Requêtes dépendantes de la localisation : Expression, évaluation et optimisation

Marie Thilliez

Thèse de Doctorat

Présentée à
L'Université de Valenciennes et du Hainaut-Cambrésis

Pour l'obtention du
Doctorat de l'Université de Valenciennes et du Hainaut-Cambrésis

Spécialité Automatique et Informatique
Des Systèmes Industriels et Humains

Discipline : Informatique

par
Marie Thilliez

Requêtes Dépendantes de la Localisation en Environnements Mobiles : Expression, Evaluation et Optimisation

Soutenue le 3 décembre 2004 devant le jury composé de :

Directeur de thèse :	Arnaud FREVILLE, Professeur à l'Université de Valenciennes
Rapporteurs :	Christine COLLET, Professeur à l'Institut National Polytechnique de Grenoble
	Philippe PUCHERAL, Professeur à l'Université de Versailles
Examinateurs :	Gérôme CANALS, Maître de Conférences à l'Université de Nancy 2
	Thierry DELOT, Maître de Conférences à l'Université de Valenciennes
	Christophe GRANSART, Chargé de recherches à l'INRETS
	Sylvain LECOMTE, Maître de Conférences à l'Université de Valenciennes

UNIVERSITE DE VALENCIENNES ET DU HAINAUT-CAMBRESIS
Le Mont-Houy, 59313 VALENCIENNES Cedex 9
Tél : (33) 03.27.51.12.34 – Fax : (33) 03.27.51.11.00

Remerciements

Une thèse se construit sur des rencontres et des échanges. Je voudrais remercier ici toutes les personnes qui ont contribué — directement ou indirectement — à l'aboutissement de mes travaux de thèse.

Tout d'abord, je remercie Arnaud Fréville pour la confiance qu'il m'a accordée durant ces trois années de thèse et pour avoir accepté d'être mon directeur de thèse.

Je tiens à remercier particulièrement mes encadrants de thèse : Thierry Delot et Sylvain Lecomte pour leurs conseils, leur exigence, leurs commentaires et leurs critiques tout au long de ces années de thèse. Je les remercie également pour leur disponibilité, leur patience, leur compréhension et leur complicité qui ont su répondre à mes questions et mes doutes et qui nous ont permis de travailler dans une ambiance conviviale.

Je remercie Christine Collet pour m'avoir fait l'honneur de rapporter mon travail, pour sa disponibilité, son investissement et ses précieux conseils qui m'ont aidée à améliorer la qualité de mon mémoire.

Je remercie Philippe Pucheral pour avoir accepté d'être rapporteur de ma thèse et pour ses remarques constructives concernant le manuscrit. Je le remercie également pour m'avoir fait l'honneur de présider le jury.

Je tiens également à remercier Gérôme Canals et Christophe Gransart qui me font le plaisir de participer à mon jury et pour nos discussions enrichissantes lors des réunions du groupe de travail « Mobilité et Ubiquité ».

Les travaux présentés dans ce mémoire ont été réalisés au sein de l'équipe « Recherche Opérationnelle et Informatique » de l'université de Valenciennes : je tiens donc, à remercier Frédéric Semet qui dirige cette équipe pour son accueil, ainsi que tous les membres de l'équipe. Je remercie tout particulièrement les anciens et actuels membres de SID : Colombe, David, Didier, Hocine, Nadia et Sergiy. Je remercie également Emmanuel et René pour les conseils qu'ils m'ont apportés au fil de ces années.

Je tiens à remercier Yohan Colmant, qui a activement participé à la réalisation du prototype, pour l'aide précieuse qu'il m'a apportée durant son stage.

Je remercie mes proches qui m'ont soutenue tout au long de ces trois années de thèse.

Je remercie particulièrement mes parents qui m'ont toujours encouragée, écoutée sans pour autant essayer d'influencer mes choix. L'intérêt qu'ils ont montré pour mon travail m'a beaucoup aidée : un clin d'œil à mon père qui m'a régulièrement « googlelisé » ces dernières années pour se tenir au courant de l'avancée de mes recherches !

Je remercie également mes grands-parents, Elise, Yann, Pierre, Laurette et Cathy pour avoir su m'écouter et pour m'avoir aidée à relativiser les diverses difficultés que j'ai rencontrées.

Finalement, je remercie infiniment Olivier pour tout : pour ses conseils techniques, ses critiques constructives, son investissement dans les expérimentations réalisées au cours de cette thèse mais également pour ses encouragements, sa compréhension, sa patience, son écoute et tout simplement parce qu'il était là, avec moi … MERCI.

Tables des matières

Table des figures

Chapitre 1

Chapitre 2

Chapitre 3

Chapitre 4

Chapitre 5

Chapitre 6

Liste des tableaux

Chapitre 2

Chapitre 3

Chapitre 4

Chapitre 5

Chapitre 1

Introduction

1. Motivations

Depuis quelques années, nous assistons à une nouvelle tendance dans le développement des applications informatiques. La généralisation des réseaux sans fil combinée à la multiplication des systèmes électroniques tels que les terminaux nomades [VAR 00] transforme en profondeur les applications informatiques. Ces différentes évolutions ont favorisé l'apparition de l'informatique ubiquitaire [GOL 01] dont l'objectif est la conception d'environnements informatiques accessibles n'importe où et quel que soit le terminal utilisé par l'utilisateur. En effet, il y a un peu plus d'une vingtaine d'années, plusieurs personnes utilisaient un seul et unique ordinateur ; aujourd'hui, une même personne a souvent à sa disposition, plusieurs ordinateurs : par exemple, un ordinateur de bureau traditionnel, un ordinateur portable, et différents terminaux nomades tels qu'un téléphone cellulaire, un assistant personnel ou PDA (PDA – Personal Digital Assistant), un appareil photo numérique ou encore un lecteur mp3.

Un des défis pour le développement de l'informatique ubiquitaire est de fournir des applications et des services adaptés aux besoins des utilisateurs selon l'endroit où ils se situent et selon l'infrastructure matérielle et logicielle dont ils disposent. En effet, un utilisateur mobile doit pouvoir accéder à l'information disponible qui l'intéresse quel que soit le terminal qu'il utilise et quelle que soit sa localisation. L'augmentation du nombre des usagers représente un potentiel important qui encourage les fournisseurs de services à mettre à la disposition de ces usagers de nouveaux services et de nouvelles applications. Par exemple, dans le contexte de la téléphonie mobile, de nombreux services sont proposés aux usagers tels que la messagerie instantanée ou la consultation de leur compte bancaire par SMS (SMS – Short Message Service). Dans ce contexte, nous avons proposé une nouvelle classe d'applications, les applications de proximité [THI 02].

2. Contexte : les applications de proximité

2.1. Définition

Nous définissons les applications de proximité comme des applications permettant à plusieurs utilisateurs physiquement proches les uns des autres de partager de l'information, des services et de communiquer. Au sein d'une application de proximité, deux catégories de participants existent :

- les participants fixes : ils sont munis d'ordinateurs de bureau traditionnels par exemple. En effet, les applications de proximité utilisent et reposent souvent sur une infrastructure fixe, éventuellement filaire, déjà existante autour de laquelle gravitent les participants mobiles munis de terminaux nomades. Cette infrastructure fixe est composée de différents ordinateurs fixes et souvent plus performants que les terminaux nomades.
- les participants mobiles sont des utilisateurs munis de terminaux nomades qui communiquent généralement grâce aux réseaux mobiles et/ou sans fil.

Les applications de proximité s'appuient donc sur l'utilisation de réseaux mobiles et/ou sans fil. Un *réseau sans fil* est un réseau dans lequel au moins deux terminaux peuvent communiquer sans liaison filaire. Les réseaux sans fil reposent sur l'utilisation d'ondes radioélectriques (radio ou infrarouges) en lieu et place des câbles habituels. Grâce à un réseau sans fil, un utilisateur a la possibilité de rester connecté en se déplaçant dans un périmètre géographique plus ou moins étendu qu'on appelle *zone de couverture*. Si différents usagers possèdent des capacités réseaux sans fil et qu'ils se situent suffisamment proches les uns des autres, les différentes zones de couverture de leurs réseaux respectifs vont se juxtaposer. Si les zones de couverture sont issues de réseaux compatibles entre eux (c'est-à-dire qu'ils permettent la communication), l'association de ces différentes zones de couverture juxtaposées, sera appelée *sphère de communication* et permettra aux différents utilisateurs de communiquer entre eux.

Certains réseaux sans fil sont fixes : ils sont basés sur une infrastructure fixe à laquelle se connectent différents usagers mobiles. La situation géographique de la zone de couverture de ce réseau sans fil ne dépend donc pas des utilisateurs mobiles mais de l'infrastructure fixe.

En revanche, certains réseaux sans fil sont mobiles : dans ce cas, les utilisateurs de terminaux nomades disposent de fonctionnalités réseaux autonomes qui leur permettent d'être entourés de leur propre réseau sans fil. Ce réseau devient alors mobile en fonction des déplacements de l'utilisateur. Différents utilisateurs peuvent communiquer via un réseau sans fil et mobile si le type de réseau utilisé par chaque utilisateur est identique et si leurs zones de couvertures se juxtaposent. Ils se trouvent donc, dans la même sphère de communication.

Dans une application de proximité, les participants peuvent échanger de l'information, communiquer dans le cas où ils sont à proximité les uns de autres. D'après le Larousse [Larousse], la *proximité* est la situation de quelqu'un ou de quelque chose qui se trouve à peu de distance de quelqu'un ou de quelque chose d'autre. La distance séparant les participants peut varier de quelques mètres à plusieurs centaines de mètres, nous utilisons ici la notion de

proximité en opposition avec les réseaux à grande échelle comme Internet ou les réseaux de téléphonie mobile tels que le GSM (GSM – Global System for Mobile Communication), le GPRS (GPRS – General Packet Radio Service) ou l'UMTS (UMTS – Universal Mobile Telecommunication System) qui n'incluent pas la notion de proximité entre les utilisateurs. Les applications de proximité sont donc davantage basées sur des réseaux dont l'envergure est limitée comme les réseaux locaux (WiFi par exemple) et les réseaux personnels tels que Bluetooth (ou IEEE 802.15.1 [IEEE 802.15]).
Chaque participant (fixe ou mobile) d'une application de proximité a la possibilité de rendre disponible certaines informations stockées sur son terminal. L'ensemble des informations partagées par les différents participants représente alors les informations disponibles et accessibles par tous les participants de l'application de proximité. Les informations disponibles sont donc distribuées sur les différents terminaux présents dans l'application.
Un exemple simple d'applications de proximité, sans infrastructure, peut se dérouler durant une conférence : plusieurs participants de la conférence choisissent de partager des documents tels que leur présentation et ces documents peuvent donc être récupérés par les participants se situant « à proximité ».

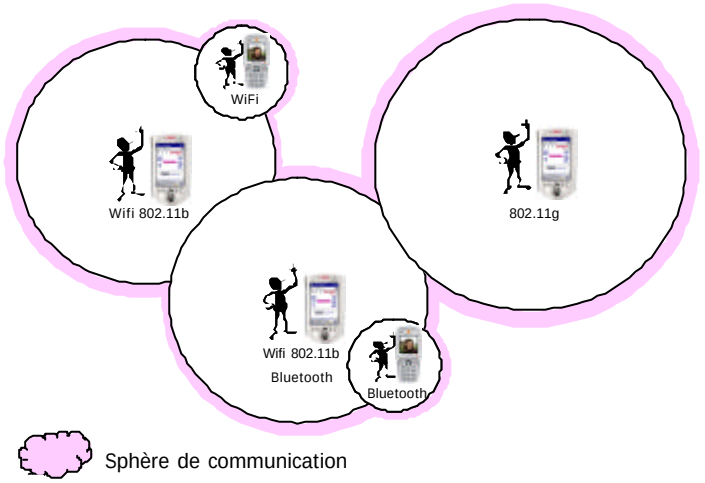


Figure 1.1. Exemple de formation d'une sphère de communication

Grâce à la formation des sphères de communication illustrée par la figure 1.1, les informations partagées et réparties sur les noeuds (i.e. terminaux) de l'application deviennent disponibles et accessibles pour les participants. Ces informations représentent des données partagées telles que des photos ou des fichiers vidéo par exemple ou des services partagés tels qu'un service de développement de photos numériques. Ces informations peuvent être obtenues par un participant de l'application de façon proactive : certaines informations sont diffusées et l'utilisateur peut paramétrer ses préférences afin de ne recevoir que les informations qui sont susceptibles de l'intéresser. D'autre part, un utilisateur peut également obtenir des informations de façon réactive : en effet, l'utilisateur, alors appelé *client*, peut émettre une requête concernant l'information qu'il recherche.

En outre, dans les applications de proximité, la plupart des participants sont munis de terminaux nomades et donc potentiellement mobiles. Lorsqu'ils choisissent d'émettre une requête au sein de l'application de proximité, la prise en compte de leur mobilité pour l'évaluation de la requête peut devenir un critère important permettant d'affiner et de personnaliser la réponse. En effet, la mobilité des utilisateurs peut être considérée en proposant aux clients d'émettre des requêtes en fonction de leurs localisations géographiques. Si un participant recherche un autre participant de l'application de proximité, considérer sa localisation dans l'évaluation de la requête n'est en général pas utile puisque les différents participants sont physiquement proches les uns des autres. Cependant, les informations partagées et recherchées au sein de l'application peuvent correspondre à des informations relatives à une localisation distante du terminal sur lequel elle est stockée. Par exemple, un utilisateur peut choisir de stocker et de partager sur son terminal, une liste de restaurants italiens sachant que ces restaurants italiens ne sont pas forcément situés dans la sphère de communication. Dans ce cas, si un participant recherche un restaurant italien, la requête peut être évaluée en fonction de la localisation du participant afin de lui fournir le restaurant italien le plus proche de lui. Ces recherches d'information en fonction de la localisation du client sont aussi appelées requêtes dépendantes de la localisation (LDQs – Location Dependent Queries).

2.2. Contraintes de l'environnement

Les applications de proximité reposent à la fois sur l'utilisation de terminaux nomades et de réseaux sans fil et/ou mobiles. Ces nouvelles technologies engendrent de nouvelles contraintes à considérer dans le développement d'applications.

Dans un premier temps, considérons les terminaux nomades. Le nombre d'usagers de ces terminaux ne cesse de croître [INS 04] depuis quelques années. Ce développement peut être attribué au moindre coût, à la facilité d'utilisation de ces terminaux en terme de portabilité en opposition avec les ordinateurs de bureau standards. Ces terminaux évoluent rapidement et actuellement, comme l'illustre le tableau décrit dans l'annexe 1, de plus en plus de terminaux regroupent plusieurs fonctionnalités [HIN 01].

Cependant, même si ces terminaux sont de plus en plus performants, il est nécessaire, pour développer des applications adaptées à ce type d'appareils portables, de prendre en considération les contraintes inhérentes à ces terminaux. Ces contraintes sont, par ordre d'importance :

- tout d'abord l'utilisation de batterie comme source d'énergie induit une autonomie limitée qui représente, selon nous, la contrainte la plus cruciale à considérer dans le développement d'applications dédiées à ce type de terminaux.
- Ensuite, la compacité de ces terminaux engendre également des limites en terme d'entrées/sorties, souvent rudimentaires sur ce type d'appareils.
- Finalement, la capacité de stockage et les ressources CPU de ces terminaux peuvent encore parfois représenter une contrainte pour le développement des applications mais

ces contraintes tendent à disparaître avec les avancées technologiques telles que l'apparition et le développement des cartes de stockages externes Secure Digital ou Compact Flash par exemple.

Finalement, les différentes caractéristiques des appareils portables sont très hétérogènes (cf. annexe 1) et cette hétérogénéité rend difficile le développement et la portabilité des applications dédiées aux terminaux nomades.

Les réseaux sans fil et/ou mobiles ont quant à eux connu, ces dernières années, un développement rapide du en particulier à une installation de ces réseaux demandant un moins grand investissement en terme d'infrastructure que les réseaux filaires. En contrepartie, ces réseaux sans fil présentent certains inconvénients :

- Les inconvénients dus à la qualité du signal : en effet, un signal radio est soumis à différentes interférences (obstacles physiques : murs, plafonds, autre ondes radio) qui peuvent limiter sa portée et sa qualité.
- Même si les débits théoriques offerts par ces réseaux sont parfois relativement élevés (11 Mbps pour le WiFi ou 54 Mbps pour le 802.11g), les débits réels constatés sont beaucoup moins importants et dépendent d'une part de la distance séparant les deux machines mais également du type des machines [HEU 03].
- Dans les réseaux sans fil et en particulier dans les réseaux locaux sans fil, l'accès facilité des utilisateurs pose le problème de la surcharge du réseau et il est donc important de considérer le partitionnement du réseau afin de limiter ces problèmes.
- Les ondes radioélectriques sont utilisées par un grand nombre d'applications (militaires, scientifiques, amateurs, etc.), et sont sensibles aux interférences. C'est la raison pour laquelle une réglementation est nécessaire dans chaque pays afin de définir les plages de fréquence et les puissances auxquelles il est possible d'émettre pour chaque catégorie d'utilisation.
- L'échange des informations via des ondes radioélectriques rend plus aisé le piratage : en effet, l'accès au réseau et aux informations disponibles sur le réseau peut être possible, a priori, à partir du moment où un utilisateur est dans la sphère de communication. Les solutions récentes telles que le protocole WTLS (WTLS – Wireless Transport Layer Security) présentent encore des failles et la sécurité et la confidentialité des données circulant sur les réseaux sans fil restent problématiques.
- Finalement, un des problèmes les plus importants dans le cadre de réseaux sans fil concerne les déconnexions fréquentes. Ces déconnexions peuvent être volontaires : dans le cas d'un utilisateur qui souhaite économiser sa batterie et préfère donc se déconnecter. Mais elles peuvent également être accidentelles : par exemple, les déconnexions peuvent être causées par les déplacements des utilisateurs qui sortent de la zone de couverture du réseau ou par des obstacles infranchissables par les ondes radioélectriques.

Finalement, comme pour les terminaux nomades, un des problèmes concernant le développement d'applications basées sur ce type de réseaux est la gestion de l'hétérogénéité.

En effet, il existe de nombreuses technologies se distinguant d'une part par la fréquence d'émission utilisée ainsi que par le débit et la portée des transmissions.

2.3. Différents problèmes à résoudre

Les applications de proximité s'exécutent dans un environnement dynamique à la fois basé sur la mobilité des utilisateurs et sur l'hétérogénéité des terminaux et des réseaux.
Cet environnement dynamique est induit par deux éléments : la mobilité des utilisateurs d'une part et l'hétérogénéité des réseaux et des terminaux d'autre part. Développer des applications adaptées à cet environnement mobile et hétérogène implique de traiter de nombreux problèmes :

- Intégration, mise à l'échelle, déploiement : chaque participant doit pouvoir s'intégrer facilement à une application et les applications doivent évoluer en fonction du nombre de participants, donc mettre en place des mécanismes de reconfiguration et de répartition de charges. De même, le déploiement de ce type d'applications doit se faire de façon dynamique, en fonction des déplacements des usagers.
- Sécurité : les transactions effectuées par les participants au cours d'une application de proximité doivent être sécurisées. Ces transactions peuvent concerner des opérations de paiement mais également l'échange de fichiers personnels par exemple.
- Confidentialité : les informations personnelles des participants doivent être protégées et un participant peut choisir de ne pas dévoiler son identité lorsqu'il partage des informations. Cependant dans de telles applications, il est également important de vérifier l'intégrité et la « légalité » des informations échangées.
- Fiabilité : les déconnexions accidentelles des participants doivent être gérées de façon à ne pas affecter le bon fonctionnement de l'application en cours.
- Personnalisation, adaptation : il est également important de mettre en place des propriétés et des mécanismes de personnalisation et d'adaptation afin de faire face aux contraintes d'hétérogénéité dans un premier temps et de prendre en compte les préférences des usagers dans un second temps.
- Localisation de l'information : dans cet environnement dynamique, la gestion des informations est un problème très important. En effet, à tout moment, les participants doivent pouvoir découvrir et localiser les informations (services, données, etc.) de leurs voisins. De même qu'ils doivent être en mesure de partager leurs propres données. Ces informations doivent par ailleurs rester cohérentes.

Parmi les problèmes à résoudre dans le cadre des applications de proximité, nous nous sommes principalement intéressés à la problématique de localisation de l'information. Parallèlement, en étudiant la problématique de localisation de l'information dans les applications de proximité, nous avons également abordé les problèmes de personnalisation, d'adaptation et de fiabilité. En effet, tout participant d'une application doit être capable de localiser de l'information quelles que soient les ressources dont il dispose.

3. Objectifs de la thèse

Aujourd'hui, les services de localisation sont, pour la plupart, dédiés à des environnements fiables et fixes. Depuis quelques années, d'autres solutions de localisation, adaptées aux terminaux nomades commencent à se développer. Cependant, ces solutions reposent essentiellement sur une centralisation des informations partagées ou sur une infrastructure hiérarchique comme dans les réseaux de téléphonie mobile. Dans une application de proximité, les informations disponibles sont disséminées sur les différents terminaux des participants de l'application de façon non hiérarchique et varient en fonction de leurs déplacements. Cet environnement dynamique et décentralisé requiert donc de définir une nouvelle approche pour la localisation.

1- Le premier objectif de la thèse est donc de proposer un service de localisation permettant de : (i) gérer la répartition des informations et la distribution du processus de localisation dans cet environnement décentralisé, (ii) prendre en compte la dynamicité de l'environnement due à la mobilité des participants de l'application (iii) supporter l'hétérogénéité de l'environnement au niveau des réseaux, des terminaux et des informations partagées (iv) prendre en compte les limites de l'environnement en terme de ressources et optimiser le processus de localisation en fonction de ces contraintes et des caractéristiques de l'utilisateur telles que son profil de mobilité ou sa localisation géographique par exemple.

2- La localisation de l'information dans un contexte mobile doit également considérer la mobilité des utilisateurs et, en particulier, prendre en compte la localisation géographique des utilisateurs mobiles afin de leur fournir une information personnalisée. Aujourd'hui, différents systèmes de localisation tels que le GPS (GPS – Global Positioning system) permettent de localiser un utilisateur. Cependant ces solutions ne sont pas encore uniformément répandues, de plus, elles ne sont utilisables que dans certains environnements : par exemple, principalement en extérieur pour le GPS. Le deuxième objectif de la thèse est donc de fournir une solution de localisation géographique permettant à un utilisateur mobile connecté de se localiser quelques soient les ressources dont il dispose et ne nécessitant pas d'infrastructure spécifique.

3- Ensuite, lorsque la localisation géographique d'un utilisateur est connue, il est alors possible de la prendre en compte dans l'évaluation des requêtes exprimées par cet utilisateur en lui permettant, en particulier, d'exprimer des requêtes dépendantes de la localisation. Le troisième objectif de la thèse concerne donc la proposition d'un modèle d'évaluation adapté permettant de considérer la localisation géographique du client dans le processus d'évaluation d'une requête. Ce modèle d'évaluation doit également prendre en compte les contraintes des ressources des terminaux nomades, en particulier en terme de consommation d'énergie.

4- Finalement, le dernier objectif de la thèse concerne la validation des solutions proposées tant au point de vue faisabilité que performance.

4. Contributions

Le travail de recherche présenté dans cette thèse a conduit aux contributions suivantes :

1. Nous avons proposé un modèle d'évaluation permettant de prendre en compte la localisation géographique des utilisateurs dans l'évaluation des requêtes. Ces requêtes sont appelées requêtes dépendantes de la localisation. Un exemple de ces requêtes peut être «quel est le restaurant le plus proche de moi ? ». Dans un premier temps, différents opérateurs permettent l'expression de ces requêtes de localisation. Ensuite, afin d'optimiser les coûts liés à l'évaluation d'une requête de localisation, différentes stratégies reposant sur plusieurs solutions d'ordonnancement sont proposées. Contrairement aux solutions d'optimisations classiques de localisation de l'information dans des systèmes fortement distribués, les critères de temps et de charge du réseau ne sont pas les seuls à prendre en compte. En effet, ces stratégies permettent également de minimiser la consommation d'énergie des terminaux nomades.
2. Afin d'évaluer une requête dépendante de la localisation, la localisation géographique du client doit être connue. Pour pallier les limites des solutions de géo-localisation existantes souvent adaptées à un environnement particulier (à l'extérieur pour le GPS), nous avons proposé une solution permettant à chaque utilisateur mobile de se localiser. Notre solution repose sur l'utilisation de métadonnées de localisation issues de l'environnement pour lui fournir une localisation approximative mais suffisante pour l'évaluation des requêtes dépendantes de la localisation. Le modèle de localisation géographique d'un utilisateur prend également en compte les ressources limitées des terminaux légers et fournit des optimisations adaptées en fonction du comportement de l'utilisateur et de son environnement.
3. Reposant sur le modèle d'évaluation des requêtes de localisation et sur notre solution de localisation géographique, nous avons proposé un service de localisation particulièrement adapté à l'environnement des applications de proximité : ISLANDS (Information and Services LocalizAtioN and Discovery Service). Ce service prend en compte la dynamicité de l'environnement et la répartition des informations disponibles. Il permet de partager et de localiser tout type d'informations au sein d'une application de proximité. Ce service repose sur un modèle d'architecture décentralisé. Une instance du service ISLANDS est donc déployée sur les différents nœuds de l'application. Ces différentes instances sont adaptées en fonction des ressources sous-jacentes du nœud. Il fournit un évaluateur de requêtes évolué capable de gérer la distribution des requêtes sur les différentes instances d'ISLANDS disponibles dans l'application de proximité. Afin d'évaluer les requêtes dépendantes de la localisation, l'évaluateur de requêtes du service ISLANDS repose sur le modèle d'évaluation des requêtes de localisation présenté ci-dessus et utilise notre solution de localisation géographique d'un utilisateur mobile.

4. Les contributions présentées dans cette thèse ont été implémentées au sein d'un prototype. Ce prototype a permis : (i) de valider le service de localisation ISLANDS au sein d'une application de proximité et donc (ii) de valider le modèle d'évaluation proposé pour les requêtes dépendantes de la localisation et (iii) de valider le modèle proposé pour la localisation géographique d'un utilisateur mobile. Ce prototype a par ailleurs été démontré lors des 19èmes Journées de Bases de Données Avancées (BDA) en 2003 [THI 03].

5. Plan de ce mémoire

La suite de ce mémoire de thèse s'articule de la manière suivante : dans le chapitre 2, nous présentons un état de l'art des services de localisation existants, ainsi que des techniques permettant la localisation géographique d'un utilisateur mobile. Finalement, nous terminons ce chapitre en présentant les différentes solutions concernant l'évaluation des requêtes dépendantes de la localisation. Le chapitre 3 présente notre approche de localisation ISLANDS proposé pour répondre au besoin des applications de proximité. Nous présentons le modèle d'architecture choisi, le modèle de données, le modèle de distribution et finalement le modèle d'évaluation. Dans le chapitre 4, nous présentons notre solution de localisation géographique d'un utilisateur mobile. Nous proposons ensuite une évaluation de cette solution et les optimisations requises afin que notre solution puisse être largement exploitée sur les terminaux nomades. Le chapitre 5 est consacré à la description du modèle d'évaluation des requêtes de localisation et aux optimisations mises en œuvre dans ce contexte. Le chapitre 6 présente, quant à lui, le prototype développé et les différentes conclusions issues de cette implémentation. Finalement, le chapitre 7 résume les travaux menés et conclut sur les perspectives des travaux présentés dans cette thèse.

Chapitre 2

Localisation & Requêtes

1. Introduction

L'augmentation du nombre d'usagers de terminaux nomades encourage le développement de nouvelles applications. Cependant, ces applications doivent répondre à de nombreuses contraintes en terme de ressources et faire face à la dynamicité de l'environnement. Dans ce contexte, nous étudions la problématique de localisation. Dans ce mémoire, le terme localisation est utilisé pour deux notions différentes. D'une part, nous nous intéressons à la localisation de l'information : comment localiser et récupérer une information donnée dans un système d'informations réparti ? D'autre part, nous utilisons également le terme localisation pour définir la localisation géographique d'un client, d'une information ou d'un service. En effet, comme nous l'avons décrit dans la section 3.2 du chapitre 1, pour permettre l'évaluation des requêtes dépendantes de la localisation, la localisation géographique de l'utilisateur ayant émis la requête est requise.

De nombreux services de localisation existent aujourd'hui et proposent une solution de localisation de l'information. Nous les présentons dans la section 2 de ce chapitre et identifions pour chaque type de services leurs avantages et leurs limites.
Cette localisation peut être, par exemple, représentée par son adresse (numéro, rue, ville, etc.) ou par des coordonnées (latitude, longitude, altitude). Nous présentons dans la section 3, les différentes catégories et solutions de localisation géographique existantes.
Enfin, la section 4 décrit les différentes solutions actuelles permettant l'évaluation des requêtes dépendantes de la localisation.

2. Services de localisation

Pour permettre la localisation de l'information dans un système réparti, différentes solutions existent. Ces solutions reposent sur l'utilisation d'un service de localisation. Un service de localisation permet à un client de localiser une information disponible dans l'environnement couvert par ce service : le client émet une requête, le service évalue cette requête et une réponse, si elle existe, est alors envoyée au client. Dans la suite, nous distinguons deux types de services de localisation : les services statiques et les services dynamiques. Les services

statiques sont actuellement les plus répandus dans les systèmes répartis existants. Ils permettent de localiser une information de manière efficace en offrant des performances intéressantes en terme de temps de réponse. Cependant au sein de ces services statiques, la gestion (enregistrement, modification, suppression) de l'information est relativement compliquée et n'est pas automatisée.
En revanche, depuis 1998, différents services, proposant une gestion dynamique de l'information, apparaissent. Ces services répondent aux nouvelles contraintes de dynamicité et de distribution de l'information apparues dans les systèmes répartis ces dernières années. Ils ont, en particulier, pour objectif de gérer les modifications des informations disponibles de façon automatique.

2.1. Services de localisation statiques

Les services de localisation statiques regroupent les services de nommage, les services de courtage et les services d'annuaires. Les services de nommage permettent de retrouver une information (un objet par exemple) en fonction de son nom (ou identifiant), nous pouvons les comparer au service de pages blanches. Le DNS (DNS – Domain Name Service) [DNS 87] est un exemple de service de nommage : il permet, en fonction du nom d'une station ou d'une adresse explicite (par exemple, une URL) d'obtenir une adresse IP. Le service de nommage de Corba (Corba Naming service) [OMG 95] : il permet, quant à lui, d'associer un nom à un objet afin de le localiser de manière transparente.
Les services de courtage permettent, quant à eux, de rechercher un service en fonction de ses propriétés. Ils fournissent un service similaire au service des pages jaunes. Un exemple de ces services de courtage est le service de courtage de Corba (Corba Trading Service) [OMG 97]. Ce service permet de rendre des services Corba accessibles par un mécanisme d'import-export : les fournisseurs de services exportent auprès de ce service les fonctionnalités qu'ils sont capables d'offrir, ces fonctionnalités permettant ensuite aux utilisateurs de sélectionner et de localiser au mieux les services capables de répondre à leurs besoins.
Finalement, les services d'annuaires proposent de rechercher une information en fonction de son nom et/ou de ses propriétés. Ils regroupent le service de pages blanches et celui des pages jaunes. LDAP (LDAP – Lightweight Directory Access Protocol) [WAH 97] est un service d'annuaires, principalement utilisé dans les entreprises pour gérer, en partie leurs systèmes d'informations. Il permet par exemple, de gérer les applications des ressources d'un réseau, l'authentification des utilisateurs, la gestion d'annuaires comme les « pages blanches » ou les carnets d'adresses.

Les services statiques ont pour principal objectif d'évaluer rapidement et efficacement les requêtes émises par les clients. Les informations disponibles sont référencées au sein du service de localisation : ces références permettent d'identifier les propriétés de l'information référencée telles que son nom et son adresse par exemple. Cependant, ces services reposent

sur la définition d'un schéma global des données or la dynamicité et les constantes évolutions des informations disponibles dans les applications de proximité ne permettent pas de définir un schéma global pour l'application. De plus, ces services de localisation sont principalement dédiés à des systèmes répartis statiques et fiables, ils ne proposent pas de mécanismes évolués pour la gestion des modifications des informations. En effet, l'enregistrement de nouvelles informations, de même que la suppression de ces références lorsque l'information n'est plus disponible, ne sont pas réalisés dynamiquement. De nombreux travaux considèrent ce problème de dynamicité et proposent des solutions dédiées aux services de localisation classiques afin de permettre une gestion de l'information plus dynamique [DEL 01]. Parallèlement à ces propositions, d'autres solutions dédiées principalement aux environnements dynamiques sont apparues, nous les présentons dans la section suivante.

2.2. Services de localisation dynamiques

Depuis quelques années, les services de localisation dynamiques se sont développés : ils permettent de répondre aux nouvelles contraintes de certains systèmes répartis actuels. En effet, avec la démocratisation de l'informatique, les informations ne sont plus nécessairement stockées sur des bases de données éventuellement réparties. En effet, ces informations sont distribuées sur de nombreux ordinateurs représentant ainsi des sources de données de structures et de tailles diverses qu'il faut faire coopérer. C'est dans ce contexte que sont apparus les services de localisation dynamiques. Ces services permettent une gestion dynamique de l'information. Ils ont pour objectif de gérer en temps réel les informations disponibles, de détecter automatiquement les changements : ils récupèrent automatiquement les nouvelles informations et gèrent leurs suppressions quand elles deviennent indisponibles. Parmi ces services de localisation dynamiques, nous distinguons deux catégories : les protocoles de découverte de services dynamiques et les services de localisation utilisés dans les réseaux pair-à-pair (P2P – Peer-To-Peer).

2.2.1. Services de découverte de services

L'augmentation du nombre d'usagers des systèmes distribués et le développement des terminaux nomades transforment ces systèmes et ajoutent de nouvelles contraintes en terme de dynamicité et de tolérance aux pannes. En effet, l'ensemble des informations et en particulier des services disponibles dans ces systèmes varie en fonction des connexions et déconnexions des différents terminaux. Les services de découverte de services ont pour objectif de permettre aux systèmes distribués d'être «plug and play ». Un appareil ou un service logiciel doit pouvoir se connecter et s'intégrer facilement au réseau. Ensuite, les clients qui souhaitent utiliser un tel service ou un tel appareil doivent pouvoir le localiser et l'utiliser. Par exemple, dans un système où un appareil photo est connecté, si une imprimante propose ses services, l'appareil photo peut alors utiliser les services proposés par l'imprimante.

Ces services de localisation reposent sur une recherche similaire à celle effectuée dans les services d'annuaires. En effet, grâce à œs services, nous sommes en mesure de rechercher un service en fonction de son nom et/ou de ses propriétés. Cependant, contrairement aux services d'annuaires classiques, les informations sont gérées dynamiquement. En effet, chaque service et/ou chaque ordinateur est enregistré dynamiquement par le service de découverte de services dès qu'il se connecte au réseau. Ce concept a été introduit depuis quelques années et aujourd'hui, plusieurs solutions ont été proposées telles que SLP [VEI 97], Jini [WAL 00], SSDP [GOL 99] et finalement Salutation [SAL 98]. Le tableau 2.1 décrit les différentes caractéristiques de ces services.

Caractéristique	SLP	Jini	Salutation	SSDP
Développé par :	IETF	Sun	Salutation Consortium	Microsoft
Réseaux	TCP/IP	Indépendant	Indépendant	TCP/IP
Langage de programmation	Indépendant	Java	Indépendant	Indépendant
Gestion centralisée	Oui (optionnelle)	Oui	Oui (optionnelle)	Oui

Tableau 2.1. Différentes caractéristiques des principaux services de découverte de services

Le protocole SLP (SLP – Service Location Protocol) est un standard proposé par l'IETF (IETF – Internet Engineering Task Force). Ce protocole est dédié à une utilisation sur des réseaux TCP/IP et a pour objectif de permettre aux applications basées sur le protocole IP de découvrir automatiquement la localisation d'un service requis. Le protocole SLP définit pour cela trois types d'agents :

- les agents « utilisateur » (UA – User Agent) permettent la découverte d'un service demandé par une application cliente. Ils représentent les clients potentiels : par exemple, si un utilisateur recherche une imprimante, un agent utilisateur est créé pour le représenter et pour effectuer la recherche d'un service « imprimante ».
- les agents de service (SA – Service Agent) permettent de fournir la localisation et les attributs d'un service. Ils représentent chaque service disponible comme un service « imprimante » par exemple.
- l'agent « annuaire » (DA – Directory Agent) collecte les informations relatives aux services présents sur le réseau.

SLP propose deux modes de fonctionnement différents :

- quand l'agent «annuaire » est présent, c'est lui qui collecte l'ensemble des informations concernant les services notifiés par les agents de service. Un agent «utilisateur » peut alors envoyer une requête à l'agent « annuaire » qui lui fournira les informations concernant le service demandé.
- quand il n'y a pas d'agent « annuaire », les agents « utilisateur » envoient leur requête en mode multicast. Les agents de service écoutent et récupèrent ces requêtes et si le service correspond, répondent directement aux agents « utilisateur ».

Jini est une technologie développée par Sun Microsystems. Son objectif est de représenter le matériel et le logiciel sous la forme d'objets Java afin de permettre à tout objet d'accéder aux services présents sur le réseau de façon flexible. Jini repose également sur un service d'annuaires similaire à l'agent « annuaire » utilisé dans SLP. Il est appelé le JLS (JLS – Jini Lookup Service). Contrairement à SLP, le service de découverte de Jini est nécessaire pour le fonctionnement de Jini : les clients doivent toujours l'utiliser pour découvrir les services et ne peuvent jamais procéder directement.

Le protocole SSDP (SSDP – Simple Service Discovery Protocol) est le protocole de découverte de services associé à l'initiative UPnP (UPnP – Universal Plug-and-Play). Cette initiative a pour objectif de permettre une connectivité simplifiée et robuste entre différents terminaux. Comme SLP, ce protocole est dédié aux réseaux TCP/IP. Il définit un protocole minimal utilisant la découverte basée sur le protocole multicast. SSDP fonctionne avec ou sans annuaire centralisé appelé SD (SD – Service Directory). Quand un service souhaite se notifier sur le réseau, il envoie tout d'abord un message d'annonce pour notifier sa présence aux autres machines. Cette annonce est :

- soit envoyée en mode multicast : dans ce cas, l'ensemble des machines présentes la reçoivent et s'il existe un annuaire, il enregistre l'annonce ;
- soit envoyée directement à l'annuaire.

Lorsqu'un client recherche un service, il peut interroger l'annuaire s'il existe, sinon il peut envoyer un message multicast.

Salutation est une architecture dont l'objectif est de proposer, découvrir et accéder aux services et à l'information disponibles indépendamment du réseau utilisé. Son objectif est de résoudre les problèmes liés à la découverte de services et à leur utilisation au sein d'environnements hétérogènes composés de types de terminaux et de réseaux différents. En effet, Salutation est indépendant de la technologie réseau utilisée et peut fonctionner sur de multiples infrastructures allant de TCP/IP à l'IrDA. A l'inverse de SSDP, il n'est pas limité au protocole HTTP. De plus, par rapport à Jini, Salutation présente l'intérêt de ne pas être limité à un seul langage. L'architecture Salutation est basée sur un gestionnaire appelé SLM (SLM – Salutation Manager) qui fonctionne comme un annuaire répertoriant les applications, les services et les machines, génériquement appelés entités du réseau. Le gestionnaire de Salutation permet à ces entités de découvrir et d'utiliser les autres entités.

Les solutions présentées ci-dessus sont les plus répandues actuellement. Cependant, de part leur centralisation (i.e. l'agent « annuaire » DA pour SLP, le service de découverte JLS pour Jini, l'annuaire SD pour SSDP, le gestionnaire SLM pour Salutation), ces systèmes requièrent tout de même une certaine fiabilité. En effet, l'environnement doit garantir la présence de machine suffisamment fiable et robuste permettant la centralisation. Quant aux solutions qui proposent une alternative décentralisée (i.e. SLP et SSDP), elles reposent sur l'utilisation intensive des protocoles réseau tels que le multicast qui nécessitent de disposer

de ressources relativement robustes et importantes afin de ne pas inonder ni le réseau, ni les terminaux [CAM 02]. Un protocole a été proposé par Campo [CAM 02] pour solutionner ce type de problèmes : le protocole PDP (PDP – Pervasive Discovery Protocol). Ce protocole a pour objectif de répondre aux contraintes d'un réseau ad hoc basé sur l'utilisation de terminaux nomades et de réseaux sans fil. Il est basé sur l'utilisation d'agents pour gérer la recherche de services dans le réseau. Chaque demande de service est effectuée par l'agent qui après avoir vérifié qu'il n'a pas l'information sur le service demandé dans son cache, diffuse une requête de service. Cette solution est adaptée aux ressources limitées du réseau et des terminaux, cependant, le protocole spécifié ne permet pas de gérer les indisponibilités accidentelles de services. En effet, dans un environnement ad hoc, les terminaux peuvent, à tout moment, être accidentellement déconnectés. Or, le protocole utilise une spécification de service qui définit une durée de vie pour chaque service. Cette durée de vie semble difficile à paramétrer et ne peut pas, de plus, considérer les déconnexions accidentelles.
D'autres solutions adaptées aux terminaux nomades ont également été proposées telles que Salutation Lite [PAS 99] et SDP (SDP – Service Discovery Protocol) de Bluetooth [SDP 99]. Salutation Lite est une version de Salutation adaptée aux terminaux contraints en terme de ressources, elle utilise donc le même mécanisme que Salutation. SDP est quant à lui, un protocole permettant de faciliter l'échange de services entre différents terminaux équipés de Bluetooth.
Ces services de découverte proposent finalement des langages d'interrogation assez restreints. En effet, la description des services est simple et permet à un client de définir si un service peut l'intéresser ou pas mais ces protocoles n'ont pas pour objectif de fournir des capacités d'interrogation étendues permettant des recherches d'informations plus précises. Or, dans les applications de proximité, la localisation de l'information nécessite des langages de requêtes relativement complexes et pas seulement un langage de découverte d'informations (ou *lookup* en anglais).

2.2.2. Localisation d'informations dans les réseaux P2P

Parallèlement aux services de découverte de services, les systèmes P2P se sont développés. Contrairement à l'environnement d'exploitation des services de découverte de services, les services P2P n'ont pas pour objectif l'intégration de terminaux nomades au sein de systèmes répartis : ils sont surtout dédiés à l'Internet et leur objectif est de permettre la fédération de toutes les ressources disponibles. Dans un réseau P2P, chaque machine est à la fois serveur et client. Chaque machine peut dont partager des ressources (i.e. être serveur) tout en recherchant une ressource particulière (i.e. être cliente). De nombreux systèmes P2P sont apparus ces dernières années et sont largement utilisés aujourd'hui : ils regroupent les services de calculs distribués tels que SETI@home [SETI], des services de messagerie instantanée tels que Jabber [Jabber] ou ICQ [ICQ] et finalement les services de stockage et de partage de fichiers tels que Napster [Napster], Kazaa [Kazaa], Gnutella [Gnutella] ou Freenet [Freenet]. Ces systèmes P2P peuvent être assimilés à des services de localisation étant donné qu'ils permettent aux utilisateurs de partager des ressources et de les localiser.

Le fonctionnement de ces systèmes s'appuie sur différentes techniques de recherche que nous pouvons classer en quatre catégories :

- les techniques de recherche qui utilisent un répertoire centralisé pour le référencement des informations disponibles.
- les techniques de recherche qui utilisent la diffusion de la requête par inondation sur le réseau (ou Query flooding en anglais).
- les techniques de recherche qui s'appuient sur groupes de nœuds dominés par un super nœud.
- Les techniques de recherche qui utilisent un système de table de hachage distribuée.

2.2.2.1. Utilisation d'un répertoire centralisé

Les premiers systèmes P2P apparus sur Internet reposaient sur l'utilisation d'un répertoire centralisé. Ces systèmes requièrent la présence d'un ensemble de machines capables de remplir le rôle de serveur. Ce serveur référence l'ensemble des informations disponibles. Lorsqu'un nœud client recherche une information, il interroge un des serveurs et grâce à la réponse du serveur, contacte ensuite directement le nœud où l'information recherchée est stockée. Ces systèmes permettent de retrouver la réponse exacte pour chaque recherche mais ils comportent des inconvénients dus à la centralisation. En effet, la centralisation du processus de recherche peut engendrer un goulot d'étranglement au niveau des serveurs lorsque de trop nombreuses requêtes sont émises. La vulnérabilité est également un des inconvénients de cette méthode de recherche : lorsqu'un serveur devient inaccessible (en cas de panne par exemple), les requêtes ne peuvent plus être évaluées. Finalement, la gestion du passage à l'échelle est difficile dans un tel système : si le nombre de clients potentiels dépasse un certain point, il devient nécessaire d'ajouter un serveur ce qui entraîne une maintenance coûteuse.

Le plus répandu des systèmes P2P basés sur l'utilisation d'un répertoire centralisé est Napster : c'est un système de partage de fichiers musicaux sur Internet. Dans Napster, l'indexation de l'information est centralisée sur quelques serveurs mais l'échange des fichiers est distribué.

2.2.2.2. Utilisation de la technique d'inondation

La technique d'inondation repose sur la diffusion de la requête sur les nœuds du réseau. Cette diffusion utilise un parcours en largeur d'abord (BFS – breadth-first traversal) comme dans le système Gnutella ou un parcours en profondeur d'abord (DFS – depth-first traversal) comme dans le système Freenet. A l'inverse des systèmes centralisés telles que Napster, ces deux méthodes sont entièrement décentralisées.

Si la technique d'inondation repose sur un parcours en largeur d'abord, une limite de profondeur p est fixée, où p représente le temps maximum TTL (TTL – Time-to-live) du message, mesuré en nombre de sauts. Le nœud source (ou nœud client) envoie sa requête à l'ensemble de ses voisins directs. Chaque nœud voisin évalue la requête et renvoie le résultat

au nœud source. Ensuite, de la même façon, il envoie la requête à l'ensemble de ses voisins (profondeur = 2) et ainsi de suite, jusqu'à ce que la profondeur de recherche ait atteint la profondeur p (i.e. le TTL maximal). Le message est alors supprimé. Un exemple de systèmes utilisant cette technique est le système Gnutella.
Si la technique d'inondation repose sur un parcours en profondeur d'abord, une limite p pour la profondeur est également fixée. Chaque nœud envoie la requête vers un seul de ses voisins et attend une réponse de ce voisin avant de la renvoyer à un autre de ses voisins si aucune réponse n'a pu être obtenue, sinon, il renvoie la réponse vers le nœud source. Le système Freenet est un exemple de systèmes utilisant l'inondation sur un parcours en profondeur d'abord.
En terme de nombre de réponses et/ou de qualité de la réponse, la technique utilisée dans Gnutella est idéale puisqu'elle envoie la requête à un maximum de nœuds possibles. Cependant en terme de consommation de bande passante et de puissance de processeur, elle est moins efficace étant donné que la requête va être traitée par un grand nombre de nœuds alors qu'elle ne pourra être satisfaite, a priori, que par un petit nombre d'entre eux. La deuxième technique utilisée dans Freenet permet de minimiser la consommation des ressources mais au détriment du temps de réponse qui peut être important. Par rapport à un système centralisé tel que Napster, ces deux solutions décentralisées ne requièrent pas de machines dédiées à la centralisation. De plus, cette décentralisation permet également d'offrir un système robuste et tolérant aux pannes. Cependant, le coût de ces recherches par inondation est très élevé en terme de nombre de messages. De plus, les réponses obtenues sont des réponses partielles étant donné que tous les nœuds ne sont pas, a priori, visités. Pour pallier ces différents problèmes, certaines évolutions ont été proposées au sein des systèmes P2P reposant sur des techniques d'inondation : par exemple, le système Gnutella intègre aujourd'hui l'utilisation de super-nœuds afin de réduire les coûts associés à la recherche d'informations.

2.2.2.3. Utilisation de super-nœuds

L'utilisation de super-nœuds a pour objectif de minimiser le nombre de nœuds visités pour évaluer la requête afin de limiter le temps de recherche. Un super-nœud [YAN 03] est un nœud qui joue le rôle de serveur centralisé pour un sous-ensemble de nœuds appelés « fils ». Dans la figure 2.1, un exemple de réseau de super-nœuds est illustré : les super-nœuds sont représentés par les ronds noirs, tandis que les nœuds fils sont représentés par les ronds blancs. Les super-nœuds sont tous connectés entre eux.
Lorsqu'un nœud client émet une requête, celle-ci est envoyée au super-nœud du nœud client. Le super-nœud évalue la requête localement : chaque super-nœud stocke un index représentant les informations partagées par ses nœuds « fils ». Cet index doit représenter suffisamment d'informations pour permettre de répondre aux requêtes. Si le super-nœud trouve une réponse, elle est envoyée au nœud client. Cette réponse contient l'adresse des nœuds qui répondent à la requête du client. Si le super-nœud ne trouve pas de réponse, la

requête est alors envoyée aux autres super-nœuds qui l'évaluent à leur tour et envoient leurs réponses au super-nœud du nœud client. C'est toujours le super-nœud du nœud client qui envoie la réponse au client.

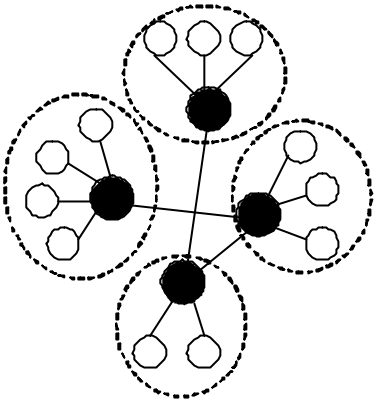


Figure 2.1. Illustration d'un réseau de super-nœuds

Ces systèmes présentent les avantages combinés des systèmes basés sur une centralisation comme Napster et ceux utilisant les techniques d'inondation comme Gnutella et Freenet. En effet, ils offrent l'efficacité des systèmes de recherche centralisés et l'autonomie, la répartition des charges et la robustesse des systèmes de recherche entièrement décentralisés. Cependant ces systèmes posent différents problèmes tels que la définition de la politique de choix des super-nœuds, le choix du nombre de fils par super-nœuds, la fiabilité du système en cas de panne d'un super-nœud et les mécanismes de gestion des index.
Deux exemples de systèmes P2P basés sur l'utilisation de groupes de nœuds sont Kazaa et Edutella [NEJ 02]. Dans Kazaa, les super-nœuds sont simplement sélectionnés en fonction de leurs ressources (connexions efficaces, ordinateurs puissants). Dans Edutella, la gestion des super-nœuds et du routage de la requête parmi ces super-nœuds est basée sur la sémantique. Chaque nœud est classé dans un groupe de nœuds en fonction de la correspondance de ses informations avec un concept donné de la taxonomie définie. La requête est elle aussi, classée en fonction des concepts de cette même taxonomie. Ensuite le routage de la requête s'effectue en fonction de la correspondance entre la description de la requête et la description du groupe de nœuds.

2.2.2.4. Utilisation des tables de hachage distribuées

Pour optimiser les performances des recherches effectuées dans les systèmes P2P, certaines solutions utilisent les tables de hachage distribuées. L'objectif est alors de minimiser le nombre de messages transmis en cherchant à localiser efficacement le ou les destinataires de la requête.
Ces tables de hachages distribuées permettent d'identifier et de localiser une information en fonction de sa clé. Dans ces systèmes, tout est identifié dans un seul espace de noms : les nœuds et les informations ont chacun un identifiant unique. Cet identifiant est utilisé lors du routage de la requête à destination d'un nœud. Chaque nœud du réseau est ensuite

responsable d'un sous-ensemble d'identifiants. Le routage des requêtes à destination d'un identifiant s'effectue de proche en proche : un nœud qui reçoit une requête la retourne aux N nœuds qu'il connaît et qui lui sont proches dans l'espace de nom (N étant défini dans la politique du système).

Les tables de hachages réparties sont notamment utilisées dans les protocoles Chord [STO 01], CAN [RAT 01] et Tapestry [ZHA 01]. Le protocole Chord utilise par exemple une représentation en anneau. Les clés et les nœuds ont des identifiants uniques sur m bits. La clé K est stockée sur le nœud dont l'identifiant correspond au plus petit ID(nœud) = ID(clé). La figure 2.2 décrit un exemple de l'espace circulaire utilisé dans Chord. Dans la figure 2.3, si la clé 50 est recherchée, les nœuds sont contactés de proche en proche pour retrouver le nœud dont l'identifiant est 60.

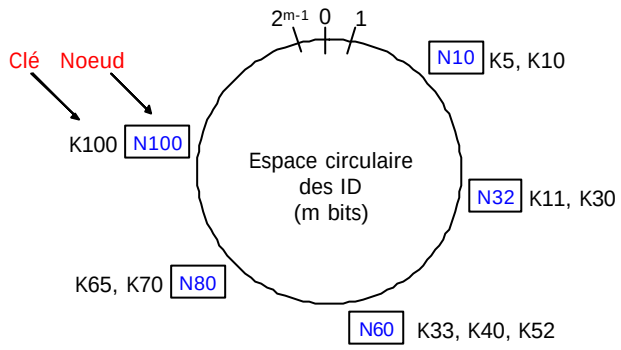


Figure 2.2. Représentation de la topologie en anneau utilisée dans Chord

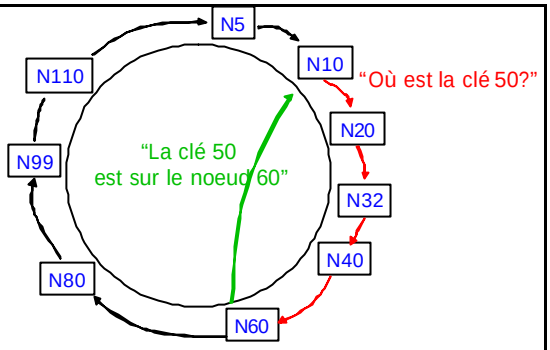


Figure 2.3. Recherche de la localisation d'une clé

2.3. Synthèse

Nous avons présenté différents services de localisation en soulignant leurs avantages et leurs limites. Ces services peuvent être séparés en deux catégories : les services de localisation statiques et les services de localisation dynamiques.

Dans le contexte des applications de proximité, les services de localisation statiques ne sont pas adaptés. En effet, leur nature statique implique une gestion des informations manuelle : c'est-à-dire que lorsqu'une modification de l'information intervient, le service doit être prévenu pour prendre en compte la modification. Ces services sont dédiés à des environnements fiables, peu sujets aux pannes et aux reconfigurations et ne proposent donc pas de solutions pour gérer les modifications des informations à la volée sans intervention humaine. Dans une application de proximité, les terminaux nomades peuvent à tout moment être déconnectés de façon accidentelle et dans cette situation, prévenir le service de localisation est impossible.

De plus, ces services sont généralement centralisés, les informations sont stockées sur un seul serveur. Dans le contexte des applications de proximité, il est impossible de garantir la présence d'un unique serveur pour chaque sphère de communication capable de gérer l'ensemble des informations présentes dans cette sphère. Certains de ces services offrent la possibilité d'envisager une distribution de l'information sur quelques serveurs : c'est par

exemple le cas de la fédération de plusieurs services de nommage Corba ou de différents services LDAP. Dans le cas d'une fédération de services de localisation, les différents serveurs sont liés entre eux par des références. Lorsqu'un utilisateur interroge un serveur qui n'a pas l'information demandée, ce serveur lui fournit la référence d'un serveur qui pourra lui envoyer une réponse. C'est alors au client de renvoyer sa requête. La distribution de la requête entre les différents serveurs n'est donc pas transparente pour le client : c'est au client de ré-envoyer sa requête. Ce manque de transparence concernant la distribution peut être très pénalisant dans un environnement contraint : puisqu'un client muni d'un terminal nomade ne dispose pas toujours d'un temps de connexion suffisamment sûrs et longs pour permettre d'attendre la réponse des différents serveurs et de gérer l'interrogation sur d'autres serveurs.
Contrairement aux services de localisation statiques, les services de localisation dynamiques sont eux dédiés aux nouvelles contraintes de dynamicité apparues dans les systèmes répartis. De ce fait, ils sont plus adaptés à l'environnement des applications de proximité. Cependant, de nombreuses limites subsistent dans les solutions actuelles.
Concernant les services de découverte de services, ils ont pour objectif de permettre une connectivité facilitée entre les différents terminaux d'un réseau. Cependant, la majorité de ces services reposent sur une centralisation du mécanisme de découverte. De la même façon que pour les services de localisation statiques, une centralisation est difficilement envisageable dans les applications de proximité. Certains services tels que SLP et SSDP proposent une alternative décentralisée. Mais, comme nous l'avons décrit dans la section 2.2.1, ces solutions inondent le réseau d'annonces de services et ne peuvent donc pas être utilisées par des terminaux nomades au sein de réseaux sans fil.

Concernant les systèmes de recherche P2P, les systèmes utilisant un répertoire central ne sont pas adaptés au contexte des applications de proximité. En effet, dans ces applications, il est impossible de garantir la présence de machines serveurs permettant une centralisation. De plus, dans un environnement peu fiable, maintenir la cohérence des tables d'indexage va s'avérer être un problème difficile à résoudre. En effet, nous parlons, dans ce cas d'information transitoire (ou *transient* en anglais) qui apparaît à un moment donné et reste disponible pendant un laps de temps plus court que le temps nécessaire à son indexation par les techniques actuelles.
Les solutions de recherche P2P reposant sur les techniques d'inondation présentent plusieurs intérêts dans le contexte des applications de proximité : elles sont extensibles et permettent des entrées/sorties facilitées pour les utilisateurs. Cependant, ces solutions ont été proposées pour l'Internet et sont donc dédiées à des environnements filaires et relativement fiables. Ces environnements permettent à un client d'attendre la réponse à sa requête pendant un temps relativement long. Ceci est impossible dans le cadre des applications de proximité où les terminaux nomades sont limités en autonomie et où le temps de réponse pour l'évaluation de la requête doit donc être aussi court que possible.
Les solutions utilisant les super-nœuds sont intéressantes parce qu'elles permettent de minimiser le temps nécessaire pour l'évaluation des requêtes. De plus, dans les applications

de proximité, où il existe parfois des machines plus puissantes et fixes, le choix de ces super-nœuds serait facilité. Cependant, l'indexation de l'information des nœuds fils sur les super-nœuds reste compliquée car l'ensemble des nœuds fils d'un super-nœud risque d'évoluer trop rapidement pour permettre cette indexation.
Finalement, les solutions utilisant des tables de hachage distribuées présentent le grand intérêt de réduire considérablement les coûts liés à l'évaluation de la requête. Cependant, dans le contexte des applications de proximité, ces techniques ne peuvent pas être mises en place car elle requiert un réseau de nœuds relativement fiable et gère difficilement la dynamicité. De plus, ces systèmes peuvent être utilisés lorsque les informations à échanger sont de même type : par exemple, des fichiers audio. En effet, la recherche ne peut reposer que sur l'utilisation d'un mot clé (le titre par exemple) pour retrouver un résultat.

Les services de localisation actuels ne permettent donc pas de répondre à l'ensemble des contraintes inhérentes à l'environnement des applications de proximité. Nous proposons donc, un service de localisation adapté à l'environnement des applications de proximité. Il permet en particulier la gestion des requêtes dépendantes de la localisation qui n'est pas possible dans les services de localisation présentés ci-dessus. Pour permettre l'évaluation de ce type de requêtes, le service de localisation doit disposer de la localisation géographique du client.

3. Techniques de localisation géographique actuelles

3.1. Introduction

Avec l'apparition des terminaux nomades, considérer la localisation géographique des utilisateurs dans les applications est devenu une problématique majeure. En effet, connaître la localisation géographique d'un usager mobile permet de personnaliser les applications : dans le contexte des applications de proximité, la localisation géographique des usagers permet, en particulier, de répondre aux requêtes dépendantes de la localisation. Actuellement de nombreuses solutions permettant de localiser géographiquement un objet ou une personne existent. Ces solutions sont souvent dédiées à des applications définies et/ou pour des environnements spécifiques. De ce fait, seules quelques unes d'entre elles sont, aujourd'hui, réellement exploitées.
Hightower & Borriello [HIG 01b] proposent une taxonomie permettant de décrire les différentes caractéristiques des solutions de localisation actuelles. Parmi les propriétés de cette taxonomie, le modèle utilisé pour la description de la localisation permet de classifier les solutions de localisation. En effet, il existe deux modèles de description d'une localisation géographique : le modèle physique (ou géométrique) et le modèle symbolique. Ces deux modèles ne fournissent pas le même type d'informations et chacun requiert des traitements différents.

- Le modèle physique permet de décrire une localisation en utilisant ses coordonnées en *n* dimensions, par exemple, un triplet de coordonnées (longitude, latitude, altitude). Ce modèle de représentation permet de fournir une compatibilité nécessaire pour le traitement de localisations issues d'environnements hétérogènes. En effet, il fournit une représentation uniforme quel que soit l'endroit où l'utilisateur se situe. Cet avantage explique que ce modèle de représentation ait été choisi dans les solutions de localisation les plus développées comme le GPS (GPS – Global Positioning System). En revanche, ce modèle nécessite, la plupart du temps, la mise en place d'une couche sémantique pour permettre l'exploitation des informations de localisation à un niveau applicatif. En effet, si un utilisateur recherche la localisation d'un ami, il ne souhaite pas obtenir les coordonnées physiques de son ami mais plutôt une description de l'endroit où il se trouve tel que le nom de la rue. La gestion de cette couche sémantique peut être coûteuse en terme de stockage sur des terminaux nomades par exemple.
- Le modèle symbolique permet de répondre au problème de sémantique du modèle physique. En effet, ce modèle décrit une localisation en terme d'entités logiques du monde réel. Ces entités sont, par exemple, des bâtiments, des rues, des villes, etc. Au niveau applicatif, la gestion de représentations symboliques est facilitée grâce à la sémantique intrinsèque de ces représentations. Cependant, l'utilisation de ce modèle est limitée lorsque les environnements de l'application sont hétérogènes. En effet, si les environnements sont différents, les représentations peuvent être très différentes et dans ce cas, la gestion devient beaucoup plus complexe, voire impossible. Par exemple, au sein d'une application, deux localisations doivent être comparées : si la première localisation est décrite grâce au triplet (n°, rue, ville) et la deuxième localisation grâce au triplet (bâtiment, étage, n°), la comparaison de ces localisations sera impossible à effectuer.

La taxonomie propose également de classer les solutions de localisation en fonction de la méthode utilisée pour effectuer la localisation. D'après [HIG 01a], trois principales techniques permettent de localiser un utilisateur : la triangulation, l'analyse de l'environnement ou la proximité. Il existe généralement de nombreuses solutions basées sur chaque méthode; dans la section suivante, nous avons choisi, pour chaque méthode, de décrire quelques solutions représentatives et/ou assez répandues.

3.2. Triangulation

Les techniques de localisation basées sur la triangulation utilisent les propriétés géométriques du triangle pour calculer la localisation d'un objet physique : trois points fixes sont utilisés pour permettre la localisation. Deux sous-catégories de triangulation existent : la *latération* utilisant des mesures de distances et l'*angulation* utilisant des mesures d'angles.

3.2.1. La latération

La *latération* permet de calculer la localisation d'un objet physique en mesurant les différentes distances qui le séparent de différentes positions de référence. Ainsi, pour calculer la position d'un objet en 2 dimensions, 3 points (ou positions) de référence non colinéaires sont nécessaires (Fig. 2.4). En 3 dimensions, 4 points non colinéaires sont requis. Certaines connaissances concernant la configuration de l'environnement peuvent permettre de réduire le nombre de points de référence requis. Par exemple, dans le système de localisation Active Bat [HAR 99] seules 3 mesures de distance sont nécessaires pour une représentation en 3 dimensions. En effet, ce système est basé sur l'utilisation de grilles de capteurs ultrasons (points de référence) placées dans les plafonds, les objets à localiser seront donc toujours en dessous des capteurs ultrasons qu'ils réussissent à capter.

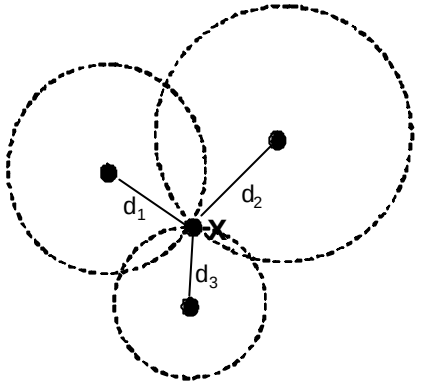


Figure 2.4. Méthode de latération pour déterminer la localisation 2D de X grâce à 3 points non colinéaires

Trois principales approches sont utilisées pour mesurer les distances requises pour la latération :

- La **mesure directe** utilise une action physique ou un mouvement pour calculer la distance séparant l'objet à localiser d'un point de référence: par exemple, un robot peut dérouler une sonde jusqu'à ce qu'elle touche un point de référence. C'est une approche facile à comprendre mais qui reste très difficile à mettre en œuvre.
- Le **temps de vol** : mesurer la distance séparant un objet physique d'un point P en utilisant la méthode du temps de vol équivaut à mesurer le temps pris pour se déplacer entre l'objet et un point de référence à une vitesse connue. On utilise souvent la différence de temps entre l'envoi et la réception d'un signal émis. Lorsque cette technique est utilisée, les deux difficultés majeures à résoudre sont les contraintes de gestion de la synchronisation temporelle et de propagation des signaux transmis : réflexion (fig. 2.5) et réfraction (fig. 2.6) des signaux.

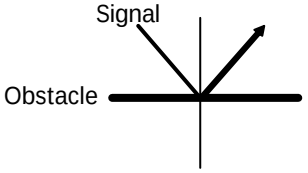


Figure 2.5. Phénomène de réflexion d'un signal

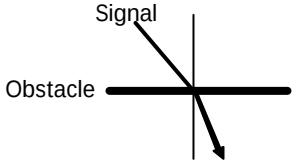


Figure 2.6. Phénomène de réfraction d'un signal

Cette méthode est utilisée par le GPS [GPS 97, DAN 00]. Grâce à sa couverture mondiale et à son développement commercial, le GPS est actuellement le système de positionnement le plus répandu. Cette technologie est basée sur une infrastructure de 24 satellites qui permet au système d'être utilisé partout dans le monde et la localisation obtenue est assez précise (de l'ordre de quelques mètres). Cependant, ce système présente quelques limites dans des environnements où il existe une concentration de bâtiment hauts (réfraction des signaux), par exemple en centre-ville ou à l'intérieur d'un bâtiment car les signaux émis sont alors, généralement bloqués par les murs et les plafonds. Le projet européen Galiléo [GAL 04] va également fournir une solution de localisation basée sur un ensemble de satellites mais cette solution ne sera opérationnelle qu'à partir de 2008. Cependant, à l'inverse du GPS, GALILEO n'a aucune vocation militaire et devrait être reçu sans aléas dans les villes et dans les régions situées à des latitudes extrêmes. D'autres systèmes utilisent cette technique de temps de vol, parmi elles, citons la solution proposée par Sinha et Das [SIN 00] qui permet d'obtenir une localisation précise en extérieur mais qui devient beaucoup moins exacte en cas de réflexions des signaux. Ce système est basé sur l'utilisation des stations de base des réseaux cellulaires de téléphonie mobile pour localiser un usager en cas d'urgence. Une autre solution, Active Bat [HAR 99], est basée sur des transmissions ultrasons et sur l'utilisation de badges par chaque employé pour permettre la localisation du personnel d'une entreprise: elle fournit une localisation très précise mais demande un investissement financier initial important. La solution Cricket [PRI 00] est basée sur le même mode de fonctionnement que la solution précédente Active Bat, elle est cependant moins coûteuse mais fournit également des localisations moins précises. La technologie PulsON [TDC 01] repose sur des transmissions radio utilisant une bande de fréquences dédiée et fournit, elle aussi, une localisation rapide et précise des utilisateurs à l'intérieur d'un bâtiment. Et citons finalement, les solutions AeroScout de Bluesoft [BLU 01] et Ekahau [EKA 04] qui utilisent une infrastructure de points d'accès Wi-Fi pour localiser des terminaux nomades à l'extérieur comme à l'intérieur des bâtiments.

- L'**atténuation** : l'intensité d'un signal émis diminue au fur et à mesure que la distance de la source augmente. La diminution relative de l'intensité originale représente l'atténuation. En utilisant une fonction, la distance parcourue peut être obtenue grâce à l'atténuation. Le système de localisation ad hoc SpotON [HIG 00] est basée sur l'utilisation d'étiquettes électroniques peu coûteuses dont la localisation est connue et sur l'estimation de la distance séparant ces étiquettes en utilisant l'indicateur de puissance du signal reçu (RSSI: Radio Signal Strength Information). Les étiquettes électroniques (ou RFID – Radio Frequency Identification) se présentent sous la forme de minuscules balises métalliques, qui réagissent aux ondes radio et transmettent ainsi des informations à distance. Dans le système SpotON, plus le nombre d'étiquettes accessibles par l'objet à localiser est important, plus la localisation sera exacte. Une autre solution proposée par Wang et al. [WAN 03] est basée sur l'utilisation d'une infrastructure de points d'accès Wi-Fi et permet une localisation précise à l'intérieur d'un bâtiment.

3.2.2. L'angulation

L'*angulation* appelée aussi méthode d'AOA (Angle Of Arrival) est une technique similaire à la latération mais des mesures d'angles sont utilisées à la place des mesures de distances: en général, pour une localisation à deux dimensions, deux mesures d'angles entre l'objet à localiser et deux points de référence sont requises ainsi que la distance séparant les deux points de référence (Fig. 2.7). Le système de navigation aérien VOR (VHF Omnidirectional Ranging) est un exemple de solution utilisant l'angulation. Le système proposé par Niculescu et Nath [NIC 03] utilise également l'angulation pour localiser des utilisateurs mobiles dans des réseaux ad hoc en utilisant leurs voisins directs.
L'angulation ne nécessite que deux points fixes pour permettre la localisation d'un utilisateur. Cependant, comme la mesure d'angles est plus difficile à gérer que la mesure de distances, les techniques de localisation utilisant l'angulation sont moins courantes.

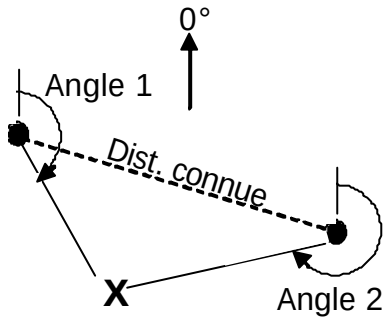


Figure 2.7. Méthode d'angulation pour déterminer la localisation 2D de X

3.3. Analyse de l'environnement

Pour obtenir la localisation d'un utilisateur mobile, une analyse de l'environnement peut également être utilisée. Cette technique est basée sur l'étude de l'environnement de l'utilisateur pour déterminer la localisation de l'utilisateur. Cet environnement est décomposé en différents points de vue appelés scènes. Généralement, les scènes observées sont simplifiées pour obtenir des caractéristiques ou des détails qui facilitent la représentation et la comparaison de la scène. Par exemple, si un utilisateur est situé dans un parc, les différentes scènes du parc ont été au préalable schématisées en fonction des éléments tels que des bancs, des arbres etc. placés dans le parc. Le système va comparer la scène observée par l'utilisateur avec les scènes stockées préalablement pour permettre une localisation de l'utilisateur.
Cette analyse peut être statique ou différentielle : dans le cas statique, la scène est comparée à un ensemble de données prédéfinies qui permettent d'obtenir la localisation de la scène;

dans le cas différentiel, les différences entre les scènes successives sont évaluées pour déduire la localisation de l'utilisateur en estimant ses déplacements.
L'avantage de cette méthode comparée à la triangulation est qu'elle ne nécessite pas d'émissions de signaux qui peuvent à la fois compromettre la confidentialité de la localisation des utilisateurs et nécessiter de l'énergie pour l'émission des signaux. Néanmoins, l'utilisation de cette méthode requiert un accès à des ensembles de données décrivant l'environnement. Et tout changement d'environnement nécessite une mise à jour de l'accès à ces ensembles de données ou une mise à jour des données elles-mêmes. Ces données peuvent correspondre à un ensemble d'éléments physiques basés, par exemple, sur des photos mais elles peuvent aussi correspondre à des caractéristiques électromagnétiques auxquelles un objet peut être soumis en fonction de sa position et de son orientation. Le système RADAR de Microsoft [BAH 00] est basé sur ce type de données : il utilise un ensemble de mesures d'intensité du signal des transmissions radio d'un réseau sans fil 802.11 observées dans différentes positions et orientations à l'intérieur d'un bâtiment. Ce système a pour objectif de permettre la localisation et la surveillance des déplacements des utilisateurs à l'intérieur d'un bâtiment.

3.4. Proximité

Cette technique détermine la localisation d'un objet si cet objet se situe à proximité d'une localisation connue. Trois approches principales existent pour déterminer la proximité :

- La détection d'un contact physique en utilisant, par exemple, des capteurs de pression ou des détecteurs de présence. Touch Mouse [HIN 99] et Contact [PAR 00] sont deux exemples de solutions utilisant cette approche.
- La gestion des points d'accès cellulaires dans les réseaux sans fil : la solution Active Badge [WAN 92] et le système Xerox ParcTAB [WAN 97] utilisent, par exemple, des cellules infrarouges dans des environnements de taille restreinte tels qu'un bureau par exemple. La solution proposée par Hills [HIL 99] est, quant à elle, basée sur les cellules d'un réseau local sans fil 802.11 dans un campus. Certaines études [ZHA 02, ILD 02] proposent d'utiliser les points d'accès cellulaires pour diffuser des informations géographiques vers les utilisateurs de terminaux nomades afin qu'ils puissent évaluer leurs positions respectives. Cependant, avec de telles solutions, l'utilisateur mobile peut difficilement mesurer la précision et l'exactitude des informations diffusées. Pour augmenter la précision de l'information diffusée vers les utilisateurs, une solution est d'augmenter le nombre de serveurs de diffusion. Mais, dans ce cas, comme l'information provient de plusieurs serveurs, il devient alors très difficile pour l'utilisateur qui reçoit les informations de plusieurs serveurs de choisir l'information à utiliser. Nous pouvons également citer la technologie Cell ID qui permet de localiser un usager de téléphone mobile connecté aux réseaux GSM en identifiant la cellule dans laquelle il se trouve.
  Certaines solutions sont dérivées de cette technique mais les points d'accès sont en fait des voisins localisés. Par exemple, dans [ERM 03, BUL 00, DOH 01], la position du

client est basée sur le calcul du barycentre de la position de ses voisins. Ces voisins peuvent être mobiles mais ils connaissent leurs localisations respectives grâce à l'utilisation du GPS. Cette méthode est intéressante mais ne répond pas aux problèmes de positionnement inhérents à l'utilisation du GPS : en effet, lorsqu'un utilisateur va se trouver dans un bâtiment, la majorité de ses voisins risque également de se trouver dans le bâtiment et donc, de ne pas bénéficier d'une localisation GPS, rendant donc impossible la localisation du client.

- L'observation des systèmes d'identification : cette méthode utilise les systèmes automatiques d'identification comme les terminaux de retrait d'argent par exemple, qui permettent de localiser un utilisateur au moment où la transaction est effectuée.

Ces trois techniques (triangulation, étude de l'environnement, proximité) sont souvent employées séparément mais sont parfois combinées: par exemple dans le cas de la solution Cricket, la principale méthode utilisée est la latération mais des méthodes de proximité sont également utilisées lorsque le nombre de points de référence n'est pas suffisant pour permettre la triangulation. De même, le GPS assisté ou A-GPS est une évolution du GPS qui permet de fournir une localisation d'un utilisateur mobile là où le GPS reste limité. Il est donc à la fois basé sur la technique de triangulation par satellites mais aussi sur la proximité en utilisant les stations de base auxquelles il est connecté.

3.5. Conclusion

Comme nous venons de le décrire, de nombreuses solutions existent déjà pour localiser un utilisateur mobile. Cependant toutes ces solutions sont souvent adaptées à un environnement bien défini et limitées dans d'autres contextes : par exemple, le GPS est rarement utilisable à l'intérieur d'un bâtiment, la solution Active Badge est, quant à elle, dédiée à de petits environnement clos de la taille d'un bureau, etc. Elles fournissent d'ailleurs des précisions très hétérogènes en fonction de l'application à laquelle elles sont dédiées.

Dans une application de proximité, l'évaluation des requêtes dépendantes de la localisation ne requiert pas une localisation de l'utilisateur extrêmement précise. En revanche, la solution de localisation utilisée doit toujours fournir une localisation de l'utilisateur quel que soit l'environnement dans lequel il se situe : à l'intérieur d'un bâtiment comme à l'extérieur. C'est pourquoi, les solutions dédiées à des environnements extérieurs (GPS) ne pourront pas être toujours exploitées. De même, les solutions dédiées à des environnements uniquement fermés comme le système Cricket par exemple, ne pourront pas non plus être toujours utilisées.

De plus, dans le contexte des applications de proximité, l'existence d'une infrastructure telle qu'un réseau d'étiquettes électroniques ou des grilles de capteurs n'est absolument pas garantie. De nombreuses solutions de localisation ne sont donc pas envisageables dans les applications de proximité. Par exemple, la solution Active Bat qui nécessite d'installer des grilles de capteurs dans les plafonds d'un bâtiment ne pourra pas être utilisée.

Enfin, les solutions sans infrastructure sont peu onéreuses mais demandent l'utilisation d'une technologie qui peut freiner son développement et donc son efficacité. Citons, par exemple, la solution SpotON qui est basée sur l'atténuation du signal entre étiquettes électroniques; pour obtenir de bons résultats, une étiquette électronique est nécessaire sur l'ensemble des terminaux nomades. En ce qui concerne, les solutions basées sur le calcul de barycentre [ERM 03, BUL 00, DOH 01], elles sont intéressantes dans le sens où elles ne requièrent aucun dispositif, cependant ces solutions ne sont basées que sur la représentation physique des localisations (coordonnées) et dépendent du bon fonctionnement du GPS.
La localisation géographique obtenue grâce à ces différentes techniques de localisation permet ensuite l'évaluation des requêtes dépendantes de la localisation.

4. Requêtes de localisation

Si le service de localisation est capable d'avoir accès aux localisations géographiques des utilisateurs et des informations, il devient alors très intéressant de faire intervenir la localisation géographique dans la localisation des informations. Dans une application de proximité, un utilisateur doit pouvoir localiser une information en fonction d'une localisation de référence. Par exemple, un participant peut émettre les requêtes : « quel est l'arrêt de bus le plus proche de moi ? » ou « quel est l'arrêt de bus le plus proche de la gare ? ». Ce type de requêtes est appelé requêtes de localisation. Ces requêtes ont commencé à apparaître avec les applications dépendantes de la localisation (LDA – Location Dependent Applications). D'après Tom Clements [CLE 04], les LDAs peuvent être classées en trois catégories : les LDAs concernant la gestion des véhicules (services de contrôle, de notification, de guidage, diagnostic de véhicules distants, d'informations sur le trafic, etc.), les LDAs de l'Internet et les LDAs sans fil. Il y a quelques années, les LDAs de l'Internet ont commencé à être introduites [BUY 99]. Dans ces applications, la localisation des utilisateurs est récupérée dans le profil des utilisateurs stockés par des services tels que my.yahoo.com et prise en compte pour personnaliser les recherches effectuées sur Internet. Par exemple, si un internaute recherchait des restaurants italiens, sa réponse était construite en fonction de la proximité des restaurants italiens par rapport à sa localisation. Dans ce cas, par exemple, seule une localisation fixe et préenregistrée de l'utilisateur est prise en compte. L'éventuelle mobilité de l'utilisateur n'est alors pas envisagée. C'est véritablement depuis l'arrivée des terminaux nomades que ces applications connaissent un essor et un regain d'intérêt considérable, de multiples scénarios [HAZ 04] peuvent alors être considérés.
Dans les applications de proximité, le service de localisation d'une application de proximité doit donc permettre, pour considérer la mobilité des utilisateurs, l'expression et l'évaluation des requêtes dépendantes de la localisation. L'expression de telles requêtes doit permettre de prendre en compte le critère de comparaison pour l'évaluation de la requête. Par exemple, pour les requêtes citées précédemment, le critère de comparaison était *« le plus proche »* mais on peut également envisager des requêtes où on souhaite rechercher un arrêt de bus à moins de 50 mètres de moi, dans ce cas le critère de comparaison devient *« à moins de 50*

mètres ». Ces requêtes se sont principalement développées grâce au développement des terminaux nomades et à la mobilité des utilisateurs. En effet, elles sont particulièrement utiles et exploitables dans des environnements ubiquitaires composés de terminaux nomades et de technologies de réseaux sans fil. Dans cette section, nous définissons les requêtes de localisation. Nous présentons ensuite les différentes solutions concernant la gestion de ces requêtes de localisation et les environnements auxquels elles sont destinées.

4.1. Définition

Avec le développement des applications dépendantes de la localisation, s'est développé le concept de requêtes de localisation. Une requête de localisation est une requête composée en partie d'attributs concernant la localisation. Par exemple, les requêtes « où est Paul Dupont ? », « Quel est le bureau Air France le plus proche de moi ? » sont des requêtes de localisation. En revanche, la requête « Quel est la compagnie aérienne affrétant le vol 9856 ? » n'est pas une requête de localisation.

Il existe deux catégories de requêtes de localisation [SEY 01] :

- les requêtes relatives à la localisation (ou LAQ – Location Aware Query)
- les requêtes dépendantes de la localisation (ou LDQ – Location Dependent Query)

Les requêtes relatives à la localisation sont des requêtes de localisation dont les attributs « de localisation » ne concernent pas la localisation du client. Un exemple de LAQ est la requête : « où est le bureau Air France le plus proche de Paul ? ».

Les requêtes dépendantes de la localisation sont des requêtes dont un des attributs de localisation concerne la localisation du client. Un exemple de LDQ est « quel est le bureau Air France le plus proche de moi ? ».

Les requêtes dépendantes de la localisation peuvent être continues ou non. Une LDQ continue permet de gérer la requête en temps réel et dans un intervalle de temps défini. Par exemple, avoir des informations sur les différents hôtels proches de nous au fur et à mesure lors d'un voyage en voiture. Une LDQ non continue permet quant à elle de retrouver des informations au moment où la requête est émise uniquement.

Au sein d'une application de proximité, nous nous intéressons particulièrement à la gestion des LDQs non continues. Ces requêtes reposent sur la gestion d'informations dépendantes de la localisation. Dans la section suivante, nous présentons les différentes solutions de gestion de ces informations et leurs impacts sur les techniques d'évaluation des requêtes dépendantes de la localisation.

4.2. Gestion des requêtes de localisation

La gestion des requêtes de localisation au sein des environnements mobiles représente un enjeu important en facilitant le développement d'applications tels que les LDAs. Cependant, les environnements mobiles sont sujets à de nombreuses contraintes telles que les déconnexions, les ressources limitées etc. Ces contraintes compliquent la gestion des

données dans ces environnements, en particulier l'accès aux données. L'accès aux données peut suivre trois modes de fonctionnement différents : l'accès aux informations utilisant les techniques de cache de données (stratégie « client »), la diffusion (stratégie « à la volée ») et l'accès aux informations en fonction de la demande (stratégie « serveur »).
Dans la suite de cette section, nous présentons les différentes solutions concernant la gestion des requêtes de localisation qui nécessite un accès aux données dépendantes de la localisation, en fonction des trois modes de fonctionnement.

4.2.1. Stratégie « client »

L'accès aux informations en utilisant la stratégie « client » requiert l'utilisation d'un cache de données sur le client. Ce cache de données permet de stocker un certain nombre d'informations susceptibles de fournir une réponse à la requête émise par le client. Dans un contexte de mobilité, le cache de données peut jouer un rôle important car il permet de répondre en partie aux problèmes de performances et de déconnexions. Dans un contexte de systèmes répartis fixes, les informations stockées dans le cache peuvent devenir obsolètes si l'information a été modifiée sur le serveur. En revanche, concernant les données dépendantes de la localisation, la gestion du cache devient beaucoup plus problématique puisque ces données peuvent devenir obsolètes lorsque l'utilisateur se déplace : ce phénomène est appelé « invalidation spatiale ». Deux techniques sont utilisées pour maintenir la cohérence du cache dans les environnements mobiles et pour permettre d'évaluer de façon satisfaisante les requêtes dépendantes de la localisation : les stratégies d'invalidation du cache qui permettent de maintenir la cohérence des informations dépendantes de la localisation stockées dans le cache, les stratégies de remplacement qui permettent de sélectionner les informations à supprimer et à remplacer et finalement les solutions de cache sémantique.
Les stratégies d'invalidation du cache reposent sur une méthode qui consiste à relier chaque information dépendante de la localisation avec sa région de validité [XU 03, ZHE 02]. Cette région définit l'espace dans lequel l'information reste valide.
Les stratégies de remplacement utilisent également les régions de validité, de même que la distance entre la région de validité de l'information, la localisation du client et la direction de mouvement du client [XU 99, ZHE 02, JUN 02].
Finalement les solutions de cache sémantique ont pour objectif d'organiser les informations stockées dans le cache afin de faciliter l'évaluation des requêtes. Les informations stockées dans un cache sémantique sont constituées des résultats partiels ou complets des requêtes précédentes appelés segments sémantiques. L'utilisation de cache sémantique présente l'avantage de stocker les paramètres de localisation dans les résultats des requêtes [REN 00, MAN 04]. De plus, ces caches permettent de décomposer l'évaluation d'une requête en fonction des informations stockées dans le cache : une première partie de la requête peut être évaluée grâce au cache tandis que l'autre est évaluée grâce à une connexion au serveur. Cette technique permet de minimiser le trafic sur les réseaux sans fil.

Dans l'environnement des applications de proximité, l'utilisation d'un cache pour évaluer les requêtes de localisation présente plusieurs avantages : d'une part, cette méthode requiert moins d'énergie et de temps que si l'accès était distant ce qui est important dans un environnement contraint en terme de ressources. D'autre part, cette méthode permet de pallier les fréquentes déconnexions possibles dans un environnement mobile.
Cependant, l'exploitation de cette méthode au sein des applications de proximité est limitée. En effet, l'espace mémoire disponible pour le cache sur les terminaux nomades est souvent restreint. Les requêtes évaluées par le cache seront donc très peu nombreuses.

4.2.2. Stratégie « à la volée »

Cette stratégie repose sur la diffusion d'informations pour permettre l'évaluation des requêtes. Le choix des informations à diffuser est sensible et doit être effectué de façon à répondre au mieux aux requêtes des utilisateurs. En effet, toutes les informations ne peuvent pas être diffusées pour ne pas surcharger le réseau et l'inonder d'informations inutiles. Cette stratégie permet de prendre en compte la mobilité de l'utilisateur et les ressources limitées du terminal en terme de stockage.
Pour permettre l'utilisation d'informations diffusées dans l'évaluation de requêtes de localisation, différentes solutions ont été proposées [XU 03, JUN 02]. Ces solutions reposent sur l'utilisation de canaux de diffusion multiples et sur l'indexation des différentes régions de validité des informations. La structure de l'index comportant les différentes régions de validité est diffusée. Le client peut alors, en fonction de sa localisation, retrouver le canal adéquat et donc les informations recherchées.

Au sein des applications de proximité, cette stratégie peut être exploitée mais elle présente rapidement des limites concernant les informations à diffuser. En effet, l'évaluation des requêtes dépendantes de la localisation reposant sur des données diffusées est limitée aux informations dépendantes de la localisation dont la granularité n'est pas fine. Ces informations sont par exemple les prévisions météo [LEE 02].

4.2.3. Stratégie « serveur »

La stratégie « serveur » permet au client d'accéder au(x) serveur(s) stockant les informations en fonction des requêtes exprimées. C'est la stratégie la plus répandue dans les environnements mobiles actuels. Cette stratégie repose sur trois problématiques que sont le placement des données, la réplication des données et la planification des données/indexage. En effet, les données doivent être placées et stockées de telles façons qu'elles puissent, a priori, être accessibles par tout le monde et de n'importe où. Pour ce faire, il est nécessaire de mettre en place des politiques de réplication des données permettant, en outre, de pallier plus aisément les problèmes causés par les pannes de serveurs. Les données doivent être indexées et planifiées de façon à permettre un accès rapide et efficace. Finalement, cette stratégie doit également prendre en compte les possibles déconnexions des clients. Ces

déconnexions peuvent en partie être gérées en paramétrant la requête : temps d'attente de la réponse, nombre de réponses, taille de la réponse etc. et/ou en proposant des solutions asynchrones permettant au client de récupérer sa réponse lors de sa reconnexion.

Concernant la gestion des requêtes dépendantes de la localisation, la stratégie « serveur » doit solutionner un problème supplémentaire. En effet, dans les solutions permettant la gestion des déconnexions, il faut s'assurer que la réponse récupérée et dépendante de la localisation est toujours cohérente avec la localisation actuelle du client [ZHE 01]. En effet, la déconnexion du client peut être due à un déplacement de celui-ci et dans ce cas, la réponse a de fortes chances d'être erronée. Trois approches concernant la gestion des données dépendantes de la localisation existent :

- l'approche qui repose uniquement sur la modification de la représentation des données : l'objectif est de stocker les données de telle façon que lorsqu'une requête est émise, elle est automatiquement routée vers les données adéquates. Dans ce cas, les requêtes ne sont pas modifiées mais les données sont stockées et accédées de manière à utiliser différentes instances de la base de données pour fournir des résultats adaptés. Madria et al. [MAD 00] propose une solution de représentation et de stockage des données hiérarchiquement en fonction de leurs localisations géographiques.
- l'approche qui repose uniquement sur la modification du langage de requêtes : les requêtes sont exprimées en spécifiant, de façon explicite, les informations concernant la localisation. Dans ce cas, des informations de localisation doivent être stockées avec les données. En revanche, le placement et l'accès aux données ne sont pas modifiés.
- l'approche qui repose à la fois sur une adaptation des requêtes et de la gestion des données. C'est cette troisième approche qui est le plus souvent utilisée. Dans l'article [DUN 98], l'environnement considéré est mobile et il repose sur l'utilisation de réseaux de téléphonie mobile. Ces réseaux sont composés de cellules représentées par un centre de commutation. Les informations sont centralisées au niveau de chaque centre de commutation. Ces informations représentent un répliqua spatial de l'ensemble des informations disponibles : c'est-à-dire que ce répliqua ne concerne que les informations relatives à la situation géographique de la cellule. De cette façon, lorsqu'un client émet une requête, celle-ci est évaluée au niveau du centre de commutation de sa cellule et donc, les résultats fournis seront dépendants de la localisation du client. La solution proposée dans [XU 00] repose elle aussi sur la topologie des réseaux cellulaires.

Les solutions proposées pour la gestion des requêtes de localisation reposant sur une « stratégie » serveur sont dédiées à des environnements sans fil. Cependant, elles s'appuient sur les réseaux de téléphonie mobile qui fournissent une certaine centralisation. Dans le cadre des applications de proximité, cette centralisation et la hiérarchie inhérente aux réseaux de téléphonie mobile n'existent pas.

Les trois stratégies présentées (client, diffusion et serveur) présentent chacune leurs avantages et leurs limites. En effet, ces trois stratégies sont complémentaires. Néanmoins, la stratégie « serveur » où l'utilisateur accède à une information distante au moment de l'émission de la requête nous semble incontournable. Cependant, dans une application de

proximité, chaque nœud de l'application choisit de partager une partie de ses informations et peut donc devenir serveur à tout moment. Une requête émise par un client doit donc interroger les nœuds distants (ou serveurs) si l'information recherchée n'est pas stockée sur le client. La diffusion peut également être utilisée dans le contexte des applications de proximité mais dans une moindre mesure.

5. Conclusion

Les premiers services de localisation, qui sont aujourd'hui largement utilisés, sont dédiés à des environnements filaires et fiables. Avec le développement du nombre d'usagers et de l'informatique ubiquitaire, sont apparus de nouveaux services de localisation. D'une part, les services de découverte de services sont adaptés à des environnements sans fil et permettent de faciliter la connexion de terminaux nomades. D'autre part, les services de recherche dans les réseaux P2P permettent, quant à eux, de répondre au grand nombre d'usagers et à la distribution de l'information sur le réseau. Cependant, nous avons présenté leurs limites dans les applications de proximité pour lesquelles de nouvelles solutions doivent être proposées. Une des limites transversale à l'ensemble des services de localisation existants est qu'ils ne permettent pas d'évaluer des requêtes dépendantes de la localisation. Ces requêtes nécessitent d'évaluer la localisation de l'utilisateur. De nombreuses solutions de localisation existent aujourd'hui pour permettre de localiser géographiquement un utilisateur mobile. Cependant, ces solutions nécessitent souvent une infrastructure absente dans les applications de proximité ou sont dédiées à des environnements particuliers (extérieur ou à l'intérieur d'un bâtiment). Dans une application de proximité, pour permettre l'évaluation d'une requête dépendante de la localisation, la localisation de l'utilisateur doit absolument être évaluée.

Ensuite, l'évaluation des requêtes de localisation peut être effectuée. Cette évaluation peut reposer sur trois stratégies différentes : « client », « diffusion » ou « serveur ». Les solutions actuelles sont dédiées à des environnements sans fil mais reposent sur l'utilisation d'une infrastructure centralisée. Elles doivent donc être adaptées pour permettre de répondre aux contraintes des applications de proximité.

Chapitre 3

ISLANDS : notre approche pour la localisation

1. Introduction

Dans une application de proximité, les informations sont réparties sur les différents terminaux présents. En effet, chaque participant peut, s'il le désire, partager une partie de ses informations et doit également pouvoir accéder à l'information disponible dans l'application. Pour illustrer ce concept d'application de proximité, nous présentons un exemple d'application déployée dans un aéroport. Les différents acteurs de cette application sont : les usagers de l'aéroport munis de leurs terminaux nomades, les différents vendeurs de l'aéroport et les serveurs d'informations de l'aéroport.

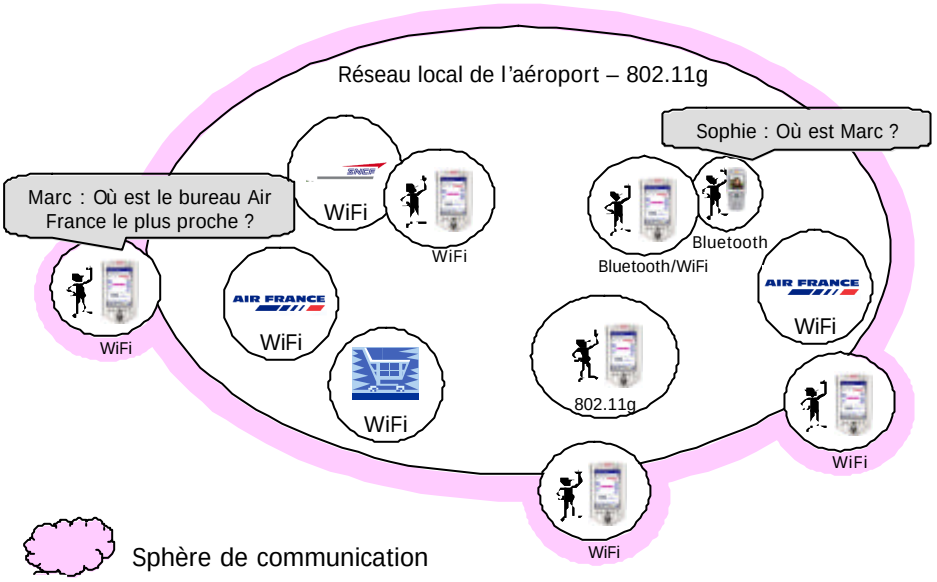


Figure 3.1. Exemple de scénario d'application de proximité dans un aéroport

Les différents réseaux présents vont former une sphère de communication : comme l'illustre la figure 3.1, la sphère de communication est basée sur l'association du réseau local WiFi 802.11g de l'aéroport permettant la diffusion des informations provenant des serveurs d'informations, des différents réseaux locaux des stands et magasins de l'aéroport et des réseaux des usagers. Dans cet exemple, la sphère de communication sur laquelle repose

l'application de proximité dépend d'une infrastructure fixe basée sur des ordinateurs fixes tels que des serveurs d'informations et/ou sur un ensemble de points d'accès WiFi. Les points d'accès (AP – Access Point) permettent de donner un accès sans fil aux différentes stations WiFi avoisinantes ; ces stations représentent tout équipement muni d'une carte d'accès WiFi. Si le point d'accès est raccordé à un réseau filaire, il permet également de fournir cet accès à l'ensemble des stations WiFi qui lui sont connectées.

Mais d'autres exemples d'applications de proximité peuvent être basés sur l'utilisation d'une sphère de communication reposant exclusivement sur un ensemble de participants mobiles. Par exemple, nous pouvons envisager des applications où les différents utilisateurs communiquent grâce au réseau WiFi en mode ad hoc (i.e. directement de nœud en nœud en fonction de la portée du réseau en opposition avec le mode infrastructure basé sur des points d'accès). Dans ce cas, l'application de proximité débute dès que deux usagers se trouvent physiquement assez proches l'un de l'autre pour être capables de capter l'autre et se termine lorsque les différents participants s'éloignent les uns des autres.

Dans l'application de proximité déployée dans un aéroport, grâce à la sphère de communication formée, les usagers de l'aéroport peuvent recevoir les informations diffusées au sein de l'aéroport. Dans les aéroports, de nombreuses données sont disponibles (horaires, retards, plans, services etc.) et sont déjà proposées de façon proactive aux usagers par le biais de panneaux d'affichage ou d'écrans de télévision par exemple. L'usager doit alors rechercher dans cet ensemble d'informations celle qui l'intéresse. En utilisant une application de proximité, cette recherche peut être facilitée, en effet, un utilisateur muni d'un terminal nomade peut paramétrer son numéro de vol sur son terminal. De ce fait, dès qu'il entre dans l'aéroport et donc dans la sphère de communication, le terminal peut filtrer les informations diffusées et ne fournir à l'utilisateur que les informations concernant son vol comme par exemple le numéro du bureau d'enregistrement.

Un participant de l'application peut également rechercher des informations de façon réactive à un besoin donné. Par exemple, dans l'exemple de la figure 3.1, Sophie vient chercher Marc à l'aéroport et souhaite le localiser. La localisation de Marc est stockée sur son assistant personnel et il a choisi de la partager. Grâce à l'application de proximité, Sophie peut émettre une requête concernant la localisation géographique de Marc et récupérer cette localisation. Un autre exemple de requête peut permettre à Marc de retrouver le bureau Air France le plus proche de lui : cette requête permet d'adapter les réponses de la requête en fonction de la localisation du client.

L'intérêt des applications de proximité est de fournir à la fois un accès proactif et réactif aux données. Les informations accédées de façon réactive ne sont pas restreintes aux seules informations fournies par l'infrastructure fixe de l'aéroport (informations sur les vols, offres des marchands, etc.) mais englobent également l'ensemble des informations partagées dynamiquement par les différents usagers de l'aéroport. En effet, ces informations sont réparties sur l'ensemble des terminaux des usagers et évoluent en fonction des entrées/sorties des usagers dans la sphère de communication. Chaque utilisateur peut donc émettre des

requêtes afin de localiser ces informations et la localisation de l'utilisateur peut être prise en compte dans l'évaluation de ces requêtes.
Les solutions actuelles pour la gestion et la localisation de données dans les réseaux sans fil sont principalement centralisées et ne sont donc pas adaptés aux applications de proximité où les informations sont réparties sur les différents terminaux des participants. Par exemple, les réseaux de téléphonie mobile qui sont aujourd'hui les réseaux sans fil les plus développés, reposent sur une gestion de données centralisées. En effet, ils se basent sur une architecture hiérarchique décrite dans la figure 3.2. Cette hiérarchie permet la centralisation au niveau de chaque centre de commutation (MSC – Mobile Switching Center) des données concernant les différents usagers connectés à une station de base (BTS – Base Transmission Station). Les données sont stockées dans deux bases de données différentes : l'enregistreur de localisations nominal (HLR – Home Location Register) gère les abonnés rattachés au MSC et l'enregistreur de localisations des visiteurs (VLR – Visitor Location Register) gère la localisation des téléphones mobiles qui traversent la zone dont s'occupe le MSC.

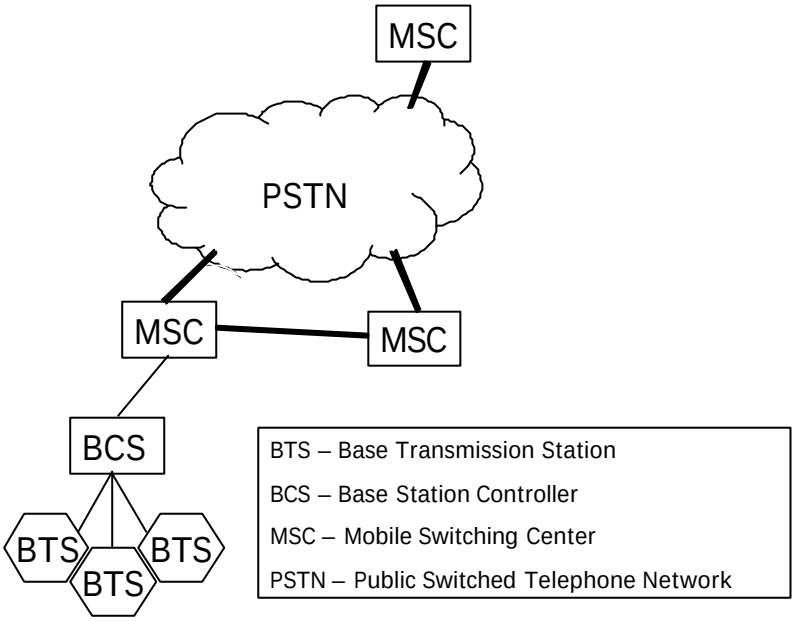


Figure 3.2. Hiérarchie d'un réseau de téléphonie mobile

Dans le cas des applications de proximité, une telle hiérarchie n'existe pas. En effet, l'hétérogénéité des réseaux utilisés et l'éventuelle absence d'infrastructure existante dans ces réseaux rend impossible une centralisation des données comme dans les réseaux de téléphonie mobile. Cette répartition de l'information explique également que les services de localisation actuels reposant sur des modèles d'architecture tels que le client/serveur ou l'architecture 3-tiers ne sont pas adaptés. D'autre part, les informations accessibles par les participants de l'application sont continuellement en cours d'évolution au gré des connexions et déconnexions des utilisateurs. En effet, les applications de proximité reposent sur un environnement très dynamique dû à l'hétérogénéité des terminaux et des réseaux utilisés d'une part et à la mobilité des utilisateurs d'autre part. De ce fait, les techniques de localisation de l'information utilisées dans les systèmes P2P ne conviennent pas aux

applications de proximité. Dans ce chapitre, nous présentons notre approche pour le service de localisation, ISLANDS[1] (ISLANDS – Information and Services LocalizAtioN and Discovery Service). Notre service fédère un ensemble d'îlots afin de permettre la localisation des informations tout en répondant aux exigences imposées par les applications de proximité. Il s'appuie sur un modèle de données ayant pour objectif la description des informations partagées et sur un modèle d'évaluation des requêtes permettant la gestion de la répartition des informations et l'évaluation des requêtes de localisation.

Ce chapitre est organisé de la manière suivante : la section 2 présente le modèle d'architecture choisi pour ISLANDS : le P2P hybride. La section 3 présente notre approche pour la localisation : le service ISLANDS. Le modèle de données défini pour représenter les informations partagées dans ISLANDS est décrit dans la section 4. Finalement, la section 5 présente les solutions proposées pour permettre un accès proactif et réactif aux informations disponibles.

2. Architecture Peer-To-Peer hybride

L'information disponible dans une application de proximité est répartie sur les terminaux des participants présents dans la sphère de communication. Pour le développement et le déploiement de ces applications, parmi les différents modèles d'architecture logicielle pouvant être utilisés pour gérer les communications entre les différents terminaux, nous avons choisi le Peer-To-Peer (P2P) hybride. Premièrement, nous présentons les caractéristiques des différents modèles et leurs limites dans le contexte des applications de proximité et justifions notre choix. Nous décrivons ensuite le modèle P2P hybride.

2.1. Justification

Il existe trois principaux modèles d'architecture pour développer des applications distribuées : l'architecture client/serveur traditionnelle, l'architecture 3-tiers et finalement l'architecture pair-à-pair. Nous décrivons dans cette section les points forts et les limites de chaque architecture pour le développement des applications de proximité.

2.1.1. Architecture client/serveur traditionnelle

L'architecture client/serveur [ORF 99] s'appuie sur un poste central appelé serveur. L'ensemble des machines des utilisateurs représente alors des clients. Sur l'ensemble des clients sont stockés indépendamment les applications clientes composées d'une interface homme/machine et les traitements nécessaires à l'application. Ces applications clientes doivent ensuite envoyer des requêtes sur le serveur unique pour accéder aux données centralisées uniquement stockées sur le serveur. Les avantages de cette architecture sont la

[1] Le nom ISLANDS a été choisi en référence aux différents « îlots » d'informations présents dans une application de proximité.

gestion facilitée des informations et de leur intégrité étant donné qu'elles ne sont stockées que sur le serveur et le contrôle d'accès aux informations. Mais les inconvénients de cette architecture sont nombreux en terme de fiabilité, de passage à l'échelle, de coût d'exploitation et de maintenance. De plus, la gestion centralisée intrinsèque de ce modèle d'architecture est en elle-même une limite importante pour les applications de proximité : en effet, centraliser l'ensemble des informations disponibles et réparties sur les différents terminaux des clients est une opération très coûteuse et les mises à jour en fonction des entrées/sorties des clients dans la sphère de communication et le maintien de la cohérence de ces informations ne sont pas envisageables dans un environnement aussi dynamique. En outre, il est impossible d'assurer la présence d'une machine « serveur » dans chaque environnement dans lequel s'exécute une application de proximité.

2.1.2. Architecture 3-tiers

L'architecture 3-tiers [EDW 99] est composée de trois couches :

1. La couche présentation associée au client qui de fait est dit "léger" dans la mesure où il n'assume aucune fonction de traitement à la différence du modèle client/serveur traditionnel.
2. La couche fonctionnelle (serveur d'applications), qui est, par exemple, un serveur Web muni d'extensions applicatives.
3. La couche persistance liée au système de gestion de base de données (SGBD)

Dans ce modèle, le client n'est muni que de fonctions d'affichage, il n'effectue donc aucun traitement, il émet uniquement des requêtes vers un serveur d'applications qui exécute les traitements et transmet les requêtes au serveur de données. Les résultats de ces requêtes seront ensuite affichés sur le client. C'est le serveur d'applications qui va effectuer tous les traitements et peut distribuer les requêtes vers d'autres serveurs additionnels comme un SGBD.

Les avantages de ce modèle par rapport au modèle client/serveur traditionnel se situent principalement au niveau du développement, du déploiement et des reconfigurations des applications. Néanmoins, les inconvénients de l'architecture client/serveur traditionnelle pour les applications de proximité dus, en particulier, à la centralisation des données, ne sont pas solutionnés dans le modèle d'architecture 3-tiers.

2.1.3. Architecture pair-à-pair

Dans une architecture pair-à-pair (P2P – peer to peer) [MIN 01], contrairement à une architecture de réseau de type client/serveur, il n'y a pas de serveur dédié. En effet, chaque ordinateur dans une telle architecture peut être à la fois client et serveur. Cela signifie que chacun des ordinateurs du réseau est libre de partager ses ressources et de mettre à disposition des informations. Il peut donc, à la fois émettre des requêtes sur les autres machines disponibles mais aussi répondre aux requêtes des autres utilisateurs. Le

développement considérable de cette architecture ces dernières années est dû aux avantages cités ci-dessous :

- Les applications développées sur ce modèle d'architecture ont pour objectif de s'auto-organiser. En effet, elles ne reposent pas sur l'utilisation de serveurs qui nécessitent d'être administrés.
- La répartition des charges est intrinsèquement facilitée étant donné que l'ensemble des requêtes émises n'est pas envoyé à un unique serveur mais est réparti sur les différentes machines présentes dans l'application.
- Ce modèle d'architecture ne repose pas sur un nombre d'utilisateurs défini au commencement des applications. Il permet donc, aux applications, une adaptation et une évolution automatiques en fonction des évolutions du nombre des utilisateurs.
- Finalement, en terme de tolérance aux pannes, étant donné que le modèle n'est basé sur aucune centralisation, la panne d'une machine n'affectera pas le bon déroulement des applications.

Cependant, ce modèle d'architecture présente également certains inconvénients : la décentralisation de ce modèle rend très difficile l'administration des applications dans un contexte où le nombre de nœuds est très important. Comme les requêtes peuvent potentiellement être envoyées à tous les participants, des problèmes de performances sont souvent rencontrés. Enfin, la sécurité est plus difficile à gérer dans les applications basées sur le modèle P2P.

2.1.4. Conclusion

Les trois modèles d'architecture décrits ci-dessus présentent chacun des avantages et des inconvénients pour le développement d'applications réparties. Cependant, les deux premiers modèles d'architecture (i.e. client/serveur et 3-tiers) reposent sur une centralisation partielle des applications (centralisation des données pour le modèle client/serveur, centralisation du traitement et des données pour le modèle 3-tiers). Cette centralisation serait très difficile, voire impossible à gérer dans une application de proximité : d'une part parce que la présence d'une machine suffisamment puissante pour permettre cette centralisation n'est pas garantie et d'autre part, parce que l'ensemble des informations disponibles évolue rapidement et de manière imprévisible. Elle ne peut donc pas être gérée efficacement de façon centralisée.

En outre, l'architecture P2P, présente, elle, de nombreux avantages pour les applications de proximité. Sa décentralisation intrinsèque est très bien adaptée à la distribution de l'information dans les applications de proximité : aucune machine serveur n'est requise et chaque utilisateur peut partager et rechercher des informations. De plus, le nombre de participants de l'application peut varier au gré des entrées/sorties des usagers dans la sphère de communication sans affecter le déroulement de l'application. Par ailleurs, en terme de fiabilité, la panne d'un des terminaux présents (dûe par exemple à une déconnexion accidentelle) dans l'application n'affecte pas le déroulement de l'application.

Pour les raisons évoquées ci-dessus, nous avons choisi de baser le développement des applications de proximité sur le modèle d'architecture P2P. Cependant, des solutions doivent être apportées afin de pallier les différents inconvénients de ce modèle. En effet, ce modèle souffre souvent de performances plutôt médiocres en terme de temps de réponse pour une recherche d'informations. Ces mauvaises performances sont dûes en particulier à la distribution de l'information et à l'interrogation distribuée sur les différents nœuds du réseau. Nous avons donc adapté ce modèle à l'environnement des applications de proximité afin d'améliorer par la suite les performances obtenues pour le service de localisation.

2.2. Peer-To-Peer hybride

Pour faciliter la gestion des applications de proximité et plus particulièrement la localisation des informations dans ces applications, nous avons choisi d'utiliser le modèle d'architecture P2P hybride. Nous présentons, dans cette section, les caractéristiques de ce modèle d'architecture et son utilisation au sein d'une application de proximité.

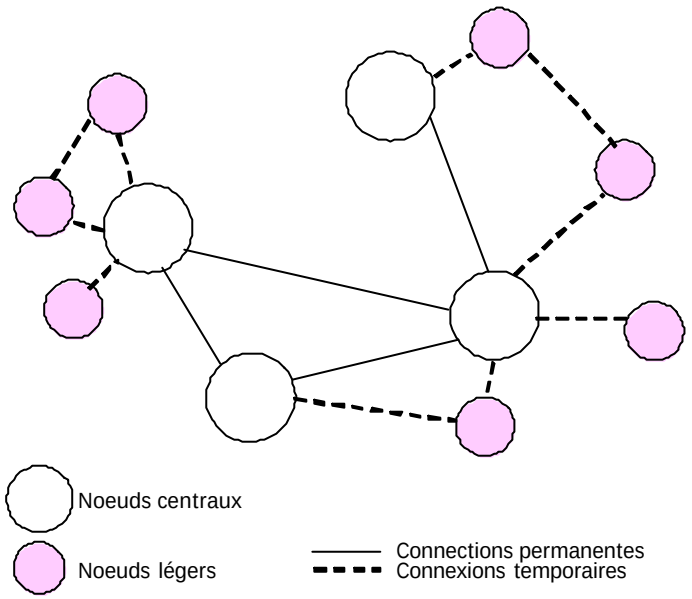


Figure 3.3. Modèle d'architecture P2P hybride dans les applications de proximité

Dans la littérature, deux types d'architecture P2P existent : le P2P pur et le P2P hybride [YAN 01]. D'après Yang et al., le modèle P2P hybride se distingue du modèle P2P pur, par la centralisation d'une fonctionnalité. En effet, dans le modèle P2P pur, l'ensemble des participants est à la fois client et serveur et aucune centralisation n'existe. Dans le modèle P2P hybride, une fonctionnalité est centralisée : par exemple, dans le système Napster, l'indexation des informations disponibles est centralisée sur différents serveurs mais l'échange des fichiers reste cependant distribué. Cette centralisation est gérée par un ensemble de super-nœuds qui jouent le rôle de serveur centralisé pour un sous-ensemble de nœuds [YAN 03]. Chaque nœud du système est connecté à un unique super-nœud et les super-nœuds sont connectés entre eux.

Nous avons choisi de développer les applications de proximité en nous basant sur le modèle d'architecture P2P hybride. Cependant notre définition du P2P hybride diffère un peu, du point de vue centralisation, de celle proposée par Yang et al. En effet, la distinction P2P pur vs. P2P hybride présentée par Yang et al. est principalement basée sur les systèmes P2P actuels utilisés sur Internet. Or dans les applications de proximité, l'environnement est différent d'Internet. Premièrement, les types de terminaux connectés au sein d'une application de proximité sont très hétérogènes et il ne s'agit pas uniquement d'ordinateurs de bureau comme c'est principalement le cas sur Internet. Deuxièmement, sur Internet, les différents nœuds connectés peuvent toujours communiquer directement entre eux. Dans les applications de proximité, la couverture des réseaux sans fil locaux et personnels ne permet pas toujours aux différents nœuds présents dans l'application d'être directement connectés.
Le modèle d'architecture P2P hybride utilisé dans notre environnement distingue également deux types de nœuds mais cette distinction n'est pas uniquement basée sur la centralisation d'une fonctionnalité. En effet, dans l'environnement des applications de proximité, il nous semble important de différencier les différents nœuds en fonction de leur comportement mais aussi de leurs ressources. Comme l'illustre la figure 3.3, nous avons donc choisi de distinguer deux catégories de nœuds dans les applications : les nœuds centraux et les nœuds légers. Les nœuds centraux correspondent à des machines robustes et fixes tandis que les nœuds légers correspondent aux terminaux nomades. Contrairement à la définition du P2P hybride de Yang et al., les nœuds légers peuvent être connectés à différents nœuds centraux en fonction des connexions existantes. De la même façon, les nœuds centraux ne sont pas obligatoirement tous connectés entre eux. Les nœuds centraux centralisent, a priori, une quantité importante d'informations. Les nœuds centraux sont majoritairement choisis au déploiement de l'application. Par exemple, lorsqu'une application de proximité est déployée au sein d'un aéroport, les nœuds centraux sont représentés par les différentes machines constituant l'architecture fixe de l'aéroport : serveurs d'informations, ordinateurs des vendeurs etc. Le nombre de ces nœuds centraux varie donc peu au cours de l'application. L'ajout de nœuds centraux au cours du déroulement d'une application de proximité est cependant envisageable. Par exemple, un utilisateur entre dans l'aéroport et il est muni d'un ordinateur portable relativement puissant. Il peut alors décider de participer à l'application en temps que nœud central et mettre à disposition ses ressources. Cependant, par défaut, lorsqu'un nouveau participant intègre une application de proximité, il est considéré comme nœud léger. Contrairement aux nœuds centraux, les nœuds légers correspondent à des appareils portables et mobiles qui entrent et sortent de l'application au gré des déplacements des usagers. Dans l'exemple de l'application de proximité déployée dans un aéroport, l'ensemble des nœuds légers correspond à l'ensemble des terminaux nomades des usagers.

3. Le service ISLANDS

Suivant le modèle P2P hybride, les informations au sein d'une application de proximité sont réparties sur les différents nœuds. En effet, chaque nœud choisit de partager une partie de ses

informations stockées localement. Le service de localisation ISLANDS a pour objectif de permettre la localisation de ces informations par tout participant de l'application de proximité. Dans cette section, nous présentons comment le service ISLANDS prend en compte cette répartition des informations sur les différents nœuds de l'application. Nous décrivons ensuite, l'architecture interne d'une instance d'ISLANDS.

3.1. Fédération des instances d'ISLANDS

L'objectif du service de localisation ISLANDS est de former un espace de recherche commun permettant aux différents participants d'accéder à l'information disponible. Pour obtenir un espace de recherche commun, des solutions sont déjà utilisées actuellement : parmi elles, la fédération de différents services de nommage permettant un accès à tous les objets enregistrés ou la fédération de différents services d'annuaires par l'utilisation des « referrals » spécifiés dans LDAP. Ces solutions reposent sur le stockage sur un site donné de l'adresse des services distants référencés. S'inspirant de ces solutions, nous avons choisi, comme l'illustre la figure 3.4, de déployer une instance du service de localisation ISLANDS sur l'ensemble des nœuds (centraux et légers) présents dans l'application et d'obtenir une fédération de ces différentes instances d'ISLANDS. Les différentes instances sont référencées entre elles en fonction des connexions existantes entre les différents nœuds.

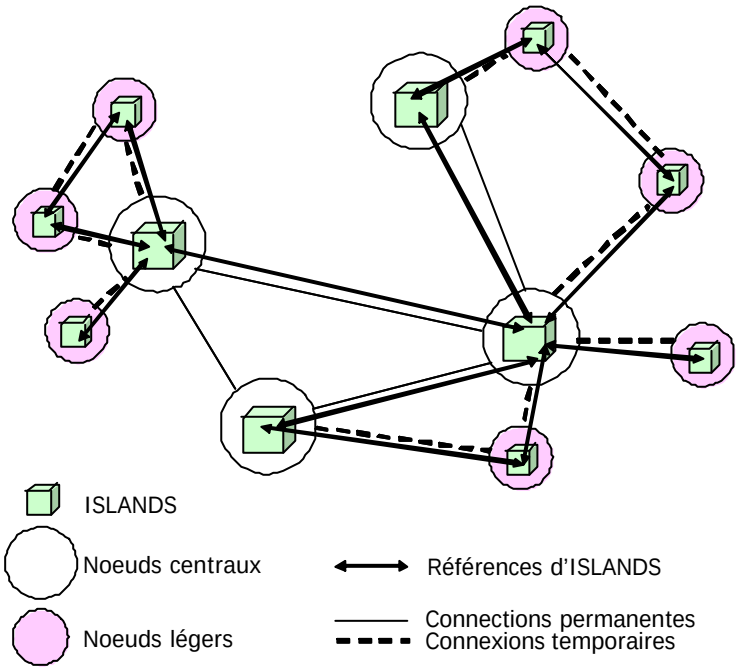


Figure 3.4. La fédération des instances du service de localisation ISLANDS

D'autre part, chaque instance du service de localisation ISLANDS prend en compte la distinction entre les différents nœuds. Le service est personnalisé et adapté en fonction du type (central ou léger) et des ressources (puissance CPU, espace mémoire, etc.) du nœud sur lequel il est hébergé. Nous présentons dans la section suivante l'architecture interne d'une instance du service ISLANDS.

3.2. Architecture interne d'une instance d'ISLANDS

L'architecture d'une instance du service ISLANDS est décrite dans la figure 3.5. Chaque instance repose sur un ensemble de composants qui peuvent être adaptés et/ou simplifiés si le nœud sur lequel l'instance est déployée, est contraint en terme de ressources. Par exemple, s'il s'agit d'un nœud central, le service est en mesure de référencer une large quantité d'informations et d'évaluer des requêtes complexes. En revanche, lorsque le service est sur un nœud léger, il permet principalement à l'utilisateur d'exprimer une requête et de lui renvoyer la réponse. Dans une moindre mesure, un service de localisation sur un nœud léger est capable d'interroger l'information partagée localement. Deux composants principaux sont à considérer dans le service de localisation ISLANDS : le gestionnaire de stockage et l'évaluateur de requêtes. Le gestionnaire de stockage permet de gérer un ensemble de métadonnées décrivant les informations partagées par le nœud : le modèle de données utilisé est décrit dans la section 4. L'évaluateur de requêtes, présenté, quant à lui, dans la section 5, prend en charge la requête dès que celle-ci est exprimée par l'utilisateur. En effet, une interface utilisateur permet de générer une requête en fonction des différents champs saisis par le client. Cette requête est ensuite analysée pour identifier s'il s'agit d'une requête dépendante de la localisation :

- s'il ne s'agit pas d'une requête dépendante de la localisation : la requête est tout d'abord évaluée localement. Si l'évaluation locale n'est pas satisfaite, la requête est alors distribuée sur les autres instances distantes. Lorsqu'une requête est émise par un nœud léger, la plupart du temps, son évaluation est effectuée sur des nœuds distants afin de récupérer une réponse. Le principe d'évaluation des requêtes est détaillé dans la section 5.1.
- s'il s'agit d'une requête dépendante de la localisation : l'évaluateur contacte le module de positionnement et récupère la localisation géographique du client. Le fonctionnement de ce module de positionnement est décrit dans le chapitre 4. La requête est ensuite évaluée suivant le modèle d'évaluation décrit dans le chapitre 5.

Finalement, l'évaluateur récupère l'ensemble des réponses et les sélectionne en fonction des besoins du client. Par exemple, si le client ne souhaite que 10 réponses, l'évaluateur sélectionne alors les 10 meilleures réponses et les envoie au client.

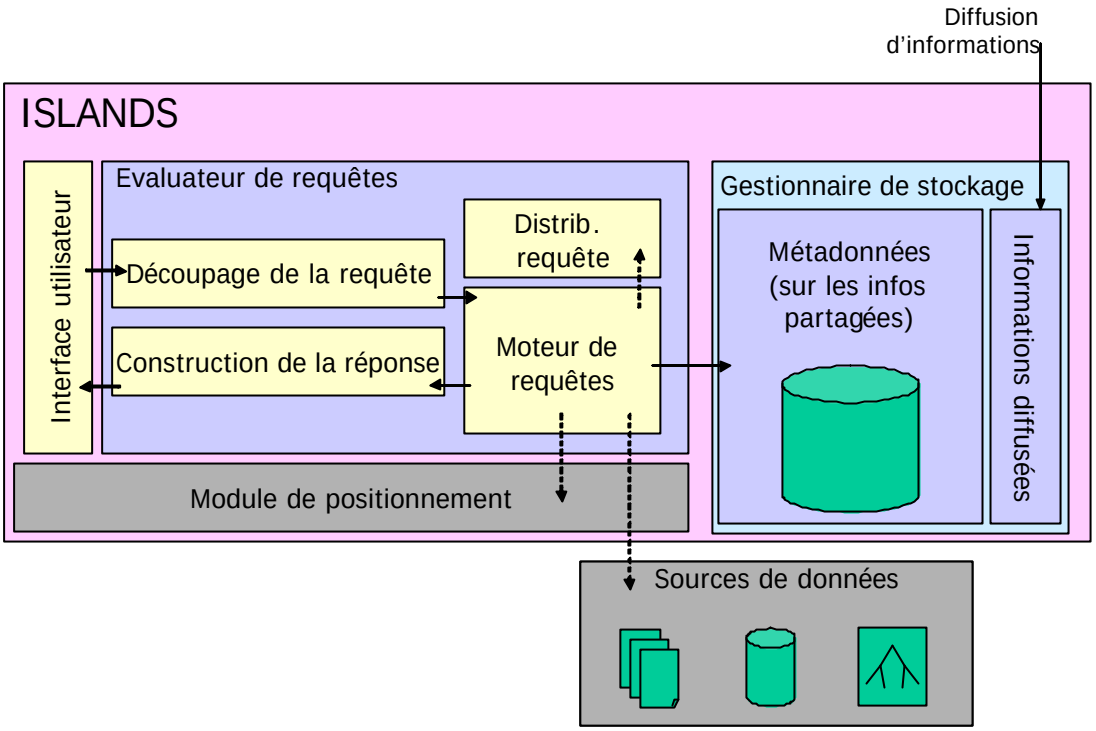


Figure 3.5. Architecture interne d'une instance du service ISLANDS

4. Représentation des informations dans ISLANDS

Au sein des applications de proximité, le service de localisation permet de localiser toute information disponible. Les informations disponibles dans une application de proximité peuvent être très hétérogènes : il ne s'agit pas uniquement, comme dans les systèmes P2P présents sur Internet d'un seul type de fichiers tel que des fichiers audio par exemple. Dans une application de proximité, les participants peuvent choisir de partager des informations hétérogènes telles que des fichiers (images, audio, texte etc.) ou des informations contenues dans une base de données (par exemple les prix des différents articles d'un vendeur). Un participant peut, par exemple, choisir de partager un ensemble de photos, son CV et sa localisation géographique. Ces informations sont donc décrites au sein du service de localisation ISLANDS afin d'être accessibles.

Dans le service ISLANDS, les informations que les utilisateurs ont choisies de partager sont représentées par des métadonnées. Les métadonnées sont des « données à propos des données » : par exemple, une bibliothèque décrit l'ensemble de ses ouvrages disponibles par une compilation de métadonnées présentant les différentes publications : c'est le catalogue de la bibliothèque. L'utilisation de métadonnées permet de ne pas devoir dupliquer les données à partager au sein du service de localisation. Le service de localisation ne stocke alors que les descriptions des données partagées : les métadonnées. De plus, les métadonnées facilitent la localisation d'informations hétérogènes, stockées différemment en permettant d'uniformiser la représentation de ces informations et donc la recherche associée.

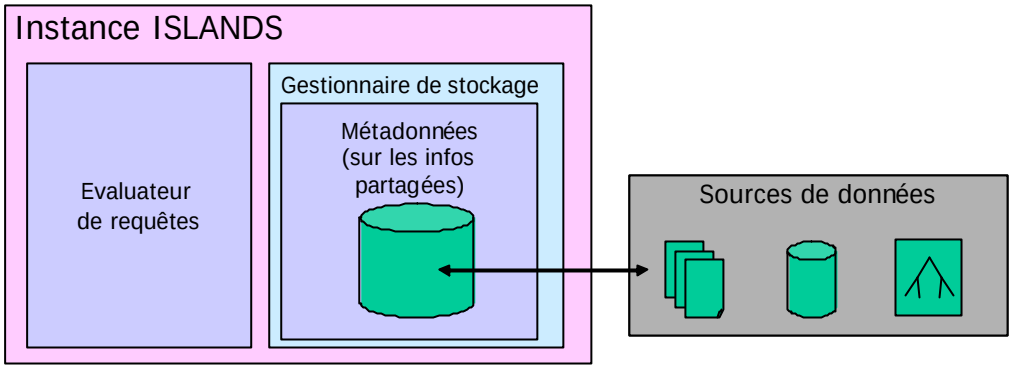


Figure 3.6. Utilisation des métadonnées dans ISLANDS

Comme l'illustre la figure 3.6., un nœud peut disposer de différents supports de stockage tels qu'un SGBD, des fichiers ou un service d'annuaires. Les données partagées et stockées dans ces différents supports sont représentées localement grâce aux métadonnées au sein de l'instance du service de localisation ISLANDS. Ces métadonnées sont, par exemple, stockées au sein d'un SGBD afin d'exploiter les mécanismes d'optimisation offerts et permettant ainsi une interrogation rapide et efficace en terme de performances. Les métadonnées décrivent les données et fournissent un lien permettant ensuite d'accéder à ces données.

4.1. RDF – Resource Description Framework

Nous avons sélectionné comme modèle de description des métadonnées : le framework RDF (RDF – Resource Description Framework [RDF]). RDF est une proposition du W3C ayant pour objectif d'assurer l'interopérabilité entre les applications qui échangent de l'information sur Internet. RDF augmente la facilité de traitement automatique des ressources Web. Aujourd'hui, RDF est largement utilisé pour décrire les ressources partagées au sein de systèmes P2P tels que Edutella [NEJ 02]. La syntaxe utilisée pour décrire les métadonnées décrites suivant RDF est basée sur le format XML (XML – eXtensible Markup Language [XML]). RDF est basé sur quatre concepts principaux :

- La ressource : une ressource représente toute information décrite par une métadonnée. Ces ressources sont identifiées par une URI (URI – Unique Resource Identifier). L'URI se décompose en trois parties : le mécanisme permettant d'accéder à la ressource (le protocole HTTP par exemple), la machine sur laquelle se situe la ressource et le nom de la ressource sur cette machine. Une URI peut par exemple, être une URL (URL – Uniform Resource Locator) pour représenter une ressource Web ou un chemin d'accès au fichier *maphoto.jpeg* pour représenter la ressource *maphoto.jpeg* qui est une image. L'URI est utilisé dans ISLANDS pour permettre l'interrogation de la donnée décrite en fonction de son support de stockage. Si la ressource décrite par la métadonnée est un fichier : dans ce cas, l'URI représente le chemin d'accès au fichier. Si la ressource décrite est une donnée stockée dans un SGBD, dans ce cas, l'identifiant fournit un moyen

d'accès à la donnée : l'identifiant du SGBD couplé à la requête permettant la récupération de l'information décrite par la métadonnée. L'URI permet ainsi d'accéder à la donnée à partir de la métadonnée.

- Le prédicat : le prédicat représente une propriété de la ressource. Par exemple, le « type » peut être une propriété de la ressource *maphoto.jpeg*.
- L'objet : l'objet représente la valeur correspondant au prédicat. L'objet peut soit être un littéral, soit une autre ressource. Pour la ressource maphoto.jpeg, l'objet correspondant au prédicat « type » peut, par exemple, être « portrait ».
- La déclaration : une déclaration RDF est un triplet (ressource, prédicat, objet). Une ressource spécifique associée à une propriété définie ainsi que la valeur de cette propriété (i.e. objet) pour cette ressource est une déclaration RDF. Dans une déclaration, la ressource sera aussi appelée sujet. Si nous utilisons l'exemple de la ressource *maphoto.jpeg*, son URI est : File:///C:/Documents%20and%20Settings/Marie/Mes%20documents/Mes%20images/ maphoto.jpeg. Cette ressource a comme prédicat « style », l'objet « portrait ». Un second exemple : la ressource identifiée par l'URI « http://marie.thilliez.free.fr » a comme prédicat «creator », l'objet «Marie Thilliez ». Une déclaration peut être schématisée à l'aide d'un graphe comme l'illustre la figure 3.7. Dans le tableau 3.1, cette déclaration est décrite suivant la syntaxe RDF/XML.

```
<?xml version="1.0">
<rdf:rdf xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://description.org/schema/"
    <rdf:description about="http://marie.thilliez.free.fr">
        <s:creator>Marie Thilliez</s:creator>
    </rdf:description>
</rdf:rdf>
```

Tableau 3.1 Exemple de représentation d'une déclaration RDF en XML

Grâce au format XML, la lisibilité du document est facilitée par l'utilisation des espaces de nom. Ces espaces de nom permettent en particulier de définir les schémas utilisés. En effet, les noms des propriétés présentes dans une déclaration doivent être associés à un schéma. Ceci peut être réalisé, comme dans l'exemple, en qualifiant les noms d'éléments avec un préfixe de l'espace de nom pour relier sans ambiguïté la définition de la propriété avec le schéma RDF correspondant.

L'utilisation de RDF pour décrire les métadonnées au sein des applications de proximité présente plusieurs intérêts. Tout d'abord RDF est un standard du W3C, il est aujourd'hui bien utilisé dans le domaine des systèmes P2P. Ce développement est en partie dû à l'utilisation de XML comme langage de représentation. En effet, XML est aujourd'hui largement utilisé mais seul, il ne permet cependant pas une gestion aisée de métadonnées en particulier à cause de sa structure hiérarchique. En effet, l'insertion de nouvelles descriptions au sein d'une structure hiérarchique est difficile : il faut définir le niveau hiérarchique correspondant à chaque description à insérer. En revanche, RDF permet une représentation plate des métadonnées et de ce fait, fournit une solution facilement extensible.

Les métadonnées décrites en utilisant RDF peuvent représenter un large éventail d'informations dans la mesure où il est possible de décrire, pour chaque domaine d'applications de nouveaux schémas de métadonnées. Un schéma RDF est un ensemble de ressources RDF qui permettent de décrire des propriétés d'autres ressources RDF appartenant à un domaine d'application particulier et suivant un vocabulaire précis. Finalement RDF facilite le passage à l'échelle : RDF permet une représentation plate des différentes ressources. Les déclarations RDF sont de simples enregistrements comportant différents champs (la ressource, les prédicats et les valeurs). Ces déclarations sont donc indépendantes les unes des autres et il est alors facile d'en insérer de nouvelles ou d'en supprimer.

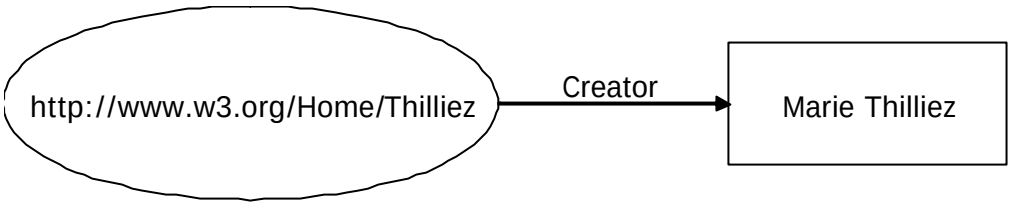


Figure 3.7. Exemple de représentation d'une déclaration RDF à l'aide d'un graphe

4.2. Schéma RDF pour ISLANDS

Dans le cadre des applications de proximité, contrairement aux systèmes P2P sur Internet où le type d'informations partagées est défini (par exemple, des fichiers audio pour Kazaa), les informations potentiellement partagées par les différents participants ne peuvent pas être assimilées à un domaine particulier. Par exemple, dans l'application de proximité de l'aéroport, un usager choisit de partager des photos de vacances tandis qu'un autre usager choisit de partager son CV et finalement les vendeurs partagent les descriptions de leurs produits issues d'un SGBD. Définir un schéma RDF précis pour une application de proximité est donc impossible. Pour les applications de proximité, l'objectif est de permettre la description de toutes les informations disponibles : les métadonnées doivent donc suivre un schéma RDF général et générique permettant la représentation de l'ensemble des informations. Dans ce contexte, l'initiative de métadonnées du Dublin Core (DCMI – Dublin Core Metadata Initiative) [DCMI], décrite dans l'annexe 2, définit un ensemble de 15 éléments (ou prédicats) pour la description de ressources grâce au RDF. La DCMI a pour principal objectif de permettre la représentation standardisée des ressources présentes sur le Web. Cependant, de part la sémantique générique utilisée, les différents éléments définis peuvent être utilisés au sein d'une application de proximité pour la description des données disponibles. Ces éléments sont :

- Les éléments décrivant le contenu :
    - Le titre de la ressource
    - Le sujet de la ressource qui peut être représenté par un ensemble de mots-clé
    - La description de la ressource qui est un résumé de la ressource

  - Le type de la ressource qui se réfère au vocabulaire de types défini par le DCMI [DCMIType]. Une ressource peut être, par exemple, une image, une donnée issue d'une base de données, du texte, etc.
  - La couverture représente la «portée » de la ressource : elle peut être représentée par une localisation spatiale, par une période de temps etc.
  - La source permet de référencer une ressource dont dépend la ressource décrite
  - La relation permet de référencer des ressources liées
- Les éléments concernant la propriété intellectuelle :
  - Le créateur de la ressource
  - L'éditeur de la ressource
  - Les collaborateurs de la ressource
  - Les droits concernant la ressource
- Les éléments concernant l'instanciation :
  - L'identifiant de la ressource qui correspond le plus souvent à l'URI de la ressource.
  - Le format de la ressource qui est définit par le type numérique de la ressource (par exemple jpeg).
  - La date correspond à la date de création de la ressource ou à la date de mise à disponibilité de la ressource.
  - La langue utilisée pour la ressource

Un exemple de représentation de la donnée Maphoto.jpeg est décrit dans le tableau 3.2. Néanmoins, dans le contexte des applications de proximité, un utilisateur doit pouvoir rechercher une information en fonction de sa localisation : par exemple, il doit être capable de retrouver l'arrêt de bus le plus proche de lui. Pour répondre à ce type de requêtes, la ressource « arrêt de bus » doit être décrite par une métadonnée dans le service de localisation.

En outre, cette métadonnée doit également permettre d'identifier la localisation de l'arrêt de bus. Nous proposons donc l'ajout d'un élément *localisation* à l'ensemble des 15 éléments définis par la DCMI. La description de cet élément est en annexe 4 de ce mémoire. Les éléments proposés par la DCMI sont représentés de la même façon qu'une ressource RDF et le format utilisé est XML. Il est donc facile d'ajouter un élément au schéma proposé par la DCMI. Cet élément a pour objectif de référencer une ressource dont le type est *localisationType*. Les ressources dont le type est *localisationType* représentent une localisation géographique. Dans le vocabulaire de types défini par le DCMI et décrit en Annexe 3, les types suivants sont définis : *Collection*, *Dataset*, *Event*, *Image*, *Interactive Resource*, *Moving Image*, *Physical Object*, *Service*, *Software*, *Still Image* et *Text*. Ces différents types permettent principalement de représenter les ressources présentes sur Internet, nous y ajoutons donc le type *localisationType*. Analogiquement à la description des éléments de la DCMI, les types sont également exprimés en tant que ressources RDF en XML.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.1/">
<rdf:Description rdf:about="C:\\Documents and Settings\\Marie\\Mes documents\\Mes
images\\maphoto.jpeg">
  <dc:creator>Marie Thilliez</dc:creator>
  <dc:title>Ma photo</dc:title>
  <dc:format>jpeg</dc:format>
  <dc:identifier>File:///C:/Documents%20and%20Settings/Marie/Mes%20documents/
Mes%20images/maphoto.jpeg </dc:identifier>
</rdf:Description>
</rdf:rdf>
```

Tableau 3.2. Exemple de représentation d'une déclaration RDF grâce aux éléments définis par la DCMI

De plus, pour permettre aux requêtes d'être évaluées sur l'ensemble des instances du service de localisation ISLANDS présentes dans une application de proximité, nous avons également ajouté un type *ISLANDSReference* permettant de spécifier une ressource qui fournit le chemin d'accès d'une instance distante du service de localisation ISLANDS. Ces métadonnées sont rafraîchies régulièrement en fonction des connexions/déconnexions du nœud. Ces métadonnées sont utilisées par l'évaluateur de requêtes pour permettre une évaluation distribuée de la requête sur l'ensemble de la sphère de communication de façon transparente pour l'utilisateur. Les deux types *localisationType* et *ISLANDSReference* sont décrits en annexe 5.

5. Accès à l'information

Comme nous l'avons déjà dit l'accès à l'information disponible au sein d'une application de proximité peut être réalisé de façon réactive ou proactive. Un client est capable d'émettre une requête pour localiser une information et il peut également recevoir sur son terminal les informations diffusées au sein de l'application de proximité. Dans cette section, nous présentons comment ces différents accès à l'information sont gérés.

5.1. Principe d'évaluation des requêtes

Le langage d'interrogation des requêtes utilisé permet l'interrogation des métadonnées stockées dans l'instance d'ISLANDS. Dans le cas d'un nœud central, pour des raisons de performances, l'utilisation d'un SGBD est favorisée. Dans ce cas, par exemple, les différentes métadonnées RDF sont stockées au sein du SGBD et interrogées en utilisant SQL. Dans le cas d'un nœud léger ne bénéficiant pas d'un SGBD, les métadonnées RDF peuvent, par exemple, être stockées dans des fichiers XML et interrogées grâce à XQuery.
Après avoir été exprimée, la requête est prise en charge par l'évaluateur de requêtes de ISLANDS. L'évaluation d'une requête classique, non dépendante de la localisation, se déroule en deux étapes :

- Etape 1 – l'évaluation locale : la requête est tout d'abord évaluée localement sur le nœud client. L'évaluateur interroge les métadonnées stockées dans l'instance locale d'ISLANDS. Si l'information recherchée est décrite par les métadonnées, l'évaluateur

récupère la valeur de l'élément *identifiant* de la ressource RDF. Cet élément permet de fournir un chemin d'accès à l'information recherchée. L'évaluateur peut ainsi récupérer l'information recherchée. Si la requête est satisfaite, la réponse est transmise au client. Sinon l'évaluateur effectue l'étape 2. Si le nœud client est un nœud léger très restreint en terme de ressources (mémoire et CPU), il peut ne pas avoir d'informations à partager et donc reposer sur une version d'ISLANDS dégradée. En particulier, l'évaluateur de requêtes ne permet alors que la gestion de l'étape 2 de l'évaluation d'une requête.

- Etape 2 – la distribution de la requête : l'évaluateur de la requête distribue aux autres services de localisation en fonction des métadonnées décrivant des ressources de type ISLANDSReference. La requête est alors, envoyée aux autres instances d'ISLANDS accessibles, et évaluée sur ces instances. Les réponses sont ensuite sélectionnées par l'évaluateur de requêtes de l'instance d'ISLANDS présent sur le nœud client pour être finalement envoyées au client.

Concernant la distribution de la requête vers d'autres services ISLANDS sur d'autres nœuds, l'évaluateur de requêtes choisit la distribution la plus adaptée en fonction des contraintes de temps de réponse et de qualité de réponse souhaitées. En effet, cette distribution est réalisée dans le but d'obtenir un compromis satisfaisant entre qualité de réponse et ressources utilisées pour cette évaluation distribuée. Plus une requête est distribuée sur des nœuds distants, plus les coûts en terme de communication, de traitement et de temps de réponse augmentent. Les différentes stratégies de distribution sont présentées et discutées dans le chapitre 6 de ce mémoire.

Ce principe d'évaluation est adapté à l'évaluation des requêtes classiques : par exemple, dans l'application de proximité se déroulant dans un aéroport, la requête « Quel est le bureau d'enregistrement du vol A7861 ? » est évaluée suivant le principe décrit ci-dessus. Un autre exemple de ces requêtes est «Où est Sophie ? » : dans ce cas, l'évaluateur recherche une ressource de type *localisationType* dont le créateur est Sophie.

5.2. Requêtes de localisation

Au sein d'une application de proximité, pour personnaliser l'information recherchée par un client en fonction de sa localisation géographique, les requêtes de localisation sont utilisées. Deux exemples différents de requêtes de localisation sont :

- Exemple 1 : «quel est l'arrêt de bus le plus proche de moi ? ». Dans ce cas, il s'agit d'une requête *dépendante* de la localisation. La localisation de référence est celle du client.
- Exemple 2 : «quel est l'arrêt de bus le plus proche de la gare ? ». Dans ce cas, il s'agit d'une requête *relative* à la localisation. La localisation de référence est alors celle de la gare.

5.2.1. Expression

Afin d'exprimer ces requêtes de localisation, nous avons introduit trois opérateurs simples et faciles d'utilisation. Ces opérateurs permettent aux utilisateurs d'exprimer des contraintes de localisation. Les trois opérateurs sont : *inside*, *closest* et *close* [THI 04a]. Ces opérateurs permettent d'exprimer à la fois des requêtes dépendantes de la localisation et des requêtes relatives à la localisation. En effet, l'opérateur prend en paramètre une localisation : par défaut la localisation du client qui émet la requête ou une localisation de référence spécifiée par le client.

L'opérateur *inside* est utilisé pour retrouver une information à l'intérieur d'une "zone géographique" définie. Ainsi, il peut être utilisé pour retrouver l'ensemble des vendeurs du premier étage d'un bâtiment. Le type de la « zone géographique » est défini comme paramètre de l'opérateur *inside* : dans l'exemple, le paramètre est « étage ».

L'opérateur *closest* est utilisé quant à lui pour retrouver une information donnée qui soit la plus proche de l'expéditeur de la requête (ou d'une localisation spécifiée). Il peut être utilisé, par exemple, pour retrouver l'arrêt de bus le plus proche de moi (ou le plus proche de la gare).

Finalement, l'opérateur *close* est une extension de l'opérateur *closest*. Cet opérateur est utilisé pour retrouver plusieurs éléments proches de l'expéditeur de la requête (ou d'une localisation spécifiée). Il est également possible de préciser un paramètre de distance. Ce paramètre optionnel est un entier représentant le nombre de mètres maximal séparant l'information recherchée de l'expéditeur de la requête (ou de la localisation spécifiée). Un exemple de requête dépendante de la localisation utilisant l'opérateur *closest* est décrit dans le tableau 3.3 : la requête est alors « rechercher l'arrêt de bus le plus proche de moi ».

```
Select busStop.Title, busStop.Description
From busStop
Where closest(BusStop.localisation);
```

Tableau 3.3. Exemple de requêtes dépendantes de la localisation exprimé en SQL

5.2.2. Evaluation

Après avoir exprimé la requête, pour l'évaluer, trois étapes distinctes sont exécutées :

- La localisation de référence est évaluée. Pour l'exemple 1, la localisation du client est récupérée. Dans le cas de l'exemple 2, la localisation de la gare est récupérée.
- La requête non dépendante de la localisation est évaluée. Dans les exemples, l'évaluation de cette requête consiste à récupérer l'ensemble des arrêts de bus et leurs localisations respectives. L'évaluation de cette requête s'effectue en deux étapes comme il est décrit dans la section 3.3.1.1.
- Lorsque les deux premières étapes sont effectuées, l'étape 3 consiste à évaluer le critère de localisation en fonction des réponses obtenues pour les étapes 1 et 2. Pour l'exemple,

cette étape consiste à retrouver l'arrêt de bus dont la localisation est *la plus proche* de la localisation de référence.

5.3. Diffusion de l'information

Les solutions de diffusion adaptées aux environnements sans fil [IMI 97, JUN 02] ne sont pas adaptées à l'environnement des applications de proximité. Ces solutions reposent sur la diffusion d'un index permettant aux utilisateurs mobiles de choisir le canal adapté en fonction de l'information qu'ils recherchent et ainsi d'économiser ses ressources. Cependant, le contexte de ces solutions est centralisé : l'information à diffuser est stockée de façon centralisée et permet donc la création d'un index. Dans les applications de proximité, l'information à diffuser provient de différents nœuds centraux. La solution utilisée dans les applications de proximité repose donc sur l'utilisation de groupes multicast. Ces groupes multicast sont créés en fonction du type d'informations diffusées dans ce groupe. Lorsqu'un utilisateur souhaite recevoir une information particulière, il peut alors s'abonner au groupe multicast adéquat et récupérer l'information qui l'intéresse. Cette solution permet de limiter les ressources utilisées par les terminaux nomades pour récupérer l'information diffusée [IMI 94]. Par exemple, dans l'application de proximité se déroulant dans un aéroport, un groupe multicast peut être créé pour la diffusion des bureaux d'enregistrement associés à chaque vol. Avant d'arriver à l'aéroport, l'utilisateur peut ainsi paramétrer son service de localisation en spécifiant qu'il est intéressé par les renseignements concernant les vols et en indiquant son numéro de vol. Dès son arrivée à l'aéroport, son service de localisation se joint aux groupes qui intéressent l'usager et il peut alors récupérer le bureau d'enregistrement de son vol.

6. Conclusion

Dans ce chapitre, nous avons défini le modèle d'architecture choisi pour les applications de proximité : le P2P hybride. Dans ce modèle d'architecture, on distingue deux types de nœuds : central et léger. Les nœuds sont principalement différenciés en fonction de leurs ressources et de leur comportement (mobile ou fixe). Le service de localisation ISLANDS proposé pour la localisation des informations au sein d'une application de proximité est donc adapté à ce modèle d'architecture. Nous choisissons de distribuer une instance d'ISLANDS sur chaque nœud. Les instances sont référencées entre elles au moyen de métadonnées en fonction des connexions entre les différents nœuds.

Le service gère à la fois un accès proactif aux données en permettant la diffusion de certaines informations mais également un accès réactif. Dans ce contexte, ISLANDS permet d'exprimer et d'évaluer une requête sur l'espace de recherche constitué des différents « îlots » d'informations. Un type particulier de requêtes est géré par ISLANDS : les requêtes dépendantes de la localisation. Pour l'évaluation de ces requêtes, la localisation géographique de l'utilisateur est requise. Cette localisation est fournie par le module de positionnement. Ce module est une interface entre les différents systèmes de positionnement

présents sur le terminal tels qu'un module GPS. En outre, ce module repose également sur une solution de localisation utilisant l'environnement du nœud client qui permet ensuite d'évaluer les requêtes dépendantes de la localisation. Dans le chapitre suivant, nous décrivons cette solution permettant la localisation géographique d'un utilisateur mobile.

Chapitre 4

Localisation de l'utilisateur mobile

1. Introduction

Actuellement de nombreuses techniques permettent de déterminer la localisation d'un utilisateur mobile. Cependant ces solutions ne sont pas disponibles sur tous les terminaux nomades utilisés aujourd'hui. Par exemple, un PDA de type Pocket PC peut être muni d'un module GPS mais celui-ci est optionnel et actuellement, de nombreux iPAQs ne sont donc pas équipés de dispositifs permettant une évaluation de leur localisation. De façon générale, le GPS qui est, aujourd'hui, le système de positionnement le plus utilisé [DAN 00, GPS 97], n'est pas disponible sur tous les terminaux nomades et n'est pas utilisable dans certains environnements : à l'intérieur d'un bâtiment. De plus, la localisation fournie par des techniques comme le GPS est représentée sous la forme d'une localisation physique (latitude, longitude, altitude) qui n'est pas facile à exploiter au niveau applicatif. En effet, l'utilisation d'une telle technique de localisation requiert une couche applicative permettant d'interpréter les coordonnées obtenues afin de fournir des informations exploitables par l'utilisateur. Nous proposons dans ce chapitre une solution de localisation permettant de localiser un utilisateur mobile et ne nécessitant aucune infrastructure et aucun dispositif spécifique [THI 04c]. D'après la classification de Hightower [HIG 01a] présentée dans le chapitre 2, notre solution est à la fois basée sur la proximité et sur l'atténuation du signal. En effet, notre algorithme est basé sur l'exploitation des métadonnées de ses nœuds voisins donc « à proximité » de l'utilisateur. S'inspirant des travaux d'Ermel et al. [ERM 03], en fonction des informations et plus précisément des localisations issues de ses différents voisins, l'algorithme calculera une «moyenne » pour définir la localisation de l'utilisateur mobile. De plus, notre algorithme exploite également les particularités des réseaux sans fil et est plus particulièrement basé sur l'atténuation du signal puisque, parmi les métadonnées utilisées, l'une d'entre elles est basée sur l'atténuation du signal mesuré.

Dans ce chapitre, nous décrivons dans un premier temps, les attributs des métadonnées utilisées par notre solution. Nous détaillons ensuite le principe de l'algorithme de localisation proposé en l'illustrant d'un exemple. Notre solution fournit une solution approximative d'un utilisateur mobile, nous décrirons donc en deuxième partie, les simulations effectuées pour les mesures de précision. Finalement, l'objectif est d'exploiter

notre solution de localisation dans un environnement contraint en terme de ressources : en effet, il est important de minimiser à la fois l'utilisation des ressources des terminaux nomades et le nombre de messages échangés sur le réseau. Nous proposons, donc, en troisième partie, différentes stratégies d'optimisation. Ces stratégies permettent à l'algorithme de localisation d'évaluer la position de l'utilisateur mobile tout en optimisant l'utilisation des ressources en fonction de l'environnement et du comportement de l'utilisateur mobile.

2. Estimation de la localisation d'un utilisateur mobile

Pour estimer la localisation d'un utilisateur mobile muni de son terminal nomade, nous proposons une solution permettant de fournir une localisation absolue de cet utilisateur : c'est-à-dire que la localisation obtenue n'est pas exprimée relativement à un point de référence spécifique. Un des objectifs de notre solution est de fournir une localisation à tout utilisateur mobile muni d'un terminal nomade avec des capacités de réseaux sans fil. Notre solution ne repose pas sur l'utilisation d'une infrastructure (telle qu'un réseau de capteurs par exemple), il peut donc être utilisé par tous et n'importe où. Il peut fournir une représentation de la localisation de l'utilisateur sous deux formes différentes : soit, une représentation physique (latitude, longitude, altitude), soit une représentation symbolique (par exemple, nom de la rue, du quartier, de la ville etc.). Dans cette section, nous décrirons dans un premier temps les métadonnées de l'environnement sur lesquelles nous nous appuyons, en décrivant en particulier les différents attributs utilisés. Nous détaillerons ensuite l'algorithme proposé en illustrant avec un exemple.

2.1. Description des métadonnées

Chaque utilisateur mobile souhaitant se localiser sera appelé nœud client dans la suite du chapitre. Grâce à ses capacités de communication, le nœud client est connecté en mode ad hoc à différents nœuds voisins i.e. la connexion les séparant est directe et n'utilise pas d'infrastructure telle qu'un point d'accès WiFi. Pour chaque noeud voisin du noeud client, une métadonnée est utilisée. Ces métadonnées, décrites en utilisant les différents éléments proposés par l'initiative DCMI, représentent à la fois les caractéristiques physiques du nœud voisin (comme sa localisation ou son type : central ou léger) et les caractéristiques de la connexion sans fil séparant le nœud client du nœud voisin. Nous décrivons ci-dessous les différents attributs utilisés par la solution de localisation géographique de l'utilisateur.
Il existe trois attributs décrivant les caractéristiques du nœud voisin :

- L'attribut *locationDescription* décrit la position du noeud distant. Cet attribut représente une métadonnée de type *localisationType*. D'après la figure 4.1, nous remarquons que deux descriptions peuvent être utilisées pour décrire la position d'un noeud: la description symbolique (bâtiment, étage …) et/ou la description physique (latitude, longitude et altitude). Cette position est représentée en XML afin de préserver la

sémantique des localisations et faciliter son traitement par la suite. Cette localisation peut ne pas être définie sur certains nœuds, en particulier sur certains nœuds légers. En effet, les nœuds légers ne bénéficient pas automatiquement de techniques de localisation et leur attribut *locationDescription* est donc, inexistant ou obsolète. L'attribut suivant est justement utilisé afin d'évaluer la fraîcheur de l'attribut *locationDescription*.

- L'attribut *locationLastUpdate* représente la date de la dernière mise à jour du fichier référencé par l'attribut *locationDescription*. Cet attribut est représenté par l'élément « *date de création* » défini par la DCMI pour la ressource *locationDescription*.
- L'attribut *mobilityProfile* décrit le profil de mobilité du noeud distant. Ce profil de mobilité est représenté sous la forme d'une vitesse (par exemple, s'il s'agit d'un piéton, cet attribut représente le nombre moyen de mètres par minute parcourus par l'usager). Pour chaque noeud voisin, on définit sa catégorie : très mobile, assez mobile, peu mobile ou immobile. Et en fonction du type de locomotion (voiture, vélo, piéton etc.), la vitesse du nœud voisin est calculée. Cet attribut est représenté par l'élément *« sujet »* défini par la DCMI. Le type du nœud distant (central ou léger) est également défini dans l'élément *« sujet »*.

```
<locationDescription>
  <symbolicLocation>
   <city name = « Valenciennes »>
    <district name = « downtown »>
     <street name = « Bd Watteau »>
      <number name = « 25 »>
      </number>
     </street>
    </district>
  </city>
 </symbolicLocation>
 <physicalPosition>
   <lat> 27.7 </lat>
   <long> -15.1 </long>
   <alt> 197.4 </alt>
 </physicalPosition>
</locationDescription>
```

Figure 4.1. Exemple de contenu pour l'attribut locationDescription

Il existe également deux attributs permettant de décrire la connexion entre le nœud voisin et le nœud client :

- L'attribut *connectionRange* est calculé en fonction du niveau du signal réseau (séparant le nœud client du noeud distant). Il est utilisé dans l'algorithme d'estimation de la localisation du nœud client pour estimer la distance séparant le nœud client du nœud voisin. En effet, dans les réseaux sans fil, une relation existe entre le niveau du signal radio et la distance séparant les deux nœuds : plus le signal est atténué (i.e. faible), plus la distance entre les deux nœuds est importante. Cependant, la distance estimée, basée sur le niveau du signal réseau de la connexion ne représente pas exactement la distance euclidienne entre les deux nœuds. En effet, le signal s'atténue dans un premier temps en fonction de la distance entre les deux nœuds mais également au contact des différents obstacles (tels que les murs, les plafonds, les fenêtres etc.) situés entre les deux nœuds. Cet attribut est représenté par l'élément *« couverture »* de la métadonnée.

- Enfin, l'attribut *connectionState* est un booléen permettant de connaître l'état de la connexion avec le noeud distant : s'il est encore connecté ou non au moment de l'exécution de l'algorithme d'estimation de la localisation du client. Cet attribut est représenté par l'élément *« description »* de la métadonnée.

Les attributs concernant les caractéristiques du nœud voisin sont obligatoirement stockés sur le nœud voisin. Chaque nœud gère localement les informations relatives à sa localisation. Ensuite, chaque nœud effectue, à intervalles réguliers, une mise à jour de son environnement. Ces différentes mises à jour vont permettre d'évaluer le profil de mobilité du nœud (i.e. en fonction du nombre de changements dans l'environnement du nœud) et d'adapter ensuite les intervalles de temps entre deux mises à jour. Ces mises à jour permettent principalement de connaître les différents voisins directs d'un nœud. Les différents attributs *connectionRange* sont donc ensuite calculés pour l'ensemble des voisins. Les attributs *connectionState* sont eux mis à jour en temps réel en fonction de l'état des connexions entre le nœud client et ses nœuds voisins.

2.2. Algorithme d'estimation de la localisation

L'ensemble des métadonnées issues de ses nœuds voisins va permettre au nœud client de se localiser en utilisant notre algorithme d'estimation de la localisation. Dans la suite, nous expliquons le fonctionnement général de l'algorithme pour ensuite le présenter de façon détaillée.

2.2.1. Principe de l'algorithme

L'algorithme calcule un degré d'approximation pour chaque fichier de localisation récupéré (i.e. l'attribut *locationDescription*). Le calcul de ce degré repose sur les attributs *locationLastUpdate*, *connectionRange* et *mobilityProfile*. Ce degré permet de classer les fichiers de localisation récupérés des nœuds voisins en fonction de leur «distance » avec la réelle localisation du nœud client. C'est-à-dire que plus le degré d'approximation est petit, plus petite est la «distance » séparant la localisation du nœud client de celle du nœud voisin. L'algorithme fournit donc, en sortie, une liste de couples (fichier de localisation, degré d'approximation) triés en fonction des degrés d'approximation. Cette liste de couples représentera la localisation du nœud client, elle peut ensuite être traitée de façon à l'utiliser dans une application. Nous verrons dans le chapitre suivant comment cette localisation est utilisée dans l'évaluation des requêtes dépendantes de la localisation.

2.2.2. Description de l'algorithme

Détaillons maintenant l'algorithme permettant d'obtenir cette liste de couples et illustré par la fonction évaluation dans la figure 4.2. Dans cet algorithme, une variable *Nb* est spécifiée comme contrainte d'environnement et indique le nombre maximal de localisations à traiter : il y aura au plus *Nb* couples (localisation, coefficient) dans la liste finale. Plus la variable *Nb*

est petite, plus le temps d'exécution de l'algorithme sera court. Cette variable permet également de limiter la taille de la liste finale dans un environnement fortement connecté où le nombre de nœuds voisins est important.

Après avoir initialisé la variable *Nb*, la liste de tous les nœuds voisins (i.e. tous les nœuds accessibles directement depuis le nœud client) est récupérée. Cette liste est mise à jour et stockée régulièrement sur le nœud client qui garde ainsi une connaissance de l'environnement qui l'entoure. Les nœuds voisins, décrits sur cette liste, peuvent donc légèrement différer par rapport aux nœuds voisins réellement connectés. Dans un deuxième temps, l'ensemble de ces nœuds voisins est filtré pour ne récupérer que ceux encore connectés au moment de l'évaluation de la localisation.

```
Function Evaluation return LocationList
{
  Nb := LocationListNb() ;
// retourne le nombre maximal de localisations désirées  dans la réponse LocationList
  RefPeerList = getPeerList() ;
// récupère l'ensemble des voisins référencés
  ConnectedPeerList := Connected(RefPeerList);
// l'attribut connectionState > 0
  nbConnectedPeers := nbPeer(connectedPeerList) ;
  LocationList := null ;
  If ConnectedPeerList is not null then
   {
    Int d[nbConnectedPeers] ;
    Int r[nbConnectedPeers] ;
    For i=1 to nbConnectedPeers loop
     {
      d[i] := coeff_d(connectedPeerList, i) ;
// définit le coefficient d pour chaque noeud voisin connecté : la difference entre la date actuelle et l'attribut
locationLastUpdate
      r[i] := coeff_r(connectedPeerList, i) ;
// définit le coefficient r en fonction de l'attribut connectionRange
      m[i] : = coeff_m(connectedPeerList, i) ;
// définit le coefficient m basé sur l'attribut mobilityProfile
      ApproxDegree := approxDegree(d[i], r[i], m[i]) ;
// retourne a*r[i] si le noeud i est un noeud central ou m[i]*d[i] + b*r[i] si le noeud i est un noeud léger
      locationList := concat(locationList, (location(i), approxDegree(i))) ;
    }
   If Nb > nbConnectedPeers then Nb := nbConnectedPeers ;
   LocationList := Sort(LocationList, Nb);
// retourne une liste de Nb couples tries en fonction du degré d'approximation
   Return LocationList ;
}
```

Figure 4.2. Algorithme d'évaluation de la localisation d'un utilisateur mobile

Si le nœud client est connecté à d'autres noeuds, les variables m_i, d_i et r_i sont définies pour chaque noeud voisin i connecté. L'ensemble des variables m_i correspond au profil de

mobilité des noeuds distants (dans le cas d'un noeud central, cette variable sera nulle). L'ensemble des variables d_i correspond à la différence entre la date actuelle et l'attribut *LocationLastUpdate*. Donc, plus d est grand, plus les données de localisation associées datent et moins elles sont fiables. Si le noeud distant est un noeud central, les variables m_i et d_i sont nulles parce que le noeud est a priori immobile et donc la localisation de ce nœud est donc toujours à jour. L'ensemble des variables r_i désigne la « proximité » entre les nœuds dont le calcul est basé sur l'utilisation de l'attribut *connectionRange*.
Ensuite, pour chaque noeud connecté, on calcule le degré d'approximation : $a*r_i$ pour un noeud central ou $m_i*d_i + b*r_i$ pour un noeud léger. Les coefficients *a* et *b* permettent de fixer des priorités (entre noeuds centraux et noeuds légers) pour calculer les meilleures localisations en fonction de l'environnement. Finalement, une liste de *Nb* couples (fichier de localisation, degré d'approximation) est triée en fonction du degré d'approximation de chaque localisation. *Nb* correspond soit à la variable d'environnement définie en début d'algorithme, soit au nombre de noeuds connectés si celui ci est inférieur à *Nb*.
Cette évaluation de la localisation renvoie une liste de fichiers de localisation triés en fonction de leur degré d'approximation. Ce degré est très important car il permet d'utiliser de façon optimale la liste obtenue au niveau applicatif. Si la localisation d'un des nœuds voisins du nœud client est elle-même issue de notre algorithme de localisation, c'est-à-dire qu'elle est représentée par une liste de couples (fichier de localisation, degré d'approximation), l'algorithme ne prend en compte que le premier de ses fichiers de localisation. Le degré d'approximation associé à cette localisation est égal à la somme du degré d'approximation issu de la liste et du coefficient calculé par l'algorithme (à partir des variables m_i, d_i et r_i).
Dans de rares cas, quand le noeud n'est connecté à personne, l'algorithme d'évaluation de la localisation retourne une liste vide; la localisation du noeud client ne peut pas être évaluée. Cependant, étant donné que la requête ne peut alors être évaluée que sur les données stockées sur le terminal du client, la limite de notre solution n'a pas beaucoup d'impact dans ce cas. En revanche, notre solution présente quelques limites si le nœud client est connecté à différents nœuds mais que ces nœuds ne connaissent pas leurs localisations respectives. Dans ce cas, une solution interactive peut être proposée pour permettre à l'utilisateur d'entrer sa localisation.

2.2.3. Exemple

Pour illustrer l'algorithme d'estimation de la localisation du client, un exemple est décrit ci-dessous. L'environnement du noeud client à l'exécution de l'algorithme est présenté dans le tableau 4.1 qui décrit l'ensemble des métadonnées des noeuds légers et centraux voisins du noeud client. Les coefficients a et b permettant de définir les priorités à considérer (nœuds centraux vs. nœuds légers) sont paramétrés à 1 pour ne privilégier aucun des 2 types de noeuds : centraux et légers. Nb est paramétré à 3 : au maximum, trois couples (fichier de localisation, degré d'approximation) seront traités et stockés dans la liste finale représentant la localisation du client. La date à laquelle l'évaluation de la localisation est effectuée, est le

14 août 2004 à 14h43. Le coefficient *connectionRange* est estimé en mètres : il représente le nombre de mètre approximatif séparant deux nœuds en fonction de la qualité du signal. Le coefficient *mobilityProfile* représente la vitesse moyenne de l'utilisateur estimée en mètres par minute.

Noeuds légers voisins	*Noeuds centraux voisins*
ID = NL1 locationDescription = «NL1.xml » locationLastUpdate = 14-08-2004 13h00 connectionRange = 2 mobilityProfile = 10 connectionState = faux	ID = NC1 locationDescription = «NC1.xml » locationLastUpdate = sysdate connectionRange = 60 mobilityProfile = 0 connectionState = vrai
ID = NL2 locationDescription = «NL2.xml » locationLastUpdate = 14-08-2004 14h33 connectionRange = 3 mobilityProfile = 10 connectionState = vrai	ID = NC2 locationDescription = «NC2.xml » locationLastUpdate = sysdate connectionRange = 90 mobilityProfile = 0 connectionState = faux
ID = NL3 locationDescription = «NL3.xml » locationLastUpdate = 14-08-2004 14h16 connectionRange = 30 mobilityProfile = 10 connectionState = vrai	ID = NC3 locationDescription = «NC3.xml » locationLastUpdate = sysdate connectionRange = 30 mobilityProfile = 0 connectionState = vrai

Tableau 4.1. Exemple de métadonnées de l'environnement

Fonction Evaluation

Nb := 3 ;

//Dans un premier temps, l'algorithme récupère l'ensemble des voisins référencés par le noeud client.

RefPeerList = {NL1, NL2, NL3, NC1, NC2, NC3} ;

//Ensuite, parmi la liste précédente, l'ensemble des noeuds connectés est récupéré.

ConnectedPeerList = {NL2, NL3, NC1, NC3} ;

nbConnectedPeers := 4 ;

//L'ensemble des variables d, r et m est défini (d représente la différence en minutes entre la date actuelle et l'attribut LocationLastUpdate) :

d = [10,27,0,0] ; r = [3,30,60,30] ; m = [10,10,0,0] ;

*//Pour chaque noeud voisin connecté, un degré d'approximation est calculé (a*r[i] pour un noeud central ou m[i]*d[i] + b*r[i] pour un noeud léger).*

locationList = {(NL2.xml,103), (NL3.xml,300), (NC1.xml,60),(NC3.xml,30)} ;

//Ensuite, la liste est triée en fonction des degrés d'approximation pour obtenir le résultat final.

LocationList = {(NC1.xml,30), (NC3.xml,60), (NL2.xml,103)} ;

La solution de localisation présentée ci-dessus permet de fournir la localisation d'un utilisateur mobile en fonction de son environnement. La localisation obtenue est représentée par une liste de couples (fichier de localisation XML, degré d'approximation) triés en fonction des degrés d'approximation. Les fichiers de localisation XML contiennent une représentation physique de la localisation du client et/ou une représentation symbolique. Le degré d'approximation permet quant à lui d'évaluer le fichier de localisation : plus le degré est petit, plus la localisation décrite par le fichier est exacte. Il est ensuite important d'évaluer la précision de la localisation globale obtenue. En effet, notre solution fournit une liste de localisations et il est intéressant d'évaluer la précision de la localisation fournie par cette liste.

3. Evaluation de la solution d'estimation de la localisation

Le principal objectif de notre algorithme est de fournir une localisation à tout utilisateur mobile sans requérir d'infrastructures matérielle et logicielle préalables. Cette localisation a ensuite pour but d'être utilisée et gérée facilement au sein d'une application. Les localisations estimées grâce à cet algorithme ne sont pas exactes mais la précision moyenne a dû être évaluée afin, dans un premier temps, de valider notre approche et dans un deuxième temps, d'identifier la marge d'erreur de notre algorithme et de permettre aux applications d'utiliser les localisations estimées de façon optimale. Pour évaluer la précision des localisations estimées par notre algorithme, différentes expérimentations ont été effectuées : dans la suite, nous décrivons le simulateur utilisé pour effectuer les tests et présentons l'ensemble des résultats obtenus.

3.1. Description des simulations

Pour évaluer la précision de la localisation calculée par notre algorithme, différentes simulations ont été effectuées. L'objectif est de simuler le calcul de localisation d'un client mobile et d'obtenir la précision de cette localisation en fonction de la position exacte du client mobile.

Pour ces simulations, nous avons modélisé un environnement constitué d'un bâtiment et d'un périmètre externe autour de ce bâtiment représentant par exemple, une gare et le parking et les voies ferrées qui l'entourent. Dans cet environnement, un certain nombre de clients mobiles sont placés aléatoirement avec une vitesse qui leur est également associée aléatoirement (de 0 à 5 mètres par seconde). Nous avons fait varier le nombre de clients mobiles au cours des simulations afin d'obtenir un spectre étendu de résultats et de tests. L'ensemble des clients mobiles (ou nœuds légers) représentés dans l'environnement a une localisation à l'instant où un des clients souhaite se localiser. Cependant ces localisations peuvent être obsolètes : pour chaque nœud mobile, l'attribut *locationLastUpdate* est défini aléatoirement (i.e. la fraîcheur de la localisation est fixée aléatoirement dans l'intervalle de 0 à 10 minutes).

Pour chaque client mobile, l'algorithme est exécuté afin d'obtenir, pour chaque client mobile, sa localisation représentée par une liste de couples {(localisation.xml, degré d'approximation)}. L'algorithme débute par le calcul de l'attribut *ConnectionRange*. L'attribut *ConnectionRange* basé sur l'intensité du signal entre le client et un de ses voisins est calculé en fonction de la distance entre le client et le voisin et des différents obstacles qui les séparent. En effet, la modélisation de l'environnement comprend la modélisation des obstacles de l'environnement tels que les murs et pour chaque type d'obstacles un coefficient d'atténuation lui est associé. Ces différents coefficients d'atténuation ont été paramétrés grâce à différentes expérimentations effectuées avec deux ordinateurs portables munis de carte WiFi et communicant en mode ad hoc. Le degré d'approximation peut ensuite être calculé pour chaque localisation d'un nœud voisin. La liste de couples (localisation, degré d'approximation) est alors obtenue. Ensuite à chaque fichier de localisation XML de la liste représentant la localisation du client mobile, un poids est associé. Ce poids équivaut à l'inverse du degré d'approximation de chaque fichier de localisation XML. Pour évaluer la précision de la localisation calculée par l'algorithme (liste de couples), la distance entre la position exacte du client mobile et le barycentre pondéré des différentes localisations des fichiers XML compris dans la liste de couples est calculée.

Soit L la localisation du client mobile issue de l'algorithme :

$L = \{(loc_1.xml, coeff\_Approx_1) ; (loc_2.xml, coeff\_Approx_2) ; ... ; (loc_n.xml, coeff\_Approx_n) \}$

De cette liste, la liste L' composée des fichiers de localisation XML et des poids associés est déduite : $L' = \{(loc_1.xml, poids_1) ; (loc_2.xml, poids_2) ; \ldots ; (loc_n.xml, poids_n)\}$. Chaque localisation XML est décrite à l'aide d'une latitude, d'une longitude et d'une altitude.

Donc, pour le calcul du barycentre pondéré, nous calculons ses coordonnées (latitude, longitude et altitude) comme suit :

$$lat_{barycentre} = \frac{\sum poids_i * lat_i}{\sum poids_i}$$ pour i allant de 1 au nombre d'éléments dans la liste L.

Donc, nous obtenons autant de barycentres pondérés que d'éléments dans la liste L. Pour chaque barycentre, nous calculons la distance entre la position exacte du client mobile et la position du barycentre.

3.2. Expérimentations : résultats et synthèse

La surface totale de l'environnement choisi est d'environ 7000 m². Le nombre total de nœuds dans l'environnement varie de 10 à 500 et le nombre de nœuds centraux varie quant à lui de 5 à 20. Nous avons choisi de limiter le nombre de nœuds à 500 : ce nombre représente le nombre de nœuds centraux et de nœuds légers localisés présents. Le nombre réel de nœuds légers présents dans l'environnement peut cependant être plus important mais les nœuds légers non localisés n'interviennent pas dans l'algorithme d'estimation de la localisation et ne sont donc pas représentés dans les simulations. Les différentes configurations obtenues permettent de représenter divers environnements : des environnements denses avec un grand nombre de participants et des environnements épars avec un nombre relativement bas de

participants (qui peuvent représenter les états de la gare à différents moments de la journée) ; des environnements bien connectés avec une seule sphère de communication qui permet une communication entre tous les participants et des environnements faiblement connectés où différentes sphères de communication coexistent et où tous les participants ne sont pas capables de communiquer entre eux. Pour chaque environnement, le placement des nœuds est réalisé aléatoirement. Notre solution de localisation est testée par tous les nœuds légers de l'environnement pour cinq configurations différentes de placement aléatoire des noeuds.

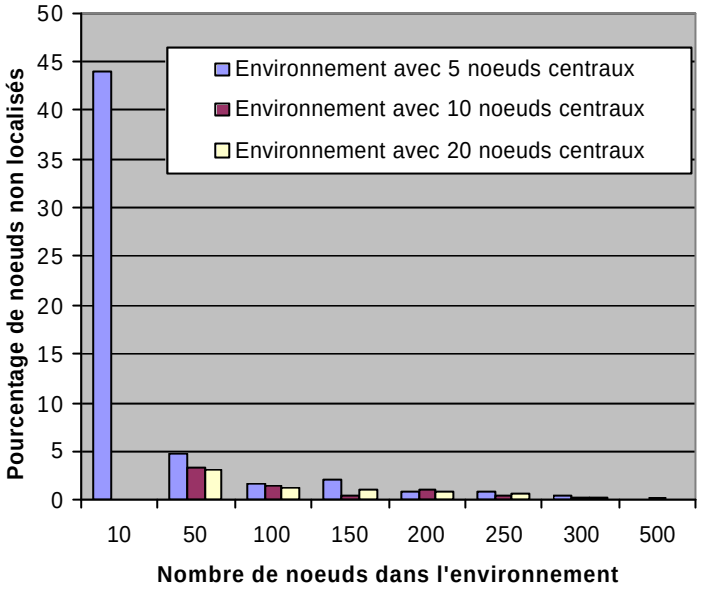


Figure 4.3. Evolution du pourcentage de noeuds non localisés en fonction du nombre de noeuds présents dans l'environnement

Dans un premier temps, nous avons mis en évidence que dans un environnement peu connecté, la localisation d'un client basée sur notre algorithme ne peut pas toujours être calculée. En effet, dans un environnement faiblement connecté (i.e. 10 nœuds pour 7000 m²), les clients ont moins de chances d'être connectés à d'autres nœuds. Comme l'illustre la figure 4.3, le pourcentage de noeuds non localisés dans un environnement de 10 noeuds est de plus de 40%. En effet, s'il y a 10 nœuds sur une surface de 7000 m², la probabilité qu'un nœud n'ait aucun voisin est importante. Cependant cette limite est à relativiser : si la localisation d'un utilisateur mobile ne peut pas être estimée à cause d'un environnement faiblement connecté, les requêtes émises par l'utilisateur risquent elles aussi de ne pas pouvoir être évaluées. En effet, dans nos simulations nous n'avons pas représenté l'ensemble des nœuds présents mais s'il existe seulement 10 nœuds localisés, le nombre total de nœuds a de fortes probabilités d'être également faible. Dans ce cas, si un utilisateur émet une requête, la requête risque donc de ne pas pouvoir être distribuée et évaluée sur d'autres nœuds. Le pourcentage de nœuds non localisés devient faible (inférieur à 5%) dès lors que le nombre de nœuds localisés est supérieur à 50 noeuds et il devient négligeable à partir de 200 nœuds localisés.

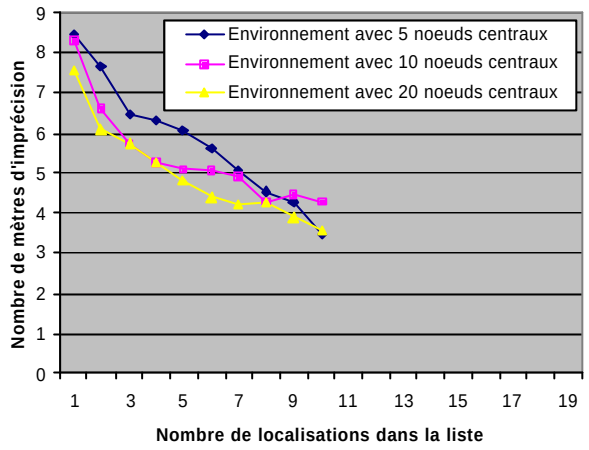


Figure 4.4.a. Environnement avec 50 noeuds

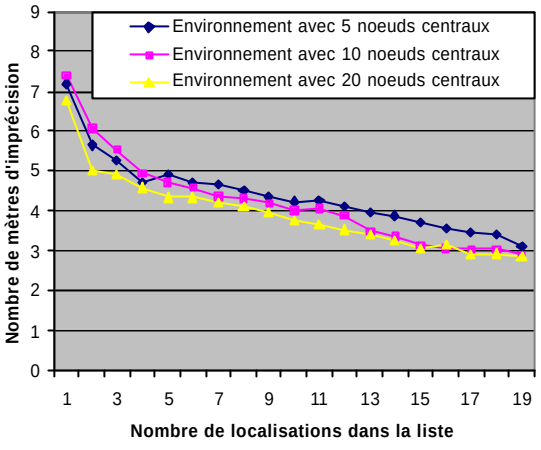


Figure 4.4.b. Environnement avec 100 noeuds

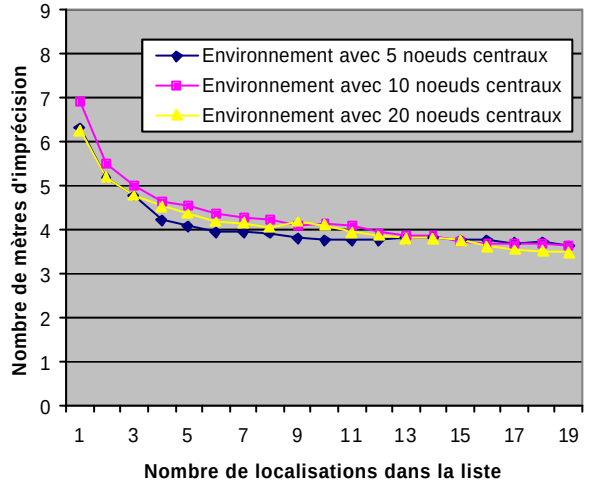


Figure 4.4.c. Environnement avec 200 noeuds

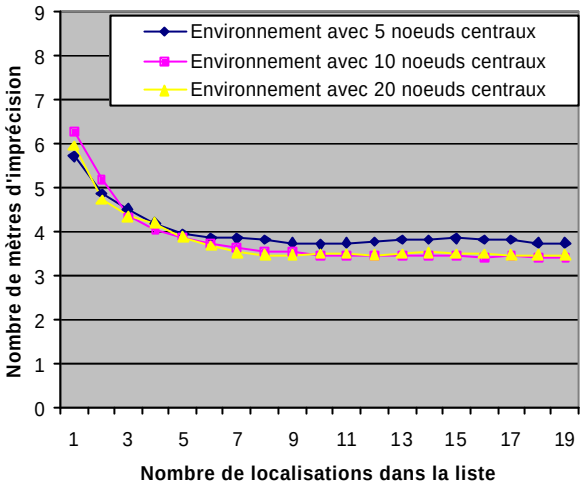


Figure 4.4.d. Environnement avec 300 noeuds

Les figures 4.4 a-b-c-d mettent en évidence que la précision des localisations calculées augmente en fonction du nombre de nœuds présents dans l'environnement. D'après ces figures, nous pouvons également constater que le nombre de noeuds centraux présents dans l'environnement influence le pourcentage de nœuds localisés dans les environnements où le nombre de nœuds est petit (environnements de 50 nœuds et de 100 nœuds). En revanche, dans des environnements bien connectés, l'influence du nombre de nœuds centraux devient moins importante.

De plus, nous pouvons également remarquer que la précision de la localisation augmente en fonction du nombre de couples (fichier de localisation, coefficient d'approximation) récupérés dans la liste de localisation. Néanmoins, nous nous apercevons qu'au-delà de 7 couples, la précision a tendance à se stabiliser. Cette atténuation est aussi illustrée par la figure 4.5.

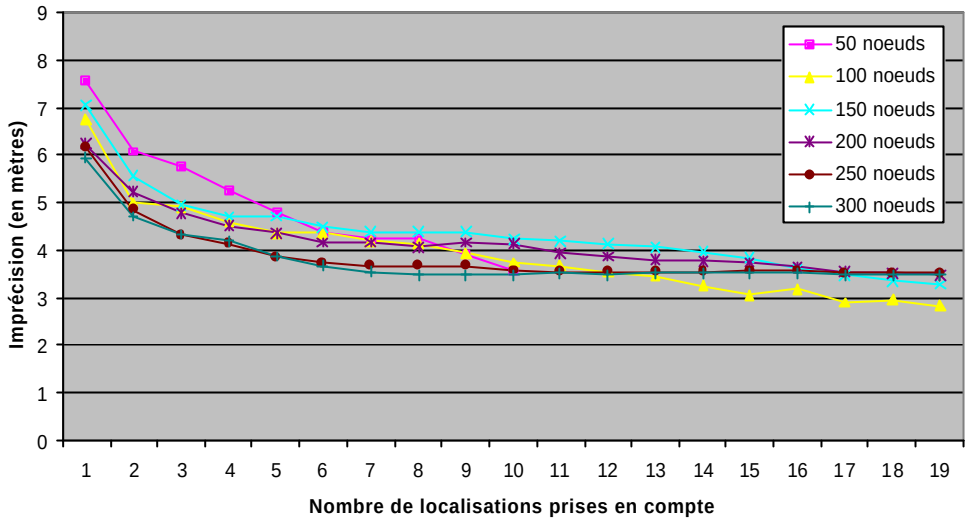


Figure 4.5. Evolution de l'imprécision dans un e nvironnement avec 20 nœuds centraux

Dans nos expérimentations, nous n'avons pris en compte que les nœuds localisés de notre environnement et les localisations de ces nœuds sont considérés comme exactes. Nous avons donc effectué des tests supplémentaires afin de mesurer l'impact de l'utilisation de localisations approximatives telles que celles issues de notre solution. Comme cela est illustré dans la figure 4.6, la localisation d'un utilisateur est légèrement moins précise lorsque certains nœuds sont localisés grâce à notre solution. Cependant ces résultats restent toutefois satisfaisants.

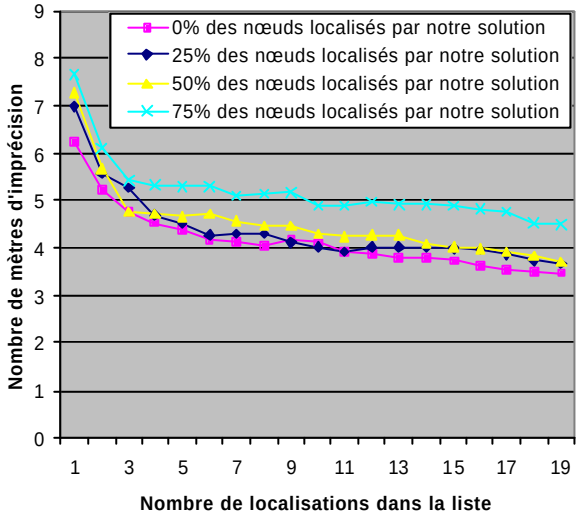


Figure 4.6. Impact de l'utilisation de nœuds localisés par notre solution dans un environnement de 200 nœuds avec 20 nœuds centraux.

3.3. Conclusion

Dans le tableau 4.2, nous avons choisi de comparer notre solution de localisation au GPS selon différents critères de la taxonomie de Hightower et al. [HIG 01b]. En effet, le GPS représente la solution de localisation existante la plus proche de notre solution : elle n'est pas

dédiée à une application particulière et donc ne nécessite pas d'infrastructure lourde et a pour objectif d'être utilisée n'importe où.

	GPS	*Notre solution de localisation*
Représentation de la localisation	- physique - absolue	- physique et/ou symbolique - absolue
Méthode utilisée	Triangulation	Proximité & atténuation du signal
Précision	Quelques mètres	Quelques mètres
Limites	- A l'intérieur d'un bâtiment - Nécessite un module GPS sur le terminal	- Nécessite des voisins localisés connectés

Tableau 4.2. Comparaison de notre solution avec le GPS selon la taxonomie de Hightower et al.

Concernant la représentation de la localisation, notre solution a l'avantage de pouvoir fournir une représentation symbolique de la localisation, ce qui permet l'évaluation des requêtes de localisation utilisant l'opérateur *inside* comme nous l'expliquons dans le chapitre 5. La précision des localisations obtenues en utilisant notre algorithme est aussi satisfaisante que celle obtenue par le GPS et suffisante pour permettre une évaluation satisfaisante des requêtes dépendantes de la localisation. Finalement notre solution présente une limite si aucun voisin localisé n'est accessible. Cependant, comme nous l'avons expliqué précédemment cette limite peut être relativiser puisque dans ce cas, l'environnement du client mobile est certainement très faiblement connecté et donc, les requêtes émises par le client ne peuvent pas non plus être évaluées.

4. Optimisations de l'algorithme d'estimation de la localisation

4.1. Problématique

Dans la section précédente, nous avons démontré que notre solution d'estimation de la localisation d'un utilisateur mobile fournissait des localisations approximatives dont la précision est satisfaisante. Nous avons donc implémenté cette solution de positionnement dans notre prototype qui sera présenté de façon détaillée dans le chapitre 6. Cette implémentation nous a permis de mettre en évidence que certaines contraintes autres que celle de la précision, sont à considérer pour notre solution de localisation. En effet, la solution a été implémentée, dans un premier temps, de façon naïve :

- Le noeud client diffuse une demande à l'ensemble de la sphère de communication.
- Chaque noeud voisin connecté lui renvoie ses métadonnées disponibles: son type, son fichier de localisation, la date de dernière mise à jour de sa localisation et son profil de mobilité.
- Le noeud client récupère localement l'ensemble de ses informations.
- Il récupère également le niveau du signal réseau pour chaque voisin auquel il est connecté.
- Il peut alors exécuter localement la fonction évaluation pour obtenir sa localisation.

En utilisant cette stratégie naïve, les temps d'exécution et la consommation d'énergie liés à l'estimation de la localisation sont très importants. Les contraintes d'autonomie et de temps de connexion sont prépondérantes pour optimiser l'utilisation de notre approche sur des terminaux nomades. Nous avons donc proposé deux stratégies d'optimisation ayant pour objectif de minimiser les ressources utilisées pour estimer la localisation d'un utilisateur mobile tout en garantissant une localisation satisfaisante de l'utilisateur. Ces stratégies ont pour objectif de minimiser les coûts associés à l'évaluation de la localisation d'un utilisateur mobile et sont choisies en fonction du comportement et de l'environnement du client.

4.2. Stratégies d'optimisation

4.2.1. Stratégie des noeuds voisins

Dans la première stratégie, l'objectif est de déléguer une partie du traitement effectué a priori sur le nœud client, sur les nœuds voisins. En effet, le nœud client peut être entouré de nœuds voisins disposant de ressources plus importantes tels que des nœuds centraux par exemple. Cependant, il est également important de ne pas inonder les nœuds voisins de traitements diverses et chaque nœud voisin a donc la possibilité de refuser d'effectuer le traitement demandé par le nœud client. Cette première stratégie, appelée *stratégie des nœuds voisins*, se déroule comme suit :

- Premièrement le client diffuse un message sur la sphère de communication pour permettre à ses nœuds voisins de l'identifier et leur demander de lui renvoyer le degré d'approximation correspondant à leur fichier de localisation. Si le nœud voisin ne possède pas son fichier de localisation, dans ce cas, le message est ignoré par le nœud voisin.
- Comme précédemment ce degré d'approximation est calculé à partir de la fraîcheur du fichier de localisation, de la qualité de la connexion séparant le nœud client du nœud voisin et du profil de mobilité du nœud voisin. Chaque noeud voisin calcule donc le degré d'approximation de son fichier de localisation, qui peut être assimilé à la « distance » qui le sépare du noeud client.
- Chaque noeud voisin envoie ce degré d'approximation au noeud client.
- Le noeud client sélectionne ensuite les noeuds voisins en fonction de leurs degrés d'approximation respectifs: il sélectionne ses voisins les plus proches. En exploitant les résultats obtenus dans la section précédente, si le nœud client a plus de sept voisins, il peut ne sélectionner que les sept plus proches voisins : en effet, au-delà de sept fichiers de localisation, la précision se stabilise. Si le nœud client a moins de sept voisins, dans ce cas, il sélectionnera l'ensemble de ses voisins.
- Il peut alors demander les fichiers de localisation aux noeuds voisins sélectionnés.
- Les noeuds voisins envoient leurs fichiers de localisation au noeud client.
- Le noeud client associe finalement les fichiers de localisation aux coefficients obtenus auparavant pour obtenir la liste de couple (fichier de localisation, degré d'approximation) représentant sa localisation.

Cette stratégie fournit différents avantages par rapport à la stratégie naïve décrite précédemment. Premièrement, le calcul du degré d'approximation est effectué par les nœuds voisins du nœud client : le nœud client n'a donc plus, comme dans la stratégie naïve, à calculer l'ensemble des degrés d'approximation pour l'ensemble de ses voisins. Ce calcul ne représente pas un important surplus de charge de travail pour les nœuds voisins, en particulier pour les nœuds voisins centraux, étant donné que les attributs utiles pour le calcul de ce degré sont stockés localement et que chaque nœud voisin doit calculer un seul degré. Finalement, le nœud client n'a plus à traiter, contrairement à la stratégie naïve, l'ensemble des fichiers de localisation de ces nœuds voisins mais uniquement, au maximum, les sept meilleures localisations.

4.2.2. Stratégie seuil de sélection

Dans la seconde stratégie, appelée *« stratégie seuil de sélection »*, l'objectif est d'utiliser un seuil de sélection pour limiter le nombre de nœuds voisins sélectionnés par le nœud client pour calculer sa localisation. Cette stratégie est composée de deux étapes distinctes. Ces deux étapes peuvent être exécutées séparément. La première étape est réalisée régulièrement et permet de maintenir un seuil de sélection à jour. Elle est effectuée à chaque fois que le nœud effectue une mise à jour de son environnement. Nous avons vu précédemment, que ces mises à jour permettaient à chaque nœud de garder une connaissance de leur environnement. Elles sont effectuées à intervalles réguliers, définis en fonction du profil de mobilité du nœud.

La première étape de la stratégie « seuil de sélection » se déroule comme suit :

- Dans un premier temps, le noeud client diffuse un message sur la sphère de communication pour permettre à ses nœuds voisins de l'identifier.
- Chaque noeud voisin connecté lui renvoie alors son type (léger ou central) et le type de réseau utilisé (WiFi par exemple) pour la connexion entre le nœud client et le nœud voisin.
- En fonction du nombre de noeuds voisins (équivalent au nombre de réponses reçues), de leurs types et du type de réseau utilisé entre chaque noeud distant et le noeud client, celui-ci calcule un seuil de sélection. Ce seuil de sélection va être calculé pour permettre dans la deuxième étape de la stratégie de ne sélectionner que les meilleurs nb_{noeuds} nœuds voisins. Le nombre nb_{noeuds} de nœuds voisins qu'on souhaite sélectionner est paramétrable dans le calcul du seuil expliqué ci-dessous. Précédemment, nous avons constaté que si $nb_{noeuds} = 7$, la précision de la localisation obtenue sera quasiment optimale.

Calcul du seuil de sélection

Nous présentons dans ce paragraphe comment le nœud client calcule le seuil de sélection. En fonction du nombre de noeuds dont il souhaite récupérer la localisation (i.e. 7 nœuds par exemple), le noeud client peut calculer un seuil de sélection qui est égal à la «distance »

maximale autorisée entre le noeud client et un noeud voisin pour sélectionner le fichier de localisation d'un noeud voisin.
Le calcul de ce seuil de sélection est basé sur les densités moyennes de noeuds voisins autour de lui calculées en fonction des portées maximales moyennes des types de réseaux utilisés. En effet, le noeud client connaît pour chaque noeud voisin, le type de réseau utilisé entre eux pour communiquer. Il peut alors définir le nombre de noeuds voisins accessibles pour chaque type de réseaux disponibles et en déduire une densité. Le nœud client peut alors, en fonction de ces densités, estimer une distance représentant le rayon d'une sphère dans laquelle il y aura au moins nb_{noeuds} de nœuds voisins.
Dans la suite, nous illustrons le calcul du seuil en nous appuyant sur un exemple d'environnement où un seul type de réseau, tel que WiFi, est utilisé. Le nœud voisin est donc entouré d'un ensemble de nœuds voisins accessibles par un réseau R. La portée moyenne maximale (en mètres) de ce réseau est notée P. Les nœuds voisins appartiennent à la sphère S ayant comme centre le nœud client, comme rayon r (égal à P) et comme volume $V = \frac{4}{3} * p * r^3$. Soit n le nombre de nœuds contenus dans la sphère S sans compter le nœud client.
Soit nb_{noeuds} le nombre de fichiers de localisation de noeuds voisins souhaités pour le calcul de la localisation du noeud client.
Si $nb_{peers} \leq n$, le volume nécessaire et suffisant pour communiquer avec nb_{noeuds} sera $V_{suff} = \frac{V * nb_{peers}}{n}$ et donc $seuil = \left[\frac{3 * V_{suff}}{4 * p} \right]^{\frac{1}{3}}$.
Sinon, $seuil = r$.

Nous avons décrit le calcul du seuil de sélection dans un environnement où un seul type de réseau est présent, étendons maintenant ce calcul à un environnement composé de différents types de réseaux. Nous illustrons cette fois, le calcul en nous basant sur un exemple d'environnement décrit dans la figure 4.7. Dans cet environnement, le noeud client est entouré de noeuds qu'il peut atteindre grâce à deux types de réseaux R_1 et R_2.
Notons :

- P_1 et P_2, les portées moyennes maximales (en mètres) de R_1 et R_2, respectives.
- S_1 et S_2, les sphères représentant R_1 et R_2. Ces sphères ont comme centre le nœud client et comme rayon, respectivement, P_1 et P_2.
- V_1 et V_2, les volumes des sphères S_1 et S_2 où $V_1 = \frac{4}{3} * p * {P_1}^3$ et $V_2 = \frac{4}{3} * p * {P_2}^3$.
- n_1 et n_2, le nombre de noeuds respectivement contenus dans S_1 et S_2. Si on considère que S_1 est la plus petite des deux sphères comme dans la figure 4.7, n_2' représente une estimation du nombre de noeuds voisins dans la sphère S_1 communiquant grâce au réseau R_2 et donc $n_2' = \frac{(n_2 * V_1)}{V_2}$.

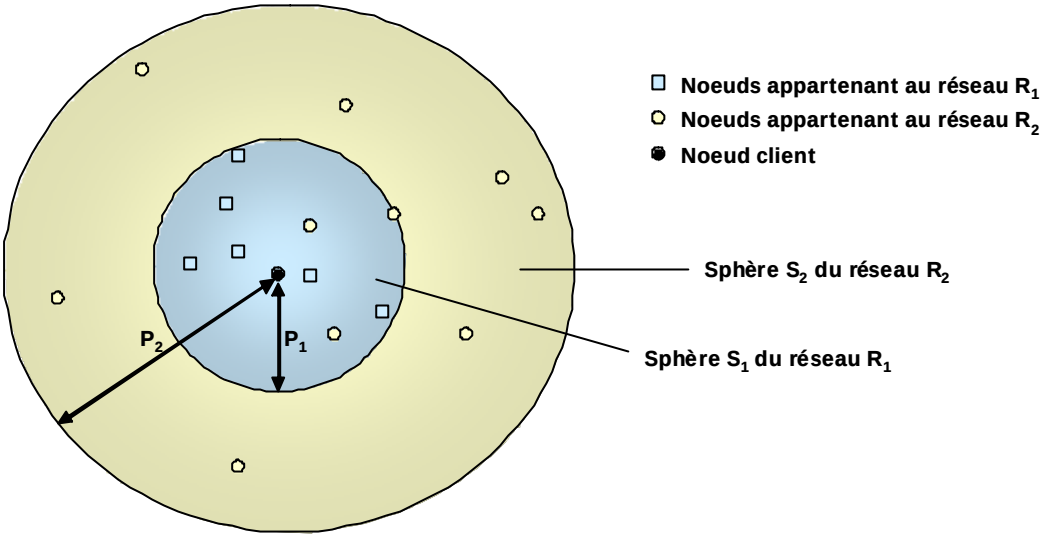


Figure 4.7. Exemple d'environnement d'un noeud

Si $nb_{peers} \leq n_1 + n_2'$, le volume nécessaire et suffisant pour communiquer avec nb_{noeuds} sera $V_{suff} = \frac{V_1 * nb_{peers}}{n_1 + n_2'}$ et donc $seuil = \left[\frac{3 * V_{suff}}{4 * p}\right]^{\frac{1}{3}}$.

Si $nb_{peers} > n_1 + n_2'$, le volume nécessaire et suffisant pour communiquer avec nb_{noeuds} sera $V_{suff} = V_1 + \frac{\left(nb_{peers} - (n_1 + n_2')\right) * (v_2 - v_1)}{n_2 - n_2'}$ et donc $seuil = \left[\frac{3 * V_{suff}}{4 * p}\right]^{\frac{1}{3}}$.

Sinon, $seuil = R_2$.

Dans le calcul du seuil de sélection, nous utilisons également les types des noeuds voisins: central ou léger afin d'affiner la sélection par la suite. Chaque réponse des voisins contient le type du noeud voisin. Si le noeud client a de nombreux nœuds voisins qui sont des nœuds centraux (au moins supérieur à 7), le noeud client pourra alors baser le calcul du seuil de sélection uniquement sur le nombre de noeuds centraux voisins et ignorer les nœuds légers voisins qui offrent généralement des localisations moins précises et/ou moins à jour.

La deuxième étape de la stratégie « seuil de sélection » est exécutée au moment précis où le noeud client a besoin d'estimer sa localisation :

- Le noeud client diffuse le dernier seuil de sélection calculé sur la sphère de communication.
- Chaque noeud voisin connecté va alors calculer le degré d'approximation correspondant à sa localisation. Ce degré représente une estimation de la distance qui le sépare du noeud client.
- Si ce degré d'approximation est inférieur au seuil de sélection, le noeud voisin renvoie son fichier de localisation ainsi que le degré d'approximation calculée.

- Le noeud client reçoit ainsi l'ensemble des couples (fichier de localisation, coefficient d'approximation) qui vont constituer sa propre localisation.

Les avantages de cette stratégie sont tout d'abord dus à son découpage en deux étapes. En effet, la première étape est réalisée régulièrement et n'influe donc pas sur les temps d'exécution de l'évaluation d'une requête de localisation. De plus, comme dans la stratégie des nœuds voisins, le nœud client n'a pas à calculer l'ensemble des degrés d'approximation. Finalement, le calcul du seuil permet de ne sélectionner que les meilleurs nœuds voisins et le nœud client ne reçoit et ne traite donc qu'un minimum de fichiers de localisation.

4.3. Résultats et synthèse

Nous présentons dans cette section les résultats obtenus pour les deux stratégies d'optimisation présentées ci-dessus. En guise de référence, nous avons calculé les résultats obtenus dans le cas de la stratégie naïve de l'algorithme de localisation proposé.

Les différentes stratégies ont été testées dans différents environnements et plus particulièrement en faisant varier le nombre de nœuds voisins connectés au nœud client. Les résultats de ces différents tests sont exprimés en temps de réponse. Les temps de réponse sont basés sur des mesures effectuées préalablement concernant les temps de communication et les temps de lecture et d'écriture de fichiers. Pour ces mesures, les noeuds légers étaient représentés par des iPAQ 3950, les noeuds centraux par des ordinateurs portables PIII et le type de réseau utilisé était un réseau 802.11b. Comme cela est illustré dans la figure 4.8, nous avons constaté, dans un premier temps, que les temps de transfert obtenus ne dépendent pratiquement pas de la distance séparant les deux appareils. Ceci s'explique par le fait que les débits réels obtenus entre deux terminaux nomades sont beaucoup plus faibles que les débits réels annoncés et donc, la diminution du débit est quasiment nulle. En revanche, nous avons remarqué l'impact des ressources sous-jacentes des terminaux sur les temps de transfert et la consommation d'énergie. En effet, les débits obtenus entre un iPAQ et un ordinateur portable ou entre deux iPAQs sont bien moins élevés que ceux obtenus entre deux ordinateurs portables. Les débits dépendent du terminal nomade utilisé et de son système d'exploitation [HEU 03].

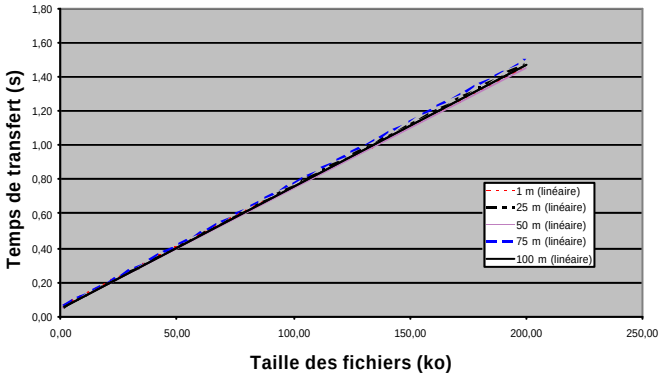


Figure 4.8. Temps de transfert en fonction de la distance

Pour les temps de transfert, nous avons également remarqué que lorsque plusieurs messages sont reçus ou envoyés par un terminal nomade, le temps de transfert total est pratiquement

équivalent, que l'ensemble des messages soit traité en parallèle ou séparément (en série). En effet, les ressources des terminaux nomades ne leur permettent pas de gérer l'envoi de messages en parallèle. De ce fait, les coûts liés à l'envoi de plusieurs messages sont relativement élevés.

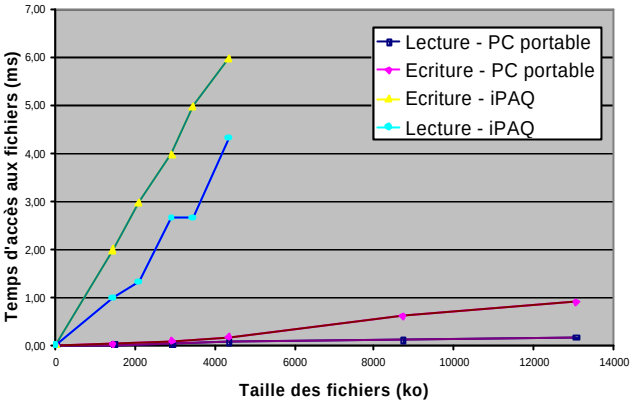


Figure 4.9. Temps d'accès en écriture et en lecture

D'autres tests concernant les temps d'accès en écriture et en lecture sur les terminaux nomades, ont été effectués et sont illustrés dans la figure 4.9. En effet, en moyenne les temps d'écriture obtenus sur un iPAQ 3950 sont de l'ordre de 5,5 Mbps (or 6,5 Mbps sur une SD card) et les temps de lecture sont de l'ordre de 10 Mbps.

Quant à la consommation d'énergie, elle représente également une contrainte importante pour évaluer ces stratégies. Cependant, les mesures expérimentales obtenues sur un iPAQ 3950 sont très imprécises et ne permettent pas de quantifier l'énergie consommée pour les différentes stratégies. En effet, les jauges de batterie sur l'iPAQ se décrémentent par palier de 10%, ce qui rend très difficile une estimation des opérations peu coûteuses. Cependant, certaines de nos expérimentations nous ont permis d'identifier une adéquation entre les temps obtenus pour exécuter une opération et la consommation d'énergie. Comme l'illustre la figure 4.10, nous pouvons remarquer que la consommation d'énergie croît linéairement en fonction du temps utilisé. Nous pourrons donc, en nous inspirant des temps de réponse obtenus pour chaque stratégie, déduire les effets en terme de consommation d'énergie.

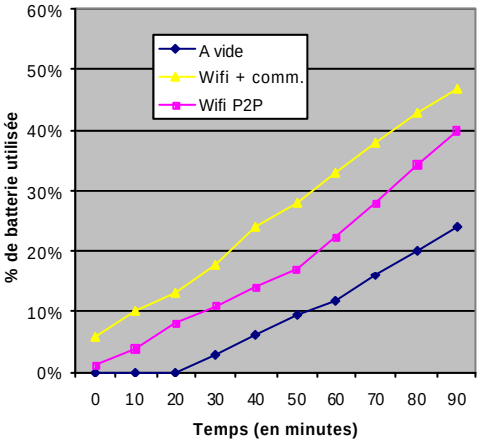


Figure 4.10. Evolution de la consommation d'énergie en fonction du temps

La figure 4.11 illustre les temps de réponses obtenus en utilisant la stratégie naïve de l'algorithme d'estimation de la localisation. Les colonnes foncées représentent les temps dédiés aux communications et les colonnes claires représentent les temps dédiés aux traitements effectués sur le noeud client. Nous remarquons que plus le nœud client a de voisins, plus le temps de réponse de l'algorithme est important et plus sa consommation d'énergie l'est également. De ce fait, dans un réseau fortement connecté, l'utilisation de notre solution initialement implémentée suivant cette stratégie surcharge rapidement à la fois le terminal nomade du nœud client, de traitements mais également le réseau, de communications.

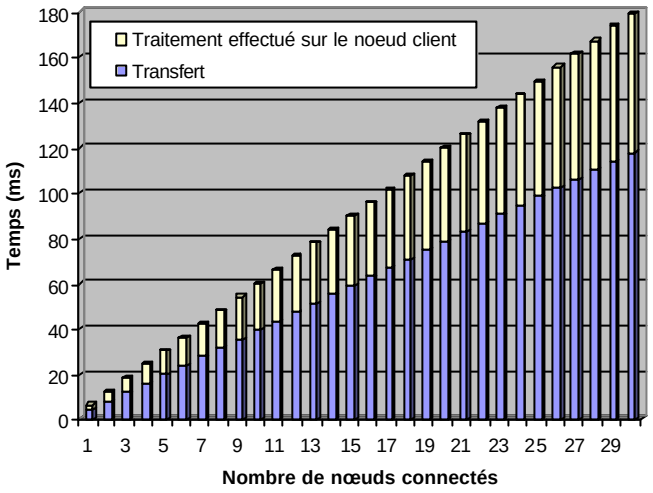


Figure 4.11. Stratégie naïve

Les figures 4.12 et 4.13 représentent respectivement les temps de réponses obtenus en utilisant la stratégie des noeuds voisins et la stratégie seuil de sélection. Pour l'exécution de ces deux stratégies, le nombre maximum de fichiers de localisation récupérés a été fixé à 7.
Dans les figures 4.12 et 4.13, les colonnes foncées représentent le temps utilisé pour l'ensemble des opérations effectuées sur les noeuds légers voisins: c'est le temps maximum obtenu si tous les noeuds connectés sont des noeuds légers et s'ils ont tous accepté d'effectuer les traitements demandés par le noeud client. En effet, les temps associés au traitement éventuel effectué sur les noeuds légers doivent intervenir dans nos mesures mais les noeuds légers sont bien entendu libres de refuser d'exécuter le travail localement. Les temps de traitement sur les noeuds centraux voisins n'apparaissent pas car ils sont négligeables comparés aux temps de communication et de traitement sur les noeuds légers. En effet, les expérimentations réalisées ont montré que les coûts en lecture et en écriture sur un PC portable Pentium III étaient en moyenne 50 fois moins importants que les coûts en lecture et en écriture sur un PDA.
La figure 4.13 présente les résultats obtenus pour la seconde étape de la stratégie du seuil de sélection. Nous ne comparons que la seconde étape étant donné que c'est la seule à être exécutée au moment de l'évaluation de la localisation. De plus, la première étape est exécutée par tous les nœuds pour connaître leur environnement. Seul le calcul du seuil de sélection est ajouté au temps de mise à jour de l'environnement.

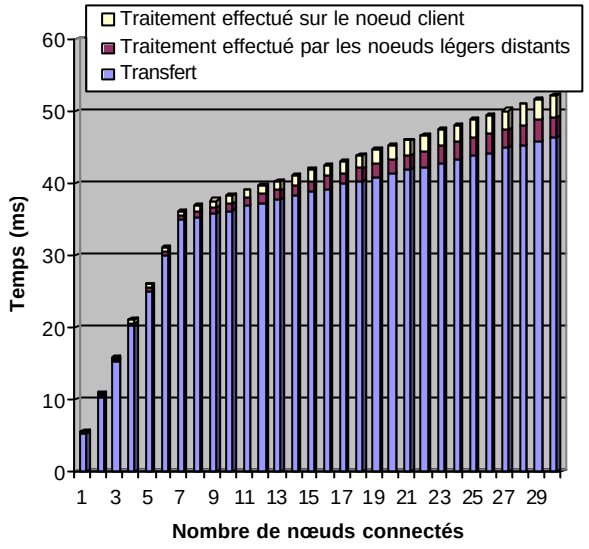


Figure 4.12. Stratégie des noeuds voisins

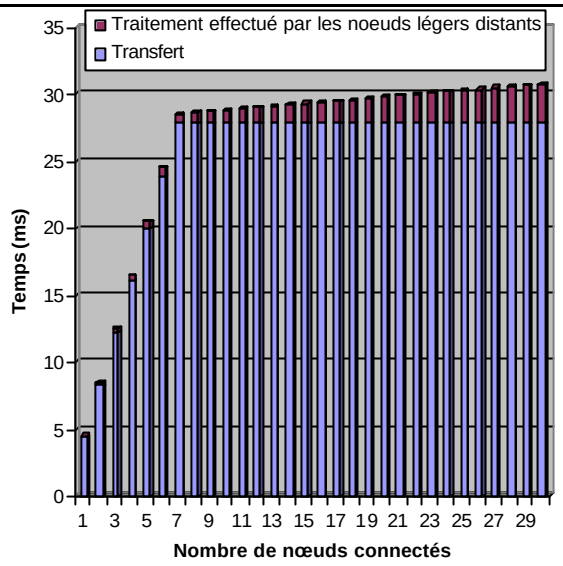


Figure 4.13. Stratégie seuil de sélection

Nos expérimentations montrent que les deux stratégies proposées permettent de réduire sensiblement les temps de réponse obtenus pour exécuter l'algorithme d'estimation de la localisation. Et surtout, elles mettent en évidence que les temps de transfert se stabilisent pour les deux stratégies. Et ce, même si les noeuds voisins sont nombreux. Cette stabilisation, dûe au paramétrage du nombre de localisations désirées, est très importante dans le cas d'environnements où le nombre de noeuds connectés est très important afin d'économiser la consommation de ressources du terminal client.

La stratégie seuil de sélection fournit une estimation de la localisation de l'utilisateur mobile en moins de temps que la stratégie des nœuds voisins. Cependant, le calcul régulier du seuil de sélection pour un nœud léger devient inutile si cet utilisateur n'a pas besoin de se localiser régulièrement et les ressources alors consommées sont perdues. De plus, si le client est extrêmement mobile, la première étape de la stratégie seuil de sélection doit être réalisée très souvent pour maintenir un seuil cohérent. De même, si l'environnement qui entoure le nœud client est essentiellement composé de nœuds très mobiles, le seuil devient rapidement obsolète même si la première étape est effectuée souvent.

Nous pouvons donc déduire que la stratégie seuil de sélection est surtout adaptée à un environnement où le nombre de nœuds légers très mobiles est relativement petit par rapport aux nœuds centraux et aux nœuds légers peu mobiles. De plus, cette stratégie est principalement dédiée aux nœuds clients qui émettent souvent des requêtes de localisation et qui ont donc besoin de se localiser régulièrement. En revanche, si le nœud client est très mobile et n'émet que rarement des requêtes de localisation, la stratégie des nœuds voisins est la plus appropriée. Il est donc important d'identifier le comportement de l'utilisateur et son environnement pour que le module de positionnement soit en mesure de sélectionner la stratégie la plus adaptée.

5. Conclusion

Le module de positionnement présenté dans ce chapitre permet d'utiliser les techniques de localisation existantes telles que le GPS si celles-ci sont disponibles sur le terminal de l'utilisateur mobile. Cependant, les techniques existantes ne permettent pas à tout utilisateur mobile de se localiser. Nous avons donc décrit dans ce chapitre une solution permettant l'estimation de la localisation d'un utilisateur mobile : par exemple à l'intérieur d'un bâtiment, le GPS ne fonctionne pas. Notre solution est basée sur l'environnement du nœud client souhaitant se localiser. Nous avons démontré que la précision de la localisation fournie par notre algorithme était suffisante pour un large nombre d'applications dédiées aux utilisateurs de terminaux nomades. Cependant, elle présente certaines limites dans un environnement peu connecté : quand le nœud client a peu, voire, aucun nœud voisin localisé. De plus, comme notre solution utilise les métadonnées issues des nœuds voisins et plus particulièrement de leurs fichiers de localisation, si ces fichiers de localisation ne sont décrits que dans une seule représentation (physique ou symbolique), la localisation du nœud client ne sera elle aussi décrite que d'une seule façon.
Cet algorithme ayant pour objectif d'être utilisé sur des terminaux nomades et reposant sur des réseaux sans fil, deux stratégies d'optimisation ont été proposées afin de minimiser à la fois la consommation des ressources du terminal et du réseau mais aussi le temps de réponse. En fonction du comportement de l'utilisateur mobile, le module de positionnement choisit la stratégie d'optimisation la plus adaptée.
La localisation de l'utilisateur mobile issue du module de positionnement peut ensuite être utilisée par d'autres applications comme le service de localisation pour l'évaluation des requêtes dépendantes de la localisation. Dans le chapitre suivant, nous présentons le mécanisme d'évaluation des requêtes de localisation.

Chapitre 5

Evaluation des requêtes de localisation

1. Introduction

Dans ce chapitre, nous nous intéressons à l'évaluation des requêtes dans ISLANDS et plus particulièrement à l'évaluation des requêtes de localisation. Comme nous l'avons décrit dans le chapitre 2, il existe deux catégories de requêtes de localisation :

- les requêtes relatives à une localisation qui est différente de celle du client (LAQ – Location Aware Query). Un exemple est «quel est l'arrêt de bus le plus proche de la gare ? ». Dans ce cas, la localisation de référence est la localisation de la gare.
- les requêtes dépendantes de la localisation du client (LDQ – Location Dependent Query). Un exemple est «quel est l'arrêt de bus le plus proche de moi ? ». La localisation de référence est alors la localisation du client.

Nous avons présenté dans le chapitre 3, les différents opérateurs de localisation permettant d'exprimer des requêtes de localisation : *close*, *closest* et *inside*. Ces différents opérateurs permettent d'exprimer des requêtes de localisation dans un langage d'interrogation donné. Une fois que le client a exprimé sa requête, celle-ci est prise en charge par le service de localisation ISLANDS pour être évaluée. Pour cela, le service prend en compte la distribution des informations sur les différents nœuds de l'application. En effet, la requête est envoyée à d'autres nœuds distants afin de calculer le résultat. Cependant, dans le contexte des applications de proximité, les échanges de messages (i.e. de requêtes et/ou de réponses) et les évaluations de la requête sur de nombreux nœuds sont coûteuses. Il est donc indispensable de proposer des stratégies d'optimisation visant à minimiser les temps de réponse et la consommation des ressources tout en garantissant une certaine qualité de réponse.

Le chapitre est organisé de la manière suivante : la section 2 présente les différentes opérations à effectuer pour l'évaluation des requêtes de localisation et décrit l'évaluation des différents opérateurs de localisation [THI 04d]. La section 3 est, quant à elle, dédiée aux optimisations proposées dans le cadre de l'évaluation des requêtes de localisation. Plusieurs stratégies sont proposées pour minimiser les coûts nécessaires à l'évaluation, en particulier en proposant des optimisations concernant la distribution de la requête.

2. Evaluation des requêtes de localisation

2.1. Principe d'évaluation des requêtes de localisation

Après avoir été exprimée, une requête de localisation peut alors être évaluée par le service de localisation ISLANDS. Comme nous l'avons décrit dans le chapitre 3, l'évaluation de ces requêtes est composée de trois étapes décrites dans la figure 5.1 :

- Etape A : l'évaluation de la localisation de référence
- Etape B : l'évaluation de la requête classique : non dépendante de la localisation
- Etape C : l'évaluation de l'opérateur de localisation

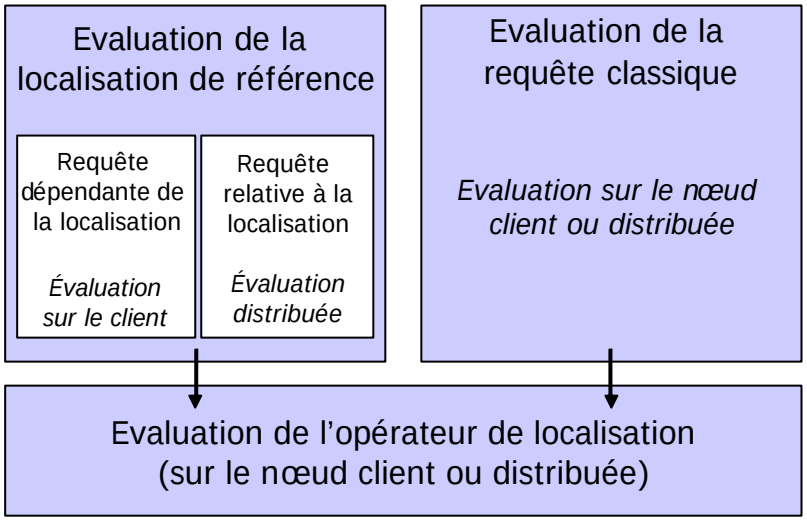


Figure 5.1. Etapes de l'évaluation d'une requête dépendante de la localisation

Illustrons ces trois étapes avec un exemple de requête dépendante de la localisation (LDQ) : « quel est l'arrêt de bus le plus proche de moi ? » et respectivement un exemple de requête relative à la localisation (LAQ) « quel est l'arrêt de bus le plus proche de la gare ? ».

- Etape A : l'évaluation de la localisation de référence consiste à récupérer la localisation du client pour la LDQ ou la localisation de la gare pour la LAQ. La localisation du client est évaluée localement grâce au module de positionnement : le résultat est alors soit un fichier de localisation XML, soit une liste de couples (fichier de localisation XML, degré d'approximation) si notre solution de localisation géographique présentée dans le chapitre 4 est utilisée. L'évaluation de la localisation de la gare correspond à l'évaluation d'une requête classique au sein d'ISLANDS. Elle est effectuée tout d'abord au sein de l'instance locale d'ISLANDS. Si aucune réponse n'est obtenue, la requête est alors distribuée et évaluée sur d'autres nœuds. Le résultat est un fichier XML représentant la localisation de la gare.
- Etape B : Dans notre exemple, l'évaluation de la requête classique (i.e. non dépendante de la localisation) consiste, quant à elle, à retrouver l'ensemble des arrêts de bus et leurs localisations respectives afin de permettre l'évaluation de l'opérateur de localisation. Cette requête est évaluée tout d'abord localement sur le client et est ensuite distribuée et évaluée sur d'autres nœuds de la sphère de communication.

- Etape C : L'évaluation de l'opérateur de localisation repose sur les résultats obtenus par l'évaluation de la localisation de référence et par l'évaluation de la requête classique. L'opérateur de localisation peut être évalué successivement sur chaque nœud ayant évalué la requête classique et disposant de la localisation de référence. Il peut aussi être uniquement évalué sur le nœud client lorsque celui-ci reçoit des réponses de la requête classique. Pour l'exemple, l'évaluation de l'opérateur de localisation consiste à sélectionner l'arrêt de bus dont la localisation est la plus proche de la localisation du client (pour la LDQ) ou de la gare (pour la LAQ).

L'étape A est gérée par le module de positionnement dans le cas d'une requête dépendante de la localisation. Sinon, elle est gérée par l'évaluateur de requêtes et prise en compte en tant que requête classique de la même façon que l'étape B c'est-à-dire que les requêtes «quels sont les arrêts de bus et leurs localisations ?» et « quelle est la localisation de la gare ? » sont traitées de la même façon. Finalement, l'étape C est gérée par le module d'évaluation de l'opérateur de localisation que nous présentons dans la section suivante.

2.2. Module d'évaluation de l'opérateur de localisation

Durant l'évaluation de l'opérateur de localisation, la localisation de référence est comparée avec les différentes localisations obtenues de la requête classique (i.e. les différentes localisations des arrêts de bus pour l'exemple) afin d'obtenir la réponse finale de la requête de localisation (i.e. l'arrêt de bus le plus proche de moi pour l'exemple).

Dans la suite du chapitre, nous emploierons le terme « *liste de localisations issues de la requête* » pour désigner la liste des localisations correspondant aux réponses obtenues pour l'évaluation de la requête classique (étape B) et le terme « *liste de localisations de référence* » pour désigner la liste des couples (fichier de localisation, degré d'approximation) représentant la localisation de référence issue de l'étape A. La liste de localisations de référence peut être constituée d'un seul couple (fichier de localisation, 0) si la requête est une requête relative à la localisation (LAQ) ou si la localisation renvoyée par le module de positionnement est exacte (dans le cas d'un nœud fixe par exemple).

Pour chaque application de proximité, nous avons choisi de définir une DTD (DTD – Document Type Definition) que chaque fichier de localisation doit suivre. Cette DTD correspond à l'environnement dans lequel se déroule l'application de proximité (par exemple bâtiment, étage, etc. ou numéro, rue, ville, etc.). Cette DTD permet de fixer la représentation symbolique des données de représentation et de faciliter l'évaluation de l'opérateur de localisation.

2.2.1. Evaluation de l'opérateur *inside*

Décrivons tout d'abord le processus d'évaluation de l'opérateur *inside*. Cet opérateur permet de retrouver une information « à l'intérieur » d'une zone géographique définie en paramètre de l'opérateur. Ce paramètre est appelé *LocalisationType*. Par exemple, la requête «quels sont les vendeurs situés au même étage que moi ? » utilise l'opérateur *inside* et le paramètre

LocalisationType associé à cet opérateur dans la requête est « étage ». Pour cet opérateur, la comparaison entre deux fichiers de localisation n'est possible que si les deux fournissent une représentation en données symboliques. En effet, le paramètre *LocalisationType* représente un élément XML, par exemple, l'élément XML « étage ». Chaque élément de la *liste de localisations issues de la requête* est comparé à chaque localisation de la *liste de localisations de référence* : lorsqu'il y a une correspondance de la valeur de l'élément entré en paramètre, l'élément de la liste de localisations issues de la requête est sélectionné. Finalement, pour chaque localisation sélectionnée, un degré d'exactitude est calculé :

$$\text{Degré d'exactitude}(\text{elt de LReq}) = \sum_{e \in \text{LRef}} \frac{\text{Correspondance}(\text{localisation}(e), \text{elt de LReq}))}{\text{Degré d'approximat ion}(e)}$$

où LReq représente la *liste de localisations issues de la requête*, LRef *la liste de localisations de référence* et Correspondance(elt de LReq, localisation(e)) est égal à 1 s'il y a une correspondance de la valeur de l'élément *LocalisationType* entré en paramètre et à 0 sinon.

Ce degré d'exactitude est égal à l'inverse du degré d'approximation car plus la localisation est approximée, moins la réponse est exacte. Ce degré d'exactitude permet de trier les réponses en fonction de leur qualité s'il existe plusieurs réponses possibles. Ce degré permet de choisir le ou les meilleures réponses à transmettre à l'utilisateur.

2.2.2. Evaluation de l'opérateur *close*

Considérons maintenant l'opérateur *close*. Pour évaluer l'opérateur de localisation *close*, chaque élément contenu dans la *liste de localisations issues de la requête* est comparé avec les fichiers de localisation de la *liste de localisations de référence*. Le tableau 5.1 résume les différentes combinaisons possibles : la comparaison entre deux localisations est réalisée différemment en fonction du contenu des fichiers XML. Par exemple, si la localisation issue de la *liste de localisations de référence* est décrite en données symboliques, la comparaison n'est possible que si l'élément de la *liste de localisations issues de la requête* est également représenté en données symboliques ou à la fois en données symboliques et physiques.

		Contenu des fichiers XML de la *liste de localisations issues de la requête*		
		Symbolique	Physique	Symbolique & Physique
Contenu des fichiers XML de la *liste de localisations de référence*	Symbolique	Comparaison symbolique	-	Comparaison symbolique
	Physique	-	Comparaison physique	Comparaison physique
	Symbolique & Physique	Comparaison symbolique	Comparaison physique	Comparaison physique

Tableau 5.1. Comparaisons en fonction du contenu des fichiers de localisation XML

Pour chaque comparaison entre un fichier XML de la *liste de localisations de référence* et un fichier XML de la *liste des localisations issues de la requête*, la « distance » entre ces deux fichiers est calculée. Dans le cas d'une comparaison entre données physiques, la distance entre les deux localisations est la distance euclidienne calculée selon la formule :

$$\text{distance} = \sqrt{\left[(lat_1 - lat_2)^2 + (long_1 - long_2)^2 + M * (alt_1 - alt_2)^2\right]}$$

La constante *M* est utilisée pour pénaliser la différence d'altitude dans certains environnements. Par exemple, dans un aéroport, si le client recherche un bureau Air France proche de lui, le bureau se situant au même endroit que le client mais deux niveaux plus bas sera sélectionné par rapport au bureau se situant au même étage mais dont la distance le séparant du client est légèrement supérieure. Or, dans ce cas, le client préfère sans doute limiter les changements d'étage afin de rejoindre plus rapidement le bureau Air France. Nous utilisons donc une constante de pénalité *M* qui est définie par l'administrateur de l'application de proximité.

Dans le cas d'une comparaison entre données symboliques, le pourcentage pondéré de valeurs identiques permet de définir la distance séparant les deux localisations. En effet, ce pourcentage est pondéré en fonction de l'importance des éléments définis par les données symboliques. L'administrateur de l'application de proximité définit ces pourcentages pondérés en même temps que la spécification de la DTD. Par exemple, la correspondance entre les valeurs de l'élément «pays » est moins importante que la correspondance entre les valeurs de l'élément « ville ». Pour calculer ce pourcentage, la structure hiérarchique des fichiers XML est également prise en compte c'est-à-dire que si les valeurs de l'élément « pays » ne sont pas identiques, les valeurs des sous-éléments (par exemple de l'élément « ville ») ne sont pas comparées et prises en compte dans le pourcentage final.

Comparaison physique (dist. en m)	Comparaison symbolique (similarités en %)	Coefficient de similarité
<100 m	>0%	10%
<80 m	>20%	30%
<60 m	>40%	50%
<40 m	>60%	70%
<20 m	>80%	90%

Tableau 5.2. Un exemple de tableau de correspondance pour le coefficient de similarité entre 2 localisations

En fonction de la distance (pour les représentations physiques) et du pourcentage pondéré (pour les représentations symboliques), un coefficient de similarité est associé à chaque couple de localisations (localisation de la *liste de localisations issues de la requête*, localisation de la *liste de localisations de référence*). Pour les représentations physiques, plus la distance entre deux localisations est grande, plus le coefficient de similarité est petit. En revanche, pour les représentations symboliques, plus le pourcentage pondéré des similarités est grand, plus le coefficient de similarité est grand également. Une table de correspondance est établie par l'administrateur en fonction de l'environnement de l'application de proximité et de la DTD choisie, cette table permet d'obtenir un même coefficient de similarité pour les différentes représentations. Un exemple est décrit dans le tableau 5.2 : dans cet exemple, si 60% des valeurs des éléments de la représentation symbolique sont identiques, la distance entre les deux localisations est évaluée à environ 40 mètres et le coefficient de similarité est fixé à 70%.

Pour chaque élément de la *liste de localisations issues de la requête*, une distance le séparant de la localisation de référence représentée est calculée. Le calcul de cette distance est basé sur le calcul de la moyenne des distances obtenues entre un élément de la *liste de*

localisations issues de la requête et les différentes localisations de la *liste de localisations de référence* en prenant en compte les degrés d'approximation des localisations de cette liste. Cette distance est calculée selon la formule :

$$\text{Dist}(\text{elt de LReq, loc. de référence}) = \frac{\sum_{e \in \text{LRef}} (\text{distance}(\text{elt de LReq, localisation}(e)) + \text{Degré d'approximation}(e))}{\text{Nombre d'éléments de LRef}}$$

où LReq représente la *liste de localisations issues de la requête,* LRef *la liste de localisations de référence* et distance(elt de LReq, localisation(e)) représente la distance en mètres correspondant au coefficient de similarité pour le couple (localisation de QL, localisation de e ? PL) en fonction du tableau de correspondance (tableau 5.2).

Par ailleurs, l'opérateur *close* peut être utilisé avec un paramètre de distance. Par exemple, dans la requête : «quels sont les arrêts de bus à moins de 100 mètres de moi?», l'opérateur close est utilisé avec un paramètre de distance égal à 100. L'évaluateur sélectionne alors les éléments issus de la requête pour lesquels, la distance calculée est inférieure ou égale au paramètre de distance spécifié dans la requête.

Si aucun paramètre de distance n'est spécifié : par exemple dans la requête «Quels sont les arrêts de bus proches de moi? ». En fonction du profil de mobilité du client, un paramètre de distance est associé à l'opérateur *close*. Par exemple, si le client est un piéton ayant une vitesse de 50 mètres par minute, le paramètre de distance est défini en fonction de la distance qu'il parcourt en 5 minutes par exemple : 250 mètres.

2.2.3. Evaluation de l'opérateur *closest*

L'évaluation de l'opérateur *closest* repose sur le même principe que l'évaluation de l'opérateur close mais seule la meilleure réponse (i.e. la plus proche) est sélectionnée. Pour chaque élément de la *liste de localisations issues de la requête,* la distance entre cette localisation et la localisation de référence est calculée. L'évaluateur de requêtes sélectionne finalement l'élément dont la distance entre sa localisation et la localisation de référence est la plus petite.

2.3. Conclusion

Nous venons de décrire les mécanismes d'évaluation des différents opérateurs de localisation supportés dans ISLANDS. Cette évaluation s'inscrit dans une série de trois étapes nécessaires à l'évaluation des requêtes de localisation. Pour évaluer ces requêtes, la localisation de référence, ainsi que la requête classique doivent également être évaluées. Ces différentes évaluations peuvent être ordonnancées différemment et être distribuées de différentes manières sur les différents nœuds du réseau. Cet ordonnancement et la distribution des requêtes doivent être effectués dans le but de minimiser les coûts associés à l'évaluation tout en maintenant une qualité de réponse satisfaisante. Dans la section suivante, nous présentons différentes stratégies d'évaluation des requêtes de localisation et l'analyse des différents tests effectués pour ces stratégies.

3. Optimisation de l'évaluation des requêtes dépendantes de la localisation

3.1. Problématique

Le service de localisation ISLANDS évalue des requêtes dans un environnement où des terminaux sont contraints en terme de ressources. De ce fait, il doit assurer une évaluation satisfaisante de la requête c'est-à-dire maintenir la qualité des réponses tout en minimisant les ressources utilisées pour effectuer cette évaluation. Contrairement aux solutions d'optimisation d'évaluation de requêtes dans les bases de données, dans notre environnement, l'objectif n'est pas seulement de réduire le temps nécessaire à l'évaluation de la requête. Notre objectif diffère également de celui des solutions d'optimisation proposées pour les recherches d'informations dans les systèmes P2P qui cherchent uniquement à réduire l'utilisation de la bande passante pour l'évaluation d'une requête. En effet, dans l'environnement des applications de proximité, les optimisations doivent non seulement prendre en compte le temps d'évaluation de la requête et l'utilisation de la bande passante mais surtout la consommation d'énergie des terminaux nomades. Nous devons donc considérer :

- L'optimisation du nombre de nœuds visités et donc le nombre de messages (requêtes et réponses) échangés.
- L'optimisation de la charge de travail des nœuds de l'application, en particulier des nœuds légers, et donc le nombre de nœuds évaluant la requête.
- L'optimisation de l'ordonnancement des différentes étapes nécessaires à l'évaluation des requêtes dépendantes de la localisation.

Concernant la réponse, lobjectif dans les applications de proximité n'est pas de fournir systématiquement la totalité des résultats disponibles dans la sphère de communication mais plutôt de garantir une qualité de réponses. Un résultat satisfaisant est une sélection des différentes réponses obtenues par l'évaluation de la requête même si toutes les réponses disponibles dans l'application de proximité n'ont pas été récupérées.

Pour assurer des résultats satisfaisants pour une requête issue d'un nœud léger, la requête doit être distribuée sur d'autres instances d'ISLANDS (i.e. être envoyée sur d'autres nœuds). L'espace de recherche, c'est-à-dire l'ensemble des terminaux, vers lesquels la requête est propagée, doit être sélectionné afin de limiter le nombre de messages échangés et le nombre d'évaluations effectuées sur des nœuds distants. Par exemple, une requête issue d'un nœud léger a plus de chances d'obtenir une réponse grâce à son évaluation sur un ou plusieurs nœuds centraux distants étant donné que les nœuds centraux disposent de ressources plus importantes et peuvent donc partager plus d'informations que les nœuds légers. Par ailleurs, afin de garantir une qualité de réponse, il est important de sélectionner les nœuds auxquels la requête est envoyée afin d'obtenir un résultat satisfaisant tout en minimisant les coûts associés à l'évaluation et à l'envoi de la requête. Cette sélection permet de limiter le nombre de nœuds visités et donc la charge de travail des différents nœuds présents dans l'application. De plus, sachant que les nœuds centraux offrent davantage de ressources que les nœuds légers, une répartition de la charge de travail adéquate doit être mise au point.

Cette sélection de l'espace de recherche est également exploitée dans certains systèmes P2P tels que Kazaa [Kazaa] reposant sur un réseau de super-nœuds qui centralisent la description des informations partagées et qui sont les seuls à être interrogés. La restriction de l'espace de recherche permet de minimiser les ressources et est utilisée pour optimiser l'évaluation des requêtes de localisation mais également l'évaluation des requêtes classiques. En effet, l'évaluation des requêtes classiques nécessite souvent une évaluation distante sur d'autres nœuds.

Dans une application de proximité, l'espace de recherche de la requête est, a priori, représenté par l'ensemble des nœuds présents dans la sphère de communication : à la fois les nœuds centraux et les nœuds légers. Afin de limiter le nombre de nœuds visités et les coûts associés à l'évaluation d'une requête, nous définissons deux autres sélections d'espace de recherche :

- L'espace de recherche composé uniquement des nœuds centraux.
- L'espace de recherche composé par les «bons » nœuds présents dans l'application. Ces « bons » nœuds, comme dans la proposition des indices locaux de Yang et al. [YAN 02] dans le cadre de systèmes P2P Internet, sont sélectionnés en fonction du nombre de requêtes auxquelles ils ont réussi à répondre précédemment, en fonction de leur disponibilité et du nombre de leurs voisins directs.

Ces deux sélections d'espace de recherche permettent de minimiser la charge de travail des nœuds légers présents dans la sphère de communication puisqu'ils ne sont pas visités par la requête. L'espace de recherche peut également être réduit en minimisant la profondeur maximale. Nous discutons dans la section 3.4.6 de l'impact de la sélection des nœuds auxquels envoyer la requête sur la qualité de la réponse obtenue.

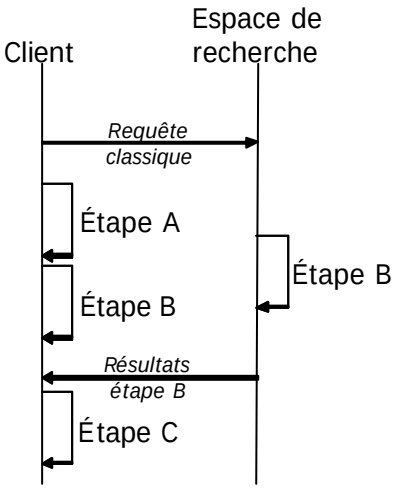


Figure 5.2.a. Exemple 1 d'ordonnancement pour l'évaluation d'une requête dépendante de la localisation

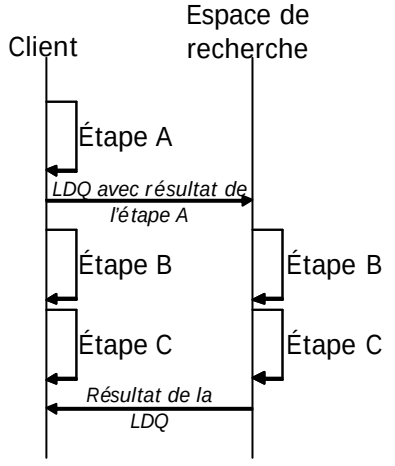


Figure 5.2.b. Exemple 2 d'ordonnancement

En ce qui concerne les requêtes de localisation, l'ordonnancement des différentes étapes à effectuer pour l'évaluation de ces requêtes participe également à l'optimisation des coûts. En effet, les différentes étapes (A – évaluation de la localisation de référence, B – évaluation de la requête classique et C – évaluation de l'opérateur de localisation) peuvent être

ordonnancées différemment. Certaines étapes peuvent être effectuées en parallèle et les stratégies de diffusion peuvent également être différentes. Par exemple, les figures 5.2. a et b représentent deux choix d'ordonnancements différents pour l'évaluation d'une requête dépendante de la localisation (LDQ).

Dans la suite, nous nous intéressons particulièrement aux requêtes dépendantes de la localisation où la localisation de référence est celle du client. Concernant les requêtes relatives à une localisation, la requête concernant la localisation de référence est envoyée et évaluée en même temps que la requête classique. Concernant les requêtes dépendantes de la localisation, nous étudions donc, dans la section suivante, les différentes possibilités d'ordonnancement pour trouver les stratégies d'optimisation les plus efficaces.

3.2. Stratégies

3.2.1. Introduction

Les stratégies d'évaluation proposées dans cette section reposent sur différents choix d'ordonnancements des différentes opérations nécessaires à l'évaluation d'une requête dépendante de la localisation. Afin d'économiser les ressources des clients légers, nous avons également envisagé l'utilisation d'un nœud de service. Ce nœud de service représente un nœud connecté au nœud client qui est capable de servir de relais entre le client et les autres nœuds de l'application. L'utilisation d'un tel nœud permet, en particulier de réduire la charge de travail du nœud client liée à la réception de multiples messages. En effet, le nœud de service permet de réceptionner l'ensemble des réponses reçues, de les traiter et de ne renvoyer que les réponses sélectionnées au nœud client. Le nœud de service fournit également en partie un support aux déconnexions du nœud client.
Dans la suite, nous proposons différentes stratégies d'évaluation des requêtes dépendantes de la localisation. Ensuite nous évaluons leurs coûts respectifs afin de définir quelle stratégie est la plus adaptée pour un environnement donné. Dans ces différentes stratégies, nous considérons que le nœud client est un nœud léger.

3.2.2. Présentation

Différentes stratégies d'évaluation des requêtes dépendantes de la localisation sont proposées dans la suite. Ces stratégies reposent sur un parcours en largeur d'abord. En effet, dans un environnement où le temps de connexion des terminaux nomades peut être relativement court et où les déconnexions accidentelles sont fréquentes, le parcours en largeur d'abord du graphe représentant l'espace de recherche permet d'obtenir plus rapidement une réponse à la requête. En fonction des différentes étapes nécessaires à l'évaluation d'une requête dépendante de la localisation, ce parcours doit être optimisé pour limiter les coûts de communication et de calcul. C'est dans cette optique que quatre stratégies d'évaluation sont proposées.

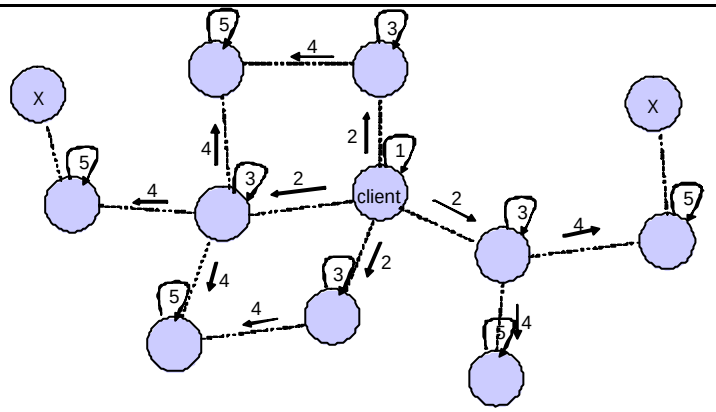
Ces stratégies sont différenciées par :

- l'utilisation ou non d'un nœud de service : les stratégies 2 et 4 utilisent un nœud de service contrairement aux stratégies 1 et 3.
- l'ordonnancement des différentes étapes nécessaires à l'évaluation des requêtes dépendantes de la localisation.

Dans la suite, nous présentons le principe de chaque stratégie en les illustrant à l'aide d'un exemple.

3.2.2.1. Stratégie 1

La figure 5.3 illustre les différentes étapes et le parcours effectué par la requête lors de l'exécution de la stratégie 1. Dans cette stratégie, le module de positionnement calcule tout d'abord la localisation du client sur le nœud client. Ensuite, la requête de localisation (avec la localisation du client en paramètre) est envoyée aux nœuds de l'espace de recherche considéré. Quand un nœud reçoit et accepte la requête, elle est évaluée par son service de localisation en utilisant les données stockées localement : chaque nœud évalue à la fois la requête classique et l'opérateur de localisation en fonction de la localisation définie en paramètre et des résultats de la requête classique. Si une réponse existe, celle-ci est envoyée au client en utilisant le chemin inverse de la requête (la figure 5.3 ne représente pas le traitement des réponses). Finalement, en utilisant un temps d'attente maximum (ou *timeout* en anglais), le nœud client construit le résultat final avec les réponses reçues en fonction des paramètres désirés par le client tels que le nombre de réponses souhaitées.



---- Connexions entre les différents nœuds

1 – Le client évalue sa localisation.

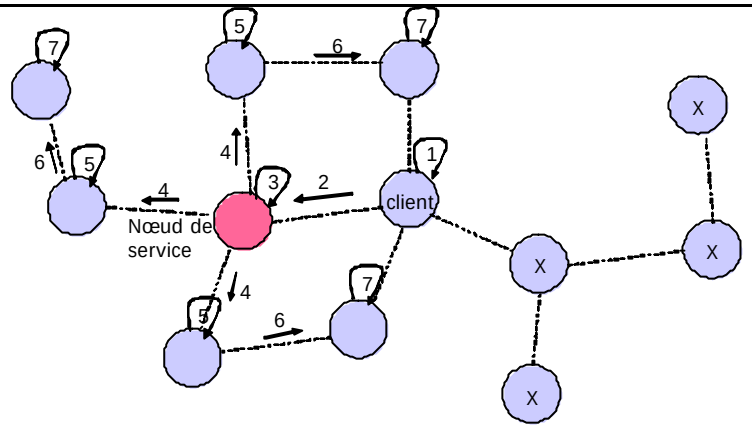
2 et 4 – La requête de localisation avec le paramètre de localisation est envoyée aux voisins (selon l'espace de recherche sélectionné).

3 et 5 – La requête est évaluée entièrement sur les nœuds voisins (requête classique + évaluation de l'opérateur de localisation).

Figure 5.3. Fonctionnement de la stratégie 1 avec une profondeur fixée à 2

3.2.2.2. Stratégie 2

La seconde stratégie diffère de la stratégie 1 par l'utilisation d'un nœud de service. Dans la seconde stratégie, dont le parcours est décrit dans la figure 5.4, comme dans la première stratégie, le module de positionnement du client calcule sa position. Ensuite, la requête (avec le paramètre de localisation renseigné) est envoyée au nœud de service. Ce nœud de service est sélectionné par le nœud client en fonction de sa qualité (nombre de réponses émises précédemment, nombre de voisins etc.) et de sa charge de travail. Le nœud de service envoie ensuite la requête aux nœuds de l'espace de recherche. Ces nœuds effectuent, s'ils le souhaitent, l'évaluation globale de la requête de localisation : évaluation de la requête classique et évaluation de l'opérateur de localisation. Les réponses sont récupérées par le nœud de service qui construit la réponse finale et la renvoie au nœud client.



1 – Le client évalue sa localisation.

2 – La requête de localisation avec le paramètre de localisation est envoyée au nœud de service.

3 – La requête est évaluée entièrement sur le nœud de service (évaluation de la requête classique et évaluation de l'opérateur de localisation)

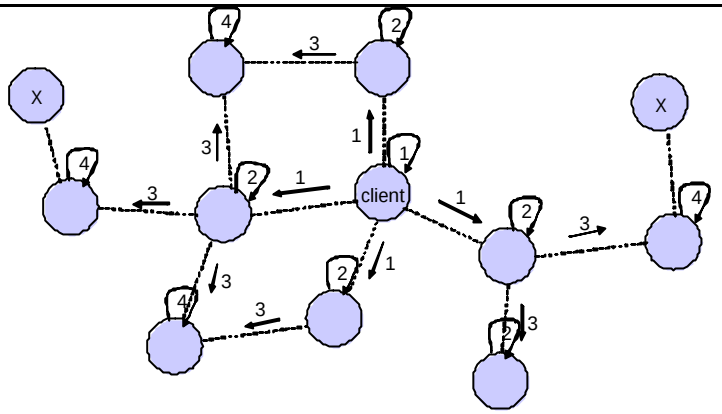
4 et 6 – La requête de localisation avec le paramètre de localisation est envoyée aux voisins du nœud de service (selon l'espace de recherche sélectionné).

5 et 7 – La requête est évaluée entièrement sur les nœuds voisins (requête classique et évaluation de l'opérateur de localisation).

Figure 5.4. Fonctionnement de la stratégie 2 avec une profondeur fixée à 2

3.2.2.3. Stratégie 3

La stratégie 3 diffère des deux premières stratégies par l'ordonnancement des étapes de l'évaluation. Comme cela est illustré dans la figure 5.5, pendant que la requête classique (sans le paramètre de localisation) est envoyée aux nœuds de l'espace de recherche sélectionné, le nœud client évalue sa localisation. Ensuite, lorsque que le nœud client reçoit les différentes réponses, l'évaluateur de requêtes du service de localisation évalue l'opérateur de localisation en fonction de la localisation calculée et des réponses reçues sur le nœud client.



1 – Le client évalue sa localisation et la requête classique est envoyée simultanément aux nœuds voisins (selon l'espace de recherche sélectionné).

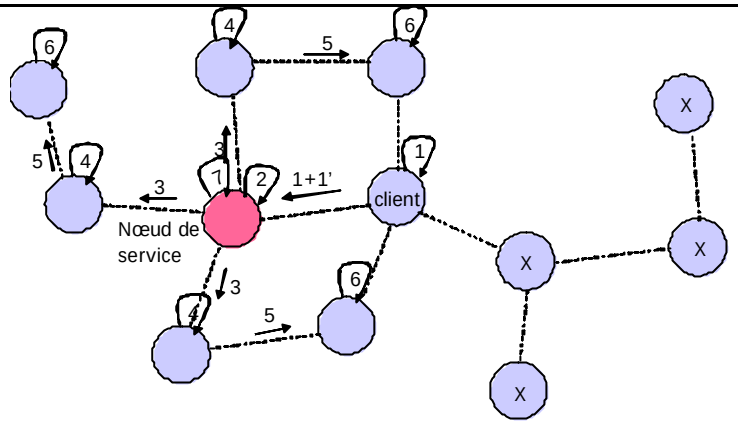
2 et 4 – La requête classique est évaluée sur les nœuds voisins.

3 – La requête classique est envoyée aux nœuds voisins.

Figure 5.5. Fonctionnement de la stratégie 3 avec une profondeur fixée à 2

3.2.2.4. Stratégie 4

La quatrième stratégie diffère de la troisième stratégie par l'utilisation d'un nœud de service. Dans cette stratégie, décrite dans la figure 5.6, le nœud client envoie la requête classique au nœud de service. Le nœud de service envoie ensuite la requête à la sélection de l'espace de recherche et récupère ensuite les réponses. Simultanément, le module de positionnement calcule la position du client et celle-ci est ensuite envoyée au nœud de service. Le service de localisation du nœud de service évalue alors l'opérateur de localisation en fonction des réponses reçues et de la localisation du client. Et finalement, la réponse finale est renvoyée au nœud client.



1 – Le client évalue sa localisation et la requête classique est envoyée simultanément au nœud de service.

1' – La localisation du client est envoyée au nœud de service.

2 – La requête classique est évaluée par le nœud de service.

3 et 5 – La requête classique est envoyée aux nœuds voisins du nœud de service (selon l'espace de recherche sélectionné).

4 et 6 – La requête classique est évaluée sur les nœuds voisins.

7 – L'opérateur de localisation est évalué sur le nœud de service.

Figure 5.6. Fonctionnement de la stratégie 4 avec une profondeur fixée à 2

3.2.3. Conclusion

Les stratégies présentées dépendent de l'ordonnancement des différentes étapes nécessaires à l'évaluation d'une requête dépendante de la localisation et du choix d'utilisation d'un nœud de service. Afin d'optimiser l'évaluation des requêtes dépendantes de la localisation, ces stratégies ont été testées afin de mettre en évidence leurs atouts et leurs limites et de définir la plus intéressante à utiliser dans une application de proximité.

3.3. Environnement de simulation

3.3.1. Introduction

Afin de déterminer la (les) meilleure(s), nos stratégies ont été évaluées dans différents environnements afin d'en déduire les coûts moyens en terme de temps de réponse, de communication et d'énergie. Pour tester ces différents coûts, nous avons développé un simulateur. En effet, tester les différentes stratégies à l'aide du prototype que nous avons développé, n'était pas envisageable car nous ne disposions pas de suffisamment de terminaux (i.e. de participants) pour tester la montée en charge dans une application de proximité.

Ces simulations ont pour objectif d'évaluer les différents coûts inhérents à l'évaluation d'une requête dépendante de la localisation. Contrairement aux optimisations dans les bases de données, nous ne considérons pas l'optimisation de l'évaluation locale de la requête sur un nœud mais plus généralement des coûts associés à la distribution de la requête au sein de l'application de proximité. Concernant l'évaluation locale d'une requête sur un terminal

nomade, différents travaux tels que [ANC 03] proposent des solutions ayant pour objectif de minimiser l'utilisation des ressources lors d'une évaluation de requêtes sur un terminal contraint.

Afin de mesurer les coûts associés à la distribution de la requête, nous proposons de mesurer les différentes opérations nécessaires à chaque stratégie et d'en déduire les coûts suivants :

- les coûts associés aux communications nécessaires pour l'évaluation en fonction des différentes stratégies. Ces coûts représentent en particulier l'utilisation de la bande passante dédiée à l'envoi de la requête vers d'autres nœuds et à la réception des réponses provenant de ces nœuds.
- les coûts associés aux différents traitements effectués sur les nœuds en terme de consommation de ressources.
- le temps de réponse associé à l'évaluation de la requête.

3.3.2. Opérations à considérer

Les différentes opérations à prendre en compte dans l'évaluation des requêtes dépendantes de la localisation sont classifiées en deux catégories différentes : les opérations de communication et les opérations de traitement. L'évaluation de la localisation du nœud client intervient également dans le coût global de l'évaluation d'une requête dépendante de la localisation. Cette opération ne peut être exécutée que depuis le nœud client et les optimisations présentées dans le chapitre précédent sont donc mises en œuvre. Cependant, nous prenons en compte le temps de cette opération car le résultat de cette opération est nécessaire dans certaines stratégies pour continuer l'évaluation : c'est le cas, des stratégies 1 et 2. Nous notons l'opération d'évaluation de la localisation par *EvalLoc*.

Parmi les opérations de communication, trois opérations sont à prendre en compte :

- l'envoi de la requête entre deux nœuds.
- la diffusion de la requête à partir d'un nœud vers l'ensemble des nœuds auxquels il est connecté dans le cas où l'espace de recherche défini dans les stratégies est représenté par tous les nœuds.
- l'envoi des réponses.

Les opérations de traitement sont :

- l'évaluation de la requête classique.
- l'évaluation de l'opérateur de localisation.
- la construction et la sélection de la réponse.

Ces différentes opérations dépendent toujours du type du nœud sur lequel elles sont effectuées. En effet, les opérations sont souvent beaucoup plus coûteuses sur un nœud léger restreint en ressources et en autonomie que sur un nœud central. Ces opérations et les variables dont elles dépendent, sont décrites dans le tableau 5.3.

Type d'opérations	*Description des différentes opérations*
Communications	EnvoiRequête(TypeNoeuds, TailleRequête) La variable *TypeNoeuds* définit entre quels types de nœuds la requête est échangée : si les deux nœuds sont centraux, dans ce cas, la variable *TypeNoeuds* est définie comme « central » sinon elle est définie comme « léger ». Le paramètre *TailleRequête* représente la taille de la requête échangée.
	DiffusionRequête(TypeNoeud, TailleRequête) Le paramètre *TypeNoeud* représente le type du nœud qui émet la diffusion.
	RéceptionRéponses(TypeNoeuds, TailleRéponse) Le paramètre *TailleRéponse* représente la taille de la réponse envoyée.
Traitements	EvalRequête(TypeNoeud, TypeStockage, TailleStockage) Le paramètre *TypeNoeud* représente le type du nœud sur lequel est évaluée la requête. Le paramètre *TypeStockage* représente le support de stockage utilisé pour stocker l'information. Le paramètre *TailleStockage* représente la quantité d'informations stockées sur le nœud.
	EvalOperateurLoc(TypeNoeud, NbListeLocalisation, NbReponses) Le paramètre *NbListeLocalisation* définit combien de couples représentent la localisation du nœud. Le paramètre *NbReponses* représente le nombre de réponses obtenues pour la requête classique.
	SélectionRéponse(TypeNoeud, NbRéponses, NbRéponsesSouhaitées) Le paramètre *NbRéponses* représente le nombre total de réponses obtenues par l'évaluation de la requête. La paramètre *NbRéponsesSouhaitées* représente le nombre de réponses souhaitées et paramétrées par le client.

Tableau 5.3. Description des différentes opérations

3.3.3. Expression des coûts

Pour les différentes opérations décrites ci-dessus, il est important d'en exprimer les coûts :
Les coûts de communication sont exprimés en terme d'utilisation de la bande passante, nous utilisons l'unité Ko (Kilo Octet).
Les coûts de traitement sont exprimés en nombre d'opérations. Ils permettent à partir du nombre d'opérations effectuées pour chaque type d'opérations de déduire en particulier l'impact des différentes stratégies sur la consommation des ressources des terminaux nomades.
Finalement, les coûts associés au temps de réponse sont exprimés en ms (milliseconde). Ils représentent le temps mis pour effectuer une opération donnée.
Le tableau 5.4 représente l'expression des différents coûts pour l'ensemble des opérations présentées dans la section précédente. La fonction *Comm()* est utilisée pour obtenir le coût

en communication d'une opération passée en paramètre. La fonction *Tps()* est quant à elle, utilisée pour obtenir le temps d'exécution d'une opération passée en paramètre. Dans ce tableau, les opérations de traitement n'ont aucun coût en terme de communication étant donné qu'elles se réalisent localement sur chaque nœud. Concernant l'opération *EvalLoc*, les coûts liés aux communications et aux traitements ne sont pas utilisés pour la comparaison des différentes stratégies. En effet, ces coûts sont les mêmes quelle que soit la stratégie utilisée. En revanche, le temps nécessaire à l'exécution de cette opération est pris en compte.

Opération	Coût de communication (en Ko)	Coût de traitement (en opérations)	Coût en temps de réponse (en ms)
EvalLoc	-	-	Tps(EvalLoc)
EnvoiRequête	Comm(EnvoiRequête)	EnvoiRequête	Tps(EnvoiRequête)
DiffusionRequête	Comm(DiffusionRequête)	DiffusionRequête	Tps(DiffusionRequête)
RéceptionRéponses	Comm(RéceptionRéponses)	RéceptionRequête	Tps(RéceptionRéponses)
EvalRequête	Aucun	EvalRequete	Tps(EvalRequête)
EvalOperateurLoc	Aucun	EvalOperateurLoc	Tps(EvalOperateurLoc)
SélectionRéponse	Aucun	SélectionRéponse	Tps(SélectionRéponse)

Tableau 5.4. Expression des différents coûts

La fonction *Comm(EnvoiRequête)* représente l'utilisation de la bande passante pour l'envoi d'une requête. Cette fonction dépend donc de la taille de la requête échangée.

La fonction *Comm(DiffusionRequête)* représente l'utilisation de la bande passante pour la diffusion d'une requête.

La fonction *Comm(RéceptionRéponses)* représente l'utilisation de la bande passante pour l'échange des réponses entre deux nœuds. Cette fonction dépend de la taille des réponses échangées.

La fonction *Tps(EvalLoc)* définit le temps d'exécution de l'évaluation de la localisation du client. Cette fonction dépend du nombre de nœuds voisins du nœud client.

La fonction *Tps(EnvoiRequête)* définit le temps mis pour l'envoi et la réception d'une requête entre deux nœuds. Cette fonction dépend de la taille de la requête et du débit réel. Ce débit dépend des types de nœuds qui échangent le message : en effet, si le message est échangé entre deux nœuds centraux, le débit mesuré est beaucoup plus grand que si un des deux nœuds est un nœud léger.

La fonction *Tps(DiffusionRequête)* définit le temps mis pour la diffusion d'une requête. Cette fonction dépend également du type de nœud qui diffuse le message et de la taille de la requête.

La fonction *Tps(RéceptionRéponse)* définit le temps mis pour l'échange de réponses entre deux nœuds. Elle dépend du type des nœuds, du débit et de la taille des réponses.

La fonction *Tps(EvalRequête)* définit le temps nécessaire à l'évaluation de la requête classique par une instance d'ISLANDS. Le temps de cette évaluation dépend dans un premier temps, du type de nœud sur lequel elle est effectuée. Cette évaluation locale

nécessite une interrogation des métadonnées RDF et une interrogation de données si la donnée recherchée est présente sur le nœud. Le temps nécessaire pour l'interrogation des métadonnées RDF dépend du support de stockage utilisé pour le stockage des métadonnées. L'interrogation des données dépend elle aussi du support de stockage de la donnée (fichiers, SGBD, service d'annuaire).

La fonction *Tps(EvalOperateurLoc)* représente le temps nécessaire à l'évaluation de l'opérateur de localisation. Ce temps est défini par le temps mis pour analyser et comparer les différents fichiers de localisations : il dépend du type de nœud sur lequel l'évaluation est effectué, du nombre de fichiers de localisations décrivant la localisation du client et du nombre de localisations récupérées de la requête classique.

La fonction *Tps(SélectionRéponse)* représente le temps nécessaire à la sélection de la réponse lorsque de multiples réponses sont obtenues et que le client ne souhaite que les 10 meilleures par exemple. La fonction *Tps(SélectionRéponse)* fournit le temps nécessaire au tri des différentes réponses et ce temps dépend du type de nœud sur lequel l'opération est effectuée (léger ou central).

Les valeurs numériques utilisées pour ces différentes fonctions, en particulier pour les fonction *Tps()* sont issues des tests expérimentaux effectués grâce au prototype présenté dans le chapitre 6.

3.3.4. Modèles de coût

Premièrement le modèle de coût pour les coûts de communication est représenté par la formule (1). Dans cette formule, nous choisissons un espace de recherche restreint (que les nœuds centraux ou que les «bons » nœuds). Si l'espace de recherche était l'ensemble des nœuds présents, la fonction Comm(envoiRequête) serait remplacée par la fonction Comm(DiffusionRequête). Ce modèle de coût est valable pour chaque stratégie.

(1) $Coût_{comm}$= ? Comm(EnvoiRequête) + ? Comm(RéceptionRéponses)

Le modèle de coût associé aux coûts d'utilisation des ressources est représenté par la formule (2). Ce modèle de coût est également valable pour chaque stratégie. L'opération EnvoiRequête est remplacée par l'opération DiffusionRequête si l'espace de recherche est représenté par l'ensemble des nœuds présents. Le coût total d'utilisation des ressources est égal à la somme des différents coûts d'opérations en terme de consommation de ressources. Les opérations sont distinguées si elles sont effectuées sur un nœud léger (NL) ou non (NC). Les coefficients A, B, C et D permettent de pondérer les différents coûts : en effet, sur les terminaux légers, les opérations de communication sont plus coûteuses que les opérations de traitement en terme de consommation de ressources. De même, sur les nœuds centraux, les opérations de communication coûtent davantage que les opérations de traitement. Finalement, les opérations effectuées sur les terminaux nomades sont beaucoup plus coûteuses que les opérations effectuées sur les nœuds centraux en terme de consommation d'énergie.

(2) $Coût_{trait}$= A(? NL EnvoiRequête + ? NL RéceptionRéponses)
+ B(? NL EvalRequête + ? NL EvalOperateurLoc + ? NL SélectionRéponse)
+ C(? NC EnvoiRequête + ? NC RéceptionRéponses)
+ D(?NC EvalRequête + ? NC EvalOperateurLoc + ? NC SélectionRéponse)

Enfin, le modèle de coûts associé au temps d'exécution de l'évaluation d'une requête dépendante de la localisation est défini dans les formules (3.1) pour la stratégie 1, (3.2) pour la stratégie 2, (3.3) pour la stratégie 3 et (3.4) pour la stratégie 4. Chaque modèle permet de définir le temps d'exécution nécessaire à l'évaluation d'une requête dépendante de la localisation pour chaque stratégie. Comme précédemment, si l'espace de recherche est représenté par tous les nœuds, l'opération DiffusionRequête est utilisée au lieu de l'opération EnvoiRequête.

(3.1) $Coût_{temps}$(S1) = Tps(EvalLoc)
+ ? $_{NL}$ Tps(EnvoiRequête) + ? $_{NL}$ Tps(RéceptionRéponses)
+ ? $_{NC}$ Tps(EnvoiRequête) + ? $_{NC}$ Tps(RéceptionRéponses)
+ ? $_{NL}$ Tps(EvalRequête) + ? $_{NL}$ Tps(EvalOperateurLoc)
+ ? $_{NC}$ Tps(EvalRequête) + ? $_{NC}$ Tps(EvalOperateurLoc)
+ ? $_{NClient}$ Tps(SélectionRéponse)

(3.2) $Coût_{temps}$(S2) = Tps(EvalLoc)
+ ? $_{NL}$ Tps(EnvoiRequête) + ? $_{NL}$ Tps(RéceptionRéponses)
+ ? $_{NC}$ Tps(EnvoiRequête) + ? $_{NC}$ Tps(RéceptionRéponses)
+ ? $_{NL}$ Tps(EvalRequête) + ? $_{NL}$ Tps(EvalOperateurLoc)
+ ? $_{NC}$ Tps(EvalRequête) + ? $_{NC}$ Tps(EvalOperateurLoc)
+ ? $_{NService}$ Tps(SélectionRéponse)

(3.3) $Coût_{temps}$(S3) = Max [Tps(EvalLoc),
? $_{NL}$ Tps(EnvoiRequête) + ? $_{NL}$ Tps(RéceptionRéponses)
+ ? $_{NC}$ Tps(EnvoiRequête) + ? $_{NC}$ Tps(RéceptionRéponses)
+ ? $_{NL}$ Tps(EvalRequête)
+ ? $_{NC}$ Tps(EvalRequête)]
+ ? $_{NClient}$ Tps(EvalOperateurLoc) + ? $_{NClient}$ Tps(SélectionRéponse)

(3.4) $Coût_{temps}(S4) = Max [Tps(EvalLoc),$

$? _{NL} Tps(EnvoiRequête) + ? _{NL} Tps(RéceptionRéponses)$

$+ ? _{NC} Tps(EnvoiRequête) + ? _{NC} Tps(RéceptionRéponses)$

$+ ? _{NL} Tps(EvalRequête)$

$+ ? _{NC} Tps(EvalRequête)]$

$+ ? _{NService} Tps(EvalOperateurLoc) + ? _{NService} Tps(SélectionRéponse)$

3.4. Résultats des tests et synthèse

3.4.1. Introduction

Pour chaque stratégie, les différentes étapes nécessaires à l'évaluation des requêtes dépendantes de la localisation ont été simulées ainsi que le parcours des requêtes et des réponses afin de définir le nombre d'opérations nécessaires. Le simulateur permet de paramétrer pour chaque stratégie la profondeur choisie et l'espace de recherche sélectionné (tous les nœuds, les nœuds centraux ou les « bons » nœuds).

Les stratégies ont été simulées dans différents environnements qui varient en fonction de la densité de nœuds centraux par rapport aux nœuds légers et de la densité des « bons » nœuds. Les nœuds étaient placés aléatoirement. Les différents nœuds légers étaient successivement sélectionnés pour être le nœud client. L'évaluation de la requête dépendante de la localisation est alors lancée à partir de ce nœud client suivant les différentes stratégies, les différents espaces de recherche et les différentes profondeurs.

Nous définissons également au sein du simulateur, l'environnement de l'application de proximité. Nous avons choisi de représenter un contexte bien connecté (en fonction de la surface et du nombre de nœud présents) afin d'analyser au mieux les performances des différentes stratégies : nous avons choisi une surface de 100m x 100m avec 50 nœuds et la portée du réseau est fixée à 40m, ce qui permet de représenter un environnement bien connecté.

Nous avons effectué différents tests en fonction de différentes densités : la densité des nœuds centraux varie de 5% à 90% et la densité des «bons » nœuds varie de 0% à 100%. Pour analyser les coûts inhérents aux différentes stratégies, nous utilisons dans la suite deux environnements représentatifs :

- Un environnement de référence composé de 50% de nœuds centraux et de 50% de nœuds légers. Les «bons » nœuds représentent 50% des nœuds centraux. Cet environnement nous sert ensuite de référence.
- Un environnement que nous appellerons «environnement réel » où la densité de nœuds centraux est de 10% et la densité de nœuds légers de 90%. La densité de « bons » nœuds parmi les nœuds centraux est toujours paramétrée à 50%. Selon nous, cet environnement reflète relativement bien les proportions réelles des applications de proximité.

En fonction de ces différents environnements, nous présentons les différents coûts obtenus pour les différentes stratégies. Pour comparer les différentes stratégies, nous ne présentons

dans la suite que les résultats obtenus pour un espace de recherche non restreint : la requête peut être envoyée à tous les nœuds (centraux et légers). La profondeur choisie est 4 car, pour l'environnement défini, elle permet d'accéder à la grande majorité des nœuds présents dans l'application. Après la présentation de la synthèse concernant les différentes stratégies, nous étudions dans la section 3.4.6, l'impact de la sélection de l'espace de recherche sur les coûts et sur la qualité de la réponse obtenue.

3.4.2. Coûts en terme d'opérations

Comme nous avons choisi comme espace de recherche l'ensemble des nœuds de l'application, le nombre d'opérations EnvoiRequête est nul et les envois de la requête sont gérés par l'opération DiffusionRequête. Nous présentons dans les figures 5.7 à 5.10 le nombre d'opérations nécessaires pour chaque stratégie. Ces stratégies sont testées dans l'environnement de référence (figures a) et dans l'environnement réel (figures b).

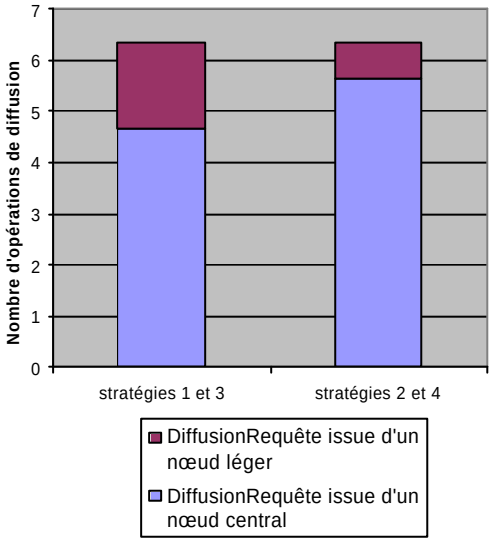


Figure 5.7.a Diffusion de la requête (env. de référence)

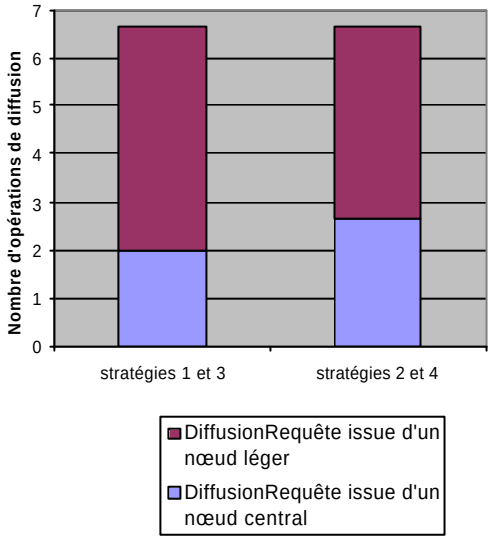


Figure 5.7.b Diffusion de la requête (env. « réel »)

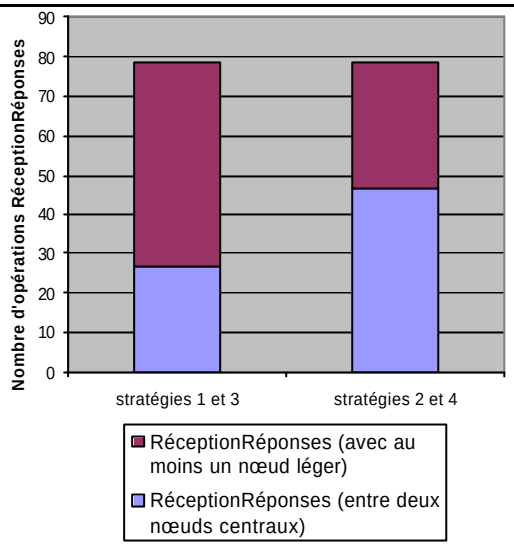


Figure 5.8.a Echange de réponses (env. de référence)

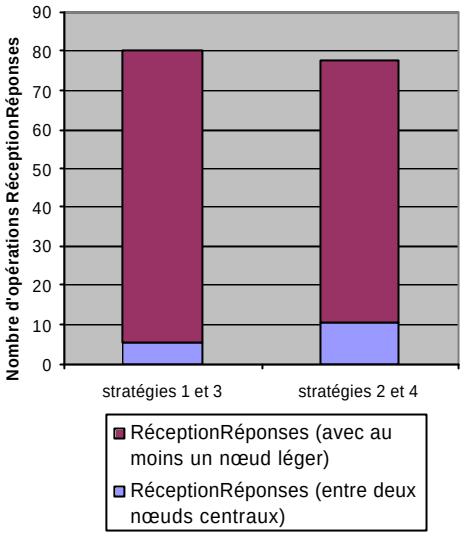


Figure 5.8.b Echange de réponses (env. « réel »)

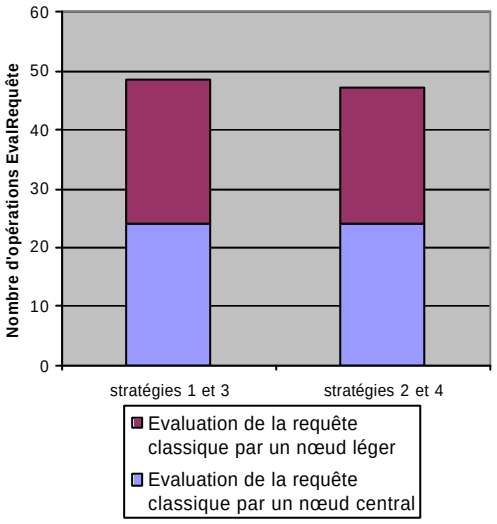


Figure 5.9.a Evaluation de la requête classique (env. de référence)

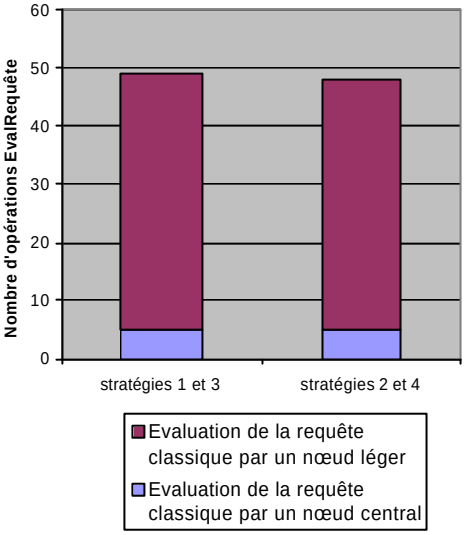


Figure 5.9.b Evaluation de la requête classique (env. « réel »)

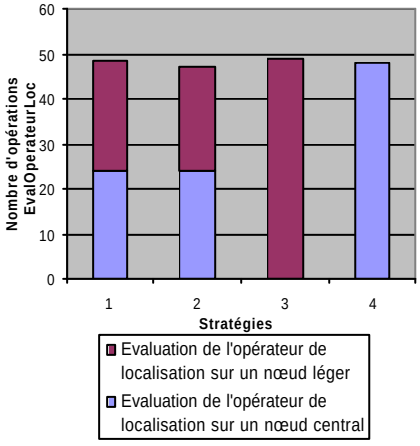


Figure 5.10.a Evaluation de l'opérateur de localisation (env. de référence)

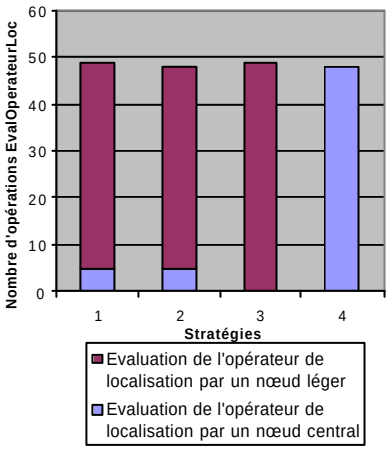


Figure 5.10.b Evaluation de l'opérateur de localisation (env. « réel »)

Nous constatons, dans un premier temps, que les ressources utilisées pour l'évaluation d'une requête dépendante de la localisation dans un environnement réel proviennent davantage des nœuds légers que des nœuds centraux. Cette constatation est logique étant donné que le nombre de nœuds centraux dans l'environnement « réel » est relativement petit.
Nous remarquons d'autre part, que les opérations de communication (DiffusionRequête et RéceptionRéponses) décrites dans les figures 5.7.a/b et 5.8.a/b, sont davantage prises en charge par les nœuds centraux dans les stratégies 2 et 4. C'est l'effet de l'utilisation du nœud de service qui reçoit l'ensemble des réponses et donc, un seul message contenant la réponse finale est finalement envoyé au nœud client qui est léger.
Le nombre d'évaluation des requêtes classiques décrit dans les figures 5.9.a/b, reste identique quelle que soit la stratégie utilisée. En effet, ce nombre d'évaluations ne peut être réduit que par la restriction de l'espace de recherche ou par le paramétrage d'une profondeur plus petite.
La différence entre les stratégies est davantage perceptible au niveau du nombre d'évaluations de l'opérateur de localisation décrit dans les figures 5.10.a/b. En effet, l'évaluation de l'opérateur de localisation est effectuée successivement sur l'ensemble des nœuds visités pour les stratégies 1 et 2. En revanche, concernant la stratégie 3, l'évaluation de l'opérateur de localisation sur l'ensemble des réponses obtenues est effectuée uniquement sur le nœud client qui est un nœud léger. A l'inverse, cette évaluation est effectuée sur le nœud service dans la stratégie 4.
Finalement, concernant la sélection des réponses, elle est gérée par le nœud client dans les stratégies 1 et 3 ou gérée par le nœud de service dans les stratégies 2 et 4.
Le nombre d'exécutions des différentes opérations nous permet d'évaluer l'impact des différentes stratégies sur la consommation d'énergie des terminaux nomades. En effet, les expérimentations réalisées grâce au prototype nous ont démontré que les opérations de communication sont extrêmement coûteuses en terme de consommation d'énergie sur les terminaux nomades. Nous comparons donc, dans la section suivante, les différents coûts liés aux communications pour les différentes stratégies.

3.4.3. Coûts de communication

Les coûts de communication générés par l'évaluation d'une requête de localisation sont séparés en deux groupes :

- les communications liées à l'envoi de la requête. Concernant les stratégies 1 et 2, les communications liées à l'envoi de la requête sont constituées de l'envoi de la requête avec le paramètre de localisation. Ce paramètre de localisation est un fichier de localisation XML ou une liste de couples (fichier de localisation, degré d'approximation). En revanche, dans les stratégies 3 et 4, seule la requête classique est envoyée aux nœuds de l'espace de recherche. De ce fait, les coûts de communication liés à l'envoi de la requête sont légèrement inférieurs pour les stratégies 3 et 4.

- les communications liées à la réception des réponses. Concernant les stratégies 1 et 2, la requête entière est évaluée sur les nœuds de l'espace de recherche (requête classique et opérateur de localisation). Les réponses envoyées sont donc les réponses finales de la requête dépendante de la localisation. En revanche, pour les stratégies 3 et 4, les nœuds de l'espace de recherche n'évaluent que la requête classique donc les réponses contiennent les résultats de la requête classique et les fichiers de localisation associés à ces résultats. De ce fait, les coûts de communication sont beaucoup plus importants pour les stratégies 3 et 4. La consommation d'énergie des terminaux nomades liés à la réception de ces réponses, pour ces deux stratégies, est donc beaucoup plus importante étant donnée la taille des informations à échanger.

Pour les stratégies 2 et 4, l'utilisation d'un nœud de service diminue légèrement le nombre de messages échangés avec au moins un nœud léger. De ce fait, la stratégie 2 est la plus intéressante en terme de coût de communication et permet de minimiser de façon maximale les ressources des terminaux nomades. Cependant, dans le cas où la sélection de l'espace de recherche est limitée aux nœuds centraux, les coûts en terme de communication sont considérablement diminués. En particulier pour la stratégie 4 où le nœud client ne reçoit alors que la réponse finale. Nous pouvons donc en déduire que si l'espace de recherche n'est pas restreint, la stratégie 2 est la plus intéressante. Sinon, la stratégie 4 est moins coûteuse en terme de communications.

3.4.4. Coûts en temps de réponse

Après la comparaison des stratégies en terme de consommation de ressources, les coûts en terme de temps d'évaluation de la requête ont également êté comparés. En effet, dans un environnement dynamique où le client qui a émis la requête est mobile, il est important de minimiser ce temps de réponse. Les stratégies 3 et 4 permettent d'effectuer l'évaluation distribuée de la requête classique parallèlement à l'évaluation de la localisation géographique du client. Elles fournissent donc, de meilleurs temps d'exécution que les stratégies 1 et 2. En outre, dans la stratégie 4, l'évaluation de l'opérateur de localisation est effectuée sur le nœud de service. L'évaluation de l'opérateur est donc effectuée plus rapidement que dans la stratégie 3 où c'est le nœud client qui effectue cette évaluation.

3.4.5. Synthèse

Une des contraintes importantes des terminaux nomades est l'autonomie limitée. En effet, il est important de minimiser les opérations effectuées par les terminaux nomades et coûteuses en terme de consommation d'énergie. Parmi ces opérations, les communications envoyées ou reçues par un terminal nomade sont particulièrement coûteuses. La stratégie 2 permet de réduire ces coûts de communication et donc l'énergie dépensée par les terminaux nomades si l'espace de recherche n'est pas réduit. En revanche, si la sélection de l'espace de recherche n'est représentée que par des nœuds centraux, dans ce cas, la stratégie 4 est alors la moins coûteuse.

En terme de temps de réponse, les stratégies 3 et 4 sont plus intéressantes. En particulier, la stratégie 4 qui permet de ne pas effectuer l'évaluation de l'opérateur de localisation sur le nœud client.
Cependant, le choix et l'utilisation d'un nœud de service présentent différentes limites. La présence d'un nœud suffisamment disponible pour accepter d'effectuer un traitement pour un autre nœud n'est pas assurée. De plus, si un tel nœud existe, il risque de devenir rapidement surchargé si le nombre de nœuds légers autour de lui est important.
Dans le cas où aucun nœud de service n'est disponible, la stratégie 1 est la plus adaptée si l'espace de recherche n'est pas restreint. En effet, les communications nécessaires à l'exécution de la stratégie 1 sont moins coûteuses que celles pour la stratégie 3. En revanche, si l'espace de recherche est réduit, dans ce cas, la stratégie 3 est alors moins coûteuse et elle permet également de minimiser le temps de réponse. Le tableau 5.5 décrit les stratégies les plus adaptées en fonction de l'espace de recherche choisi et en fonction de la présence d'un nœud de service.

	Présence d'un nœud de service	*Pas de nœud de service disponible*
Espace de recherche non restreint	Stratégie 2	Stratégie 1
Espace de recherche restreint	Stratégie 4	Stratégie 3

Tableau 5.5. Sélection des stratégies

Les différentes stratégies ont été testées sur un espace de recherche non limité regroupant à la fois les nœuds centraux et les nœuds légers de l'application et également sur deux sélections de l'espace de recherche (nœuds centraux et «bons » nœuds). Dans la section suivante, nous discutons de l'impact de ces sélections sur les différents coûts.

3.4.6. Impact de la sélection de l'espace de recherche

La sélection de l'espace de recherche et le choix de la profondeur maximale permettent de réduire de façon importante les différents coûts associés à l'évaluation d'une requête dépendante de la localisation. Nous présentons dans cette section la réduction des coûts obtenus puis nous discutons ensuite de l'impact de ces choix sur la qualité de la réponse obtenue.

Réduction des coûts

Les sélections de l'espace de recherche permettent de réduire considérablement les coûts de communication, de traitement et de temps de réponse puisque le nombre de nœuds visités est réduit. Les graphiques illustrés dans les figures 5.11 et 5.12 montrent la réduction du nombre de nœuds visités et donc du nombre d'évaluation de la requête dans les deux environnements présentés ci-dessus (environnement de référence et environnement «réél ») :

- a : l'espace de recherche est constitué de tous les nœuds de l'application
- b : l'espace de recherche est constitué uniquement des nœuds centraux

- c : l'espace de recherche est constitué uniquement des « bons » nœuds

Les stratégies 1 et 3 (respectivement les stratégies 2 et 4) sont regroupées étant donné que le nombre de nœuds visité pour ces deux stratégies est identique.

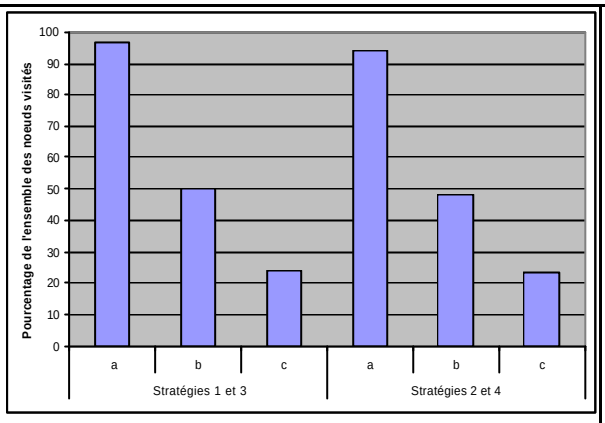


Figure 5.11. Nombre de nœuds visités en fonction des différentes sélections de l'espace de recherche dans l'environnement de référence

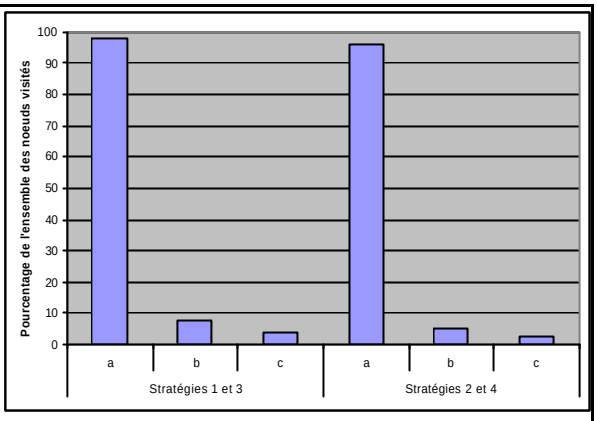


Figure 5.12. Nombre de nœuds visités en fonction des différentes sélections de l'espace de recherche dans l'environnement « réel »

Le choix de la profondeur permet lui aussi de réduire le nombre de nœuds visités et donc les coûts associés à l'évaluation d'une requête dépendante de la localisation. Cependant, dans les applications de proximité, étant donné que la sphère de communication est limitée à un espace déterminé, au-delà de quelques sauts, l'ensemble des nœuds ont été accédés et la requête ne doit plus et ne peut plus être envoyée et évaluée par d'autres nœuds. Le choix de la profondeur n'a donc un impact sur les coûts que si la profondeur est inférieure à un certain seuil dépendant de la densité de nœuds et de leur couverture réseau dans chaque environnement. Par exemple, d'après les résultats des simulations présentés dans la figure 5.13, nous pouvons remarquer que les coûts sont réduits uniquement si la profondeur est inférieure à 4. Au delà, le paramétrage de la profondeur n'a plus du tout d'impact sur les résultats et sur les coûts.

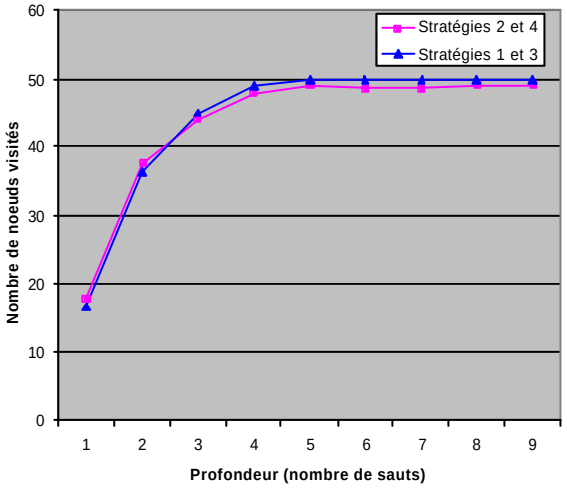


Figure 5.13. Influence de la profondeur sur le nombre de nœuds visités

Discussion sur la qualité de la réponse

Si limiter l'espace de recherche et la profondeur maximale permet de réduire de façon considérable les coûts, cela peut également engendrer une perte en terme de qualité de réponse obtenue. Ainsi dans le pire des cas, si la réponse à la requête se situe sur un nœud léger et que l'espace de recherche choisi est l'ensemble des nœuds centraux, la requête ne sera pas satisfaite. Ce cas n'est cependant pas exceptionnel dans une application de proximité : par exemple, dans l'application de l'aéroport, si Sophie recherche la localisation de Marc, celle-ci n'est a priori partagée que par le terminal nomade de Marc.

Si un pourcentage important des informations est stocké sur les nœuds centraux, le compromis entre la qualité de la réponse et les coûts nécessaires à l'évaluation de la requête en utilisant une restriction de l'espace de recherche aux nœuds centraux peut être intéressant. Cependant, dans un environnement « réel », les nœuds centraux ne sont pas toujours accessibles directement par le nœud client : dans ce cas, les nœuds de l'espace de recherche sélectionné ne sont pas tous contactés et donc la perte de qualité de la réponse est encore plus importante. Les figures 5.14 et 5.15 représentent le pourcentage de nœuds contactés parmi les nœuds de la sélection de l'espace de recherche respectivement dans l'environnement de référence et dans l'environnement « réel » considéré. Nous pourrions donc envisager de paramétrer la sélection de l'espace de recherche appropriée en fonction du nombre de connexions que possède un nœud client vers des nœuds centraux.

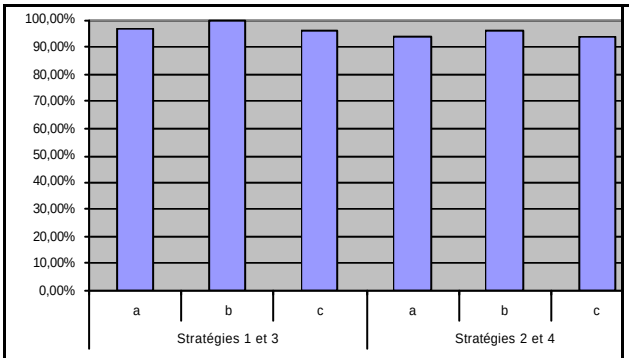


Figure 5.14. Pourcentage du nombre de nœuds visités parmi les nœuds de la sélection de l'espace de recherche dans l'environnement de référence

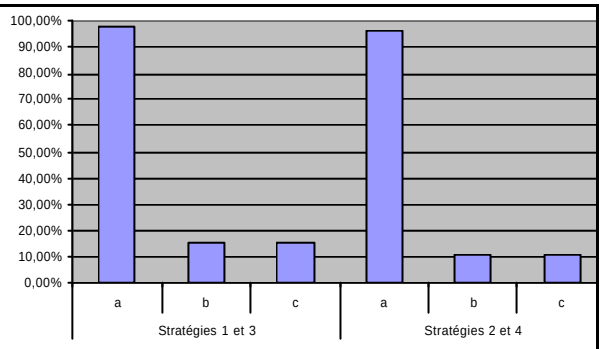


Figure 5.15. Pourcentage du nombre de nœuds visités parmi les nœuds de la sélection de l'espace de recherche dans l'environnement « réel »

Aujourd'hui, quantifier la perte de qualité des réponses obtenues reste problématique. En effet, les simulations réalisées ont été calibrées en utilisant des données expérimentales issues de notre prototype. Ces simulations nous permettent d'analyser la montée en charge. En revanche, nous ne disposons pas d'informations probantes sur le comportement des utilisateurs dans une application de proximité : quelles informations sont-ils prêts à échanger ? Combien d'informations veulent-ils échanger ? Existe-t-il des comportements représentatifs de partage de données pour les nœuds centraux ? De ce fait, nous ne sommes pas en mesure de calibrer nos simulations de façon « réelle ». Il serait intéressant, de quantifier les comportements d'utilisateurs d'une application de proximité dans un

environnement réel afin d'être capable d'exprimer la perte moyenne de qualité de la réponse en fonction de la sélection de l'espace de recherche. Des études similaires concernant les comportements des utilisateurs dans les systèmes P2P d'Internet ont été réalisées et permettent ainsi d'effectuer des simulations en considérant des modèles types de distribution de l'information : par exemple, dans [CRE 02], les simulations effectuées concernant la solution proposée d'optimisation de la recherche d'informations reposent sur une distribution 80/20 de l'information dans le système P2P. C'est-à-dire que 80% des documents disponibles sont stockés par 20% des nœuds présents.

3.5. Comparaison avec d'autres approches

Différentes solutions ont été proposées ces dernières années concernant l'optimisation de la distribution de requêtes dans les réseaux P2P. Dans cette section, nous positionnons nos travaux par rapport à ces différents travaux.

L'optimisation de la recherche d'informations dans les réseaux P2P est actuellement un domaine de recherche très actif. En effet, le développement des applications P2P et leurs lacunes au niveau des performances encouragent les propositions de nouvelles solutions optimisant à la fois les temps et les ressources alloués pour la recherche mais également les réponses obtenues. Les solutions actuelles reposent sur différentes techniques telles que les variations de politiques de parcours, l'utilisation de métadonnées telles que des statistiques ou des index, l'utilisation de cache ou encore la réplication. L'objectif principal de ces différentes techniques est de minimiser le nombre de messages transitant sur le réseau en limitant le nombre de nœuds visités pour l'évaluation d'une requête. Ce nombre de nœuds est donc, suivant les différentes techniques, sélectionné afin de répondre de façon optimale à la requête émise. Limiter le nombre de nœuds visités par une requête permet de réduire le temps de réponse et de limiter l'utilisation de la bande passante du réseau.

Nous présentons dans cette section certaines solutions basées sur une ou plusieurs des techniques ci-dessus. Premièrement, la solution proposée par Yang et al. [YAN 01], appelée itération sur la profondeur (iterative deepening en anglais) propose une adaptation du parcours en largeur initialement utilisée dans le réseau Gnutella. Le principe de cette solution est d'exécuter différents parcours en largeur d'abord en augmentant successivement les profondeurs limites de ces parcours. La recherche est arrêtée lorsque la requête est satisfaite ou lorsque la profondeur maximale est atteinte. Une politique de parcours P = {a, b, c} spécifie les profondeurs successives des parcours en largeur. Les tests effectués par les auteurs démontrent l'intérêt d'une telle méthode : celle-ci permet d'obtenir satisfaction pour la requête tout en limitant le nombre de nœuds explorés par rapport à une recherche basée sur un parcours en largeur d'abord classique. Cependant, les temps de réponse obtenus demeurent très importants étant donné que la recherche peut être réitérée plusieurs fois.

Proposée également par Yang et al. [YAN 01], la technique de parcours en largeur dirigé permet d'envoyer directement la requête à un sous-ensemble de nœuds sélectionnés afin de recevoir de nombreuses réponses rapidement. Pour sélectionner intelligemment ce sous-

ensemble de nœuds, chaque nœud maintient localement des statistiques sur ses voisins. Dans l'article [YAN 01], les statistiques proposées sont simples : par exemple le nombre de résultats obtenues par chaque voisin pour les requêtes précédentes ou la latence de la connexion avec chaque voisin. En fonction de ces statistiques, différentes heuristiques sont développées pour sélectionner les meilleurs voisins vers qui envoyer la requête. Des exemples de ces heuristiques sont : quels sont les nœuds voisins qui ont renvoyés le plus grand nombre de résultats ?, quels sont les nœuds voisins qui ont les moins longues files de messages en attente ? etc. Les expérimentations effectuées montrent que les coûts sont largement réduits étant donné que le nombre de nœuds visités peut être fortement diminué et la qualité des réponses obtenues ne décroît pas de façon significative par rapport à un parcours classique.

De nombreuses solutions utilisent des métadonnées pour optimiser l'évaluation de la requête. Des solutions telles que la stratégie des index locaux ou encore les stratégies de recherche locale présentées dans [ADA 01] sont basées sur l'utilisation d'index sur chaque nœud. Chaque nœud stocke localement un index du contenu d'autres nœuds présents dans le système. Ces solutions permettent aux requêtes d'être évaluées sur une large fraction des informations disponibles tout en limitant le nombre de nœuds réellement visités par la requête. Cependant, ces solutions présentent différentes contraintes. Les nœuds doivent, ainsi, accepter de stocker des informations supplémentaires sur leurs disques. De plus, les mises à jour des index peuvent devenir coûteuses et difficiles dans un environnement fortement dynamique.

En opposition avec les solutions basées sur l'utilisation d'index sur le contenu, Crespo et al. [CRE 02] proposent d'utiliser des index pour définir le parcours ou le « routage sémantique» de la requête sur les différents nœuds du système. Cette solution permet aux nœuds d'avoir des index de taille souvent plus restreinte que pour les index de contenu étant donné qu'ils seront proportionnels au nombre de voisins et pas aux nombres d'informations. L'index définit, pour chaque voisin, le nombre de documents accessibles. Pour faciliter la recherche, les différents documents sont classés par domaine. Un exemple d'index de routage est décrit dans le tableau 5.6 : l'index décrit est l'index du nœud A. Le nœud A a deux voisins B et C. Si nous recherchons un document sur les bases de données, le nœud A va envoyer la requête au nœud B d'abord car il a accès à un plus grand nombre de document concernant ce sujet. Cette solution permet de limiter les coûts de recherche d'information tout en assurant une réponse satisfaisante. A l'inverse des solutions utilisant les index sur le contenu, la taille des index est limitée, cependant, cette solution demande la mise en place de mécanismes de mises à jour d'index complexes et parfois coûteux si l'environnement est très dynamique.

L'article [JOS 03] résume les différentes approches d'utilisation des métadonnées dans les recherches P2P. Les avantages en terme de performances sont souvent démontrés cependant l'utilisation de métadonnées issues d'autres nœuds nécessite la mise en place de mécanisme pour maintenir la cohérence de ces métadonnées.

	Nb total de docs	Bases de données	Réseaux	Systèmes répartis
A	100	0	80	10
B	150	20	10	30
C	60	0	0	50

Tableau 5.6. Un exemple de tableau de correspondance pour le coefficient de similarité entre 2 localisations

Les solutions proposées actuellement pour optimiser la recherche dans les réseaux P2P sont principalement dédiées aux systèmes P2P de l'Internet tels que Gnutella. Elles ont donc pour principal objectif d'optimiser les coûts tout en maintenant la qualité des réponses obtenues. Les coûts à optimiser au sein des systèmes P2P sont principalement les coûts liés à l'envoi de la requête et à la réception des messages. Les solutions cherchent à minimiser l'utilisation de la bande passante et l'utilisation de la puissance CPU des différents nœuds utilisée lors de l'envoi et la réception des messages. Ces solutions ne considèrent pas comme une contrainte l'espace de stockage disponible sur les différents nœuds. Or, dans les applications de proximité, les terminaux nomades peuvent être incapable ou refuser de stocker des informations supplémentaires telles que des métadonnées décrivant les données de ses voisins. De plus, l'environnement dynamique des applications de proximité rend impossible le maintien de la cohérence d'un cache ou de métadonnées.

De plus, la recherche d'informations dans les systèmes P2P se limite souvent à une recherche de fichiers (audio par exemple). De ce fait, aucun coût d'évaluation de requête n'est pris en compte. Dans le service de localisation ISLANDS, tout type d'informations peut être recherché et un langage d'interrogation évolué est utilisé. L'évaluation de ces requêtes ajoute donc un coût non négligeable dans la recherche d'informations, en particulier si cette requête est évaluée sur des terminaux nomades.

4. Conclusion

Nous avons présenté dans ce chapitre, le modèle d'évaluation des requêtes dépendantes de la localisation. Différentes stratégies d'évaluation ont été proposées afin d'optimiser les coûts nécessaires à l'évaluation d'une requête dépendante de la localisation. Ces stratégies reposent sur différents choix d'ordonnancement des différentes étapes et sur lutilisation d'un nœud de service. Ces stratégies ont été testées afin de déduire les coûts pour chacune d'elles et de déduire l'environnement auquel elles s'adaptaient le mieux. Nous avons également discuté de l'impact de la sélection de l'espace de recherche dans ces stratégies. Il serait intéressant de pouvoir expérimenter une application de proximité en environnement réel afin d'en déduire certaines informations concernant le comportement des différents utilisateurs. En effet, nous pourrions par exemple, définir la quantité moyenne d'informations partagées, la fréquence des requêtes émises, les évolutions du nombre de nœuds en fonction des heures de la journée, etc. Ces statistiques nous permettraient d'affiner nos choix en terme de stratégies d'évaluation et de sélection de l'espace de recherche.

Chapitre 6

Prototype

Dans les chapitres précédents, nous avons décrit les différents composants du service de localisation ISLANDS adapté au contexte des applications de proximité. Dans ce chapitre, nous présentons le prototype réalisé. Ce prototype est constitué de deux parties distinctes : l'implantation du service ISLANDS et l'implantation de la partie applicative correspondant à l'application de proximité choisie. Notre prototype a été démontré au 19èmes journées de Bases de Données Avancées [THI 03b] organisées à Lyon en 2003. L'objectif principal qui a motivé son implémentation a été de valider notre approche. Ce prototype nous a ainsi permis de prouver que notre service est adapté et efficace dans une application de proximité. La première version de notre prototype nous a également permis de mettre en évidence certaines limites de notre solution, d'en définir les causes et de permettre ainsi de proposer les améliorations et les optimisations nécessaires.

Ce chapitre est organisé de la manière suivante : dans la section 1, nous présentons le contexte applicatif choisi pour le développement de notre prototype. Nous présentons dans la section 2 l'architecture globale du prototype. La section 3 décrit l'évaluation des requêtes de localisation. Finalement, la section 4 présente les différents résultats obtenus.

1. Contexte applicatif

Le contexte applicatif choisi pour le développement du prototype d'ISLANDS est le commerce électronique de proximité [THI 01]. Cette catégorie d'applications de proximité s'appuie sur un environnement composé de vendeurs et de clients potentiels tel qu'une galerie marchande par exemple. Les clients sont les usagers de la galerie marchande : s'ils sont munis d'un terminal nomade, ils peuvent participer à l'application de commerce électronique de proximité. Comme pour l'exemple d'applications de proximité se déroulant dans un aéroport, cette application repose sur une architecture fixe représentée par les différents vendeurs présents dans la galerie marchande. Les nœuds centraux sont alors représentés par les différents vendeurs présents dans la galerie marchande tandis que les nœuds légers sont représentés par les clients potentiels munis d'un terminal nomade. Dans une application de commerce électronique de proximité, un client, entrant dans la sphère de

communication va pouvoir d'une part recevoir les offres diffusées par les marchands correspondant à ses préférences et d'autre part émettre des requêtes. Ces requêtes peuvent être de simples requêtes telles que «quels sont les prix proposés pour les iPAQs H3950 ? » ou des requêtes de localisation telles que « quel est le magasin de musique le plus proche de moi ? ».

2. Description des différents composants

Dans une application de proximité, le service ISLANDS est distribué sur chaque nœud de l'application De ce fait, une version du prototype est déployée sur chaque nœud. L'architecture du prototype implémenté est décrite dans la figure 6.1. Le prototype repose sur quatre composants principaux :

- l'interface utilisateur : elle fournit à l'utilisateur un moyen de composer sa requête et de visualiser les différents résultats.
- le gestionnaire de stockage des informations : les différentes informations partagées par l'utilisateur y sont stockées.
- le module de positionnement : il fournit la localisation du client.
- l'évaluateur de requêtes d'ISLANDS : le service de localisation ISLANDS permet la diffusion d'informations par les vendeurs mais il fournit un évaluateur dédié à l'évaluation des requêtes émises par les différents participants dans l'application de proximité.

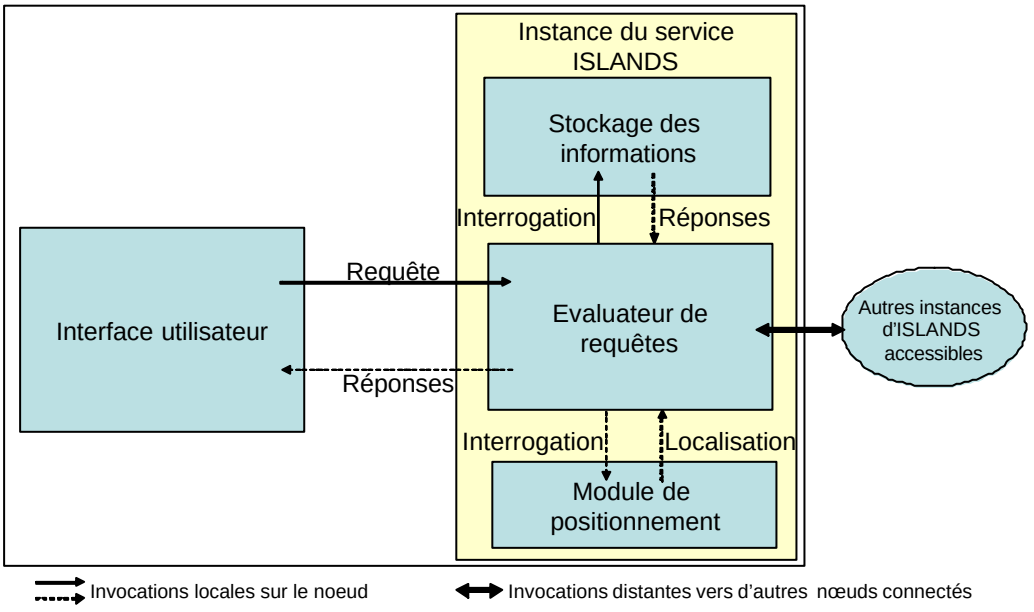


Figure 6.1. Composants de l'implémentation du prototype

Lorsqu'un client émet une requête : le client commence par saisir la requête via l'interface utilisateur disponible sur son terminal. La requête saisie est ensuite envoyée au service de localisation ISLANDS. S'il s'agit d'une requête dépendante de la localisation, le service

interroge le module de positionnement du nœud pour récupérer la localisation géographique de l'utilisateur. Ensuite, l'évaluation de la requête est gérée par le service qui interroge les informations partagées localement par le client mais également par les autres nœuds de l'application en utilisant les différentes instances d'ISLANDS.

2.1. Interface utilisateur

Dans le prototype, deux séries d'interfaces différentes ont été réalisées. La première est destinée aux nœuds centraux et la seconde est adaptée aux nœuds légers. En effet, dans une application de commerce électronique de proximité, les fonctionnalités proposées aux différents nœuds sont différentes en fonction du type de nœuds (central ou léger). De même, les contraintes d'affichage sont différentes et elles interviennent donc dans la conception des différentes interfaces. Les nœuds centraux représentent les vendeurs présents dans la galerie marchande donc les différentes fonctionnalités à développer sont :

- la saisie/modification/suppression d'une offre (promotions, soldes etc.)
- la diffusion d'offres
- la saisie/modification/suppression d'un article ou d'un produit

Les interfaces développées pour les vendeurs sont statiques : la génération dynamique d'interfaces pour les vendeurs n'est pas nécessaire. Les fonctionnalités sont fixées et définies pour l'application : les vendeurs peuvent diffuser des informations, enregistrer, modifier ou supprimer de nouveaux produits et de nouvelles offres. Un exemple d'interface utilisée par les nœuds centraux est illustré par la figure 6.2. Cette figure représente l'interface utilisée pour la saisie d'un nouveau produit par un vendeur.

Figure 6.2. Interface utilisateur pour les nœuds centraux

En revanche, la série d'interfaces dédiée aux nœuds légers repose sur un fichier de configuration. Ce fichier XML décrit les différentes étapes et enchaînements nécessaires à l'expression d'une requête par un client. Ce fichier de configuration est diffusé dans la galerie marchande et permet de générer des interfaces en fonction de l'environnement de

l'application de proximité. Un extrait du fichier de configuration utilisé dans l'application de commerce électronique de proximité est décrit dans la figure 6.3.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<CEP>
   <Request>
      <Produit type-directory="objectClass" id-directory="Product">
         <Nom type-directory="attribute" id-directory="name">
            <Vendeur type-directory="objectClass" id-directory="Store">
               <Nom type-directory="attribute" id-directory="name" />
               <Type type-directory="attribute" id-directory="type" />
            </Vendeur>
         </Nom>
         <Type type-directory="attribute" id-directory="type">
            <Vendeur type-directory="objectClass" id-directory="Store">
               <Nom type-directory="attribute" id-directory="name" />
               <Type type-directory="attribute" id-directory="type" />
            </Vendeur>
         </Type>
      </Produit>
      <...>
   </Request>
   <Localization>
      <Close>
         <Store type-directory="objectClass" id-directory="Store">
            <Name type-directory="attribute" id-directory="name" />
            <Type type-directory="attribute" id-directory="type" />
         </Store>
      </Close>
      <...>
   </Localization>
<CEP>
```

Figure 6.3. Fichier de configuration des interfaces graphiques pour la saisie des requêtes

Ce fichier contient deux parties, la première consiste à représenter les choix possibles pour la requête classique (i.e. non dépendante de la localisation), la seconde permet, quant à elle de saisir les informations relatives aux requêtes de localisation. Dans l'annexe 6, le fichier de configuration utilisé est représenté dans son intégralité.

Les différents nœuds légers représentent des clients munis de terminaux nomades. Cependant, ces terminaux offrent des capacités d'affichage hétérogènes : noir et blanc, niveaux de gris, couleurs, taille de l'écran, etc. La saisie d'une requête est donc réalisée par une série d'interfaces simples guidant l'utilisateur. La figure 6.4 représente la série d'interfaces générées grâce au fichier de configuration et utilisées pour exprimer la requête « Quels sont les fast-foods situés à moins de 50 mètres de moi ? ». L'utilisateur peut naviguer au travers de ces différentes interfaces par les boutons « Back » et « Next » comme il en a l'habitude sur les terminaux légers.

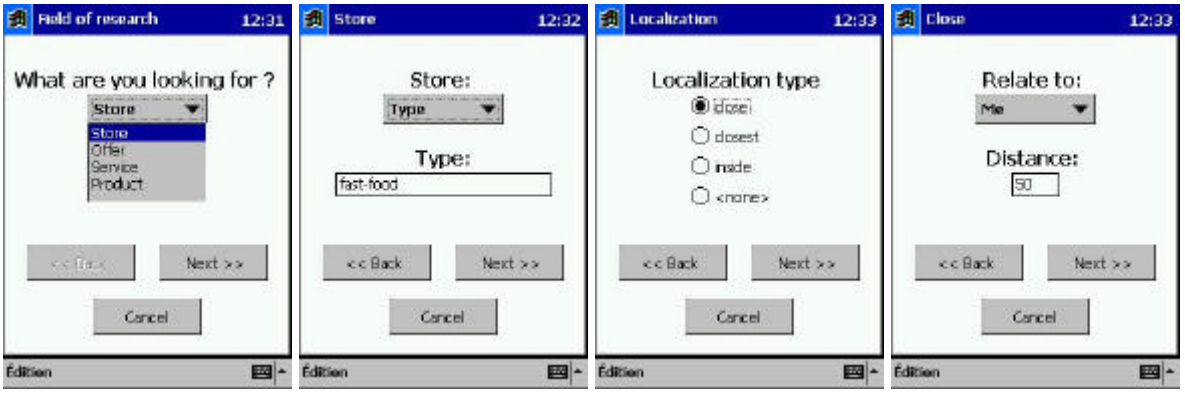


Figure 6.4. Interfaces utilisateur pour les nœuds légers

2.2. Service de localisation ISLANDS

Le service de localisation est composé de trois parties :

- le module de positionnement : il permet de récupérer la localisation du client. Cette localisation est, soit issue d'une technique de géo-localisation tel que le GPS, soit issue de notre solution de localisation décrite également dans le chapitre 4. Ce module de positionnement fournit au service de localisation une représentation XML (localisation physique et/ou localisation symbolique) de la localisation du client.
- le gestionnaire de stockage : dans le prototype implémenté, les informations des nœuds centraux sont stockées au sein de l'annuaire LDAP OpenLDAP. Les services d'annuaire nous permettent notamment de privilégier l'accès aux données en lecture plutôt qu'en écriture. Les informations des nœuds légers sont quant à elles stockées au sein de fichiers XML.
- l'évaluateur de requêtes : il est quant à lui composé de 5 composants :
    - le parseur de la requête qui permet de décomposer les requêtes, en particulier les requêtes de localisation.
    - l'évaluateur de la requête qui permet l'évaluation d'une requête sur les données stockées localement.
    - l'évaluateur de l'opérateur de localisation qui permet d'évaluer les différents opérateurs de localisation comme nous l'avons décrit dans le chapitre 5.
    - le composant permettant de gérer la distribution de la requête au sein de la sphère de communication.
    - le composant permettant la construction finale du résultat en fonction des différents paramètres précisés par le client tels que le nombre de réponses souhaitées.

L'évaluateur de requête présent dans ISLANDS peut être adapté en fonction des ressources disponibles sur le nœud qui l'héberge. En effet, dans le cas d'un nœud léger, seule une version dégradée de l'évaluateur peut être utilisée. Cette version dégradée d'ISLANDS comporte le parseur de la requête qui permet entre autre de définir s'il s'agit d'une requête dépendante de la localisation ou non et le composant permettant de gérer la distribution de la

requête. Ainsi, si le nœud client ne bénéficie que de la version dégradée, la requête est directement envoyée aux autres instances d'ISLANDS avec le paramètre contenant la localisation géographique de l'utilisateur s'il s'agit d'une requête dépendante de la localisation.
Par contre, les nœuds centraux peuvent, quant à eux, bénéficier d'une version élaborée d'ISLANDS comprenant en plus un composant permettant la diffusion d'informations sur la sphère de communication.
Dans la suite, nous présentons en détail l'évaluation d'une requête au sein d'ISLANDS puis nous décrivons l'infrastructure de communication utilisée.

3. Gestion des requêtes

Pour l'évaluation des requêtes dans ISLANDS, les données stockées sur les nœuds centraux dans des services d'annuaires sont exportées en DSML[1] (DSML – Directory Services Markup Language). Le langage DSML permet de représenter les informations contenues dans un service d'annuaires dans un format structuré dérivé du XML. Ce langage permet également l'expression et l'évaluation de requêtes : les requêtes sont alors exprimées selon le format « DSML Request ». Ce choix de langage est dû en particulier à l'utilisation des annuaires comme stockage des informations sur les nœuds centraux. Il permet de fournir un langage d'interrogation uniforme pour les informations stockées sur les nœuds légers et centraux.

```
<?xml version="1.0" encoding="UTF-8"?>
<dsml:searchRequest dn="o=PeerDirectoryService"
xmlns:dsml="urn:oasis:names:tc:DSML:2:0:core">
  <dsml:filter>
    <dsml:and>
      <dsml:equalityMatch name="objectClass">
        <dsml:value>Store</dsml:value>
      </dsml:equalityMatch>
      <dsml:equalityMatch name="type">
        <dsml:value>fast-food</dsml:value>
      </dsml:equalityMatch>
    </dsml:and>
  </dsml:filter>
  <dsml:localization>
    <dsml:close>
      <dsml:location>me</dsml:location>
      <dsml:distance>50</dsml:distance>
    </dsml:close>
  </dsml:localization>
</dsml:searchRequest>
```

Figure 6.5. Exemple d'expression de requêtes au format DSML Request étendu

Pour permettre l'expression des opérateurs de localisation, le format standard d'expression des requêtes a été étendu. L'exemple présenté dans la figure 6.5 permet d'exprimer une requête de localisation utilisant l'opérateur *close*. La description de l'expression de l'opérateur de localisation est effectuée en fonction des différents paramètres de l'opérateur.

[1] http://www.oasis-open.org/committees/dsml/

Les différentes étapes nécessaires à l'évaluation d'une requête dépendante de la localisation sont décrites dans la figure 6.6. Premièrement, la requête DSML étendue est soumise au service de localisation ISLANDS. Le parseur de requête effectue alors le découpage de la requête pour séparer la requête standard de l'opérateur de localisation. La requête standard est évaluée par l'évaluateur de requêtes DSML standard qui interroge les informations stockées localement en DSML. Parallèlement, le module de positionnement évalue la localisation de l'utilisateur et la localisation de référence est alors paramétrée dans la requête DSML étendue. Ces informations sont récupérées par l'évaluateur de l'opérateur de localisation. Si celui-ci n'obtient pas de réponse satisfaisante, la requête DSML étendue avec le paramètre de localisation renseigné est alors envoyée aux services de localisation distants. Finalement le constructeur de réponses prend en charge l'ensemble des réponses, sélectionnées en fonction des préférences de l'utilisateur.

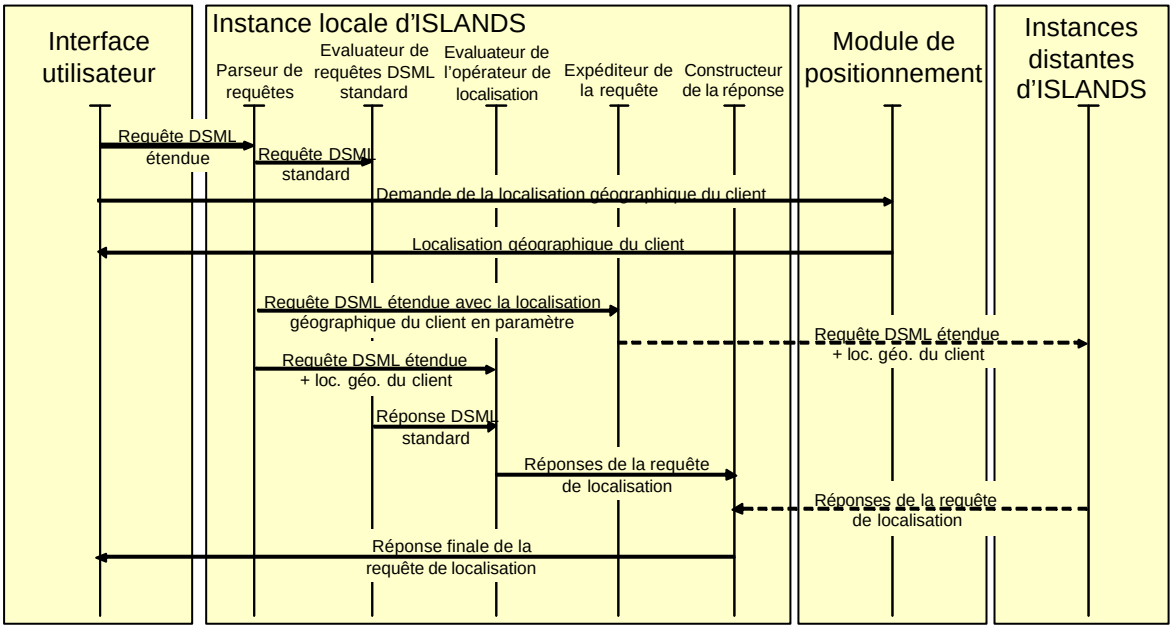


Figure 6.6. Evaluation d'une requête dépendante de la localisation

4. Description de l'infrastructure de communication

Pour développer le prototype, les différentes infrastructures de communication P2P ont été étudiées. Nous présentons dans cette section, les infrastructures existantes, leurs avantages et leurs limites. Nous expliquons aussi pourquoi nous n'avons pas utilisé d'infrastructure existante et nous décrivons les différentes technologies choisies.

4.1. Infrastructures existantes

Pour faciliter le développement des applications P2P, de nombreuses plateformes sont apparues ces dernières années. Aujourd'hui, les implémentations d'applications P2P telles que Napster ou Gnutella ne sont pas interopérables entre elles. C'est principalement pour

assurer l'interopérabilité entre ces différents systèmes que différentes plateformes ont été proposées pour faciliter le développement et le déploiement d'applications P2P. Cependant, beaucoup d'entre elles sont dédiées à un type d'applications P2P défini.

Par exemple, la plateforme XTremWeb (XW) [FED 01] est une plateforme qui permet de développer des applications P2P de calcul distribué académique sur Internet orienté hautes performances. Il permet à différents centres de recherches, universités et industriels d'installer et d'utiliser leur propre système de calcul P2P pour leurs travaux de recherches ou pour la production de calcul.

De même, la plateforme NEMO [BRA 04] (Networked Environment for Media Orchestration) est une plateforme P2P expérimentale qui a pour objectif de permettre à différents utilisateurs d'avoir accès à différents médias (musique, vidéos, etc.). Les deux objectifs principaux de cette plateforme sont d'une part de permettre l'interopérabilité des différents standards existants et d'autre part d'assurer la sécurisation et le contrôle d'accès aux médias. L'interopérabilité permet, par exemple de pouvoir découvrir un service disponible en utilisant des standards disponibles comme Jini, UPnP/SDP ou Salutation. Dans cette plateforme, les noeuds peuvent être à la fois des entités physiques comme un PDA ou des entités logiques comme des agents logiciels. Ces nœuds peuvent fournir et/ou consommer des services. NEMO est une plateforme intéressante du point de vue interopérabilité, cependant, elle est encore en phase expérimentale et est exclusivement dédiée au domaine des médias.

Toutes ces plateformes sont très intéressantes lorsqu'il s'agit d'applications P2P bien définies où les utilisateurs connaissent le type d'informations qu'ils souhaitent échanger (des médias : fichiers audio et vidéo dans le cas de NEMO). Dans le cas des applications de proximité, le type des informations échangées n'est pas défini : en effet, il peut aussi bien s'agir de fichiers textes, de fichiers audio que de services tels qu'un service d'imprimantes par exemple. A notre connaissance, une seule plateforme fournit une infrastructure permettant de développer des applications P2P sans réellement définir le types d'applications visées : la plateforme JXTA [JXTA, GON 02]. JXTA est une plateforme open source proposée par Sun visant à faciliter le développement, le déploiement et la gestion d'applications P2P. Elle a pour principaux objectifs d'assurer l'interopérabilité pour simplifier l'interconnexion entre les différents participants, la localisation et la communication entre les participants, l'indépendance vis-à-vis des plateformes logicielles et matérielles et l'ubiquité.

Comme l'illustre la figure 6.7, l'architecture de la plateforme JXTA est séparée en 3 niveaux principaux : le niveau inférieur (couche noyau) regroupe les fonctions essentielles telles que la sécurité ou la gestion de groupes et des communications entre membres de la communauté de noeuds JXTA, à un niveau intermédiaire (couche services), ces fonctionnalités de base peuvent servir à créer des services d'indexation, de recherche, de partage etc. Et enfin, basé sur ces deux niveaux, différentes applications P2P pourront être développées.

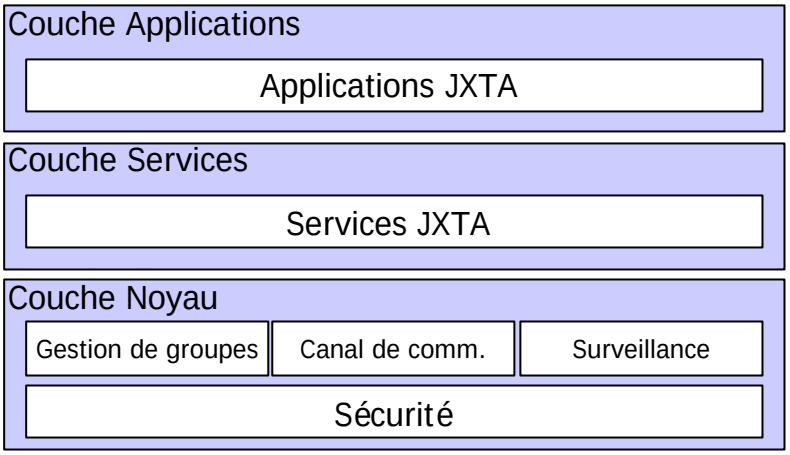


Figure 6.7. Architecture logique de la plateforme JXTA

La plateforme JXTA permet de faciliter le développement, le déploiement et la gestion d'applications P2P particulièrement en fournissant de nombreux services utiles. Cependant, sa compréhension et son utilisation étant encore très complexes, les temps réels de développement des applications ne sont finalement pas réduits [HAL 02, WAL 04]. De plus, un des objectifs de la plateforme JXTA est de permettre l'indépendance vis-à-vis des plateformes matérielles. Pourtant, le portage de la plateforme sur des terminaux nomades évolués demeure très complexe et constitue une difficulté importante à considérer dans le choix d'une telle plateforme. En effet, la version JXTA pour les terminaux nomades reposant sur J2ME n'est pas facilement utilisable. Elle repose de plus, sur l'utilisation de relais entre les nœuds JXTA classiques et les nœuds JXTA/J2ME. L'utilisation de ces relais est une contrainte et une condition supplémentaire à intégrer dans la gestion du service de localisation et demande de nombreuses modifications et adaptations du modèle de localisation proposé dans ISLANDS. En outre, seul le protocole HTTP est supporté par cette plateforme.
Ces limites ajoutées au fait que la plateforme est récente et qu'elle n'est vraiment exploitable que depuis quelques mois expliquent que nous n'ayons pas choisi cette plateforme comme base de travail pour développer notre prototype. Nous avons donc développé une infrastructure de communication pour notre service ISLANDS.

4.2. Choix d'implémentation

Le prototype a été réalisé en Java. Les nœuds légers sont représentés par des iPAQ H3950 et pour les nœuds centraux, nous avons utilisé des ordinateurs de type PIII. La machine virtuelle choisie est celle de Personal Java qui est adaptée au développement d'applications sur terminaux nomades. Le type de réseau utilisé est Wifi 802.11b et le réseau filaire a également été exploité pour la connexion entre les différents nœuds centraux. La distribution entre les différents services de localisation est gérée par l'API JNDI (JNDI – Java Naming Directory Service). Les communications entre les différents nœuds de l'application sont gérées en utilisant des sockets. En effet, l'utilisation intensive d'une technologie telle que les

Java RMIs (RMI – Remote Method Invocation) pose des problèmes de surcharge au niveau des terminaux nomades qui se bloquent alors rapidement.
Concernant la découverte de ses nœuds voisins, un serveur multicast est géré par chaque nœud de l'application. Ce serveur est activé lorsque l'utilisateur souhaite participer à l'application : rechercher ou partager de l'information. Ce serveur est implémenté à l'aide d'une socket en « écoute » sur une adresse de groupe multicast. L'adresse de groupe multicast est définie pour l'application. Lorsqu'un nœud souhaite mettre à jour son environnement, un message est envoyé sur cette adresse de groupe. Les nœuds connectés sont alors prévenus et envoient grâce à une socket unicast, leurs informations. Les mises à jour des informations concernant les voisins d'un nœud sont effectuées à intervalles définis par le profil de mobilité de l'utilisateur : plus l'utilisateur est mobile, plus l'intervalle de temps entre deux mises à jour est réduit.
La diffusion des informations au sein d'ISLANDS repose également sur l'utilisation de groupes multicasts auxquels les nœuds s'inscrivent s'ils sont susceptibles d'être intéressés par les informations diffusées dans un groupe particulier.

5. Synthèse

De nombreuses topologies peuvent être envisagées dans le cadre d'une application de proximité ; en effet, dans une application de commerce électronique de proximité, par exemple, le nombre de nœuds varie considérablement en fonction des heures de la journée. Pour évaluer notre solution de localisation, nous avons envisagé différents types d'environnements et testé notre solution dans ces environnements :

- Environnement dense où le nombre de noeuds est important (environ un noeud sur 2m² par exemple).
- Environnement peu dense
- Environnement bien connecté : où il n'existe qu'une seule sphère de communication pour l'ensemble des participants.
- Environnement peu connecté : plusieurs sphères existent et chaque participant peut ne pas être en mesure de communiquer avec tous les autres participants.
- Environnement où l'ensemble des localisations est exprimé en données physiques : ces localisations peuvent cependant être soit estimées (via notre algorithme), soit issues d'une technologie de géo-localisation.
- Environnement où l'ensemble des localisations est exprimé en données symboliques : ces localisations peuvent cependant être soit estimées (via notre algorithme), soit exactes (décrites par l'utilisateur).
- Environnement mixte où les localisations sont représentées à la fois par des données physiques et symboliques.

Pour chaque environnement, des tests d'évaluation des requêtes dépendantes de la localisation ont été effectués grâce au prototype implémenté. Les résultats obtenus sont globalement concluants. L'évaluation d'une LDQ basée sur notre solution d'estimation de la

localisation fournit de bons résultats dans la majorité des environnements. Toutefois, pour certaines configurations, notre solution peut présenter quelques limites que nous décrivons ci-dessous.

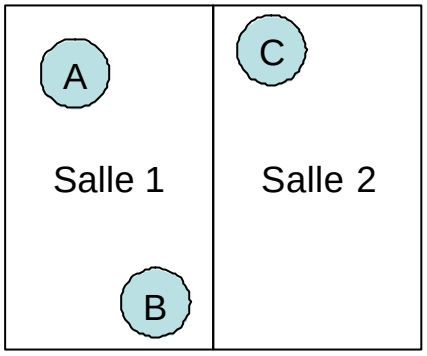


Figure 6.8.a. Exemple de répartitions

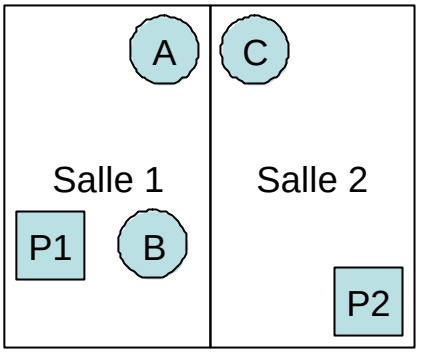


Figure 6.8.b

Une des premières constatations est que dans un environnement dense, il devient impossible de différencier les différents noeuds voisins lorsque l'on a atteint le grain de représentation le plus fin (la pièce par exemple), si la localisation est représentée en données symboliques. Cette constatation est cependant à relativiser : si on a atteint le grain de représentation le plus fin, la marge d'erreur est négligeable (quelques mètres si la structure choisie pour la représentation symbolique est finement décrite).

La deuxième constatation concerne la différence de résultats entre un traitement basé sur des localisations symboliques fixes et un traitement basé sur des localisations physiques issues d'une technologie de géo-localisation : nous obtenons des différences puisque pour la représentation en données physiques, nous calculons la distance exacte entre les différents noeuds sans prendre en compte les obstacles : murs, sols etc. Ainsi comme l'illustre la figure 6.8.a, si on recherche le voisin le plus proche de A: la réponse est B dans le cas d'une représentation symbolique et C sinon. En nous basant sur cette constatation, il peut être intéressant lorsque cela est possible, de choisir le type de données de localisation utilisé en fonction de l'environnement dans lequel on se situe, pour mieux répondre aux requêtes. Par exemple, s'il s'agit d'un grand hall, sans obstacle, où le grain utilisé pour la représentation en données symboliques n'est pas extrêmement fin, on favorise le traitement en données physiques et inversement.

La troisième constatation concerne l'estimation de la localisation basée sur des données symboliques : l'estimation utilise la « distance » qui sépare un noeud de ses différents voisins. La comparaison des fichiers est quant à elle une comparaison entre représentations symboliques. Si le noeud se trouve à la limite d'une section géographique, la réponse peut être erronée : par exemple, dans la figure 6.8.b, le noeud A recherche le produit P le plus proche de lui mais il ne connaît pas sa localisation. Donc, suivant l'algorithme d'estimation de la localisation, la localisation de A est représenté par une liste de couples (localisation, degré d'approximation) où la localisation de C risque d'être en première position avec le degré le plus petit. L'évaluation de la requête repose ensuite sur une comparaison des

données symboliques et donc, va retourner comme réponse que le produit P le plus proche est le produit P2 se situant dans la salle 2. Cependant, comme le calcul du degré d'approximation repose sur le signal entre deux machines, les obstacles sont pris en compte et cette erreur est souvent atténuée.

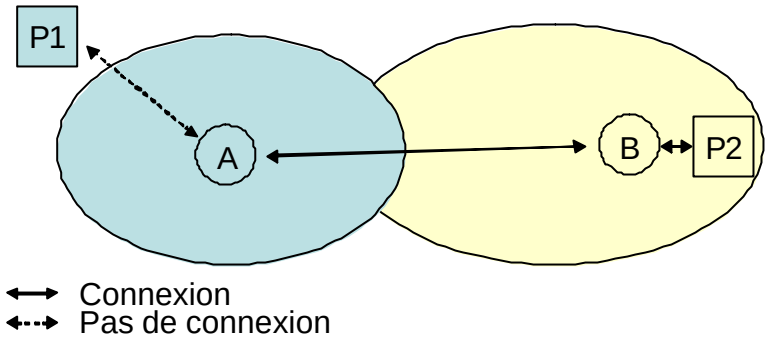


Figure 6.9. Exemple de répartition de noeuds

Et enfin, la dernière constatation est que dans un environnement faiblement connecté, notre algorithme d'estimation de la localisation peut entraîner des réponses erronées. Comme l'illustre la figure 6.9, si A recherche le produit P le plus proche de lui, sa localisation est basée sur la localisation de B puisque B est le seul noeud connecté à A, il obtient donc, en réponse, le produit P2.

6. Conclusion

Le prototype a permis dans un premier temps de valider le service de localisation ISLANDS et plus particulièrement le modèle d'évaluation des requêtes dépendantes de la localisation. Quelques limites ont été mises en évidence dans un environnement peu connecté : dans ce cas, l'application de proximité utilise plusieurs sphères de communication et l'accès à l'ensemble des informations disponibles dans l'application devient impossible.
La première version du prototype a également permis de mettre en évidence certaines limites et nous a aidé à définir quelles contraintes étaient à considérer pour proposer des optimisations adaptées. En effet, les contraintes en terme de temps de réponse et de consommation des ressources sont importantes à considérer dans les applications de proximité. En particulier, l'autonomie des terminaux nomades s'est avérée très problématique. D'autre part, les expérimentations effectuées grâce au prototype ont été utilisées pour le calibrage de nos simulations d'optimisations. Une version évoluée du prototype est actuellement en cours de développement. Cette version intègre l'ensemble des optimisations proposées dans les chapitres précédents.
Actuellement, nous étudions également la possibilité de développer et de déployer ce prototype au sein d'une gare de la région Nord Pas de Calais. L'application de commerce électronique de proximité doit être adaptée à l'environnement d'une gare. Cette expérience nous permettra de mettre en évidence et de résoudre d'autres problématiques intéressantes

telles que la montée en charge et le déploiement automatisé. Le matériel dont nous disposions ne nous a pas permis, pour le moment, d’aborder ces problématiques.

Chapitre 7

Conclusion & Perspectives

1. Conclusion

Dans cette thèse, nous avons étudié la problématique de localisation des informations au sein d'applications se déroulant dans un environnement contraint en terme de ressources, fortement réparti et très dynamique. Nous avons en particulier proposé un service de localisation adapté à l'environnement des applications de proximité. D'une part, le modèle de données proposé dans ISLANDS permet la localisation de tous types d'informations. D'autre part, une instance d'ISLANDS est déployée sur chaque nœud de l'application permettant ainsi de gérer la dynamicité de l'environnement en ne permettant l'accès à une information que si celle-ci est réellement disponible. Une instance d'ISLANDS peut, de plus, être adaptée en fonction des ressources du nœud sous-jacent. Finalement, le service ISLANDS fournit un évaluateur de requêtes adapté à l'environnement des applications de proximité. Cet évaluateur permet de gérer la distribution des requêtes au sein des différentes instances d'ISLANDS. Il repose surtout sur le modèle d'évaluation proposé permettant de prendre en compte la localisation géographique des utilisateurs. En effet, l'évaluateur de requêtes gère l'expression et l'évaluation des requêtes dépendantes de la localisation. D'une part, trois opérateurs de localisation ont été introduits afin d'exprimer ces requêtes. D'autre part, différentes stratégies ont été proposées afin d'évaluer ces requêtes en optimisant les coûts nécessaires à l'évaluation.

Pour permettre l'évaluation de ces requêtes dépendantes de la localisation, le service ISLANDS doit disposer de la localisation géographique du client qui émet la requête. Cette localisation peut être obtenue par des solutions de géo-localisation existantes. Cependant, comme ces solutions ne sont pas toujours disponibles et pas toujours utilisables, nous avons proposé une solution pour localiser un utilisateur mobile. Cette solution repose sur l'utilisation de l'environnement du client et en particulier sur ses voisins. Elle ne requiert donc pas de matériel et d'infrastructure particuliers. Cette solution a été optimisée afin de fournir la localisation d'un utilisateur mobile en minimisant le temps d'exécution et la consommation d'énergie des terminaux nomades.

Pour valider l'ensemble des solutions proposées, un prototype a été implémenté. La première version du prototype nous a permis de valider notre proposition de service de localisation ISLANDS. Il a également démontré la faisabilité de notre solution de localisation d'un

utilisateur mobile et de notre modèle d'évaluation des requêtes dépendantes de la localisation. Les mesures expérimentales issues de ce prototype ont permis d'une part de définir précisément les besoins en terme d'optimisations, et d'autre part de calibrer les simulations des optimisations proposées.
Finalement, le travail effectué dans cette thèse s'est avéré très enrichissant car il se situe à la jonction de différents domaines de recherche. En effet, nous avons pu aborder différents problèmes liés aux domaines des bases de données, des systèmes répartis et des réseaux.

2. Perspectives

Concernant la gestion des localisations géographiques, nous nous appuyons sur le format XML. En effet, l'utilisation du XML facilite l'évaluation des opérateurs de localisation dans le traitement d'une requête de localisation. En outre, il serait très intéressant d'introduire une dimension sémantique dans la gestion de ces fichiers de localisation géographique. L'introduction de sémantique dans le traitement de ces fichiers présente de nombreux intérêts : par exemple, dans l'évaluation de l'opérateur de localisation, être capable de définir que le bureau 34 est plus proche du bureau 35 que le bureau 47 permettrait d'affiner la réponse envoyée au client. Nous avons commencé à travailler au sein de l'équipe sur ce problème de gestion de la sémantique au sein des fichiers de localisation.
Nous avons proposé un modèle d'évaluation des requêtes dépendantes de la localisation reposant sur une « stratégie serveur ». C'est-à-dire que l'information est stockée par le terminal de l'utilisateur qui la partage et cette information n'est localisée et interrogée que sur ce terminal. D'autres solutions complémentaires reposant sur l'utilisation de cache et sur la réplication sont envisageables. Ces solutions permettraient de fournir une alternative à notre modèle d'évaluation. Des mécanismes de réplication ou d'indexation des informations stockées entre les différents nœuds centraux peuvent, par exemple, être proposées. La gestion de la réplication serait alors facilitée par la stabilité quasi-assurée des nœuds centraux. Cependant, concernant la gestion des informations issues des terminaux nomades, ces solutions de cache et de réplication sont beaucoup plus difficiles à mettre en œuvre. En effet, le maintien de la cohérence des données devient très problématique dans un environnement extrêmement dynamique. Il serait donc intéressant d'étudier les différentes solutions envisageables dans ce contexte et mesurées le gain en terme de coûts nécessaires pour l'évaluation en considérant les différents coûts supplémentaires engendrés par la mise en place de ces solutions.
D'autres problèmes concernant les applications de proximité ont été brièvement présenté dans ce mémoire tels que la sécurité ou la confidentialité des données. Ces problèmes sont importants mais ils n'ont pas été traités dans le cadre de cette thèse, il serait très intéressant d'y apporter les solutions adaptées.
Finalement, dans les applications de proximité, nous n'avons envisagé que l'utilisation de réseaux locaux et personnels sans fil. Aujourd'hui, le développement des réseaux de téléphonie mobile tels que le GPRS incite à étendre notre service de localisation à des

systèmes d'informations de plus grande échelle. Cette ouverture permettrait par exemple de chercher un résultat de requête dans un système d'information distant ou sur Internet lorsqu'il n'a pas pu être calculé en n'utilisant seulement les données de l'application de proximité. Les optimisations devraient alors être adaptées afin de prendre en compte les coûts de communication hétérogènes en fonction du type de réseaux utilisé. D'autre part, le service ISLANDS repose aujourd'hui sur l'utilisation d'une infrastructure existante représentée par les nœuds centraux. Même si beaucoup d'applications de proximité reposent sur une infrastructure existante, il nous semble aujourd'hui important de considérer les environnements sans infrastructure. Afin d'illustrer les deux points que nous venons d'évoquer, considérons une application permettant à différents véhicules d'accéder à des informations tout au long de leur trajet afin de les aider à déterminer dynamiquement leur itinéraire (tarif des carburants, embouteillages, etc.). Ces informations pourraient être issues soit d'un serveur de données distant, soit d'une station service ou partagées par un véhicule proche par exemple. Dans cette application, l'existence d'une infrastructure de nœuds centraux n'est absolument pas garantie. Il serait intéressant de mesurer l'impact de cet environnement sur les différentes solutions proposées dans ce mémoire et de proposer les modifications et les évolutions nécessaires.

Références Bibliographiques

[ADA 01] L. A. Adamic, R. M. Lukose, A. R. Puniyani, B. A. Huberman, "Search in power-law networks", Physical Review E 64, 2001.

[ANC 03] N. Anciaux, L. Bouganim, P. Pucheral, "Memory Requirements for Query Execution in Highly Constrained Devices", *29th Int.Conf. on Very Large Data Bases*, 2003.

[BAH 00] P. Bahl, V. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system", *Int. Conf. IEEE INFOCOM*, 2000.

[BLU 01] Bluesoft Incorporated, disponible à l'adresse : http://www.bluesoft-inc.com/, 2001.

[BRA 04] W. B. Bradley, D. P. Maher, "The NEMO P2P Service Orchestration Framework", *37th Hawaii Int. Conf. On System Sciences*, 2004.

[BUL 00] N. Bulusu, J. Heidemann, D. Estrin, "GPS-less low cost outdoor localization for very small devices", *IEEE Personal Communications, Special Issue on Smart Spaces and Environment*, 2000.

[BUY 99] O. Buyukkokten, J. Cho, H. Garcia-Molina, L. Gravano, N. Shivakumar, "Exploiting geographical location information of web pages", *Int. ACM SIGMOD Workshop on theWeb and Databases (WebDB)*, 1999.

[CAM 02] C. Campo, "Service Discovery in Pervasive Multi-Agent Systems", *Workshop on Ubiquitous Agents on embedded, wearable, and mobile devices*, 2002.

[CLE 04] T. Clements, "Telematics Talking", disponible à l'adresse : http://developers.sun.com/techtopics/mobility/midp/articles/telematics/, 2004.

[CRE 02] A. Crespo, H. Garcia-Molina, "Routing Indices For Peer-to-Peer Systems", *ICDCS*, 2002.

[DAN 00] P. H. Dana, “Global Positioning System overview”, disponible à l’adresse : http://www.colorado.edu/geography/gcraft/notes/gps/gps.html, 2000.

[DCMI] Dublin Core Metadata Initiative, disponible à l’adresse http://dublincore.org/.

[DCMIType] DCMI Type Vocabulary, disponible à l’adresse : http://dublincore.org/documents/dcmi-type-vocabulary/

[DEL 01] T. Delot, “Interrogation d’annuaires étendus : modèles, langage et optimisation”, *Thèse à l’université de Versailles, Saint-Quentin-en-Yvelines*, 2001.

[DNS 87] Domain Name System : Implementation & Specification, RFC 1035, disponible à l’adresse http://www.ietf.org, 1987.

[DOH 01] L. Doherty, K. S. J. Pister, L. E. Ghaoui, “Convex optimization methods for sensor node position estimation”, *IEEE INFOCOM*, 2001.

[DUN 98] M. H. Dunham, V. Kumar, “Location Dependent Data and its Management in Mobile Databases”, *DEXA Workshop*, 1998.

[EDW 99] J. Edwards, “3-Tier Server/Client at Work”, *Revised Edition, Wiley, ASIN 0471315028*, 1999.

[EKA 04] EKAHAU, disponible à l’adresse : http://www.ekahau.com, 2004.

[ERM 03] E. Ermel, A. Fladenmuller, G. Pujolle, A. Cotton, “Estimation de positions dans des réseaux sans-fil hybrides”, *Colloque Francophone sur l'Ingénérie des Protocoles* , 2003.

[FED 01] G. Fedak, C. Germain, V. Néri, F. Cappello, “XtremWeb : A Generic Global Computing system”, *Workshop on Global Computing on Personal Devices*, 2001.

[Freenet] The Free Network Project, disponible à l’adresse : http://freenet.sourceforge.net/.

[GAL 04] Galiléo, disponible à l’adresse : http://europa.eu.int/comm/dgs/energy_transport/galileo/

[Gnutella] Gnutella, disponible à l'adresse http://www.gnutella.com/.

[GOL 99] Y. Goland, T. Cai, P. Leach, Y. Gu, S. Albright, "Simple Service Discovery Protocol", Internet Draft draft-cai-ssdp-v1-03, disponible à l'adresse : http://www.upnp.org/download /draft_cai_ssdp_v1_03.txt, 1999.

[GOL 01] M. Golm, J. Kleinöder, "Ubiquitous computing and the need for a new operating system architecture", *Int. Conf. On Ubiquitous Computing*, 2001.

[GON 02] L. Gong, "Project JXTA: A Technology Overview", *Rapport technique Sun Microsystems*, 2002.

[GPS 97] "Global Positioning System : Theory and Practice", *Springer-Verlag*, 1997.

[HAL 02] E. Halepovic, R. Deters, "Building a P2P Forum System with JXTA", *Second Int. Conf. On Peer-To-Peer Computing*, 2002.

[HAR 99] A. Harter, A. Hopper, P. Steggles, A. Ward, P. Webster, "The anatomy of a context-aware application", *Int. ACM/IEEE Conf. On Mobile Computing and Networking*, 1999.

[HAZ 04] M. Hazas, J. Scott, J. Krumm, "Location-Aware Computing Comes of Age", *Computer*, 2004.

[HEU 03] M. Heusse, F. Rousseau, G. Berger-Sabbatel, A. Duda, "Performance Anomaly of 802.11b", *IEEE Infocom*, 2003.

[HIG 00] J. Hightower, R. Want, G. Borriello, "SpotON: An indoor 3d location sensing technology based on RF signal strength", *Rapport technique UW-CSE 00-02-02, Université de Washington*, 2000.

[HIG 01a] J. Hightower, G. Borriello, "A Survey and Taxonomy of Location systems for Ubiquitous Computing", *Rapport technique UW-CSE 01-08-03, Université de Washington*, 2001.

[HIG 01b] J. Hightower, G. Borriello, "Location Systems for Ubiquitous Computing", *IEEE Computer*, 2001.

[HIL 99] A. Hills, “Wireless andrew”, *IEEE Spectrum, 36(6):49-53*, 1999.

[HIN 99] K. Hinckley, M. Sinclair, “Touch-sensing input devices”, *Int. Conf. on Human Factors in Computing Systems*, 1999.

[HIN 01] K. Hinchley, E. Horvitz, “Toward More Sensitive Mobile Phones”, *14th ACM symp. On User Interface Software and Technology*, 2001.

[ICQ] ICQ, disponible à l’adresse : http://www.icq.com.

[IEEE 802.15] IEEE 802.15 Working Group for WPAN, disponible à l’adresse : http://grouper.ieee.org/groups/802/15/.

[ILD 02] J. Il-dong, Y. Young-ho, L. Jong-hwan, K. Kyungsok, “Broadcasting and Caching Policies for Location-Dependent Queries in Urban Area”, *Int. workshop on Wireless sensor networks and applications (WMC),* 2002.

[IMI 94] T. Imielinski, S. Viswanathan, “Adaptive Wireless Information Systems”, *SIGDBS (Special Interest Group in DataBase Systems) Conf.*, 1994.

[IMI 97] T. Imielinski, S. Viswanathan, B.R. Badrinath, “Data on Air: Organization and Access”, *IEEE Transactions on knowledge and data engineering*, 1997.

[INS 04] Instat Report, “Handset Market Rumbles On: Q2 2004”, disponible sur le site http://www.instat.com, 2004.

[Jabber] Jabber Software Foundation, disponible à l’adresse http://www.jabber.org.

[JOS 03] S. Joseph, T. Hoshiai, “Decentralized Meta-Data Strategies: Effective Peer-To-Peer Search”, *IEICE Trans. Commun., Vol. E86-B, N° 6*, 2003.

[JUN 02] Y. Y. Jung, J. Lee, K. Kim, “Broadcasting and Caching Policies for Location-Dependent Queries in Urban Areas”, *WMC*, 2002.

[JXTA] JXTA Project, disponible à l’adresse : http://www.jxta.org

[Kazaa] Kazaa, disponible à l'adresse http://www.kazaa.com/.

[Larousse] Larousse Dictionnaire Français, ISBN 2035320801.

[LEE 02] D. L. Lee, W-C. Lee, J. Xu, B. Zheng, "Data Management in Location-Dependent Information Services: Challenges and Issues", *IEEE Pervasive Computing, 1(3): 65-72*, 2002.

[MAD 00] S. K. Madria, B. Bhargava, E. Pitoura, V. Kumar, "Data Organisation for Location-Dependent Queries in Mobile Computing", *Conférence ADBIS-DASFAA*, 2000.

[MAN 04] H. Manica, M. S. de Camargo, R. R. Ciferri, C. D. A. Ciferri, "A New Model for Location-Dependent Semantic Cache Based on Pre-Defined Regions", *30ma Conferencia Latinoamericana de Informática (CLEI2004)*, 2004.

[MIN 01] N. Minar et al., "Peer-to-Peer : Harnessing the Power of Disruptive Technologies", *ISBN 0-596-00110-X*, 2001.

[MOO 65] G. E. Moore, "Cramming more components onto integrated circuits", *Electronics, Volume 38, Number 8*, 1965.

[Napster] Napster, disponible à l'adresse http://www.napster.com/.

[NEJ 02] W. Nejdl et al., "EDUTELLA: a P2P networking infrastructure based on RDF", *WWW Conf.*, 2002.

[NIC 03] D. Niculescu and B. Nath, "Ad Hoc Positioning System (APS) using AOA", *IEEE INFOCOM*, 2003.

[OMG 95] Object Management Group, "Naming Service Specification", disponible à l'adresse http://www.omg.org, 1995.

[OMG 97] Object Management Group, "Trading Object Service Specification", disponible à l'adresse http://www.omg.org, 1997.

[ORF 99] R. Orfali, D. Harkey, J. Edwards, "Client/Server Survival Guide", *Third Edition, Wiley, ISBN 0471316156*, 1999.

[PAR 00] K. Partridge, L. Arnstein, G. Borriello, T. Whitted, “Fast intrabody signaling”, *Conf. on Wireless and Mobile Computer Systems and Applications*, 2000.

[PAS 99] B. Pascoe, “Salutation-Lite : White Paper”, disponible à l’adresse : http://www.salutation.org/whitepaper/Sal-Lite.PDF, 1999.

[PRI 00] N. B. Priyantha, A. Chakraborty, H. Balakrishnan, “The Cricket location-support system”, *Int. Conf. on Mobile Computing and Networking*, 2000.

[RAT 01] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, “A Scalable Content-Addressable Network”, *Proc. SIGCOMM*, 2001.

[RDF] Resource Description Framework, disponible à l’adresse : http://www.w3.org/RDF/.

[REN 00] Q. Ren, M. H. Dunham, “Using Semantic Caching to Manage Location Dependent Data in Mobile Computing”, *Mobicom Conf.*, 2000.

[SAL 98] Salutation Consortium, “White Paper : Salutation Architecture : Overview”, disponible à l’adresse http://www.salutation.org/whitepaper/originalwp.pdf, 1998.

[SDP 99] Bluetooth Specification Part E. Service Discovery Protocol (SDP), disponible à l’adresse http://www.bluetooth.com, 1999.

[SETI] SETI@home, disponible à l’adresse : http://setiathome.ssl.berkeley.edu/.

[SEY 01] A Seydim, M. Dunham, V. Kumar, “Location Dependent Query Processing”, *Second ACM MobileDE*, 2001.

[SIN 00] K. Sinha and N. Das, “Exact Location Identification in Mobile Computing Networks”, *Int. Workshop on Parallel Processing (ICPP)*, 2000.

[STO 01] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, “Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications”, *SIGCOMM*, 2001.

[TDC 01] Time Domain Corporation, “PulsON Technology overview”, 2001.

[THI 01] M. Thilliez, "Le Commerce Electronique de Proximité", *Colloque sur les 'Mobiles-services et réseaux mobiles de 3ème Génération' Des architectures aux Services (MS3G'01)*, 2001.

[THI 02] M. Thilliez, "Les Applications de Proximité", *Colloque sur l'Informatique Mobile*, 2002.

[THI 03a] M. Thilliez, T. Delot, S. Lecomte, N. Bennani, "Hybrid Peer-To-Peer Proximity Applications", *17th Int. Conf. on Advanced Information Networking and Applications, ISBN 0-7695-1906-7,* 2003.

[THI 03b] M. Thilliez, Y. Colmant, T. Delot, "Query Evaluation in ISLANDS", *19èmes Journées de Bases de Données Avancées*, 2003.

[THI 04a] M. Thilliez, T. Delot, "A Localization Service for Mobile Users in Peer-To-Peer Environments", *Mobile and Ubiquitous Information Access, Springer-Verlag, LNCS 2954, pp. 271-282, ISBN 3-540-21003-2*, 2004.

[THI 04b] M. Thilliez, T. Delot, "Evaluating Location Dependent Queries Using ISLANDS", *Advanced Distributed Systems: Third International School and Symposium, ISSADS 2004, Guadalajara, Mexico, January 24-30, 2004. Revised Selected Papers, 3061/2004*, Springer-*Verlag Heidelberg, ISBN 3-540-22172-7/0302-9743*, 2004.

[THI 04c] M. Thilliez, T. Delot, S. Lecomte, "A Metadata-Based Positioning Solution to Evaluate Location-Dependent Queries in Wireless Environments", *20èmes Journées de Bases de Données Avancées*, 2004.

[THI 04d] M. Thilliez, T. Delot "Evaluation de requêtes dépendantes de la localisation dans les réseaux mobiles", *In G. Canals, A. Giboin, L. Nigay, A-m. Pinna, J-y. Tigli (Ed.), Premières Journées Francophones: Mobilité et Ubiquité 2004, ACM International Conference Proceeding Series, ISBN 1-58113-915-2*, 2004.

[VAR 00] U. Varshney, R. Vetter, "Emerging Mobile and Wireless Networks", *Communications of the ACM, Vol. 43, Issue 6*, 2000.

[VEI 97] J. Veizades, E. Guttman, C. E. Perkins, S. Kaplan, “Service Location Protocol, RFC 2165”, disponible à l’adresse http://www.ietf.org, 1997.

[WAH 97] M. Wahl, T. Howes, S. Kille, “Lightweight Directory Access Protocol (v3)”, *Internet RFC-2251*, 1997.

[WAL 00] J. Waldo, “The Jini Specifications”, *édité par Ken Arnold (2nd Edition) chez Addison-Wesley Professional, ASIN: 0201726173*, 2000.

[WAL 04] J. Walkerdine, I. Melville, I. Sommerville, “A Framework for P2P Application Development”, *Technical Report COMP-004-2004*, Computing Department, Lancaster University, 2004.

[WAN 03] Y. Wang, X. Jia, H. K. Lee, G. Y. Li, “An indoors wireless positioning system based on wireless local area network infrastructure”, *Int. Symp. on Satellite Navigation Technology Including Mobile Positioning & Location Services (SatNav)*, 2003.

[WAN 92] R. Want, A. Hopper, V. Falcao, J. Gibbons, “The active badge location systems”, *ACM Transactions on Informations Systems, 10(1):91-102*, 1992.

[WAN 97] R. Want, B. Schilit, N. Adams, R. Gold, K. Petersen, D. Goldberg, J. Ellis, M. Weiser, “The parctab ubiquitous computing experiment”, *Mobile Computing, chap. 2, ISBN 0-7923-9697-9*, 1997.

[XML] Extensible Markup Language, disponible à l’adresse : http://www.w3.org/XML.

[XU 99] J. Xu, X. Tang, D. L. Lee, Q. Hu, “Cache Coherency in Location-Dependent Information Services for Mobile Environment”, *1st Int. Conf. on Mobile Data Access (MDA'99)*, 1999.

[XU 00] J. Xu, D. K. Lee, “Querying location-dependent data in wireless cellular environment”, *Workshop on Position Dependent Information Services*, 2000.

[XU 03] J. Xu, X. Tang, D. L. Lee, “Performance Analysis of Location-Dependent Cache Invalidation Schemes for Mobile Environments”, *TKDE*, 2003.

[YAN 01] B. Yang, H. Garcia-Molina, “Comparing Hybrid Peer-To-Peer Systems”, *27th Int.Conf. on Very Large Data Bases*, 2001.

[YAN 02] B. Yang, H. Garcia-Molina, “Improving Search in Peer-to-Peer Networks”, *ICDCS*, 2002.

[YAN 03] B. Yang, H. Garcia-Molina, “Designing a Super-Peer Network”, *IEEE Int. Conf. on Data Engineering*, 2003.

[ZHA 01] B. Zhao, J. Kubiatowicz, A. Joseph, “Tapestry: An infrastructure for fault-tolerant wide-area location and routing”, *UC Berkeley Computer Science Division, Rapport No. UCB/CSD 01/1141*, 2001.

[ZHE 01] B. Zheng, D. L. Lee, “Processing location-dependent queries in a multi-cell wireless environment”, *MobiDE*, 2001.

[ZHE 02] B. Zheng, J. Xu, D. L. Lee, “Cache Invalidation and Replacement Strategies for Location-Dependent Data in Mobile Environments”, *IEEE Trans. on Computers, Special Issue on Database Management and Mobile Computing, 51(10): 1141-1153*, 2002.

Annexes

Annexe 1 : Quelques exemples de terminaux nomades

	HP iPAQ H5555	Sony clié PEG – NX73V	Ericsson P900	Nokia N-Gage QD
				
Utilisation principale	PDA	PDA	Téléphone	Téléphone
Processeur	Intel XScale 400MHz	Intel PXA250 200 MHz	ARM9 156MHz	ARM9 104mhz
Système d'exploitation	Ms Windows Mobile 2003 Premium pour Pocket PC	Palm OS 5.0	Symbian OS	Symbian OS
Mémoire	128Mo SDRAM, 48Mo Flash ROM	16 Mo de RAM, 32 Mo de ROM	16 Mo en interne + 32 Mo Memory Stick Duo	3,4 Mo interne + carte externe (32 à 256 Mo)
Réseaux sans fil	Bluetooth, 802.11b	Bluetooth, 802.11b, IrDA	GSM, GPRS, Bluetooth, IrDA	GSM, GPRS, Bluetooth
Environnement de développement	Java – support J2ME	Java – support J2ME	Java – support J2ME/MIDP 2.0	Java – support J2ME
Ecran	240 x 320 pixels, 16-bit couleurs	320 x 480 pixels, 65 536 couleurs	208 x 320 pixels 65536 couleurs	176 x 208 pixels 4096 couleurs
Fonctionnalités diverses	sécurité biométrique basée sur une analyse de l'index	appareil photo numérique, dictaphone, lecteur mp3	lecteur multimédia, lecteur mp3, appareil photo	console de jeux, calendrier, carnet d'adresses

Annexe 2 : Les différents éléments définis par la DCMI


```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE rdf:RDF (View Source for full doctype...)>
- <rdf:RDF xmlns:dcterms="http://purl.org/dc/terms/" xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
- <rdf:Description rdf:about="http://purl.org/dc/elements/1.1/">
<dc:title xml:lang="en-US">The Dublin Core Element Set v1.1 namespace providing access to its content by means of an RDF
Schema</dc:title>
<dc:publisher xml:lang="en-US">The Dublin Core Metadata Initiative</dc:publisher>
<dc:description xml:lang="en-US">The Dublin Core Element Set v1.1 namespace provides URIs for the Dublin Core Elements v1.1.
Entries are declared using RDF Schema language to support RDF applications.</dc:description>
<dc:language xml:lang="en-US">English</dc:language>
<dcterms:issued>1999-07-02</dcterms:issued>
<dcterms:modified>2003-03-24</dcterms:modified>
<dc:source rdf:resource="http://dublincore.org/documents/dces/" />
<dc:source rdf:resource="http://dublincore.org/usage/decisions/" />
<dcterms:isReferencedBy rdf:resource="http://www.dublincore.org/documents/2001/10/26/dcmi-namespace/" />
<dcterms:isRequiredBy rdf:resource="http://purl.org/dc/terms/" />
<dcterms:isReferencedBy rdf:resource="http://purl.org/dc/dcmitype/" />
</rdf:Description>
- <rdf:Property rdf:about="http://purl.org/dc/elements/1.1/title">
<rdfs:label xml:lang="en-US">Title</rdfs:label>
<rdfs:comment xml:lang="en-US">A name given to the resource.</rdfs:comment>
<dc:description xml:lang="en-US">Typically, a Title will be a name by which the resource is formally known.</dc:description>
<rdfs:isDefinedBy rdf:resource="http://purl.org/dc/elements/1.1/" />
<dcterms:issued>1999-07-02</dcterms:issued>
<dcterms:modified>2002-10-04</dcterms:modified>
<dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#element" />
<dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#title-004" />
</rdf:Property>
- <rdf:Property rdf:about="http://purl.org/dc/elements/1.1/creator">
<rdfs:label xml:lang="en-US">Creator</rdfs:label>
<rdfs:comment xml:lang="en-US">An entity primarily responsible for making the content of the resource.</rdfs:comment>
<dc:description xml:lang="en-US">Examples of a Creator include a person, an organisation, or a service. Typically, the name of a
Creator should be used to indicate the entity.</dc:description>
<rdfs:isDefinedBy rdf:resource="http://purl.org/dc/elements/1.1/" />
<dcterms:issued>1999-07-02</dcterms:issued>
<dcterms:modified>2002-10-04</dcterms:modified>
<dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#element" />
<dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#creator-004" />
</rdf:Property>
- <rdf:Property rdf:about="http://purl.org/dc/elements/1.1/subject">
<rdfs:label xml:lang="en-US">Subject and Keywords</rdfs:label>
<rdfs:comment xml:lang="en-US">The topic of the content of the resource.</rdfs:comment>
<dc:description xml:lang="en-US">Typically, a Subject will be expressed as keywords, key phrases or classification codes that
describe a topic of the resource. Recommended best practice is to select a value from a controlled vocabulary or formal
classification scheme.</dc:description>
<rdfs:isDefinedBy rdf:resource="http://purl.org/dc/elements/1.1/" />
<dcterms:issued>1999-07-02</dcterms:issued>
<dcterms:modified>2002-10-04</dcterms:modified>
<dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#element" />
<dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#subject-004" />
</rdf:Property>
- <rdf:Property rdf:about="http://purl.org/dc/elements/1.1/description">
<rdfs:label xml:lang="en-US">Description</rdfs:label>
<rdfs:comment xml:lang="en-US">An account of the content of the resource.</rdfs:comment>
<dc:description xml:lang="en-US">Description may include but is not limited to: an abstract, table of contents, reference to a
graphical representation of content or a free-text account of the content.</dc:description>
<rdfs:isDefinedBy rdf:resource="http://purl.org/dc/elements/1.1/" />
<dcterms:issued>1999-07-02</dcterms:issued>
<dcterms:modified>2002-10-04</dcterms:modified>
<dc:type rdf:resource="http://dublincore.org/usage/documents/ principles/#element" />
<dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#description-004" />
</rdf:Property>
- <rdf:Property rdf:about="http://purl.org/dc/elements/1.1/publisher">
<rdfs:label xml:lang="en-US">Publisher</rdfs:label>
<rdfs:comment xml:lang="en-US">An entity responsible for making the resource available</rdfs:comment>
<dc:description xml:lang="en-US">Examples of a Publisher include a person, an organisation, or a service. Typically, the name of a
Publisher should be used to indicate the entity.</dc:description>
<rdfs:isDefinedBy rdf:resource="http://purl.org/dc/elements/1.1/" />
<dcterms:issued>1999-07-02</dcterms:issued>
<dcterms:modified>2002-10-04</dcterms:modified>
<dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#element" />
<dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#publisher-004" />
</rdf:Property>
- <rdf:Property rdf:about="http://purl.org/dc/elements/1.1/contributor">
<rdfs:label xml:lang="en-US">Contributor</rdfs:label>
<rdfs:comment xml:lang="en-US">An entity responsible for making contributions to the content of the resource.</rdfs:comment>
<dc:description xml:lang="en-US">Examples of a Contributor include a person, an organisation, or a service. Typically, the name of
a Contributor should be used to indicate the entity.</dc:description>
<rdfs:isDefinedBy rdf:resource="http://purl.org/dc/elements/1.1/" />
<dcterms:issued>1999-07-02</dcterms:issued>
<dcterms:modified>2002-10-04</dcterms:modified>
<dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#element" />
<dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#contributor-004" />
</rdf:Property>
- <rdf:Property rdf:about="http://purl.org/dc/elements/1.1/date">
<rdfs:label xml:lang="en-US">Date</rdfs:label>
<rdfs:comment xml:lang="en-US">A date associated with an event in the life cycle of the resource.</rdfs:comment>
<dc:description xml:lang="en-US">Typically, Date will be associated with the creation or availability of the resource. Recommended
best practice for encoding the date value is defined in a profile of ISO 8601 [W3CDTF] and follows the YYYY-MM-DD
```

```
format.</dc:description>
  <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/elements/1.1/" />
  <dcterms:issued>1999-07-02</dcterms:issued>
  <dcterms:modified>2002-10-04</dcterms:modified>
  <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#element" />
  <dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#date-004" />
  </rdf:Property>
- <rdf:Property rdf:about="http://purl.org/dc/elements/1.1/type">
  <rdfs:label xml:lang="en-US">Resource Type</rdfs:label>
  <rdfs:comment xml:lang="en-US">The nature or genre of the content of the resource.</rdfs:comment>
  <dc:description xml:lang="en-US">Type includes terms describing general categories, functions, genres, or aggregation levels for
content. Recommended best practice is to select a value from a controlled vocabulary (for example, the DCMI Type Vocabulary
[DCMITYPE]). To describe the physical or digital manifestation of the resource, use the Format element.</dc:description>
  <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/elements/1.1/" />
  <dcterms:issued>1999-07-02</dcterms:issued>
  <dcterms:modified>2002-10-04</dcterms:modified>
  <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#element" />
  <dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#type-004" />
  </rdf:Property>
- <rdf:Property rdf:about="http://purl.org/dc/elements/1.1/format">
  <rdfs:label xml:lang="en-US">Format</rdfs:label>
  <rdfs:comment xml:lang="en-US">The physical or digital manifestation of the resource.</rdfs:comment>
  <dc:description xml:lang="en-US">Typically, Format may include the media-type or dimensions of the resource. Format may be
used to determine the software, hardware or other equipment needed to display or operate the resource. Examples of dimensions
include size and duration. Recommended best practice is to select a value from a controlled vocabulary (for example, the list of
Internet Media Types [MIME] defining computer media formats).</dc:description>
  <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/elements/1.1/" />
  <dcterms:issued>1999-07-02</dcterms:issued>
  <dcterms:modified>2002-10-04</dcterms:modified>
  <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#element" />
  <dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#format-004" />
  </rdf:Property>
- <rdf:Property rdf:about="http://purl.org/dc/elements/1.1/identifier">
  <rdfs:label xml:lang="en-US">Resource Identifier</rdfs:label>
  <rdfs:comment xml:lang="en-US">An unambiguous reference to the resource within a given context.</rdfs:comment>
  <dc:description xml:lang="en-US">Recommended best practice is to identify the resource by means of a string or number
conforming to a formal identification system. Example formal identification systems include the Uniform Resource Identifier (URI)
(including the Uniform Resource Locator (URL)), the Digital Object Identifier (DOI) and the International Standard Book Number
(ISBN).</dc:description>
  <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/elements/1.1/" />
  <dcterms:issued>1999-07-02</dcterms:issued>
  <dcterms:modified>2002-10-04</dcterms:modified>
  <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#element" />
  <dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#identifier-004" />
  </rdf:Property>
- <rdf:Property rdf:about="http://purl.org/dc/elements/1.1/source">
  <rdfs:label xml:lang="en-US">Source</rdfs:label>
  <rdfs:comment xml:lang="en-US">A reference to a resource from which the present resource is derived.</rdfs:comment>
  <dc:description xml:lang="en-US">The present resource may be derived from the Source resource in whole or in part.
Recommended best practice is to reference the resource by means of a string or number conforming to a formal identification
system.</dc:description>
  <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/elements/1.1/" />
  <dcterms:issued>1999-07-02</dcterms:issued>
  <dcterms:modified>2002-10-04</dcterms:modified>
  <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#element" />
  <dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#source-004" />
  </rdf:Property>
- <rdf:Property rdf:about="http://purl.org/dc/elements/1.1/language">
  <rdfs:label xml:lang="en-US">Language</rdfs:label>
  <rdfs:comment xml:lang="en-US">A language of the intellectual content of the resource.</rdfs:comment>
  <dc:description xml:lang="en-US">Recommended best practice is to use RFC 3066 [RFC3066], which, in conjunction with ISO 639
[ISO639], defines two- and three-letter primary language tags with optional subtags. Examples include "en" or "eng" for English,
"akk" for Akkadian, and "en-GB" for English used in the United Kingdom.</dc:description>
  <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/elements/1.1/" />
  <dcterms:issued>1999-07-02</dcterms:issued>
  <dcterms:modified>2002-10-04</dcterms:modified>
  <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#element" />
  <dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#language-005" />
  </rdf:Property>
- <rdf:Property rdf:about="http://purl.org/dc/elements/1.1/relation">
  <rdfs:label xml:lang="en-US">Relation</rdfs:label>
  <rdfs:comment xml:lang="en-US">A reference to a related resource.</rdfs:comment>
  <dc:description xml:lang="en-US">Recommended best practice is to reference the resource by means of a string or number
conforming to a formal identification system.</dc:description>
  <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/elements/1.1/" />
  <dcterms:issued>1999-07-02</dcterms:issued>
  <dcterms:modified>2002-10-04</dcterms:modified>
  <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#element" />
  <dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#relation-004" />
  </rdf:Property>
- <rdf:Property rdf:about="http://purl.org/dc/elements/1.1/coverage">
  <rdfs:label xml:lang="en-US">Coverage</rdfs:label>
  <rdfs:comment xml:lang="en-US">The extent or scope of the content of the resource.</rdfs:comment>
  <dc:description xml:lang="en-US">Coverage will typically include spatial location (a place name or geographic coordinates),
temporal period (a period label, date, or date range) or jurisdiction (such as a named administrative entity). Recommended best
practice is to select a value from a controlled vocabulary (for example, the Thesaurus of Geographic Names [TGN]) and that, where
appropriate, named places or time periods be used in preference to numeric identifiers such as sets of coordinates or date
ranges.</dc:description>
  <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/elements/1.1/" />
  <dcterms:issued>1999-07-02</dcterms:issued>
  <dcterms:modified>2002-10-04</dcterms:modified>
  <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#element" />
  <dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#coverage-004" />
  </rdf:Property>
- <rdf:Property rdf:about="http://purl.org/dc/elements/1.1/rights">
```

```
  <rdfs:label xml:lang="en-US">Rights Management</rdfs:label>
  <rdfs:comment xml:lang="en-US">Information about rights held in and over the resource.</rdfs:comment>
  <dc:description xml:lang="en-US">Typically, a Rights element will contain a rights management statement for the resource, or
reference a service providing such information. Rights information often encompasses Intellectual Property Rights (IPR),
Copyright, and various Property Rights. If the Rights element is absent, no assumptions can be made about the status of these and
other rights with respect to the resource.</dc:description>
  <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/elements/1.1/" />
  <dcterms:issued>1999-07-02</dcterms:issued>
  <dcterms:modified>2002-10-04</dcterms:modified>
  <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#element" />
  <dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#rights-004" />
  </rdf:Property>
  </rdf:RDF>
```

Annexe 3 : Le vocabulaire de types défini par la DCMI


```
<?xml version="1.0" encoding="UTF-8" ?>
  <!DOCTYPE rdf:RDF (View Source for full doctype...)>
- <rdf:RDF xmlns:dcmitype="http://purl.org/dc/dcmitype/" xmlns:dcterms="http://purl.org/dc/terms/"
xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
- <rdf:Description rdf:about="http://purl.org/dc/dcmitype/">
  <dc:title xml:lang="en-US">The DCMI Types namespace providing access to its content by means of an RDF Schema</dc:title>
  <dc:publisher xml:lang="en-US">The Dublin Core Metadata Initiative</dc:publisher>
  <dc:description xml:lang="en-US">The Dublin Core Types namespace provides URIs for the entries of the DCMI Type Vocabulary.
Entries are declared using RDF Schema language to support RDF applications. The Schema will be updated according to dc-usage
decisions.</dc:description>
  <dc:language xml:lang="en-US">English</dc:language>
  <dcterms:requires rdf:resource="http://dublincore.org/documents/dcmi-type-vocabulary/" />
  <dc:source rdf:resource="http://dublincore.org/documents/dcmi-type-vocabulary/" />
  <dc:source rdf:resource="http://dublincore.org/usage/terms/" />
  <dcterms:requires rdf:resource="http://purl.org/dc/elements/1.1/" />
  <dcterms:isReferencedBy rdf:resource="http://purl.org/dc/terms/" />
  <dcterms:issued>2000-07-11</dcterms:issued>
  <dcterms:modified>2002-05-22</dcterms:modified>
  </rdf:Description>
- <dcterms:TypeScheme rdf:about="http://purl.org/dc/terms/DCMIType">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
  <dc:description>The DCMI Type Vocabulary provides a general, cross-domain list of approved terms that may be used as values for
the Resource Type element to identify the genre of a resource.</dc:description>
  <dcterms:issued>2000-07-11</dcterms:issued>
  </dcterms:TypeScheme>
- <dcterms:DCMIType rdf:about="http://purl.org/dc/dcmitype/Collection">
  <rdfs:label xml:lang="en-US">Collection</rdfs:label>
  <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/dcmitype/" />
  <rdfs:comment xml:lang="en-US">A collection is an aggregation of items. The term collection means that the resource is described
as a group; its parts may be separately described and navigated.</rdfs:comment>
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
  <dcterms:issued>2000-07-11</dcterms:issued>
  <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#vocabulary-term" />
  <dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#Collection-001" />
  </dcterms:DCMIType>
- <dcterms:DCMIType rdf:about="http://purl.org/dc/dcmitype/Dataset">
  <rdfs:label xml:lang="en-US">Dataset</rdfs:label>
  <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/dcmitype/" />
  <rdfs:comment xml:lang="en-US">A dataset is information encoded in a defined structure (for example, lists, tables, and
databases), intended to be useful for direct machine processing.</rdfs:comment>
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
  <dcterms:issued>2000-07-11</dcterms:issued>
  <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#vocabulary-term" />
  <dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#Dataset-001" />
  </dcterms:DCMIType>
- <dcterms:DCMIType rdf:about="http://purl.org/dc/dcmitype/Event">
  <rdfs:label xml:lang="en-US">Event</rdfs:label>
  <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/dcmitype/" />
  <rdfs:comment xml:lang="en-US">An event is a non-persistent, time-based occurrence. Metadata for an event provides descriptive
information that is the basis for discovery of the purpose, location, duration, responsible agents, and links to related events and
resources. The resource of type event may not be retrievable if the described instantiation has expired or is yet to occur. Examples
- exhibition, web-cast, conference, workshop, open-day, performance, battle, trial, wedding, tea-party,
conflagration.</rdfs:comment>
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
  <dcterms:issued>2000-07-11</dcterms:issued>
  <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#vocabulary-term" />
  <dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#Event-001" />
  </dcterms:DCMIType>
- <dcterms:DCMIType rdf:about="http://purl.org/dc/dcmitype/Image">
  <rdfs:label xml:lang="en-US">Image</rdfs:label>
  <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/dcmitype/" />
  <rdfs:comment xml:lang="en-US">An image is a primarily symbolic visual representation other than text. For example - images and
photographs of physical objects, paintings, prints, drawings, other images and graphics, animations and moving pictures, film,
diagrams, maps, musical notation. Note that image may include both electronic and physical representations.</rdfs:comment>
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
  <dcterms:issued>2000-07-11</dcterms:issued>
  <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#vocabulary-term" />
  <dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#Image-002" />
  </dcterms:DCMIType>
- <dcterms:DCMIType rdf:about="http://purl.org/dc/dcmitype/InteractiveResource">
  <rdfs:label xml:lang="en-US">Interactive Resource</rdfs:label>
  <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/dcmitype/" />
  <rdfs:comment xml:lang="en-US">An interactive resource is a resource which requires interaction from the user to be understood,
executed, or experienced. For example - forms on web pages, applets, multimedia learning objects, chat services, virtual
reality.</rdfs:comment>
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
  <dcterms:issued>2000-07-11</dcterms:issued>
  <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#vocabulary-term" />
  <dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#InteractiveResource-001" />
  </dcterms:DCMIType>
- <dcterms:DCMIType rdf:about="http://purl.org/dc/dcmitype/Service">
  <rdfs:label xml:lang="en-US">Service</rdfs:label>
  <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/dcmitype/" />
  <rdfs:comment xml:lang="en-US">A service is a system that provides one or more functions of value to the end-user. Examples
include: a photocopying service, a banking service, an authentication service, interlibrary loans, a Z39.50 or Web
server.</rdfs:comment>
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
  <dcterms:issued>2000-07-11</dcterms:issued>
```

```
  <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#vocabulary-term" />
  <dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#Service-001" />
  </dcterms:DCMIType>
- <dcterms:DCMIType rdf:about="http://purl.org/dc/dcmitype/Software">
  <rdfs:label xml:lang="en-US">Software</rdfs:label>
  <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/dcmitype/" />
  <rdfs:comment xml:lang="en-US">Software is a computer program in source or compiled form which may be available for installation non-transiently on another machine. For software which exists only to create an interactive environment, use interactive instead.</rdfs:comment>
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
  <dcterms:issued>2000-07-11</dcterms:issued>
  <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#vocabulary-term" />
  <dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#Software-001" />
  </dcterms:DCMIType>
- <dcterms:DCMIType rdf:about="http://purl.org/dc/dcmitype/Sound">
  <rdfs:label xml:lang="en-US">Sound</rdfs:label>
  <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/dcmitype/" />
  <rdfs:comment xml:lang="en-US">A sound is a resource whose content is primarily intended to be rendered as audio. For example - a music playback file format, an audio compact disc, and recorded speech or sounds.</rdfs:comment>
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
  <dcterms:issued>2000-07-11</dcterms:issued>
  <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#vocabulary-term" />
  <dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#Sound-001" />
  </dcterms:DCMIType>
- <dcterms:DCMIType rdf:about="http://purl.org/dc/dcmitype/Text">
  <rdfs:label xml:lang="en-US">Text</rdfs:label>
  <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/dcmitype/" />
  <rdfs:comment xml:lang="en-US">A text is a resource whose content is primarily words for reading. For example - books, letters, dissertations, poems, newspapers, articles, archives of mailing lists. Note that facsimiles or images of texts are still of the genre text.</rdfs:comment>
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
  <dcterms:issued>2000-07-11</dcterms:issued>
  <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#vocabulary-term" />
  <dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#Text-001" />
  </dcterms:DCMIType>
- <dcterms:DCMIType rdf:about="http://purl.org/dc/dcmitype/PhysicalObject">
  <rdfs:label xml:lang="en-US">Physical Object</rdfs:label>
  <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/dcmitype/" />
  <rdfs:comment xml:lang="en-US">An inanimate, three-dimensional object or substance. For example -- a computer, the great pyramid, a sculpture. Note that digital representations of, or surrogates for, these things should use Image, Text or one of the other types.</rdfs:comment>
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
  <dcterms:issued>2002-07-13</dcterms:issued>
  <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#vocabulary-term" />
  <dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#PhysicalObject-001" />
  </dcterms:DCMIType>
- <dcterms:DCMIType rdf:about="http://purl.org/dc/dcmitype/StillImage">
  <rdfs:label xml:lang="en-US">Still Image</rdfs:label>
  <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/dcmitype/" />
  <rdfs:comment xml:lang="en-US">A static visual representation. Examples of still images are: paintings, drawings, graphic designs, plans and maps.</rdfs:comment>
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
  <dcterms:issued>2003-11-18</dcterms:issued>
  <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#vocabulary-term" />
  <dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#StillImage-001" />
  <rdfs:subClassOf rdf:resource="http://purl.org/dc/dcmitype/Image" />
  </dcterms:DCMIType>
- <dcterms:DCMIType rdf:about="http://purl.org/dc/dcmitype/MovingImage">
  <rdfs:label xml:lang="en-US">Moving Image</rdfs:label>
  <rdfs:isDefinedBy rdf:resource="http://purl.org/dc/dcmitype/" />
  <rdfs:comment xml:lang="en-US">A series of visual representations that, when shown in succession, impart an impression of motion. Examples of moving images are: animations, movies, television programs, videos, zoetropes, or visual output from a simulation.</rdfs:comment>
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
  <dcterms:issued>2003-11-18</dcterms:issued>
  <dc:type rdf:resource="http://dublincore.org/usage/documents/principles/#vocabulary-term" />
  <dcterms:hasVersion rdf:resource="http://dublincore.org/usage/terms/history/#MovingImage-001" />
  <rdfs:subClassOf rdf:resource="http://purl.org/dc/dcmitype/Image" />
  </dcterms:DCMIType>
  </rdf:RDF>
```

Annexe 4 : Définition de l'élément *Localisation*

```
- <rdf:Property rdf:about="http://purl.org/dc/elements/1.1/language">
  <rdfs:label xml:lang="FR">Localisation</rdfs:label>
  <rdfs:comment xml:lang="FR">La référence de la ressource contenant la localisation correspondant à la
ressource définie</rdfs:comment>
  <dc:description xml:lang="FR">Utilisation d'une URI référençant un fichier de localisation XML </dc:description>
  <dcterms:issued>2004-08-25</dcterms:issued>
  <dcterms:modified>2004-08-25</dcterms:modified>
  </rdf:Property>
```

Annexe 5 : Définition des types *LocalisationType* et *ISLANDSReference*

```
<dcterms:DCMIType rdf:about="LocalisationType">
 <rdfs:label xml:lang="FR">Localisation Type</rdfs:label>
  <rdfs:comment xml:lang="FR">Un type pour représenter les fichiers de localisation physique</rdfs:comment>
 <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
 <dcterms:issued>2004-08-25</dcterms:issued>
 </dcterms:DCMIType>
<dcterms:DCMIType rdf:about="ISLANDSRefe rence">
 <rdfs:label xml:lang="FR">ISLANDS Référence</rdfs:label>
  <rdfs:comment xml:lang="FR">Type d'une ressource de type service de localisation ISLANDS</rdfs:comment>
 <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
 <dcterms:issued>2004-08-25</dcterms:issued>
</dcterms:DCMIType>
```

Annexe 6 : Fichier de configuration

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<CEP>
  <Request>
        <Store type-directory="objectClass" id-directory="Magasin">
              <Name type-directory="attribute" id-directory="name">
                    <Type type-directory="attribute" id-directory="type" />
              </Name>
              <Type type-directory="attribute" id-directory="type">
                    <Name type-directory="attribute" id-directory="name" />
              </Type>
        </Store>
        <Product type-directory="objectClass" id-directory="Produit">
              <Name type-directory="attribute" id-directory="name">
                    <Store type-directory="objectClass" id-directory="Magasin">
                          <Name type-directory="attribute" id-directory="name" />
                          <Type type-directory="attribute" id-directory="type" />
              </Store>
        </Name>
              <Description type-directory="attribute" id-directory="description">
                    <Store type-directory="objectClass" id-directory="Magasin">
                          <Name type-directory="attribute" id-directory="name" />
                          <Type type-directory="attribute" id-directory="type" />
                    </Store>
              </Description>
        </Product>
        <Offer type-directory="objectClass" id-directory="Publicite">
              <Title type-directory="attribute" id-directory="name">
                    <Product type-directory="objectClass" id-directory="Produit">
                       <Name type-directory="attribute" id-directory="name" />
                       <Description type-directory="attribute" id-directory="description"
  />
                    </Product>
              </Title>
              <Description type-directory="attribute" id-directory="annonce">
                    <Product type-directory="objectClass" id-directory="Produit">
                       <Name type-directory="attribute" id-directory="name" />
                       <Description type-directory="attribute" id-directory="description"
  />
                    </Product>
              </Description>
        </Offer>
  </Request>
  <Localization>
        <Close>
              <Store type-directory="objectClass" id-directory="Magasin">
                    <Name type-directory="attribute" id-directory="name" />
                    <Type type-directory="attribute" id-directory="type" />
              </Store>
        </Close>
        <Closest>
              <Store type-directory="objectClass" id-directory="Magasin">
                    <Name type-directory="attribute" id-directory="name" />
              </Store>
        </Closest>
        <Inside>
              <Building type-directory="objectClass" id-directory="Building">
                <Mine>
                  <Stage type-directory="objectClass" id-directory="Stage">
                    <Name type-directory="attribute" id-directory="name">
                      <Section type-directory="objectClass" id-directory="Section">
                        <Name type-directory="attribute" id-directory="name">
                          <Store type-directory="objectClass" id-directory="Room">
                            <Name type-directory="attribute" id-directory="name" />
                          </Store>
                        </Name>
                      </Section>
                    </Name>
                  </Stage>
              </Mine>
              <Name type-directory="attribute" id-directory="name">
                <Stage type-directory="objectClass" id-directory="Stage">
                  <Name type-directory="attribute" id-directory="name">
                    <Section type-directory="objectClass" id-directory="Section">
```

```
                        <Name type-directory="attribute" id-directory="name">
                          <Store type-directory="objectClass" id-directory="Room">
                              <Name type-directory="attribute" id-directory="name" />
                            </Store>
                          </Name>
                        </Section>
                      </Name>
                    </Stage>
                  </Name>
                </Building>
                <Stage type-directory="objectClass" id-directory="Stage">
                  <Mine>
                    <Section type-directory="objectClass" id-directory="Section">
                      <Name type-directory="attribute" id-directory="name">
                        <Store type-directory="objectClass" id-directory="Room">
                          <Name type-directory="attribute" id-directory="name" />
                        </Store>
                      </Name>
                    </Section>
                  </Mine>
                  <Name type-directory="attribute" id-directory="name">
                    <Section type-directory="objectClass" id-directory="Section">
                      <Name type-directory="attribute" id-directory="name">
                        <Store type-directory="objectClass" id-directory="Room">
                          <Name type-directory="attribute" id-directory="name" />
                        </Store>
                      </Name>
                    </Section>
                  </Name>
                </Stage>
                <Section type-directory="objectClass" id-directory="Section">
                  <Mine>
                    <Store type-directory="objectClass" id-directory="Room">
                      <Name type-directory="attribute" id-directory="name" />
                    </Store>
                  </Mine>
                  <Name type-directory="attribute" id-directory="name">
                    <Store type-directory="objectClass" id-directory="Room">
                      <Name type-directory="attribute" id-directory="name" />
                    </Store>
                  </Name>
                </Section>
                <Store type-directory="objectClass" id-directory="Room">
                  <Name type-directory="attribute" id-directory="name" />
                </Store>
              </Inside>
            </Localization>
</CEP>
```

Résumé :

Depuis quelques années, l'évolution des terminaux nomades et des réseaux mobiles et/ou sans fil favorise le développement de nouveaux services et de nouvelles applications dédiées aux usagers mobiles. Parmi ces applications, nous définissons le concept d'applications de proximité. Celles-ci permettent à différents usagers physiquement proches les uns des autres de partager certaines de leurs informations et de localiser les données disponibles. Ces applications s'inscrivent dans un contexte fortement distribué et dynamique où l'ensemble de l'information disponible est répartie sur les terminaux des différents participants et subit de fréquentes variations dûes à la mobilité des participants. Dans cet environnement, les services de localisation actuels présentent d'importantes limites. Nous avons donc proposé ISLANDS (**I**nformation and **S**ervices **L**ocaliz**A**tio**N** and **D**iscovery **S**ervice), un service de localisation adapté à l'environnement des applications de proximité, en particulier à la distribution de l'information. Ce service ISLANDS repose sur notre modèle d'évaluation des requêtes dépendantes de la localisation. Un exemple de requêtes dépendantes de la localisation est « quel est l'arrêt de bus le plus proche de moi ? ». Le modèle d'évaluation proposé permet de considérer la mobilité des utilisateurs et en particulier leur localisation géographique dans l'évaluation des requêtes. Ce modèle d'évaluation a été optimisé afin de pouvoir être pleinement exploité dans un environnement contraint en terme de ressources. Le processus d'évaluation d'une requête dépendante de la localisation s'articule autour de différentes étapes dont l'évaluation de la localisation géographique du client qui émet la requête. Aujourd'hui, les techniques de localisation ne sont pas toujours exploitables : par exemple, le GPS qui est la solution la plus répandue, ne fonctionne généralement pas à l'intérieur d'un bâtiment, nous proposons donc une solution de localisation reposant sur les métadonnées de l'environnement. Notre solution permet de localiser un utilisateur de façon approximative mais avec une précision suffisante pour évaluer des requêtes dépendantes de la localisation. Cette solution a été optimisée de façon à minimiser la consommation des ressources sur les terminaux nomades et à réduire le nombre de communications entre les participants. Un prototype d'ISLANDS a été implémenté et démontré aux 19èmes journées de bases de données avancées (BDA'03) dans le but de valider nos propositions.

Mots-clés : Requêtes Dépendantes de la Localisation, Service de Localisation, Mobilité

Abstract :

For a few years, the evolution of handheld devices and wireless and/or mobile networks has implied the development of new services and new applications dedicated to mobile users. Among these applications, we define the proximity applications concept. These applications allow to different users physically close to each other to share and locate available data. These applications are running in a strongly distributed and dynamic context in which, the available information is distributed in different participants terminal and is continually modified due to the participants mobility. In this environment, the current localisation services have severe limits. Thus, we have proposed ISLANDS (**I**nformation and **S**ervices **L**ocaliz**A**tio**N** and **D**iscovery **S**ervice), a localisation service adapted to the data distribution. This service relies on our evaluation model for location dependent queries (LDQ). An example of LDQ is "which is the closest bus stop to me ?". The proposed evaluation model allows to consider the mobility of users and specially their geographic localization during the query evaluation. This evaluation model has been optimized to be used in constrained environments. The evaluation process of a LDQ relies on several steps : among them, the evaluation of the geographic localization of client who submitted the query. Today, the localization technologies are not always available : for example, GPS which is the most widespread solution, do not always run inside a building, thus, we propose a localization solution based on environment metadata. Our solution allows to locate a user in a approximatively way but sufficient to the location dependent queries evaluation. This solution has been optimised to minimize the resource consumption of nomadic devices and to reduce the communications between participants. A prototype of ISLANDS has been implemented and demonstrated at the french conference : Bases de Données Avancées (BDA'03). This prototype had permit to validate our propositions.

Keywords : Location Dependent Queries, Localization Service, Mobility