central et du serveur auxiliaire ou de leurs modems radio entrainera une cessation de la collecte des données en mode radio sans fil et fera basculer tous les petits équipements radio sur le mode GPRS coûteux.

Développement future

Le modèle proposé dans ce mémoire n'a pas été développé en intégralité et seuls les éléments clés ont été réalisés afin de donner une preuve de conception et donner une idée à de futures développeurs quant à la manière de procéder pour concevoir un tel système. Il reste cependant quelques points que je voudrais souligner pour en tenir compte dans un développement futur. Ces points sont les suivants :

- Améliorer l'algorithme de communication et trouver une solution pour compresser les lots de messages à envoyer vers le relayeur afin de réduire le temps de transfert.

- Introduire un algorithme de cryptage tel qu'AES ou autres qui peuvent être assez performants et moins encombrants pour la taille globale des données à envoyer.

- Développer un outil parallèle à notre application afin de pouvoir trouver l'emplacement optimal des véhicules relayeurs dans une ville avec sa vrais forme et bordures (nous avions supposés dans notre étude de coût une forme carrée et rectangulaire pour la superficie de la ville) afin de réduire le nombre de véhicules relais requis pour couvrir toute la ville sans beaucoup augmenter le délai d'attente de transmission des petits objets géo-localisables à faible portée.

Mon présent travail est une piste parmi plusieurs à suivre en ce qui concerne la géo-localisation mais qui offre plusieurs avantages par rapport à des solutions existantes, soit plus coûteuses ou/et plus complexes. Nous avons vu à travers ce document qu'il est possible d'implémenter ce système avec un faible coût si on le compare avec d'autres techniques comme WSN. J'ai aussi démontré, via une implémentation concrète de mon modèle de gestion des objets géo-localisables, qu'il est simple d'utiliser OSGI et la programmation modulaire ce qui nous permet de tirer profit des atouts de cette approche mentionnés dans ce mémoire. Le développement via OSGI non seulement m'a permis de faciliter le développement et l'intégration des bundles de mon application mais constitue aussi une première étape permettant de greffer d'avantage de services et de fonctionnalités au système proposé. L'utilisation d'un tel réseau de nœuds intelligents donne plus de flexibilité à notre système en permettant à nos objets d'être autonomes et à notre système d'utiliser moins de puissance de calcul. Étant donné que le système est capable de gérer plusieurs objets hétérogènes à la fois, et que chaque groupe d'objets constitue un ensemble de services, nous sommes en mesure d'interconnecter plusieurs services ou réseaux d'objets entre eux pour offrir d'avantage de services dans un environnement coopératif. À

titre d'exemple, le réseau de voitures civiles peut être connecté au service d'ambulances et ce dernier au service de police. Ainsi il devient possible de réagir plus rapidement et efficacement en cas d'incident. Les patients, les enfants de bas âges et même des animaux domestiques peuvent être représentés dans le système pour être facilement localisés. Le système peut être ouvert à tous si des applications, similaires à la mienne, de la communauté « logiciels libres » sont offerts au large publique. Ou bien des organismes gouvernementaux pourraient offrir de tels systèmes afin de mieux protéger les gens dans la société et éviter les coûts exorbitants qui peuvent être liés aux dangers de la route ou des complications de la santé physique, etc. tel que fut le cas pour la gratuité du signal GPS pour éviter des accidents d'avions. Cependant le large public n'aura qu'à payer pour l'équipement de géo-localisation, alors que certains particuliers, comme ceux atteints d'Alzheimer, pourraient voir ce coût pris en charge par leurs assurances maladie.

BIBLIOGRAPHIE

[D.Fontana & al., 2001] The New L2 Civil Signal: LCDR Richard D. Fontana, GPS Joint Program Office. Wai Cheung, Science Applications International Corporation.. Paul M. Novak, Science Applications International Corporation. Thomas A. Stansell, Jr., Stansell Consulting, 2001.

[A.Assad, 2007] A Real-time Laboratory Testbed For Evaluating Localization Performance of Wifi Technologies, Muhammad Ali Assad. Faculty of Worcester Polytechnic Institute, Electrical and Computer Engineering, 2007.

[R.Tesoriero & al,. 2008] A Location-aware System using RFID and Mobile Devices for Art Museums, R. Tesoriero, J. A. Gallud, M. Lozano, V. M. R. Penichet. Fourth International Conference on Autonomic and Autonomous Systems, 2008.

[U.Bischoff & al., 2006] Constraint-based distance estimation in ad-hoc wireless sensor networks, U. Bischoff, M. Strohbach, M. Hazas, and G. Kortuem.. In EWSN, 2006.

[M.Russel & al., 2002] The Precision Revolution :GPS and the future of aerial warfare, M.Russel, James M.Hasik, ISBN 1557509735, 2002

[C.Pellerin, 2006] United States Updates Global Positioning System Technology: New GPS satellite ushers in a range of future improvements, Cheril Pellerin , **www.america.gov**, 2006,

[D.Rousseau, 2009] Radio Canada, Maladie d'Alzheimer : La police propose des bracelets GPS, Daniel Rousseau, SPVM, http://www.radio-canada.ca/regions/Montreal/2009/12/22/008-Alzheimer-GPS-police.shtml, 2009.

[Alzheimer-rive, 2009] Société Alzheimer rive-sud, Rapport annuel 2008-2009, http://www.alzheimerrivesud.ca/documents/RapportannuelPDF.pdf, 2009.

[J.Faure, 2003] BE Canada numéro 239 - Ambassade de France au Canada / ADIT, Johann Faure, Le devoir-13/11/2003, http://www.bulletins-electroniques.com/actualites/17431.htm, 2003.

[Triangulation, 2006] La triangulation et les mesures de distance, Observatoire de Paris, http://expositions.obspm.fr/lumiere2005/triangulation.html, 2006.

[USAirForce, 2010] US Air Force official website, Global Positioning System, http://www.af.mil/information/factsheets/factsheet.asp?id=119 , 2010.

[TriangTrimble, 2010] Trimble, GPS tutorial : Triangulating from satellites, http://www.trimble.com/gps/howgps-triangulating.shtml, 2010.

[ErrTrimble, 2010] Trimble, GPS tutorial : Correcting errors, http://www.trimble.com/gps/howgps-error.shtml, 2010.

[H.Dana, 94] Peter H. Dana, The Geographer's Craft Project, Department of Geography, The University of Colorado at Boulder, http://www.colorado.edu/geography/gcraft/notes/gps/gps.html, 1994.

[PNT, 2008] National Executive Committee for Space-Based PNT : GLOBAL POSITIONING SYSTEM STANDARD POSITIONING SERVICE PERFORMANCE STANDARD, http://pnt.gov/public/docs/2008/spsps2008.pdf, 2008.

[PNT-SA, 2010] National Executive Committee for Space-Based PNT : Selective Availability, http://pnt.gov/public/sa/, 2010

[kowoma, 2009] The GPS system: sources of errors in GPS, http://www.kowoma.de/en/gps/errors.htm, 2009.

[B.Langley, 1999] Richard B.Langley, GPS World, Dilution Of Precision, University of New Brunswick, http://www.sbg.ac.at/mat/staff/revers/lectures/2006_2007/GPS/gpsworld.may99.pdf, 1999.

[R.Thompson & al., 2009] Ryan J. R. Thompson, Asghar Tabatabaei Balaei, and Andrew G. Dempster School of Surveying and Spatial Information Systems, University of New South Wales, Australia. Dilution of precision for GNSS interference localisation systems, 2009.

[E W.Weisstein, 2007] Doppler Effect, Eric W. Weisstein, 1997-2007, scienceworld.wolfram.com

[T.Chalko, 2009] Tom Chalko MSc, PhD, Estimating Accuracy of GPS Doppler Speed Measurement using Speed Dilution of Precision (SDOP) parameter, http://nujournal.net/SDOP.pdf, 2009.

[T.Chalko, 2007]Tom Chalko MSc, PhD, High accuracy speed measurement using GPS (Global Positioning System), http://nujournal.net/HighAccuracySpeed.pdf, 2007.

[L.Denis, 2000] Lynn Denis: LE GPS ET LE DGPS EN QUELQUES MOTS, Pêches et Océans Canada, Garde côtière, http://www.ccg-gcc.gc.ca/folios/00021/docs/dgps_guide-fra.pdf, 2000.

[WAAS, 2010] Federal Aviation: Administration, http://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gnss/waas/, 2010.

[A-GPS, 2010] Assisted GPS, A-GPS: an overview, information or turorial about the basics of Assisted GPS (Global Positioning System) or A-GPS used to provide location based services used for cellular technology and cellular networks, http://www.radio-electronics.com/info/cellulartelecomms/location_services/assisted_gps.php , 2010.

[GPSBase, 2010] GPS Base station for post-treatment, www.gsf.qc.ca, 2010.

[AGPS-intro, 2008] AGPS Introduction And Application, http://www.gorentech.co.il/itemfiles/397_AGPS%20Introduction.pdf , 2008.

[CSR, 2010] CSR, Introducing eGPS, http://www.csr.com/egps/, 2010.

[CSR-MOTOROLA, 2008] CSR, Press release : CSR and Motorola announce intention to form EGPS Forum to advance and improve location technologies, New group to encourage mobile industry leaders to work together to accelerate development of EGPS solutions, http://www.csr.com/pr/pr321.htm, Cambridge, UK and Libertyville, Ill., USA – 28 January, 2008.

[GALILEO, 2009] GALILEO, European Satellite Navigation System, GALILEO as a major technological, economic and political challenge, *http://ec.europa.eu/dgs/energy_transport/galileo/intro/challenge_en.htm*, 2009.

[GALILEO-GPS, 2009] GALILEO, European Satellite Navigation System: GPS and Galileo... Progress through Partnership, http://ec.europa.eu/dgs/energy_transport/galileo/intro/gps_en.htm, 2009.

[GALILEO-EC, 2010] European Commission, Enterprise and Industry: Galileo - What do we want to achieve ?, http://ec.europa.eu/enterprise/policies/space/galileo/index_en.htm, 2010.

[GALILEO-ESA, 2010] European Space Agency, Galileo: Europe's global navigation satellite system, http://www.satellite-navigation.eu/, 2010.

[EGNOS, 2010] European GNSS Supervisory Authority: What is EGNOS ?, http://www.gsa.europa.eu/go/egnos/what-is-egnos, 2010.

[EGNOS-FAQ, 2006] ESTB/EGNOS FAQ, http://www.egnos-pro.esa.int/ESTB-EGNOS%20FAQ%2017%20May%2006.pdf, 2006.

[Cospas-Sarsat, 2009] Système international de satellites pour les recherches et le sauvetage : Programme International Cospas-Sarsat, http://www.cospas-sarsat.org/index.php?option=com_content&view=article&id=180&Itemid=101&lang=fr, 2009.

[Spaceandtech, 2001] Space and Tech: GLONASS Summary, www.spaceandtech.com, 2010.

[GLONASS, 2010] Russian Space Agency : GLONASS, www.glonass-ianc.rsa.ru, 2010.

[V.Glotov, 2009] Dr. Vladimir Glotov, Division Head, Central Research Institute of Machine Building. http://147.102.106.44/ILRS_W2009/docs/PP02B_GLOTOV_GLONASS.pdf, Metsovo, Greece, 2009.

[Wilson & al., 2007] P. Wilson, D. Prashanth, and H. Aghajan. Utlizing RFID signaling scheme for localization of stationary objects and speed estimation of mobile objects. In IEEE International Conference on RFID, 2007.

[Bischoff & al., 2006] U. Bischoff, M. Strohbach, M. Hazas, and G. Kortuem. Constraint-based distance estimation in ad-hoc wireless sensor networks. In EWSN, 2006.

[GPRS, 2004] GPRS tutorial, MorganDoyle Limited, http://www.item.ntnu.no/fag/tm8100/Pensumstoff2004/GPRS_Tutorial.pdf , 2004.

[X.Lagrange, 2000] Xavier Lagrange, Les réseaux radiomobiles, HERMES Science Publications ISBN : 2-7462-0110-0, 2007.

[COS, 99] COST 231: Evolution of land mobile radio (including personal) communications, Final report, Information, Technologies and Sciences, European Commission, 1999.

[mes799, 2007] MESURES 799: L'antenne le composant de base, www.mesures.com, 2007.

[R.Want, 2006] Pervasive Computing: An Introduction to RFID technologies , http://www.perada.eu/documents/articles-perspectives/an-introduction-to-rfid-technology.pdf , 2006.

[GAO, 2010] GAO RFID Inc, Asset tracking RFID technologies, http://www.gaorfidassettracking.com/RFID_Asset_Tracking_Technologies/ , 2010.

[MSSI, 2005] The RFID Revolution Starts Now! : Sapphire DART Ultra Wideband Precision Asset Location System, http://www.multispectral.com/pdf/Sapphire_Revolution.pdf , 2005.

[WiFiorg, 2010] WiFi alliance: WiFi certified-discover and learn http://www.wi-fi.org/discover_and_learn.php, 2010.

[IEEE-std, 2008] IEEE, IEEE Standards Association, standards.ieee.org, 2008.

[C.Eklund & al., 2002] C. Eklund, R. Marks, K. Stanwood, and S. Wang. IEEE standard 802.16: a technical overview of the wirelessmantm air interface for broadband wireless access. Communications Magazine, IEEE, 40(6):98–107, Jun 2002.

[OSGI, 2010] OSGI Alliance : OSGI technology, http://www.osgi.org/About/Technology, 2010.

[FelixApache, 2010] Apache Felix: OSGI R4 platform, http://felix.apache.org/site/index.html, 2010.

ANNEXES

A. Le code du bundle « gpsservice »

Cette section décrit le code utilisé dans les classes Java du bundle « gpsservice » étudié dans la section 4.1.1 et qui offre le service de lecture des données de localisation depuis un GPS et l'envoi des données sur un port radio. Ce code a été adapté pour être compatible avec les implémentations d'OSGI tel que Knopflerfish.

Gpsservice.java

```
package org.knopflerfish.GPS_Service.gpsservice;

public interface Gpsservice {
      public String LirePosition();
      public void envoyerMSG(String msg, String port);
      public void Arret();
}
```

Activator.java

```
package org.knopflerfish.GPS_Service.gpsservice.impl;
import java.util.Hashtable;
import org.osgi.framework.BundleActivator;
import org.osgi.framework.BundleContext;
import org.osgi.framework.Constants;
import org.osgi.framework.ServiceRegistration;
import org.knopflerfish.GPS_Service.gpsservice.Gpsservice;
public class Activator implements BundleActivator {
public static BundleContext bc = null;
private GpsserviceImpl serv=null;

public void start(BundleContext bc) throws Exception {
System.out.println(bc.getBundle().getHeaders().get
(Constants.BUNDLE_NAME) + " Demmarage du service du positionnement sur
le port local COM3...");
Activator.bc = bc;
serv = new GpsserviceImpl("COM8");
ServiceRegistration registration =
bc.registerService(Gpsservice.class.getName(), serv, new Hashtable());

System.out.println("Enregistrement du service GPS: Gpsservice");
}
public void stop(BundleContext bc) throws Exception {
System.out.println(bc.getBundle().getHeaders().get
(Constants.BUNDLE_NAME) + " arret en cours...");
serv.Arret();
Activator.bc = null;
}
}
```

ComGPS.java

```
package org.knopflerfish.GPS_Service.gpsservice.impl;

import java.io.IOException;

public class ComGPS {

      PortCom PC = null;

      public ComGPS(String nomport)
      {

            try {
                  PC = new PortCom(nomport);
            } catch (Exception e) {
```

```
                    System.out.println("echec ouverture port:"+nomport);
                    e.printStackTrace();
              }


       }

       public String Affiche(){

       String msg=PC.Lire(">QPV<");
     Traitement x = new Traitement(">QPV<");
     x.TraitementMsg(msg);

     String PDOP=PC.Lire(">QSSF11<");
     int PDOP_niv = (PDOP.charAt(7)=='1') ? 1 : 0;

     String GPS_DATA="";

GPS_DATA=(x.latitude()+";"+x.longitude()+";"+x.signal_qualite+";"+x.vel
ocity()+";"+PDOP_niv+";"+x.precision_source+";"+x.precision_niveau+";"+
x.id);
       return GPS_DATA;

       }

public void arret()
{
       try {
             System.out.println("Arret de communication");
             PC.Fermercnx();
       } catch (IOException e) {
             // TODO Auto-generated catch block
             System.out.println("Fin de communication impossible");
             e.printStackTrace();
       }
}

}
```

GpsserviceImpl.java

```
package org.knopflerfish.GPS_Service.gpsservice.impl;
import java.io.IOException;

import org.knopflerfish.GPS_Service.gpsservice.Gpsservice;

public class GpsserviceImpl implements Gpsservice {
      String port="";
```

```
	private ComGPS GPS=null;

public GpsserviceImpl(String x){
	port=x;
	GPS= new ComGPS(port);
}
public String LirePosition() {

return GPS.Affiche();
}

public void envoyerMSG(String msg, String port){
	PortCom port2 = null;
	try {
		port2 = new PortCom(port);
	} catch (Exception e) {
		System.out.println("impossible d'envoyer le message sur le
port radio");
		e.printStackTrace();
	}
	port2.Ecrire(msg);
	System.out.println("message envoyé sur le port radio COM10:
"+msg);
	try {
		port2.Fermercnx();
		System.out.println("port radio COM10 fermé: ");
	} catch (IOException e) {
		System.out.println("impossible de fermer le port radio
COM10: ");
		e.printStackTrace();
	}

}

public void Arret()
{
	GPS.arret();
}

}
```

PortCom.java

```
package org.knopflerfish.GPS_Service.gpsservice.impl;
import java.io.*;
import javax.comm.CommPort;
import javax.comm.CommPortIdentifier;
import javax.comm.SerialPort;
```

```
public class PortCom {

      /** The input stream */
        protected BufferedReader is;
        /** The output stream */
        protected PrintStream os;
        CommPort port;
        int BAUD=115200;

      public  PortCom(String nomport) throws Exception {

    // Prendre port ID
    CommPortIdentifier portId =
      CommPortIdentifier.getPortIdentifier(nomport);

    // Ouvrir port
    // initialisation du port avec un timer de 10 secondes
    port = portId.open("Connexion au GPS", 10000);
    System.out.println("ouverture en cours ");
    SerialPort monPort = (SerialPort) port;
    System.out.println("ouverture accomplie ");

    /* parametrage du port serie avec la vitesse en BAUD la longueure
des données etc*/
    monPort.setSerialPortParams(BAUD, SerialPort.DATABITS_8,
      SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);

 // Acquisition des flux d'entrée et de sortie IS et OS
    try {
      is = new BufferedReader(new
InputStreamReader(port.getInputStream()));
    } catch (IOException e) {
      System.err.println("ne peut lire le flux entrant: lecture-
seule");
      is = null;
    }
    os = new PrintStream(port.getOutputStream(), true);
}

public String Lire(String cmd){
             os.println(cmd);
             try {
                  return is.readLine();
            } catch (IOException e) {
            e.printStackTrace();
            }
            return null;
        }

public void Ecrire(String msg){
             os.println(msg);
       }
```

```
public void Fermercnx() throws IOException{
   System.out.println("fermeture flux de sortie");
   os.close();
   System.out.println("fermeture flux d'entrée");
   is.close();
   System.out.println("fermeture du port");
   port.close();
   System.out.println("fin de communication sur le port");
}
}
```

Traitement.java

```
package org.knopflerfish.GPS_Service.gpsservice.impl;

class Traitement {
      public static String id="";
      public static String precision_source="s/o";
      public static int precision_niveau=0;
      private static boolean available = false;
      private static String latitude;
      private static String longitude;
      private static int north;
      private static int east;
      private static String Mph;
      private static String heading;
      public static String signal_qualite;

      private static String MSG;
      private static String MsgType;
      private static String[] Tab=null;

       Traitement(String TaipCmd) {

            MsgType = TaipCmd;
      }

static void TraitementMsg(String InMsg) {
           MSG = InMsg;

           if(MSG != "") {


           available = true;

                 if(MsgType.equals(">QPV<"))
                       {
                            Tab=MSG.split(";");
```

```
                                id=Tab[1];
                                Tab=id.split("=");
                                id=Tab[1];
                                Tab=id.split("<");
                                id=Tab[0];
                                latitude = MSG.substring(10, 17);
                                north = (MSG.charAt(9)=='+') ? 1 : -1;

                                longitude = MSG.substring(18, 26);
                                east = (MSG.charAt(17)=='+') ? 1 : -1;

                                Mph = MSG.substring(26, 29);
                                heading = MSG.substring(29, 31);

                                //conversion auto de char vers int -48
pour 0
                                int choix=MSG.charAt(32)-48;

                                switch (choix){
                           case 0:  precision_source ="2D
GPS";precision_niveau=3; break;
                           case 1:  precision_source ="3D GPS";
precision_niveau=4;break;
                           case 2:  precision_source ="2D DGPS(WAAS ou
EGNOS)";precision_niveau=5; break;
                           case 3:  precision_source ="3D DGPS(WAAS ou
EGNOS)"; precision_niveau=6;break;
                           case 6:  precision_source ="DR(Dead
Reckoning)";precision_niveau=2; break;
                           case 8:  precision_source ="DR(Dead Reckoning)
DEGRADe";precision_niveau=1; break;
                           case 9:  precision_source ="Origine inconnue";
precision_niveau=0;break;

                                   }
                                choix=MSG.charAt(33)-48;
                                switch (choix){
                           case 0:  signal_qualite ="signal non
disponible"; break;
                           case 1:  signal_qualite ="signal age de plus
que 10sec";break;
                           case 2:  signal_qualite ="signal fraichement
releve, moins de 10sec"; break;
                                                   }

            }
            }



      }
```

```
float latitude() {
            if(!available) return 0;
            // convert from ddmm.mmmm to dd.dddddd
            return Float.valueOf(latitude).floatValue()/100000 * north;
      }

float longitude() {
            if(!available) return 0;
            // convert from ddmm.mmmm to dd.dddddd
            return Float.valueOf(longitude).floatValue()/100000  *
east;
      }

Double velocity() {
            if(!available) return 0.0;
            // Mph vers Km/h
            return Double.valueOf(Mph).doubleValue() * 1.6093;
      }

float heading() {
            if(!available) return 0;
            return Float.valueOf(heading).floatValue();
      }
}
```

B. Le code du bundle « gpsserviceuser »

Dans cette section, je décris le code utilisé pour le bundle « gpsserviceuser » dans la section 4.1.2. Ce bundle permet d'utiliser le service « gpsservice » pour lire les données de localisation et les envoyer sur un port radio, vers notre serveur centrale.

Activator.java

```
package org.knopflerfish.GPS_Service.gpsserviceuser.impl;

import org.knopflerfish.GPS_Service.gpsservice.Gpsservice;
import org.knopflerfish.GPS_Service.gpsserviceuser.impl.Activator;
import org.osgi.framework.BundleActivator;
import org.osgi.framework.BundleContext;
import org.osgi.framework.Constants;
import org.osgi.framework.ServiceReference;

public class Activator implements BundleActivator {
      public static BundleContext bc = null;
      Gpsservice service=null;
```

```
      public void start(BundleContext bc) throws Exception {
      System.out.println(bc.getBundle().getHeaders()
      .get(Constants.BUNDLE_NAME) +
      " demmarage...");
      Activator.bc = bc;
      ServiceReference reference =
      bc.getServiceReference(Gpsservice.class.getName());
       service = (Gpsservice)bc.getService(reference);
      System.out.println("Utilisation du service GPS gpsservice");
      String msg="";
      msg=service.LirePosition();
      System.out.println("Affichage données avec gpsserviceuser: "
+msg);
      service.envoyerMSG(msg+"\n", "COM10");

      bc.ungetService(reference);
      }
      public void stop(BundleContext bc) throws Exception {
      System.out.println(bc.getBundle().getHeaders()
      .get(Constants.BUNDLE_NAME) +
      " arret bundle utilisateur...");
      //serv.Arret();
      Activator.bc = null;
      }
   }
```

C. Le code du bundle « guipda »

Dans cette section je décris le code pour le bundle « guipda » qui permet de créer une interface utilisateur visuelle GUI pour consulter graphiquement les données GPS sur un dispositif de géo-localisation comme montré dans la section 4.1.3.

Activator.java

```
package org.knopflerfish.GPS_Service.guipda.impl;

import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import org.knopflerfish.GPS_Service.gpsservice.Gpsservice;
import org.knopflerfish.GPS_Service.guipda.impl.Activator;
import org.osgi.framework.BundleActivator;
import org.osgi.framework.BundleContext;
import org.osgi.framework.Constants;
import org.osgi.framework.ServiceReference;
```

```
public class Activator implements BundleActivator {
      public static BundleContext bc = null;
      Gpsservice serv=null;
      public void start(BundleContext bc) throws Exception {
      System.out.println(bc.getBundle().getHeaders()
      .get(Constants.BUNDLE_NAME) +
      " demmarage interface graphique GPS...");
      Activator.bc = bc;
      ServiceReference reference =
      bc.getServiceReference(Gpsservice.class.getName());
      serv = (Gpsservice)bc.getService(reference);
      System.out.println("Utilisation du service de localisation GPS
par le bundle guipda ....");


final GuiDFrame fenetre = new GuiDFrame(0,0,"",0.0);

   System.out.println("ouverture de la fenetre");
   fenetre.setVisible(true);



javax.swing.Timer t = new javax.swing.Timer(5000, new ActionListener()
{
          public void actionPerformed(ActionEvent e) {

                FormatPos x=null;
                x=new FormatPos(serv.LirePosition());
                System.out.println(x.Latitude());

                     if (x.Precision()>3  && x.PDOP()==0)
                  {

fenetre.refresh(x.Latitude(),x.Longitude(),x.Qualite(),x.Vitesse(),Colo
r.yellow,x.Source());
                  }


             if (x.Precision()>3  && x.PDOP()==1)
             {

fenetre.refresh(x.Latitude(),x.Longitude(),x.Qualite(),x.Vitesse(),Colo
r.green,x.Source());
             }


                  if (x.Precision()<=3 && x.Qualite() =="signal
fraichement releve, moins de 10sec")
                  {

fenetre.refresh(x.Latitude(),x.Longitude(),x.Qualite(),x.Vitesse(),Colo
r.red,x.Source());
```

```
                    }

                            if ( x.Qualite() =="signal age de plus que
10sec" || x.Qualite() =="signal non disponible")
                            {
fenetre.refresh(x.Latitude(),x.Longitude(),x.Qualite(),x.Vitesse(),Colo
r.gray,x.Source());

                            }
          }
        });
      t.start();
      bc.ungetService(reference);
      }
      public void stop(BundleContext bc) throws Exception {
      System.out.println(bc.getBundle().getHeaders()
      .get(Constants.BUNDLE_NAME) +
      " arret bundle GUI PDA...");
      serv.Arret();
      Activator.bc = null;
      }
   }
```

FormatPos.java

```
package org.knopflerfish.GPS_Service.guipda.impl;

public class FormatPos{

      private float latitude;
      private float longitude;
      private String signal_qualite;
      private String precision_source;
      private int precision_niveau;
      private double vitesse;
      private int PDOP;
      private String message="";
      private String[] Tab=null;



      public FormatPos(String msg){
            message=msg;
            System.out.println(message);
            Tab=message.split(";");
      }

      private String Lire(int pos){
            String position=Tab[pos];
```

```
            return position;
      }


      public float Latitude(){

            latitude=Float.valueOf(Lire(0)).floatValue();
            return latitude;
            }
public float Longitude(){

            longitude=Float.valueOf(Lire(1)).floatValue();
            return longitude;
            }
public String Qualite(){

      signal_qualite=Lire(2);
      return signal_qualite;
      }
public double Vitesse(){

      vitesse=Double.valueOf(Lire(3)).doubleValue();
      return vitesse;
      }
public int PDOP(){

      PDOP=(Lire(4)=="1") ? 1 : 0;
      return PDOP;
      }
public String Source(){
      precision_source=Lire(5);
      return precision_source;
}

public int Precision(){
      Integer I= new Integer(Lire(6));
      precision_niveau=I.intValue();
      return precision_niveau;
}

}
```

GuiDFrame.java

```
package org.knopflerfish.GPS_Service.guipda.impl;

import java.awt.*;
import javax.swing.*;

public class GuiDFrame extends JFrame
```

```
{

      private String lattxt="";
      private String longtxt="";
      private String vittxt="";
      private String qulttxt="";
      private Color code=Color.red;

    JLabel lab_lo=null;
     JLabel lab_la;
     JLabel lab_vit;
     JLabel titr;
     JLabel signa;
     JLabel untitled_17;
     JTextField lo;
     JTextField la;
     JTextField vi;
     JTextField sourc;

public void refresh(float a,float b,String c,double d, Color coul,
String source)
      {

            la.setText(""+a);
            lo.setText(""+b);
            vi.setText(""+d);
            signa.setText(c);
            signa.setBackground(coul);
            sourc.setText(source);



      }

public GuiDFrame(float lat,float lon,String signal,double vit) throws
InterruptedException{
            super();

            lattxt=String.valueOf(lat);
            longtxt=String.valueOf(lon);
            qulttxt=signal;
            vittxt=String.valueOf(vit);

            AliFrame();//On initialise notre fenï¿½tre

      }

public void AliFrame()
   {
     getContentPane().setLayout(null);
     setupGUI();
     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   }
```

```
void setupGUI()
   {
      lab_lo = new JLabel();
      lab_lo.setLocation(55,91);
      lab_lo.setSize(87,25);
      lab_lo.setForeground(new Color(0,0,0)) ;
      lab_lo.setBackground(new Color(204,204,255));
      lab_lo.setText("      Longitude ");
      lab_lo.setToolTipText("longitude actuelle");
      lab_lo.setOpaque(true);
      getContentPane().add(lab_lo);

      lab_la = new JLabel();
      lab_la.setLocation(262,91);
      lab_la.setSize(95,25);
      lab_la.setForeground( new Color(0,0,0) );
      lab_la.setBackground( new Color(204,204,255) );
      lab_la.setText("         Latitude ");
      lab_la.setToolTipText("latitude actuelle");
      lab_la.setOpaque(true);
      getContentPane().add(lab_la);

      lab_vit = new JLabel();
      lab_vit.setLocation(161,182);
      lab_vit.setSize(77,25);
      lab_vit.setForeground( new Color(-16777216) );
      lab_vit.setBackground( new Color(-3355393) );
      lab_vit.setText("      Vitesse");
      lab_vit.setOpaque(true);
      lab_vit.setToolTipText("vitesse approximative actuelle,
relativement fiable en 3D GPS et DGPS avec PDOP faible");
      getContentPane().add(lab_vit);

      titr = new JLabel();
      titr.setLocation(87,20);
      titr.setSize(223,37);
      titr.setText("Localisation en temps réel via port USB");
      titr.setToolTipText("Réalisé par A.Goundafi, UQAC 2010");
      titr.setOpaque(true);
      getContentPane().add(titr);

      signa = new JLabel();
      signa.setLocation(81,257);
      signa.setSize(233,45);
      signa.setBackground(code);
      signa.setOpaque(true);
      signa.setText("                              Fiabilité du signal");
      signa.setToolTipText("Fiabilité du signal, rouge=pas fiable,
jaune= peu fiable (3d-GPS/DGPS et PDOP élevé), vert=assez fiable (3D-
GPS/DGPS PDOP faible)");
      getContentPane().add(signa);
```

```
    untitled_17 = new JLabel();
    untitled_17.setLocation(163,311);
    untitled_17.setSize(86,25);
    untitled_17.setBackground( new java.awt.Color(-3355393) );
    untitled_17.setText("    Source GPS ");
    untitled_17.setOpaque(true);
    getContentPane().add(untitled_17);

    lo = new JTextField(longtxt,5);
    lo.setLocation(31,112);
    lo.setSize(130,27);
    lo.setHorizontalAlignment(JTextField.CENTER);
  lo.setBackground(Color.LIGHT_GRAY);
    lo.setForeground(Color.black);
    getContentPane().add(lo);

    la = new JTextField(lattxt,5);
    la.setLocation(243,111);
    la.setSize(129,27);

    la.setHorizontalAlignment(JTextField.CENTER);
    la.setBackground(Color.LIGHT_GRAY);
    la.setForeground(Color.black);
    getContentPane().add(la);

    vi = new JTextField(vittxt,5);
    vi.setLocation(130,201);
    vi.setSize(132,29);

    vi.setHorizontalAlignment(JTextField.CENTER);
    vi.setBackground(Color.BLUE);
    vi.setForeground(Color.white);
    getContentPane().add(vi);

    sourc = new JTextField();
    sourc.setLocation(142,332);
    sourc.setSize(124,25);
    sourc.setText("");
    sourc.setHorizontalAlignment(JTextField.CENTER);
    sourc.setBackground(Color.gray);
    sourc.setForeground(Color.yellow);
    getContentPane().add(sourc);



    setTitle("Visualisation données GPS");
    setSize(407,506);
    setVisible(true);
    setResizable(false);

}
```

```
}
```

D. Le code du bundle « serveurbundlelecture »

Dans cette section je décris le code du bundle installé du côté serveur central « serveurbundlelecture ». Ce bundle comme décrit dans la section 4.2.1, intercepte les données reçues sur le port radio du serveur central, tri et sauvegarde ces données dans une base de données locale.

Activator.java

```
package org.knopflerfish.serveurbundlelecture.impl;
import org.osgi.framework.BundleActivator;
import org.osgi.framework.BundleContext;
public class Activator implements BundleActivator {
      public static BundleContext bc = null;
      private ThreadLecture thread = null;
      public void start(BundleContext bc) throws Exception {
      System.out.println("Demarrage du bundle, interception de
communication radio...");
      Activator.bc = bc;
      thread = new ThreadLecture();
      thread.start();
      }
      public void stop(BundleContext bc) throws Exception {
      System.out.println("arret du bundle, fin communication
radio...");
      thread.stopThread();
      thread.join();
      }
   }
```

FormatPos.java (version 2)

```
package org.knopflerfish.serveurbundlelecture.impl;

public class FormatPos{

      private float latitude;
      private float longitude;
      private String signal_qualite;
      private String precision_source;
```

```
      private int precision_niveau;
      private double vitesse;
      private int PDOP;
      private String message="";
      private static String[] Tab=null;
      private String utilisateur="";



public FormatPos(String msg){
            message=msg;
            Tab=message.split(";");
      }

private String Lire(int pos){
            String position=Tab[pos];
            return position;
      }


public float Latitude(){

            latitude=Float.valueOf(Lire(0)).floatValue();
            return latitude;
            }
public float Longitude(){

            longitude=Float.valueOf(Lire(1)).floatValue();
            return longitude;
            }
public String Qualite(){

      signal_qualite=Lire(2);
      return signal_qualite;
      }
public double Vitesse(){

      vitesse=Double.valueOf(Lire(3)).doubleValue();
      return vitesse;
      }
public int PDOP(){

      PDOP=(Lire(4)=="1") ? 1 : 0;
      return PDOP;
      }
public String Source(){
      precision_source=Lire(5);
      return precision_source;
}

public int Precision(){
      precision_niveau= Integer.parseInt(Lire(6));
      return precision_niveau;
```

```
}
public String Utilisateur(){
      utilisateur=Lire(7);
      return utilisateur;

}
}
```

PortCom.java (version 2)

```
package org.knopflerfish.serveurbundlelecture.impl;
import java.io.*;
import javax.comm.CommPort;
import javax.comm.CommPortIdentifier;
import javax.comm.SerialPort;

public class PortCom {

      /** Flux entrant */
        protected BufferedReader is;
        /** Flux de sortie */
        protected PrintStream os;
        CommPort port;
        int BAUD=115200;

public  PortCom(String nomport) throws Exception {

    // portId reçoit l'identificateur du port choisit
    CommPortIdentifier portId =
      CommPortIdentifier.getPortIdentifier(nomport);

    // Ouverture port
    port = portId.open("Connexion au GPS", 10000);
    System.out.println("ouverture en cours ");
    SerialPort monPort = (SerialPort) port;
    System.out.println("ouverture accomplie ");

    // configuration du port série
    monPort.setSerialPortParams(BAUD, SerialPort.DATABITS_8,
      SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);

    try {
      is = new BufferedReader(new
InputStreamReader(port.getInputStream()));
    } catch (IOException e) {
      System.err.println("ne peut lire le flux entrant: lecture-
seule");
      is = null;
    }
    os = new PrintStream(port.getOutputStream(), true);
```

```
}
public String Lire(){

            String in;

                  try {
                        if (is.ready()==true){
                        in=is.readLine();
                        return in;
                        }
                  } catch (IOException e) {
                        System.out.println("impossible de lire sur le
port");
                        e.printStackTrace();
                  }
                  return ("RAS");
}

public void Fermercnx() throws IOException{
System.out.println("fermeture flux de sortie");
os.close();
System.out.println("fermeture flux d'entrée");
is.close();
System.out.println("fermeture du port");
port.close();
System.out.println("fin de communication sur le port");
}
}
```

ThreadLecture.java

```
package org.knopflerfish.serveurbundlelecture.impl;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
//import com.mysql.jdbc.*; /*import désactivé car non supporté
actuellement dans la version 2.x de knopflerfish*/

public class ThreadLecture extends Thread {
private boolean running = true;
private PortCom port=null;
FormatPos message=null;

private void enregistrer(float latitude, float longitude, double
vitesse, String source, String ID) {
      try {
```

```
            Statement stmt;
            ResultSet rs;

            //Enregistrement du pilote JDBC pour MySQL.
            Class.forName("com.mysql.jdbc.Driver").newInstance();

            String url =
               // "jdbc:mysql://sunensweb.uqac.ca/agoundaf";
            "jdbc:mysql://db4free.net:3306/goundafi";
            Connection con =
                DriverManager.getConnection(
                // url,"agoundaf", "123sesame");
     url,"abdelali", "sesame");
            //Affichage information de connection
            System.out.println("URL: " + url);
            System.out.println("Connection: " + con);

            //creation d'un objet statement
            stmt = con.createStatement();
boolean id=false;
String x1=String.valueOf(latitude);
String x2=String.valueOf(longitude);
String ch=x1+"? "+x2;
System.out.println("chaine: " + ch);
          id=  stmt.execute(

                    "SELECT id from position where point='"+ch+"'");

          if (id)  {
            stmt.executeUpdate(
                          "UPDATE position " +
                          "set `point`='"+ch+"',
`vitesse`='"+vitesse+"', `precision`='"+source+"', `GPS_ID`='"+ID+
                          "' WHERE `id`="+id);
                  con.close();
          }
          else
          {
            stmt.execute(

                        "INSERT INTO position (`point`, `vitesse`,
`precision`, `GPS_ID`)VALUES ('"+ch+"', '"+vitesse+"', '"+source+"',
'"+ID+"')");
                        con.close();
          }
    }catch( Exception e1 ) {
      e1.printStackTrace();
    }
}

public ThreadLecture() {
 try {
      port=new PortCom("COM12");
```

```
} catch (Exception e) {
      System.out.println("Impossible de creer le port: ");
      e.printStackTrace();
}
}
public void run() {
String msg = "";
System.out.println("tentative de lecture sur le port COM12: ");

while (running) {

      msg = port.Lire();
      try {
            if (msg!="RAS"&&port.is.ready()==true){
                  System.out.println("succé lecture nouvelles données
sur COM12: "+msg);
                  message=new FormatPos(msg);
                  if (message.Precision()>3  && message.PDOP()==1)
                    {
                        //enregistrer(message.Latitude(),
message.Longitude(), message.Vitesse(), message.Source(),
message.Utilisateur()); /*opération BD bloquée*/
                        System.out.println("Données fiables,
enregistrées sur BD ");
                    }
            }
      } catch (IOException e1) {
            e1.printStackTrace();
      }
      System.out.println("fin de boucle");


try {
Thread.sleep(5000);
} catch (InterruptedException e3) {
System.out.println(" ERREUR Thread: " + e3);
}
}
System.out.println("Arret thread");
}



public void stopThread() {
System.out.println("Arret thread en cours");
this.running = false;
try {
      port.Fermercnx();
} catch (IOException e4) {
      System.out.println("Erreur fermeture connexion");
      e4.printStackTrace();
}
```

```
}
}
```

E. Les pages web de l'application GoogleMap du côté serveur

Dans cette section je montre les codes utilisés pour notre application du coté serveur accessible via des pages web. Les codes utilisés sont écrits en PHP, HTML, Javascript en utilisant l'API GoogleMap et le langage SQL pour les opérations sur notre base de données.

Index.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Page d'authentification</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<style type="text/css">
<!--
body {
margin-top: 5cm;
}
-->
</style></head>

<body>
<div align="center">
 <p><strong>Veuillez entrer votre nom d'usager, mot de passe et l'application demandée </strong></p>
 <p> 
  </p>
</div>
<form action="login.php" method='post'>
<table width="251" border="0" align="center">
 <tr>
  <td>Utilisateur :</td>
  <td><input type="text" name="login" maxlength="250"></td>
 </tr>
 <tr>
  <td>Mot de passe </td>
  <td><input type="password"name="pass" maxlength="10"></td>
 </tr>
 <tr>
  <td>Application</td>
  <td><select name="select">
  <option>police</option>
  <option>vehicules</option>
  <option>sante</option>
 </select></td>
 </tr>
 <tr>
  <td colspan="2" align="center"><input type="submit" value="acceder"></td>
```

```
 </tr>
</table>
</form>
</body>
</html>
```

Login.php

```
<?
/* Ouverture de la connexion à notre base de donnée*/
$link = mysql_connect("localhost", "agoundaf", "gpsuqac") or die("Could not connect: " . mysql_error());
mysql_selectdb("agoundaf",$link) or die ("Ne peut utiliser dbmapserver : " . mysql_error());
/*si tous les champs du formulaire sont remplis on continue le traitement*/
if(isset($_POST) && !empty($_POST['login']) && !empty($_POST['pass'])) {
 extract($_POST);
 /* on recupère le mot de passe de la table qui correspond au login du visiteur*/
 $sql = "select pass from authentification where login='".$login."'";
 $req = mysql_query($sql) or die('Erreur SQL !<br>'.$sql.'<br>'.mysql_error());
 $data = mysql_fetch_assoc($req);
//si le mot de passe ne concorde pas on remet le formulaire d'authentification
 if($data['pass'] != $pass) {
        echo('<p align="center">Mauvais login / mot de passe. Merci de recommencer</p>');
  include('index.html'); // On inclut le formulaire d'identification
  exit;
 }
 else //si le mot de passe concorde, on verifie si l'utilisateur a un droit d'accés à l'application choisie
        {
        $sql = "select id from applications where login='".$login."' and app='".$select."'";
        $req = mysql_query($sql) or die('Erreur SQL !<br>'.$sql.'<br>'.mysql_error());
        $data = mysql_fetch_assoc($req);

        if ($data)
                {
                session_name($login);
                session_start();
        $_SESSION['login'] = $login;
        echo '<p align="center">Vous etes bien logué</p>';
        include('carte.html'); /* On inclut la page de l'application, dans cet exemple on inclus directement
                la carte des positions d'objets géolocalisables stockés dans notre base de données du serveur central et on
ferme la page courante*/
        exit;
                }
        else
                {
                echo("<p align=\"center\">Vous n'etes pas autorisé à utiliser l'application: $select</p>");
                include('index.html'); // On inclut le formulaire d'identification si l'utilisateur n'est pas autorisé à acceder à
l'application
                exit;
                }
        }
}
else {
        echo ('<p align="center">Vous devez remplir tous les champs.</p>');
        include('index.html'); // On inclut le formulaire d'identification si on a pas remplis tous les champs d'identification
        exit;
}
?>
```

Carte.html

```
<?
$login=$_SESSION['login'];
session_name($login);
session_start();

/*
si la variable de session login n'existe pas cela siginifie que le visiteur
n'a pas de session ouverte, il n'est donc pas logué ni autorisé à
acceder à l'espace membres
*/
if(($_SESSION['login']==NULL) || ($_SESSION['login']==""){
        echo "<p align=\"center\">Vous n'êtes pas autorisé à acceder à cette zone</p>";
        include('index.html');
        exit;
        }
else
        {
        $utisateur=$_SESSION['login'];
        echo "<p align=\"center\">Bienvnue: $utisateur</p>";
        }
?>
<html>
<head>
<title>Projet UQAC GPS</title>
<script
src="http://maps.google.com/maps?file=api&v=2&key=ABQIAAAAilMtvyf4fUjlRHOz4coiGhR_oxCzrNtMB78bo91aW47W28DBtBSx3
MqJ1q941AUWFCov1nzJV16yGw" type="text/javascript"></script>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<style type="text/css">
</style>
</head>
<body>
<form method="post" action="logoff.php">
 <div align="right">
<?
$cache=$_SESSION['login'];
echo ("<input name=\"cache\" type=\"hidden\" value=\"$cache\">");
?>
  <input type="submit" name="Submit" value="D&eacute;connexion">
 </div>
</form>
<p align="center"> </p>
<p align="center"><strong>
 <object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=5,0,0,0" width="813" height="23"
align="texttop">
  <param name="movie" value="text1.swf">
  <param name="quality" value="high">
  <param name="BGCOLOR" value="#FFFFFF">
  <embed src="text1.swf" width="813" height="23" align="texttop" quality="high"
pluginspage="http://www.macromedia.com/shockwave/download/index.cgi?P1_Prod_Version=ShockwaveFlash"
type="application/x-shockwave-flash" bgcolor="#FFFFFF" ></embed>
 </object>
</strong></p>
<div align="left" id="map" style="width: 1360px; height: 768px">
 <script type="text/javascript">
//<![CDATA[

var iconR = new GIcon();
```

```
iconR.image = "http://labs.google.com/ridefinder/images/mm_20_red.png";
iconR.shadow = "http://labs.google.com/ridefinder/images/mm_20_shadow.png";
iconR.iconSize = new GSize(12, 20);
iconR.shadowSize = new GSize(22, 20);
iconR.iconAnchor = new GPoint(6, 20);
iconR.infoWindowAnchor = new GPoint(5, 1);

var iconV = new GIcon();
iconV.image = "http://labs.google.com/ridefinder/images/mm_20_green.png";
iconV.shadow = "http://labs.google.com/ridefinder/images/mm_20_shadow.png";
iconV.iconSize = new GSize(12, 20);
iconV.shadowSize = new GSize(22, 20);
iconV.iconAnchor = new GPoint(6, 20);
iconV.infoWindowAnchor = new GPoint(5, 1);

var iconG = new GIcon();
iconG.image = "http://labs.google.com/ridefinder/images/mm_20_gray.png";
iconG.shadow = "http://labs.google.com/ridefinder/images/mm_20_shadow.png";
iconG.iconSize = new GSize(12, 20);
iconG.shadowSize = new GSize(22, 20);
iconG.iconAnchor = new GPoint(6, 20);
iconG.infoWindowAnchor = new GPoint(5, 1);

// Fonction de création de points et utilisation des icones
function Marker_rouge(point) {
var marker = new GMarker(point, iconR);
return marker;
}

function Marker_vert(point) {
var marker = new GMarker(point, iconV);
return marker;
}

function Marker_gris(point) {
var marker = new GMarker(point, iconG);
return marker;
}

var map = new GMap2(document.getElementById("map"));
map.addControl(new GLargeMapControl());
map.addControl(new GMapTypeControl());
map.addControl(new GScaleControl());


/* Les fonctions suivantes créent des points de couleurs rouge, vert et gris */
function createMarkerR(point, text)
{
var marker1 = new Marker_rouge(point);
/* Ajoute un texte à l'infobulle qui s'affiche en cliquant sur le point */
var html = text;
GEvent.addListener(marker1, "click", function() {marker1.openInfoWindowHtml(html);});
return marker1;
};

function createMarkerV(point, text)
{
var marker1 = new Marker_vert(point);
/* Ajoute un texte à l'infobulle qui s'affiche en cliquant sur le point */
var html = text;
GEvent.addListener(marker1, "click", function() {marker1.openInfoWindowHtml(html);});
return marker1;
```

```
};

function createMarkerG(point, text)
{
var marker1 = new Marker_gris(point);
/* Ajoute un texte à l'infobulle qui s'affiche en cliquant sur le point */
var html = text;
GEvent.addListener(marker1, "click", function() {marker1.openInfoWindowHtml(html);});
return marker1;
};

<?php
 function pointStringToCoordinates($pointString) {
      $coordinates = explode(" ", $pointString);


      return array("x" => (float)$coordinates[0], "y" => (float)$coordinates[1]);
   }


$link = mysql_connect("localhost", "agoundaf", "gpsuqac") or die("Could not connect: " . mysql_error());
mysql_selectdb("agoundaf",$link) or die ("Can\'t use dbmapserver : " . mysql_error());

$rs1 = mysql_query("SELECT * FROM position",$link);

if (!$rs1)
{
echo "no results ";
}
/* ajout de la classe pointLocation incluse dans le fichier PointInPolygon.php qui permet de tester
si un point est inclus dans un polygone comme une route par exemple*/
include_once("pointInPolygon.php");



while($row1 = mysql_fetch_array($rs1))
{
 $qzone="SELECT * FROM  zone";

 $rs2 = mysql_query($qzone,$link);
 $trouve=false;


 while($row2 = mysql_fetch_array($rs2))
 {
 $chainepoint=$row1['point'];
 $point = $chainepoint;
 $chainezone=$row2['zone'];
 $chainezone=explode(',', $chainezone);
$polygon=$chainezone;



 $pointLocation = new pointLocation();
 $inclus=$pointLocation->pointInPolygon($point, $polygon);
 $coord=pointStringToCoordinates($row1['point']);



 if (($inclus==1) and ($row1['vitesse']>$row2['limite']))
  {
echo "var point = new GLatLng(" . $coord['x'] . "," . $coord['y'] . ");\n";
```

```
 echo "var marker = createMarkerR(point, 'vitesse:" . addslashes($row1['vitesse']) ." <br> Source:". addslashes($row1['precision']) .
" <br> date:". addslashes($row1['dat']) ."<br> _____ <cite><em>Copyright
A.Goundafi</em></cite>');\n";
 echo "map.addOverlay(marker);\n";
 echo "\n";
 $trouve=true;
 break;
  }


 else if($inclus==1 and $row1['vitesse']<=$row2['limite'])
 {
 echo "var point = new GLatLng(" . $coord['x'] . "," . $coord['y'] . ");\n";
 echo "var marker = createMarkerV(point, 'vitesse:" . addslashes($row1['vitesse']) ." <br> Source:". addslashes($row1['precision']) .
" <br> date:". addslashes($row1['dat']) ."<br> _____ <cite><em>Copyright
A.Goundafi</em></cite>');\n";
 echo "map.addOverlay(marker);\n";
 echo "\n";
 $trouve=true;
 break;
 }
 }//fin while zones

 if($trouve==false)
 {
 $coord=pointStringToCoordinates($row1['point']);
 echo "var point = new GLatLng(" . $coord['x'] . "," . $coord['y'] . ");\n";
 echo "var marker = createMarkerG(point, 'vitesse:" . addslashes($row1['vitesse']) ." <br> Source:". addslashes($row1['precision']) .
" <br> date:". addslashes($row1['dat']) ."<br> _____ <cite><em>Copyright
A.Goundafi</em></cite>');\n";
 echo "map.addOverlay(marker);\n";
 echo "\n";
 }
}//fin while points

mysql_close($link);
?>
map.setCenter(new GLatLng(48.39555, -71.08789), 11, G_HYBRID_MAP);
//]]>
 </script>
</div>

</body>
</html>
```

pointInPolygon.php

```
<?php
class pointLocation {
   var $pointOnVertex = true; // Check if the point sits exactly on one of the vertices

   function pointLocation() {
   }


      function pointInPolygon($point, $polygon, $pointOnVertex = true) {
      $this->pointOnVertex = $pointOnVertex;
```

```
    // Transform string coordinates into arrays with x and y values
    $point = $this->pointStringToCoordinates($point);
    $vertices = array();
    foreach ($polygon as $vertex) {
      $vertices[] = $this->pointStringToCoordinates($vertex);
    }

    // Check if the point sits exactly on a vertex
    if ($this->pointOnVertex == true and $this->pointOnVertex($point, $vertices) == true) {
      return "1";
    }

    // Check if the point is inside the polygon or on the boundary
    $intersections = 0;
    $vertices_count = count($vertices);

    for ($i=1; $i < $vertices_count; $i++) {
      $vertex1 = $vertices[$i-1];
      $vertex2 = $vertices[$i];
      if ($vertex1['y'] == $vertex2['y'] and $vertex1['y'] == $point['y'] and $point['x'] > min($vertex1['x'], $vertex2['x']) and
$point['x'] < max($vertex1['x'], $vertex2['x'])) { // Check if point is on an horizontal polygon boundary
        return 1;
      }
      if ($point['y'] > min($vertex1['y'], $vertex2['y']) and $point['y'] <= max($vertex1['y'], $vertex2['y']) and $point['x'] <=
max($vertex1['x'], $vertex2['x']) and $vertex1['y'] != $vertex2['y']) {
        $xinters = ($point['y'] - $vertex1['y']) * ($vertex2['x'] - $vertex1['x']) / ($vertex2['y'] - $vertex1['y']) + $vertex1['x'];
        if ($xinters == $point['x']) { // Check if point is on the polygon boundary (other than horizontal)
          return 1;
        }
        if ($vertex1['x'] == $vertex2['x'] || $point['x'] <= $xinters) {
          $intersections++;
        }
      }
    }
    // If the number of edges we passed through is even, then it's in the polygon.
    if ($intersections % 2 != 0) {
      return 1;
    } else {
      return 0;
    }
  }

  function pointOnVertex($point, $vertices) {
    foreach($vertices as $vertex) {
      if ($point == $vertex) {
        return true;
      }
    }

  }

  function pointStringToCoordinates($pointString) {
    $coordinates = explode("?", $pointString);
    return array("x" => $coordinates[0], "y" => $coordinates[1]);
  }
}

?>
```

La fonction pointInPolygon existe sur internet en source libre. J'ai adapté cette fonction pour pouvoir lire mes coordonnées (Longitude, Latitude) et les polygones stockés dans ma base de données (les polygones représentent des routes). Cette fonction permet de tester si un point géographique (x,y) appartient à un polygone (x1y1, x2y2, x3y3, x4y4, etc.).

Logoff.php

```
<?
        session_name($cache);
        $_SESSION = array();
        session_destroy();
        echo ("<p align=\"center\">Déconnexion: $cache</p>");
        header ("Location: index.html");// On redirige vers la page d'authentification
?>
```