Accès à l'Information en Ubiquité Numérique

David Touzet, Frédéric Weis, Michel Banâtre

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Accès à l'Information en Ubiquité Numérique

David Touzet, Frédéric Weis, Michel Banâtre

N˚4490

Mai 2002

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--4490--FR

Accès à l'Information en Ubiquité Numérique

David Touzet, Frédéric Weis, Michel Banâtre*

Thème 1 — Réseaux et systèmes
Projet ACES

Rapport de recherche n° 4490 — Mai 2002 — 47 pages

Résumé : Du domaine de l'informatique mobile a émergée, voici quelques années, une nouvelle discipline s'attachant à intégrer, de manière transparente, les environnements numériques au monde réel : l'*ubiquité numérique*. En dotant les applications d'une "conscience" de leur environnement physique, les travaux menés dans ce cadre cherchent à proposer aux utilisateurs des services adaptés à leur situation. Tirant parti des récents progrès technologiques dans les domaines des communications sans fil et de l'informatique embarquée, l'ubiquité voit, chaque année, naître un nombre toujours croissant de prototypes. Le présent document est consacré à l'étude des architectures mises en œuvre par ces prototypes et, plus particulièrement, aux techniques utilisées pour accéder à l'information.

Mots-clé : Mobilité, ubiquité numérique

(Abstract: pto)

* {dtouzet,fweis,banatre}@irisa.fr

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : 02 99 84 71 00 - International : +33 2 99 84 71 00
Télécopie : 02 99 84 71 71 - International : +33 2 99 84 71 71

Information Access for Ubiquitous Computing

Abstract: A new discipline, called *ubiquitous computing* has appeared, for a few years, in the mobile computing area. It aims to merge, in a transparent way, computing environments with the physical world. For this purpose, ubiquitous computing studies attempt to offer users services that are adapted to their situation by investing applications with a kind of physical environment "consciousness". Taking part from the rise of embedded systems and wireless communications, more and more prototypes are actually developed. This document reviewes the architectures these prototypes are based on and studies the mecanisms used to access information.

Key-words: Mobility, ubiquitous computing

1 Introduction

L'informatique mobile connaît actuellement un essor considérable dû aux développements conjugués des communications sans fil et des technologies embarquées. Ces évolutions récentes peuvent, par exemple, être illustrées par le succès de la téléphonie mobile. Le système GSM [1] [1] permet en effet à deux utilisateurs nomades de communiquer entre eux dès lors qu'ils évoluent dans la zone de couverture. L'accès aux systèmes d'information tire également parti de ces développements. Le protocole WAP [2] [2] permet ainsi à un utilisateur mobile d'accéder au Web par le biais d'un simple téléphone portable. Les architectures mobiles développées jusqu'alors cherchent pour la plupart à déployer un ensemble de services homogènes sur la totalité de leur zone de couverture. Le système WAP, par exemple, propose à ses utilisateurs l'accès au même ensemble de serveurs Web quelque soit leur position géographique.

Un certain nombre d'études se distinguent aujourd'hui de cette approche classique. Pour ce faire, elles proposent d'exploiter la mobilité des utilisateurs et, plus généralement, le contexte de ces derniers afin d'adapter les services rendus à leur situation. Ce type de travaux fait aujourd'hui l'objet d'un axe de recherche particulier appelé ubiquité numérique. L'exploitation de ce principe permet d'envisager le déploiement de nouveaux types de services.

Plusieurs documents de synthèse ont déjà été consacrés au domaine. Ainsi, Korkeaaho dresse le paysage des grandes classes d'applications contextuelles [3]. Chen et Kotz proposent, quant à eux, une classification des prototypes réalisés en fonction du type de contexte manipulé [4]. Enfin, Couderc étudie les apports de l'ubiquité numérique au problème de l'adaptation en mobilité [5]. Le présent document est plus particulièrement dédié à l'étude de l'accès à l'information en ubiquité numérique. Il a pour objectif, d'une part, de montrer en quoi cet accès diffère de ceux utilisés par les systèmes traditionnels et, d'autre part, de présenter les techniques mises en œuvre par les prototypes réalisés dans le domaine de l'ubiquité.

Nous revenons, dans une première partie, sur la définition de l'ubiquité numérique, sur ces principaux champs d'application ainsi que sur les problèmes qu'elle soulève. Dans un second temps, nous proposons de dresser un état de l'art des systèmes d'ubiquité numérique existants en nous intéressant plus particulièrement à l'accès à l'information au sein de ces derniers. La troisième partie est, quant à elle,

1. Global System of Mobile communications
2. Wireless Application Protocol

consacrée à la comparaison des techniques d'accès utilisées en informatique mobile et en ubiquité numérique.

2 Ubiquité numérique : principes, applications et problèmes soulevés

Les bases de l'ubiquité numérique ont été posées par Weiser dans son article de 1993 intitulé *Some Computer Science Issues in Ubiquitous Computing* [6]. Il y définit l'ubiquité numérique comme une technologie *invisible* à des utilisateurs avec lesquels elle entretient des interactions permanentes. Dans cet article, Weiser dresse le constat suivant : exploiter les capacités d'un ordinateur nécessite aujourd'hui toute l'attention de son utilisateur. Les systèmes nomades actuels, conçus pour être utilisés par des utilisateurs mobiles, ne peuvent s'accommoder de ce type de contrainte. Un nouveau mode d'utilisation des machines, plus intuitif et ne nécessitant pas d'interactions explicites, devient alors nécessaire afin de permettre aux utilisateurs de consacrer la totalité de leur attention aux tâches qui les occupent.

L'objectif de l'ubiquité numérique est donc de réaliser l'intégration transparente des environnements numériques au monde physique. Pour concrétiser cette intégration, Weiser imagine un monde au sein duquel est disséminée une multitude de calculateurs embarqués, équipés de capacités de communication, fournissant un pendant numérique à leur environnement physique. Cette intégration des mondes physiques et numériques doit offrir aux utilisateurs des modes d'interaction plus naturels avec leur milieu. En poussant le raisonnement à terme, ces interactions doivent avoir lieu sans que l'utilisateur ait conscience d'utiliser les services de calculateurs avoisinant (on parle d'interactions *invisibles*).

Afin d'intégrer un calculateur à son environnement, il est nécessaire de le munir d'outils dédiés à la perception de ce dernier. Cette perception doit, de plus, être accompagnée d'une structuration appropriée des informations captées afin que celles-ci puissent être exploitées. La détection et l'analyse des situations sont deux tâches inconscientes que nous effectuons tout au long d'une journée. Par exemple, en rentrant dans une pièce, nous pouvons sans effort découvrir combien de personnes s'y trouvent, leur identité si elle nous est connue, ou encore la fonction de la pièce (bureau, salle de réunion ...). Cependant, lorsque ces tâches sont confiées à un ensemble de calculateurs, le traitement devient nettement plus délicat. En effet, la complexité du monde réel est telle qu'il est difficile pour un calculateur d'identifier les sources d'information pertinentes. Les applications d'ubiquité numérique délèguent donc les tâches d'acquisition de l'environnement d'exécution, également appelé *contexte*, à des

systèmes spécialisés tels que le GPS[3] [7] (pour la localisation des utilisateurs). Les types de contexte gérés par de tels systèmes ne constituent qu'un sous-ensemble limité du contexte perçu par les humains. Chen et Kotz proposent de distinguer quatre classes de contexte pouvant être exploitées par les applications d'ubiquité numérique [4] :

- le contexte *numérique*, regroupant les paramètres tels que la présence (ou l'absence) d'accès réseau, la bande passante disponible, les périphériques accessibles (imprimante, écran) ...
- le contexte *utilisateur*, traitant des informations relatives à un utilisateur : son identité, ses préférences, sa localisation ...
- le contexte *physique*, relatif à l'environnement de l'utilisateur (conditions climatiques, niveau de bruit, luminosité ...).
- enfin, le paramètre *temps*, qui fait référence à l'heure et la date. Ce dernier type de contexte, combiné aux types précédents, permet l'obtention d'historiques de contexte.

Nous pouvons considérer, à titre d'exemple, un musée équipé d'un système de guide virtuel. Pour cela, supposons que chacun des visiteurs du musée soit équipé d'un calculateur portable de type PDA[4]. Une des fonctionnalités d'un tel système peut alors être l'affichage spontané, sur ces PDAs, d'informations relatives aux œuvres situées à proximité des utilisateurs. Pour ce faire, le système doit être capable de détecter devant quelle(s) œuvre(s) se trouve chacun d'entre eux. L'ensemble des œuvres à proximité d'un utilisateur constitue, pour ce type de service, le contexte utile à l'application.

La suite de cette partie est tout d'abord consacrée à une présentation des principales classes d'applications tirant aujourd'hui parti du concept d'ubiquité numérique. Nous étudions, dans un second temps, les différents types de problèmes soulevés par cette nouvelle famille d'applications.

2.1 Principales applications de l'ubiquité numérique

Des premiers prototypes, dont le déploiement est resté confiné aux laboratoires, aux développements plus récents, adaptés à des environnements variés (un musée, une ville, un site de fouilles ...), l'ubiquité numérique a déjà trouvé des applications dans de nombreux domaines. Nous en proposons ici une rapide revue.

3. Global Positioning System
4. Personnal Digital Assistant

2.1.1 Les précurseurs

Datant du début des années 90, les premiers travaux en ubiquité numérique ont été développés au sein des laboratoires Olivetti (*Active Badge* [8]) et Rank Xerox (*ParcTab* [9]). Ils proposent principalement des services liés à la localisation de leurs utilisateurs au sein d'un bâtiment.

Active Badge est un système de localisation pour des personnes situées dans un immeuble. Les utilisateurs sont munis d'un badge signalant leur présence (et donc celle de leur porteur) à une infrastructure de capteurs. Leur localisation peut être exploitée par différentes applications telles que l'affichage des positions sur un plan ou encore le transfert des appels téléphoniques vers le combiné le plus proche du destinataire.

ParcTab propose aux utilisateurs de jouer un rôle plus actif en leur permettant d'interagir directement avec le système. Dans cet objectif, chacun d'entre eux dispose d'un terminal mobile, appelé *Tab*, relié via un réseau cellulaire infrarouge à une infrastructure de communication. Plusieurs applications exploitent cette architecture. Les utilisateurs se voient ainsi proposer la possibilité d'accéder à des informations contextuelles (un répertoire étant associé à chaque pièce), de sélectionner des périphériques en fonction de leur éloignement physique, ou encore de déclencher des applications lorsqu'un certain contexte est détecté.

2.1.2 Les systèmes de localisation

Cette classe d'application, dérivée des travaux précédents, regroupe l'ensemble des systèmes principalement axés sur la localisation physique des utilisateurs. A l'image de *Cooltown* [10], *Impulse* [11], *Mobisaic* [12] et *comMotion* [13] (ce dernier étant également utilisé en tant qu'aide-mémoire, voir ci-après), certains d'entre eux délivrent, de façon générique, une information contextuelle en fonction de la localisation des utilisateurs.

D'autres prototypes ont été conçus dans l'optique d'applications particulières. C'est, par exemple, le cas des guides virtuels *GUIDE* [14], *Cyberguide* [15], *Metraunot* [16], *C-Map* [17], ou encore *Hippie* [18]. Ce dernier exploite, conjointement à un historique des informations visualisées, les positions successives des utilisateurs afin d'en déduire leurs préférences, ce qui permet, par la suite, d'adapter le contenu des informations délivrées.

L'assistance au *shopping* représente une autre classe d'applications dédiées. Elle peut être déclinée sous plusieurs formes. *DealFinder* [19] est, par exemple, un système de comparaison d'offres interrogeable contextuellement par rapport à la localisation

de ses utilisateurs. *ShopNavi* [20] et *Personal Shopping Assistant* [21] sont, pour leur part, des applications vouées à accompagner les clients au sein d'un magasin. Elles exploitent, à l'instar de *Hippie*, un contexte plus riche que la simple localisation des utilisateurs. De fait, elles s'appuient sur la proximité physique des articles à vendre et, dans le but de réaliser des offres publicitaires personnalisées, sur un profilage des clients.

Certains systèmes sont exclusivement basés sur l'exploitation de la proximité physique des objets. C'est notamment le cas de *Chameleon* [22] qui présente à l'utilisateur des informations relatives aux objets qu'il désigne.

2.1.3 Les aide-mémoires

Les aide-mémoires constituent une autre classe d'application traditionnelle de l'ubiquité numérique. Ils sont conçus pour soutenir la mémoire de leur utilisateur dans deux types de situation : d'une part, l'enregistrement et l'indexation d'événements passés, d'autre part, la notification de la survenance de situations particulières. La définition de ces situations intègre généralement une notion de contexte bien plus large que la simple localisation des utilisateurs (les utilisateurs voisins, l'activité courante ...).

Forget-me-Not [23] et *StartleCam* [24] relèvent de la première classe d'aide-mémoires. Ils consignent certaines classes d'événements, ainsi que leur contexte respectif, afin de reconstituer une sorte d'emploi du temps de leurs utilisateurs. De cette façon, ces derniers ont la possibilité, en interrogeant le système, de retrouver le contexte de situations passées à partir de simples détails. Les systèmes de notification, tels que *CybreMinder* [25], *Stick-e Notes* [26] ou encore *comMotion* [13], ont, pour leur part, à charge de scruter le contexte courant de leur utilisateur. Lorsqu'apparaît une situation prédéfinie, ils doivent en avertir l'utilisateur, le plus souvent par le biais de signaux sonores et/ou lumineux. La notification peut, de plus, être accompagnée de la visualisation d'un document attaché à la situation détectée.

FieldNote [27], bien qu'il ne soit pas, à proprement parler, un système d'aide-mémoire, peut être rapproché de cette classe d'applications. Destiné à assister les tâches d'observations en environnement extérieur, il annote automatiquement, au moyen du contexte détecté (localisation, heure, température ...), les relevés effectués par les utilisateurs. Il permet, de plus, de consulter les annotations effectuées par d'autres observateurs dans des conditions similaires.

2.1.4 Les applications de rencontres

Cette dernière classe d'applications exploite la proximité physique entre utilisateurs afin d'enrichir les rencontres au moyen de services contextuels. Elle fait l'objet d'un intérêt accru depuis l'apparition des technologies de communication sans fil de proximité telles que *Bluetooth* [28]. Ces dernières permettent, il est vrai, de mettre en œuvre des interactions entre terminaux mobiles voisins sans nécessiter d'infrastructure de communication.

Ces interactions peuvent, comme c'est le cas pour *Proxy Lady* [29], prendre la forme de simples notifications déclenchées par la découverte de voisinages jugés intéressants. Ce type de service peut alors être rapproché des aide-mémoires présentés précédemment.

Cependant, les rencontres peuvent également donner lieu à des interactions plus complexes telles que le partage spontané d'information. C'est, par exemple, ce que propose *Proem* [30] par le biais d'échanges de profils utilisateur. C'est également la fonction de *SIDE Surfer* [31] qui, sur la base de profils utilisateur générés automatiquement, offre de télécharger spontanément les documents pertinents présents dans le voisinage physique des terminaux mobiles. Les *Wearable Communities* [32] vont plus loin en proposant de mettre en place un système de coopération automatisée entre utilisateurs voisins.

Ces différents systèmes exploitent tous des communications directes entre terminaux mobiles. Des applications de rencontres ont également été réalisées à partir d'infrastructures de communication fixes. On peut citer l'exemple du prototype *Pirates!* [33] qui, dans le domaine du ludique, permet à des joueurs physiquement proches de s'affronter dans des joutes navales.

2.2 Les problèmes soulevés

La mise en œuvre d'applications contextuelles soulève un certain nombre de problèmes qui ne sont pas traités par les systèmes mobiles classiques. L'introduction du contexte des utilisateurs comme paramètre à part entière des applications est bien évidemment l'élément le plus novateur. Les applications doivent, d'une part, être en mesure de capturer les différents éléments de contexte qui les intéressent (tels que la localisation ou l'heure) et, d'autre part, pouvoir les synthétiser afin d'en déduire un contexte global.

Les applications d'ubiquité numérique doivent plus particulièrement apporter des réponses spécifiques à deux classes de problèmes. La première concerne la confidentialité des informations manipulées par le système. Il est possible, dans ce domaine, de

distinguer deux aspects : d'une part, la sécurité des informations de contexte captées par le système et, d'autre part, celle des données que le système est amené à diffuser. La seconde classe de problèmes concerne, quant à elle, l'architecture proprement dite des applications.

Confidentialité du contexte. Le contexte d'un utilisateur est généralement composé (au moins en partie) d'un ensemble d'informations personnelles : son identité, mais aussi ses préférences, sa localisation, ou encore un historique de ces différentes variables. Ces informations, peu ou pas exploitables lorsque manipulées séparément, revêtent un caractère particulièrement sensible une fois regroupées. Afin de pouvoir garantir aux utilisateurs d'un système que les données les concernant ne seront accessibles qu'aux individus autorisés, la manipulation et le stockage de ces dernières doivent être conçus avec le plus grand soin. Ce type de problème de sécurité n'est, bien sûr, pas l'apanage de l'ubiquité numérique; il se trouve cependant ici accentué par le caractère personnel des données manipulées.

Confidentialité des informations. Outre la confidentialité des données concernant les utilisateurs, l'ubiquité numérique doit composer avec un problème de sécurité plus spécifique. En effet, l'information et les services dispensés par les systèmes d'ubiquité numérique sont fortement liés au contexte de l'utilisateur. Ils n'ont de valeur que dans un certain contexte et, bien souvent, leur accès est restreint aux seuls utilisateurs pouvant justifier de ce contexte. L'accès à l'information n'est donc plus seulement conditionné par le paramètre statique qu'est l'identité des utilisateurs, comme c'est souvent le cas dans les systèmes classiques. Il faut également prendre en compte le facteur dynamique que représente le contexte de ces derniers. Kindberg et Zhang se sont déjà intéressés à cette problématique en considérant plus particulièrement l'aspect localisation [34].
A titre d'exemple, on peut imaginer d'autoriser spontanément les personnes présentes dans une pièce à utiliser le vidéo-projecteur de la salle. Cependant, afin d'éviter qu'un utilisateur extérieur à la pièce puisse manipuler le projecteur, il est nécessaire de disposer d'un mécanisme permettant de vérifier la réalité physique des contextes présentés par les utilisateurs au système. De plus, une fois qu'un utilisateur a pu accéder par ce biais à des ressources sécurisées, il convient de maintenir un contrôle de son contexte afin de pouvoir modifier les droits d'accès lorsque ce dernier évolue.

Architecture des systèmes. La mise en correspondance des mondes physiques et numériques, telle que décrite par Weiser, passe par une architecture complètement

répartie au sein de laquelle interagissent une multitude de calculateurs intégrés à notre environnement. Bien que cette vision ne soit pas encore d'actualité, les développeurs dans le domaine de l'ubiquité numérique ont, d'ores et déjà, plusieurs types d'architecture à leur disposition. Ils doivent notament effectuer des choix concernant l'infrastructure de communication (globale, locale voire hybride), ou encore la stratégie de distribution des services et des informations proposés par le système.

La prochaine partie est consacrée à l'étude des architectures mises en œuvre par les différentes applications d'ubiquité numérique présentées précédemment. Chaque type d'architecture y est décrit dans les grandes lignes avant d'être illustré par un aperçu des systèmes l'ayant adopté. Points forts et faiblesses de ces différentes architectures sont alors mis en lumière.

3 Techniques d'accès à l'information

Toute application d'ubiquité numérique peut être considérée selon les trois aspects suivants : le service offert aux utilisateurs (cf. paragraphe 2.1), le type de contexte manipulé par l'application (se reporter au document de Chen et Kotz [4]) et, enfin, l'architecture utilisée pour déployer le service. Nous proposons, dans cette partie, une étude des systèmes d'ubiquité axée sur les différences entre architectures. En ubiquité, comme d'ailleurs en informatique nomade, les applications se trouvent bien souvent distribuées entre les terminaux mobiles des utilisateurs et une infrastructure fixe. Traditionnellement, cette répartition se répercute au niveau de la gestion des services et du stockage de l'information, offrant ainsi un critère intéressant pour la comparaison de différents prototypes. Cependant, la prise en compte, en ubiquité, du contexte utilisateur comme paramètre à part entière permet d'élargir passablement l'ensemble des éléments comparatifs. De fait, la capture de son contexte par une application, quelle que soit la nature de celui-ci, peut, à l'image des services, être distribuée entre terminaux mobiles et infrastructure. Là encore, différentes politiques de répartition peuvent être observées. Il en est de même pour l'exploitation du contexte par les mécanismes de sélection et d'accès à l'information, exploitation qui peut s'opérer, selon les cas, soit au niveau des terminaux mobiles, soit au niveau de l'infrastructure. Cette tâche peut, de plus, donner lieu à différentes mises en œuvre suivant qu'elle soit acquittée par un processus indépendant ou bien intégrée à un mécanisme de nommage conventionnel.

L'ensemble des critères sus-cités, bien que satisfaisant, n'offre pas, à nos yeux, l'approche permettant de rendre compte au mieux de la spécificité des applications d'ubiquité, à savoir l'intégration du contexte au modèle applicatif. C'est pourquoi

nous sommes nous plus particulièrement intéressés à la corrélation, observable parmi les différents prototypes étudiés, entre contexte et politique de répartition. Nous avons, dans cet objectif, dégagé deux catégories d'applications : d'une part celles dont l'architecture est indépendante du contexte d'utilisation et, d'autre part, celles dont la politique de distribution (des services et de l'information) tient compte de ce contexte. Dans le premier cas, nous qualifions les applications d'applications *affranchies du contexte* alors que, dans le second, il s'agit d'applications *réparties contextuellement*.

Cette classification peut aisément être illustrée à l'aide du facteur localisation. Dans les systèmes dits distribués, l'information sera stockée par des machines directement situées là où l'information est pertinente. A l'inverse, dans un système centralisé, l'information pourra être stockée indifférement sur une ou plusieurs machines appartenant à l'infrastructure, ou encore être répliquée intégralement sur chacun des calculateurs mobiles appartenant au système. Prenons, par exemple, le cas d'un système permettant d'accéder à des informations météorologiques. Dans le cas d'une architecture affranchie du contexte, l'ensemble des données va être regroupé sur un même serveur. Par contre, si on considère une architecture distribuée contextuellement, les informations climatiques vont être réparties géographiquement (par exemple par région ou par ville).

Notre essaierons, au fil de cette section, de mettre en évidence une évolution des architectures exploitées par les applications d'ubiquité, de mises en œuvre indépendantes du contexte d'exploitation vers une répartition contextuelle de plus en plus importante.

3.1 Architectures affranchies du contexte

Nous nous attachons, dans cette première partie, aux systèmes au sein desquels l'information n'est pas distribuée par rapport au contexte manipulé. Deux cas de figure peuvent être rencontrés : soit l'information est stockée au niveau d'une infrastructure indépendante du contexte (composée d'une ou plusieurs machines), soit toute l'information est répliquée sur chacun des calculateurs du système.

3.1.1 Systèmes à réplication totale

La réplication complète des informations sur l'ensemble des terminaux est une technique mise en œuvre par relativement peu d'applications. Elle est utilisée par les systèmes, tels que les guides virtuels, proposant l'accès à un ensemble de données

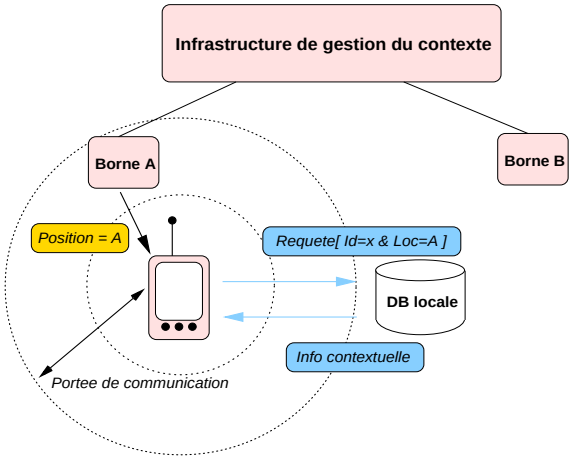


FIG. 1 – *Accès à des données répliquées localement*

communes. C'est notamment le cas de *Cyberguide* [15, 35], de *Chameleon* [22], ou encore de l'application de guide basée sur les *Stick-e Notes* [26, 36, 37].

L'architecture de ces systèmes est dérivée de celle présentée par la figure 1. Les données manipulées par le système sont répliquées sur chaque calculateur mobile. Le contexte peut être acquis localement (acquisition de l'orientation par un compas électronique), à l'aide d'une infrastructure dédiée (un système GPS pour la localisation) voire par une combinaison de ces deux sources. A partir des éléments de contexte collectés, le terminal mobile est en mesure de contruire une requête qu'il soumet localement à sa base de données. Il obtient ainsi les informations relatives à son contexte. L'accès à l'information, qu'il s'agisse de la synthèse du contexte ou du stockage des données, est donc intégralement géré au niveau des terminaux mobiles. Ce type d'architecture a été déployé par au moins trois prototypes.

Le guide virtuel développé à partir des *Stick-e Notes* de Brown recourt à la réplication complète de l'information. Les utilisateurs du système sont chacun munis d'un terminal mobile capable de capter son contexte (localisation, œuvres à proximité) à l'aide de l'infrastructure appropriée (GPS, technologie *Active Badge*, système de code-barres ...). Les informations ayant trait à la visite étant répliquées sur chacun des terminaux mobiles, le système est donc en mesure, lorsqu'est détecté un contexte correspondant à un lieu ou un objet digne d'intérêt, de présenter à l'utilisateur les informations correspondantes.

Cyberguide est un système de guide pouvant être exploité, en recourant à l'infrastructure appropriée, aussi bien en intérieur qu'en extérieur. Un prototype a d'ailleurs été développé pour chacun de ces deux environnements (visite d'une ville et d'un laboratoire). L'architecture et les fonctionnalités restent, dans les deux cas, identiques. Le système est organisé en quatre composants, le *cartographe*, la *bibliothèque d'information*, le *positionnement* et les *services de communication*, dont les interfaces masquent la variété des technologies employées pour les différentes implémentations. Le module de *positionnement* cache ainsi les distinctions entre le système GPS utilisé en extérieur et la localisation cellulaire par infrarouge utilisée en intérieur.
L'application permet à chaque utilisateur de consulter, via son terminal mobile, la carte du lieu visité, carte sur laquelle apparaît sa position ainsi que celles des différents points d'intérêt. La sélection de l'un d'entre eux permet d'accéder à un ensemble d'informations lui étant associées. Ces dernières, répliquées sur chacun des terminaux mobiles, sont localement gérées par la *bibliothèque d'information*.

Enfin, *Chameleon* est un système d'information contextuel exploitant la proximité physique des objets. Les points d'intérêt gérés par le système sont marqués au moyen d'étiquettes ou de codes-barres. Le prototype réalisé ne propose à son utilisateur qu'une mobilité réduite, ce dernier ne manipulant qu'un écran mobile relié par un câble à sa station de travail. C'est sur cette dernière que sont stockées les données manipulées par l'application. Dans la mesure où le terminal manipulé par l'utilisateur n'est qu'un périphérique de sortie, la station ne peut être considérée comme un serveur d'information. La position spatiale de l'écran est utilisée à la façon d'un périphérique d'entrée 6D : la localisation de l'écran en 3D à laquelle s'ajoutent son orientation, son inclinaison ainsi que son assiette. L'écran est par ailleurs muni d'une caméra numérique permettant l'identification des objets pointés par l'utilisateur.

Avantages. Le point fort de ce type d'architecture est, bien évidement, de disposer localement des données à présenter aux utilisateurs. En effet, ces systèmes sont autonomes en ce qui concerne l'accès à l'information et restent fonctionnels malgré l'absence de moyens de communications (par exemple, lorsqu'un utilisateur sort des zones de couverture). Même en présence d'un accès réseau, l'accès à l'information ne souffre pas des délais imposés par la latence de ce dernier.
La réplication des données permet par ailleurs au système de rester opérationnel en cas de défaillance de l'un des calculateurs. Il reste toutefois dépendant du bon fonctionnement de l'infrastructure de gestion du contexte. Enfin, le fait de disposer localement de l'ensemble de l'information permet également d'éviter tous risques

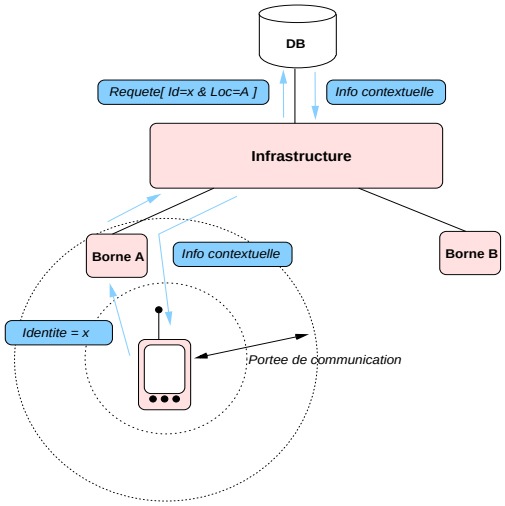
d'interception de données confidentielles lors de communications.

Inconvénients. Le passage à l'échelle est bien évidement le problème majeur de ce type d'architecture. Il n'est, par exemple, pas envisageable de stocker de grosses bases de données sur des calculateurs de type PDA. En effet, puissance de calcul et espace de stockage des terminaux mobiles restent modestes au regard de ce qui est disponible sur les stations de travail. Du fait d'une dépendance vis-à-vis des batteries embarquées, leurs ressources énergétiques sont également limitées. Un iPAQ Pocket PC H3630 de Compaq ne dispose ainsi que d'une autonomie théorique (*i.e.* d'après le constructeur) de douze heures. Cet ensemble de contraintes ne peut donc être ignoré lors de la mise en œuvre d'applications utilisant ce type d'architecture.
De plus, la mise à jour d'un système d'information répliqué, dans le cas où les données sont partagées par tous les utilisateurs (par exemple pour un système de guide), devient rapidement impossible lorsque croît le nombre de réplicas. Ce type d'architecture est donc réservé à des déploiements très restreints et ne peut être mis en œuvre à plus grande échelle.

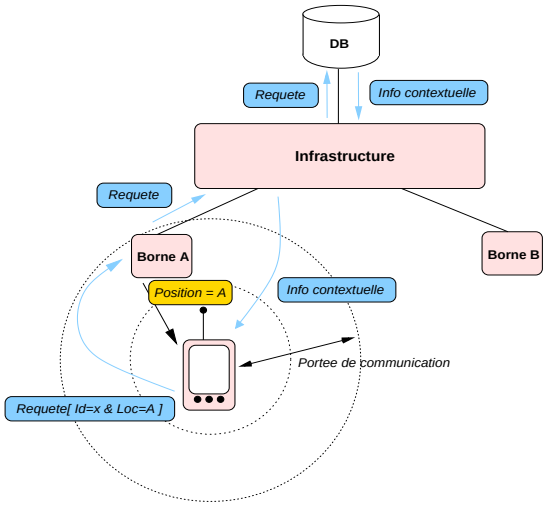
3.1.2 Systèmes à infrastructure dissociée du contexte

Nous venons de le mentionner, outre les contraintes inhérentes à leur architecture, les systèmes s'appuyant sur une stratégie de réplication complète ne peuvent guère être déployés que pour des applications de partage de données communes. Parallèlement à ce premier schéma s'est donc développée une autre classe d'architecture tirant parti d'infrastructures non plus limitées aux tâches relevant de la capture du contexte mais impliquées dans la gestion des informations. Performant et relativement souple, ce type d'architecture a été employé aussi bien pour des systèmes de localisation (*ParcTab, C-Map*) que pour des applications d'aide-mémoire (*Forget-me-Not, CybreMinder*).

La figure 2 illustre les techniques d'accès à l'information mises en œuvre lorsque les données sont gérées par une infrastructure décorrélée du contexte d'exploitation. Le système qui y est décrit dispose d'une unique base de données, accessible via l'infrastructure, regroupant l'ensemble des informations utiles à l'application. La communication s'opère grâce à un réseau cellulaire, les calculateurs mobiles étant dotés de capacités de communication à courte portée, assurant également la localisation des terminaux. D'autres configurations peuvent être rencontrées qui exploitent des bases de données réparties (toujours indépendantes du contexte d'utilisation), des systèmes de communication globaux ainsi que différentes technologies de localisation.



(a) Contexte synthétisé par l'infrastructure



(b) Contexte synthétisé par le terminal

FIG. 2 – *Accès aux données via une infrastructure non contextuelle*

Deux cas de figures apparaissent figure 2 illustrant les cas où la synthèse du contexte est effectuée par l'infrastructure (figure 2(a)) ou les terminaux mobiles (fi-

gure 2(b)). Dans la première configuration, l'identité des utilisateurs, diffusée en tant qu'élément de contexte par chaque terminal, est captée par la borne la plus proche. Cette dernière, détenant le couple (identité, localisation), la localisation étant déduite de l'identité de la borne, est alors en mesure de construire la requête contextuelle appropriée à la situation de l'utilisateur. Dans la seconde configuration, l'infrastructure fournit aux terminaux, via ses bornes émettrices, l'ensemble des variables contextuelles gérées par des capteurs globaux. Les terminaux reçoivent ainsi leur localisation de la borne la plus proche. Ils peuvent alors générer la requête associée au couple (identité, localisation), la retourner à la borne qui, elle-même, la soumet au système d'information global via l'infrastructure de communication. Dans les deux cas, la requête obtenue peut alors être soumise à la base de données qui retourne l'information correpondant au contexte communiqué. Cette information peut enfin être délivrée à l'utilisateur. La suite de cette partie va permettre d'illustrer la mise en œuvre de ce type d'architecture à travers la description des prototypes qui l'implémentent.

Le système *ParcTab* [9, 38, 39] propose un ensemble d'applications de bureau. Il repose sur un réseau infrarouge cellulaire dans lequel chaque cellule est associée à une pièce. Afin d'être localisés par l'infrastructure, et ce même en l'absence de communication, les calculateurs mobiles, appelés *Tabs*, émettent périodiquement un message d'identification. Ces messages, captés par les bornes infrarouges, sont transmis, via un réseau local classique, à un service de localisation central. Le contexte est donc ici géré au niveau de l'infrastructure. Les *Tabs* communiquent avec les serveurs applicatifs, pour la plupart exécutés sur des stations de travail, par l'intermédiaire de l'agent, lui-même situé sur l'infrastructure, qui est associé à chacun d'entre eux. Parmi les services développés, se trouvent être un système de fichiers localisés (associant un ensemble de fichiers à chaque pièce), une application de tableau partagé ou encore un *pager* contextuel.

Le système *Active Badge* [8], développé par Olivetti, utilise le même type d'architecture. Pour mémoire, il s'agit d'un système de localisation sur lequel est notamment basée une application de transfert automatique des appels téléphoniques. La localisation est réalisée par un réseau de capteurs qui détecte les signaux infrarouges émis par les badges portés par les utilisateurs. Les données ainsi collectées sont regroupées, via un réseau local, au niveau d'un serveur de localisation qui est interrogé par les applications clientes.

Metraunot [16] est un système de guide, mis en œuvre dans le cadre d'un campus, permettant à ses utilisateurs de s'orienter. La localisation est effectuée par les utilisateurs en scannant les codes-barres disposés à chacun des endroits gérés par l'application. Certains de ces codes-barre sont utilisés pour l'annonce d'événements :

en les scannant, l'utilisateur les intègre spontanément à son emploi du temps. Les informations manipulées par le système sont stockées au sein de trois bases de données accessibles via Internet : la première est affectée aux localisations, la seconde à la navigation entre lieux et la dernière aux événements. Les calculateurs mobiles accèdent à ces bases au moyen d'une interface de communication sans fil. *Metraunot* confie la gestion du contexte aux terminaux qui ont à charge, à partir des données de localisation et des emplois du temps, de synthétiser un contexte puis d'interroger le système d'information global.

C-Map [17] est un système de guide pour expositions. Il permet aux utilisateurs de repérer sur une carte les points d'intérêt proches. Il se démarque des autres guides par la prise en compte des préférences des utilisateurs (sous forme de mots-clés), ce qui lui permet de générer dynamiquement des visites personnalisées. La localisation des utilisateurs repose sur le système *Active Badge* [8]. Les informations, centralisées au sein d'une unique base de données, sont accédées en deux temps depuis les calculateurs mobiles : un réseau local sans fil permet de communiquer avec des bornes qui sont elles-mêmes reliées au reste du système via un réseau filaire. Le modèle applicatif est de type client/serveur, chaque utilisateur étant représenté par deux agents, un client exécuté sur le calculateur mobile et un serveur situé sur l'infrastructure. L'agent client appelle les méthodes de l'agent serveur en lui transmettant une chaîne de caratères exécutée en tant que script.

CybreMinder [25] est, quant à lui, un système d'aide-mémoire de type notification qui prend la forme d'un mécanisme de délivrance de messages. A chacun des messages créés est associée une date d'expiration. Dans l'attente de celle-ci ou de la délivraison, les messages sont stockés par un serveur au niveau de l'infrastructure. Les notifications peuvent intervenir sous plusieurs formes, suivant les préférences et la situation de l'utilisateur. Elles peuvent ainsi prendre la forme d'un mél ou d'un SMS[5], pouvant être directement délivrés sur des terminaux mobiles. Elles peuvent également être effectuées au moyen de l'imprimante ou de l'écran le plus proche du destinataire, auquel cas le contexte physique de ce dernier est pris en compte non seulement dans le déclenchement de la notification mais aussi dans le processus même de cette dernière. En ce qui concerne la gestion du contexte, *CybreMinder* repose sur le *Context Toolkit* [40] développé par Dey.

Mobisaic [12] est un système d'information sensible à la localisation des utilisateurs. Il s'appuie sur une architecture Web : les informations sont stockées sur des serveurs HTTP[6] classiques [41]; les clients peuvent y accéder depuis leur terminal

5. Short Message Service
6. HyperText Transfer Protocol

mobile par le biais de communications sans fil entre le terminal et une infrastructure de communication. *Mobisaic* introduit deux mécanismes, les URLs [7] dynamiques et les documents actifs, afin d'intégrer la notion de contexte à l'architecture Web. Une URL dynamique est une URL dans laquelle il est possible d'inclure des variables d'environnement (telles que la localisation de l'utilisateur) en tant que paramètres. De cette façon, une même URL, de type :

```
http://www/offices/$(location).html$
```

permet de désigner différentes pages Web suivant la valeur de la variable *location*. La résolution d'une URL dynamique (contenant une ou plusieurs variables environnementales) en URL statique est effectuée au niveau du terminal client, de telle sorte qu'il est possible d'utiliser des serveurs Web classiques.
Le second mécanisme proposé, les documents actifs, doit permettre au client de disposer d'une information en adéquation avec son contexte lorsque ce dernier évolue. En effet, la seule utilisation d'URLs dynamiques ne permet pas de mettre à jour spontanément les informations affichées par le *browser* Web du client lorsque le contexte de ce dernier (par exemple sa localisation) change. Lors de la réalisation d'une page Web active, il revient au développeur d'assigner à la page l'ensemble de variables environnementales auxquelles elle est sensible (identité de l'utilisateur, sa localisation ...). Lorsqu'un client charge un document dynamique, il doit *s'abonner* aux différentes variables relatives à ce document. Les abonnements des différents clients du système sont gérés au niveau de l'infrastructure. Lorsqu'un élément de contexte auquel un utilisateur est abonné vient à évoluer, l'infrastructure transmet à ce dernier la nouvelle valeur. Le terminal client peut ainsi procéder à une nouvelle résolution de l'URL dynamique du document visualisé en utilisant la nouvelle valeur de la variable contextuelle. La requête HTTP adaptée au nouveau contexte de l'utilisateur peut alors être transmise à l'infrastructure.

Forget-me-Not [23] est un système d'aide-mémoire dont l'objectif est la mémorisation spontanée du contexte rencontré lors des différentes activités de ses utilisateurs. Il s'appuie sur l'architecture *ParcTab* présentée précédement. Le contexte considéré inclut la date, l'heure, l'identité des personnes présentes, ainsi que l'activité en elle-même (envoi/reception de méls, téléphone). Les *logs* ainsi récoltés sont centralisés au sein d'une unique base de données, les *Tabs* ne servant que d'interface avec le reste du système. Les utilisateurs cherchant à retrouver le contexte de situations passées s'en servent pour interroger la base de données.

StartleCam [24] propose le même type de service que *Forget-me-Not*. Cependant, au lieu de s'intéresser, comme ce dernier, aux activités numériques de son utilisateur,

7. Universal Resource Locator

il se focalise sur ses moments de stress, ceux-ci étant décelés au moyen d'un capteur physiologique. Outre ce capteur, chaque utilisateur est équipé d'un terminal mobile doté d'un accès Internet sans fil ainsi que d'une caméra numérique. Lorsqu'est détectée une situation de stress, le système effectue une série de photos qu'il adresse à un serveur accessible par Internet, les photos pouvant également être stockées sur un disque dur local.

Le prototype *comMotion* [13] est un système de type notification exploitant exclusivement la localisation de ses utilisateurs. Chacun d'entre eux dispose d'un PC portable doté d'un accès sans fil à Internet ainsi que d'un GPS. Le système, capable de découvrir par apprentissage les lieux caractéristiques de chaque utilisateur, génère, pour chacun d'entre eux, une *to-do-list* destinée à contenir les notes liées aux notifications créées par l'utilisateur. Ces notes sont stockées sur le PC portable de leur utilisateur, leur délivrance pouvant intervenir sous forme vocale ou textuelle. Le système permet également de "poster" des rappels (*i.e.* des messages liés à un lieu et/ou une date), sous forme de méls, aux utilisateurs. Ces rappels sont alors transmis à un serveur avant d'être transférés, dès que cela est possible, sur le portable du destinataire.

FieldNote [27] est un système d'annotation automatique de contexte pour les travaux d'observations en environnement extérieur. Les utilisateurs disposent, dans ce cadre, de terminaux mobiles dotés d'interfaces de communication sans fil. Lors de la saisie d'observations, le système annote, de façon spontanée, les relevés effectués par les utilisateurs. Il utilise, pour ce faire, les capteurs dont il dispose localement : un GPS pour la localisation ou encore un thermomètre pour la température. Les relevés, une fois annotés, sont transmis, via une connection Internet de type GSM, à une base de données centralisée. De la même façon qu'il transmet ses propres observations, un utilisateur peut consulter celles que d'autres observateurs ont pu effectuer dans des conditions similaires. En cas d'absence d'interface de communication sans fil, le système peut fonctionner par réplication, sur chaque terminal mobile, de tout ou partie de la base de données globale. On retrouve, dans ce cas de figure, le schéma de fonctionnement exposé dans la partie précédente.

Dans le domaine des guides virtuels, *Hippie* [18] apparaît comme l'un des prototypes les plus aboutis. Les informations associées à la visite sont accessibles de deux façons : depuis Internet, au moyen d'une navigation thématique, ou directement sur place, par le biais d'une navigation physique entre les différents points d'intérêt. Dans le second cas, les utilisateurs disposent de terminaux mobiles reliés à un réseau local sans fil. Leur localisation repose sur une infrastructure d'émetteurs infrarouges diffusant vers les terminaux mobiles l'identité du point d'intérêt auprès duquel ils

sont disposés, l'orientation des visiteurs étant, quant à elle, acquise au moyen du compas électronique adjoint aux terminaux. Chaque terminal est ainsi en mesure de fournir le couple (position, orientation) à l'infrastructure qui, en retour, lui transmet l'information correspondante. Outre la localisation, *Hippie* exploite activement les préférences des utilisateurs, préférences découvertes par analyse d'une part de la navigation physique et, d'autre part, des informations consultées. Cette analyse, effectuée au niveau de l'infrastructure, permet de réaliser des suggestions de visites correspondant aux centres d'intérêt des utilisateurs ainsi qu'une personnalisation de la présentation qui est faite des différentes attractions.

Avantages. Les avantages de ce type d'architecture sont multiples. Tout d'abord, elles offrent un mécanisme unique permettant d'accéder à la fois aux informations contextuelles et non contextuelles. Ensuite, le fait de centraliser l'ensemble des informations permet une administration du système plus simple. Enfin, l'utilisation d'une architecture combinant les aspects communication et gestion du contexte (par opposition aux architectures dédiées au contexte, voir figure 1) permet d'exploiter les technologies de communications de proximité en tant que systèmes de localisation.

Inconvénients. Les systèmes que nous venons d'étudier souffrent des limitations inhérentes aux systèmes centralisés. Ainsi, la base de données centralisée constitue un potentiel goulot d'étranglement. De plus, elle rend le système particulièrement sensible aux pannes puisqu'en cas de défaillance de la base, ce dernier est entièrement paralysé (la disponibilité des informations pouvant cependant être accrue par la mise en œuvre d'un système de réplication).
En outre, chaque accès à l'information (contextuelle ou non) nécessite désormais l'utilisation de ressources réseau (dans la mesure où toutes les informations sont délocalisées). Dans cette situation, il devient nécessaire de considérer attentivement les problèmes de confidentialité des informations transmises aux utilisateurs. En effet, et contrairement aux systèmes à base de réplication totale, l'information transite ici par différents réseaux (au sein de l'infrastructure d'une part, des bornes aux calculateurs mobiles d'autre part) avant de parvenir à l'utilisateur.
Enfin, nous pouvons observer que ce type d'architecture se démarque de la vision totalement distribuée qu'avait Weiser des environnements d'ubiquité numérique.

3.2 Architectures distribuées contextuellement

Les premières réalisations dans le domaine de l'ubiquité numérique n'ont pas tenu le contexte, et plus particulièrement la localisation, comme un paramètre pertinent

pour la répartition des services. L'influence de ce facteur s'est pourtant accrue au fil des années, au point de conditionner la conception de certaines applications récentes. Deux grandes classes de systèmes peuvent être rattachées aux architectures distribuées contextuellement : les systèmes à infrastructure, dont l'information est localisée, et les systèmes mobiles personnels.

3.2.1 Systèmes d'information localisés

Les prototypes relevant de ce type d'architecture sont articulés autour des différentes localisations administrées par les applications. Les services et les informations proposés doivent ainsi être *physiquement* gérés à l'endroit où ils sont effectivement utiles aux utilisateurs.

La figure 3 présente le schéma de fonctionnement de tels systèmes. L'information est initialement distribuée par rapport au contexte. Sur la figure, les données relatives aux différents lieux sont ainsi directement stockées sur place. Lorsqu'un calculateur mobile s'annonce auprès d'une borne, celle-ci peut construire et soumettre à la base de données locale la requête adaptée. Les informations de localisation (et, plus généralement, les éléments constants du contexte de la borne) n'ont pas besoin d'être inclus dans la requête. Elles sont en effet inutiles puisque seules les informations relatives à ce contexte fixe sont stockées dans la base locale. La requête peut donc être assemblée à l'aide des seuls éléments de contexte variables (tels que l'identité de l'utilisateur présent).

La présence d'une infrastructure de communication globale n'est pas indispensable au bon fonctionnement du système. Cependant, en son absence, les informations non contextuelles ne peuvent être accédées par les utilisateurs. Les informations non contextuelles ne sont diffusées que suite à une requête explicite de la part d'un utilisateur. Dans ce cas, la borne qui reçoit la requête doit être en mesure de localiser la base de données stockant les informations demandées. Elle peut ainsi lui adresser la requête de l'utilisateur. La réponse est alors retournée à celui-ci via la borne locale.

GUIDE [14, 42, 43, 44, 45] est un système de guide conçu pour la visite de villes. Les utilisateurs, munis de calculateurs mobiles, se voient proposer des informations (sous forme de pages Web) relatives aux lieux visités et aux attractions proches (musées, boutiques ...). La communication et la localisation des utilisateurs sont toutes deux effectuées par la disposition de cellules WaveLAN [46], de manière à ce que celles-ci recouvrent les différents points d'intérêt de la ville. Un utilisateur est ainsi localisé en identifiant la borne avec laquelle il communique (leurs zones de couverture doivent donc être disjointes).
Les informations dispensées sont fonctions à la fois de la localisation et des préférences

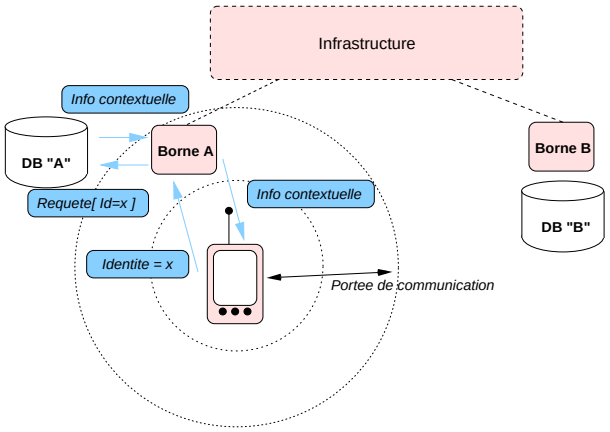


FIG. 3 – *Accès aux données via une infrastructure contextuellement distribuée*

des utilisateurs. Elles sont cependant organisées et stockées suivant la localisation en utilisant des serveurs géographiquement affectés aux différents points d'intérêt de la ville. La diffusion des informations proprement dite repose sur la technique dite des *Broadcast Disks* [47]. Le principe du *Broadcast Disk* est la diffusion continue par les serveurs des informations qu'ils gèrent. Dans *GUIDE*, les serveurs de lieu diffusent donc régulièrement, suivant un calendrier, un ensemble d'information ainsi que leur calendrier de diffusion. Les terminaux mobiles n'ont ainsi plus qu'à attendre la diffusion des informations qui les intéressent. Cependant, dans le cas où l'information attendue n'appartient pas au calendrier de diffusion d'un serveur, le terminal a la possibilité d'envoyer directement une requête à ce dernier afin que l'information recherchée soit intégrée au calendrier.

A la différence du système *Mobisaic* [12] qui génère et stocke l'ensemble des pages potentiellement accessibles, *GUIDE* dispose d'un mécanisme de composition dynamique de pages Web [42, 45]. Ce mécanisme permet de construire des documents "à la volée" en fonction des paramètres contextuels d'un utilisateur. Pour ce faire, les développeurs de pages Web ont la possibilité d'inclure dans leur code HTML[8] des balises telles que :

```
<GUIDETAG INSERT NEIGBOURS>
```

Cette balise permet, par exemple, d'insérer dans un document le code correspondant aux lieux intéressants situés dans le voisinage de l'utilisateur. Alors que les URLs

8. HyperText Markup Language

dynamiques de *Mobisaic* désignent des documents existants (*i.e.* déjà composés), les pages Web de *GUIDE* sont construites dynamiquement lorsqu'elles sont accédées par les utilisateurs.

Cooltown [10, 48, 49, 50], développé par Hewlett Packard, est un système cherchant à attacher une représentation virtuelle aux lieux, personnes et objets de la vie courante. Dans cet objectif, chacune de ces entités est associée à un ensemble de pages Web (on parle alors de *Web presence*) dont l'accès est contrôlé par la proximité physique. La représentation virtuelle des objets est assurée par le déploiement d'un serveur HTTP sur les périphériques pouvant le supporter (c'est, par exemple, le cas de certaines imprimantes). La représentation des autres équipements, ainsi que celle des objets les plus simples (tels que les tableaux d'un musée), est assurée par des serveurs externes. C'est également le cas des lieux dont la présence virtuelle est assurée par des serveurs appellés *Place Managers* [48]. Ces derniers sont d'ailleurs en charge de la représentation des objets non autonomes présents localement. On peut, par exemple, imaginer que le *Place Manager* d'une des salles d'exposition d'un musée soit chargé de la représentation des toiles présentes dans la salle. Dans la mesure où la localisation des serveurs externes n'est pas corrélée à celle des entités représentées, l'architecture de *Cooltown* peut être considérée comme hybride au niveau de la répartition des données.
Les utilisateurs du système sont munis d'un terminal mobile doté de capacités de communication sans fil de courte portée. C'est la proximité physique qui leur permet de "capter" l'URL des entités avoisinantes. Deux cas de figure peuvent se présenter : soit l'entité diffuse, par l'intermédiaire d'un émetteur, l'URL qui lui est associée, soit elle diffuse un identificateur que les terminaux mobiles devront faire traduire en URL par un *résolveur*. Dans le second cas, le choix par plusieurs utilisateurs de *résolveurs* différents permet d'obtenir différentes URLs à partir d'un même identificateur.
Enfin, à l'instar de *GUIDE*, *Cooltown* dispose d'un mécanisme de composition dynamique de documents [48]. En effet, les pages Web présentées aux utilisateurs du système sont construites au moment de l'accès. L'identité des utilisateurs mais aussi l'heure, le lieu (l'utilisateur est-il physiquement présent ou accède-t-il à l'URL via un poste de travail?) et les capacités des terminaux mobiles sont pris en compte lors de la composition des pages Web. Pour ce faire, la grammaire du langage HTML a été enrichie de nouvelles instructions permettant de contrôler le contenu d'un document en fonction de variables contextuelles.

Impulse est un système dit de localisation. Il propose un service d'information contextuelle à ses utilisateurs en exploitant les données disponibles sur le Web. Pour ce faire, chaque utilisateur est équipé d'un terminal mobile, de type *Palm*, lui-même

muni d'un capteur GPS et d'une interface de communication sans fil. Un utilisateur est représenté par un agent utilisateur qui gère ses préférences (stockées sous forme de mots-clés). Le système exploite un serveur *Wherehoo* [51] indexant les sites Web au moyen, d'une part, de leur localisation et, d'autre part, des mots-clés les décrivant. Les lieux et objets enregistrés auprès de ce serveur sont représentés par des "fournisseurs". L'agent utilisateur construit, au moyen de la localisation et des préférences utilisateur, des requêtes HTTP qu'il transmet au serveur *Wherehoo*. Celui-ci lui retourne les URLs des fournisseurs qui lui sont physiquement proches et dont la description correspond aux mots-clés spécifiés. L'agent utilisateur peut alors rapatrier ces pages en s'adressant directement aux fournisseurs concernés.

Avantages. L'accès simplifié aux informations contextuelles constitue le premier apport de ce type d'architecture. La localité des informations permet ainsi au système de fonctionner avec une infrastructure limitée à des serveurs de données locaux. L'aspect tolérance aux pannes peut également être mis en avant. En effet, en cas de défaillance d'un des serveurs d'information, le système reste globalement fonctionnel : seuls les utilisateurs rattachés à la zone du serveur défaillant sont privés de service (l'information stockée par le serveur défaillant étant toutefois inaccessible pour l'ensemble des utilisateurs). Enfin, la répartition des informations entre plusieurs sites rend les tâches d'administration localement moins lourdes. Pour ces raisons, ce type d'architecture est particulièrement bien adapté aux systèmes déployés à grande échelle.

Inconvénients. La principale difficulté soulevée par cette architecture concerne l'accès à l'information non contextuelle, accès qui s'avère impossible en cas d'absence d'une infrastructure de communication globale. Même en présence de cette dernière, ces accès s'avèrent plus complexes que dans les exemples précédents. En effet, les différents pôles de stockage de l'information n'ont *a priori* pas connaissance de la politique de répartition de l'information. Un mécanisme de localisation leur est donc nécessaire pour effectuer des accès aux informations non contextuelles.
Par ailleurs, la répartition des données, lors de la mise en place du système (et éventuellement pour ses mises à jour), exige de ses administrateurs une bonne connaissance du contexte d'utilisation. Ce type de contrainte ne s'applique pas aux systèmes dans lesquels l'information n'est pas répartie par rapport au contexte.

3.2.2 Systèmes mobiles personnels

Partant du constat que certains des objets modélisés par les applications d'ubiquité peuvent se voir facilement déplacés d'un lieu à un autre et que, de ce fait, les informations s'y rapportant ne peuvent être rattachées statiquement à une localisation, plusieurs prototypes ont appliqué le principe de répartition contextuelle à une échelle plus fine que celle de la simple localisation. Ces développements ont concordé avec la récente apparition des technologies de communication sans fil de proximité (également appelées PAN[9]) telles que *Bluetooth* [28]. Ces technologies permettent aux objets munis de capacité de calcul d'être intégrés à des environnements numériques sans avoir à être représentés par des serveurs intermédiaires et, ce, quelque soit leur localisation.

Il est possible de considérer cette classe d'applications comme un prolongement des systèmes d'information localisés, prolongement dans lequel les différents nœuds du système (mobiles ou non) sont autonomes. Le fonctionnement des calculateurs mobiles s'affranchit désormais de toute infrastructure, qu'elle soit de communication ou d'acquisition du contexte. En outre, l'indépendance des calculateurs mobiles se traduit par un stockage local des informations qui leur sont propres. Il revient de plus à chacun des terminaux de gérer son propre contexte. En l'absence d'infrastructure, cette gestion repose intégralement sur la perception du voisinage physique, notamment celle des utilisateurs voisins. C'est pourquoi ce type de système trouve ses principales applications dans le domaine des interactions de personne à personne. D'autres applications peuvent toutefois tirer parti de ce type d'architecture. Il est en effet possible, à l'instar des indices utilisés dans le jeu *Cluefinder* [52], d'utiliser des émetteurs indépendants comme sources d'informations complémentaires.

Le mode de fonctionnement de ce type d'architecture, illustré par la figure 4, est le suivant. Lorsque deux terminaux sont à portée de communication l'un de l'autre, quelque soit le lieu de rencontre (sauf si l'un des deux nœuds est fixe), ils peuvent s'adresser des requêtes. Ces dernières seront, bien sûr, fonction de l'identité de chaque utilisateur mais peuvent aussi incorporer des objets (ou personnes) présents dans le voisinage physique de chacun des deux nœuds. On peut ainsi obtenir des requêtes de type :

```
(Id = x) & (Voisin = {y,z}) & (Objet = {k})
```

où y,z sont des personnes physiquement proches de x et k un objet (tel qu'un indice dans *Cluefinder*) appartenant à son voisinage physique.

9. Personal Area Network

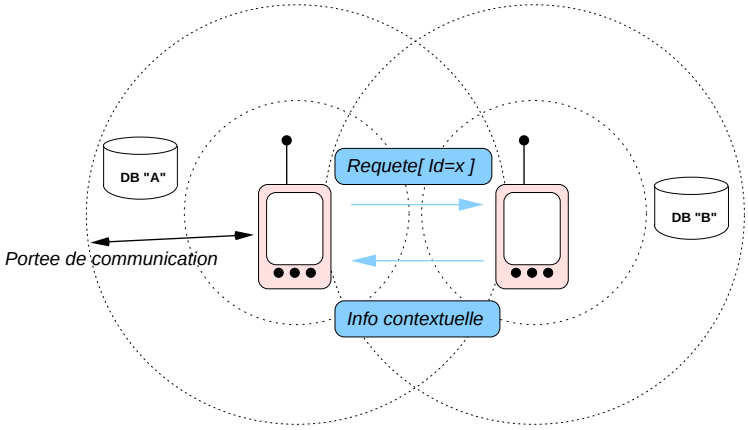


FIG. 4 – *Accès aux bases de données personnelles mobiles*

Le but du système *Proxy Lady* [29] est de faciliter l'amorce d'échanges informels entre personnes lors de rencontres opportunes. Pour ce faire, les utilisateurs de ce système sont munis d'un PDA équipé d'une interface de communication de proximité. Ils ont la possibilité d'associer des informations aux autres utilisateurs du système. Ces informations prennent actuellement la forme de méls qui sont soit transférés, à partir du client de messagerie d'une station de travail vers un PDA (où ils sont stockés), soit directement créés sur le PDA. La proximité physique d'un utilisateur intéressant (*i.e.* auquel est attaché un ou plusieurs messages) est détectée lorsque les PDAs respectifs des utilisateurs sont à portée de communication l'un de l'autre. Dans cette situation, l'utilisateur intéressé est prévenu par le biais d'un signal (sonore et lumineux) et les items associés à l'utilisateur rencontré lui sont spontanément présentés. Bien qu'il n'y ait pas ici d'échanges d'information entre calculateurs mobiles (les méls ne sont pas transférés entre PDAs), les interactions entre ceux-ci sont bel et bien directes. De plus, les informations stockées sur chaque PDA sont des informations personnelles.

L'idée d'interactions directes entre calculateurs mobiles est également à l'origine de l'étude SIS (Systèmes d'Information Spontanés) [53, 54]. Les travaux menés dans le cadre de cette étude ont pour objectif de proposer une plateforme logicielle adaptée aux interactions de voisinage. Cette plateforme regroupe un ensemble d'outils parmi lesquels se trouve un système de prédiction du temps de communication entre nœuds mobiles [55]. Le prototype *SIDE Surfer* [56, 31], développé dans le cadre des SIS, permet de réaliser des échanges directs d'information pertinente entre PDAs mobiles. Le prototype a été, dans un premier temps, appliqué à des interactions de type Web :

il s'agit d'échanger des pages Web décrites au moyen de mots-clés. Le système est en mesure de profiler automatiquement chaque utilisateur par l'analyse des mots-clés caractérisant les documents stockés sur son PDA : les centres d'intérêt des utilisateurs peuvent ainsi être découverts. Lors des rencontres physiques, *SIDE Surfer* est capable de rechercher au sein du voisinage physique les informations pertinentes (vis-à-vis du profil de son utilisateur) et de les rapatrier spontanément. Cet exemple nous permet de constater qu'il est possible d'enrichir les requêtes échangées par les calculateurs mobiles au moyen de paramètres tels que les centres d'intérêt des utilisateurs.

A l'instar des SIS, l'objectif du projet *Proem* [57] est de construire une plate-forme de développement adaptée aux interactions (directes ou non) entre calculateurs mobiles. Un ensemble de services est ainsi proposé aux développeurs : découverte et annonce de la présence de nœuds voisins, gestion de communautés d'utilisateurs... Plusieurs applications ont été réalisées dans ce cadre. La première d'entre elles, répondant également au nom de *Proem* [30], met en œuvre des échanges de profil entre personnes physiquement proches. Là encore, les utilisateurs sont équipés de PDAs capables de communications de proximité. A la différence de *SIDE Surfer*, les profils utilisateur, stockés sur les PDAs, doivent ici être renseignés par les intéressés. Les utilisateurs ont la possibilité de visualiser et d'enregitrer le profil des personnes rencontrées. Ils peuvent également définir des "filtres", basés sur des règles de type *condition* $\Rightarrow$ *action*, sur ces dits profils. De cette façon, ils sont alertés dès qu'est rencontrée une personne dont le profil satisfait une des conditions spécifiées.
Le jeu *Cluefinder* [52] est également basé sur le système *Proem*. Outre les interactions directes entre joueurs, il exploite la présence des sources d'information complémentaire que représentent les indices.

Avantages. Déployer une application en utilisant ce type d'architecture présente un premier avantage, celui d'un coût réduit. En effet, les nœuds mobiles étant autonomes, il n'est plus nécessaire de disposer d'une infrastructure. L'autonomie des calculateurs mobiles permet, par ailleurs, d'envisager le déploiement d'applications dans des zones non couvertes par les infrastructures, voire même en milieu hostile.
Enfin, cette classe d'architecture offre l'avantage de ne pas nécessiter d'administration dans la mesure où elle est complètement distribuée. Les tâches d'administration sont en fait réparties entre les différents utilisateurs du système qui doivent gérer eux-mêmes leurs données personnelles.

Inconvénients. L'absence de toute infrastructure implique une prise en compte, lors de la conception, des contraintes inhérentes aux calculateurs mobiles destinés

à accueillir les applications. Le problème de l'espace de stockage ne s'avère pas ici aussi critique que dans les systèmes à réplication totale puisque chaque terminal n'est supposé stocker que les données propres à son utilisateur. Cependant, l'obligation d'effectuer localement l'ensemble des traitements reste problématique vis-à-vis, notament, des ressources énergétiques limitées. Chaque terminal est, par exemple, en charge de scruter l'apparition du contexte courant parmi l'ensemble des contextes conditionnant le déclenchement d'actions.
Par ailleurs, et c'est là une différence avec les systèmes de réplication, les applications doivent ici se contenter d'une perception du contexte limitée au voisinage physique. Certaines variables, telles que les conditions météorologiques, ne peuvent ainsi pas être captées directement sur des calculateurs mobiles (*i.e.* sans recourir à une infrastructure spécialisée).

3.3 Synthèse

De la revue de systèmes d'ubiquité effectuée dans la partie précédente ont émergées quatre grandes classes d'architectures intégrant, à des degrés divers, le contexte à leur stratégie de répartition. Nous avons mis en lumière, au sein de ces quatre classes, une progression, notamment liée à des aspects technologiques, de l'impact du contexte sur l'architecture des applications. Nous nous proposons maintenant de caractériser chacune de ces quatre classes à l'aide des critères évoqués en début de partie. Pour ce faire, nous revenons, dans un premier temps, sur la description des critères adoptés.

3.3.1 Description des critères

La principale singularité des applications d'ubiquité par rapport aux systèmes dits "classiques" est l'adaptation des services rendus au contexte des utilisateurs. Outre le contexte *numérique*, Chen et Kotz distinguent un contexte *utilisateur*, un contexte *physique* auquel il convient d'ajouter le facteur *temps* [4]. Avant de pouvoir l'exploiter, les applications doivent acquérir une description du contexte. Nous avons choisi de distinguer les prototypes dont les terminaux mobiles sont autonomes en matière de capture du contexte de ceux qui dépendent d'infrastructures spécialisées. Localisation et conditions météorologiques (en général) sont deux exemples de variables contextuelles nécessitant le recours à de telles infrastructures. La localisation, utilisée par la plupart des systèmes, peut être acquise par de nombreuses technologies. Il peut s'agir, pour une application à grande échelle, du système GPS [7] ou, en associant des localisations aux différentes cellules, d'un réseau cellulaire tel que le

GSM [1]. Pour des applications au déploiement plus restreint, il est possible d'avoir recours à des systèmes infrarouges de type *Active Badge* [8], à des systèmes vidéo [58], à des technologies de sols actifs [59, 60], ou encore à des réseaux cellulaires à courte portée de type WaveLAN [46]. Un certain nombre de variables contextuelles reste toutefois obtensibles par le biais de capteurs pouvant être associés à des terminaux mobiles. C'est, par exemple, le cas de la température ambiante (avec un thermomètre [27]), de l'orientation (à l'aide d'un compas électronique [15]), du niveau de stress de l'utilisateur (au moyen de capteurs physiologiques [24]) ou encore du voisinage physique (en exploitant les interfaces de communication sans fil à courte portée [10, 30, 31]).

La sélection contextuelle des services exige des systèmes d'ubiquité que soit effectuée une "synthèse" des diverses sources d'information permettant d'obtenir un état global de la situation. Suivant les prototypes, ce traitement est effectué par les terminaux ou l'infrastructure. Quelques travaux ont été réalisés dans le domaine de la synthèse de contexte. On peut notament penser au *Context Toolkit* [40] qui propose une gestion du contexte s'appuyant sur trois types de composants : les *widgets* (interfaçant chacun des capteurs), les *agregators* (regroupant les informations d'un ou plusieurs *widgets*) et les *interpreters* (permettant d'interpréter les informations captées). Dans le domaine, plus ciblé, de la localisation, plusieurs auteurs ont également proposé des services unifiant les multiples technologies existantes afin de permettre aux applications de disposer d'un service de localisation uniforme [61, 62]. Cependant, les mécanismes de gestion du contexte présentés par ces différentes études ne s'avèrent adaptés qu'aux seules infrastructures.

Quelle que soit la façon dont est géré le contexte, les prototypes présentés peuvent être étudiés en fonction des mécanismes d'accès à l'information utilisés. Ainsi, le nommage de l'information permet de faire apparaître deux familles d'applications : celles à nommage contextuel et celles à nommage non contextuel. Dans un système dispensant des informations contextuelles, chaque entité (telle qu'une personne, un lieu, certains objets) est représentée par un document pouvant revêtir plusieurs formes. Les différentes versions d'un document permettent de disposer de plusieurs représentations d'une même entité. Dans ce cadre, le nommage non contextuel identifie de manière commune les différentes formes d'un document. Considérons l'exemple d'un musée disposant d'un système de guide virtuel. Chaque œuvre du musée peut être illustrée par plusieurs fiches s'adressant à des publics différents (par exemple enfant ou adulte). Les multiples formes de la fiche associée à une œuvre doivent, dans le cas du nommage non contextuel, être désignées par un même identificateur. Le contexte n'est donc pas pris en compte par le mécanisme de nommage. Un méca-

nisme complémentaire doit donc être chargé d'associer à chaque document accédé la forme seyant au contexte courant. C'est, par exemple, le cas pour *Cooltown* [10] dont le nommage est effectué au moyen d'URLs classiques : chacune d'entre elles y désigne un modèle de document à partir duquel sont générées, lors des accès, les pages Web correspondant au contexte. Le nommage de *GUIDE* [14] fonctionne suivant les mêmes principes. D'autres systèmes, tels *Impulse* [11] et *C-Map* [17], appliquent un nommage non contextuel à un ensemble de documents déjà constitués.

Le nommage contextuel fait, quant à lui, directement intervenir la notion de contexte dans la désignation des documents. L'exemple précédent peut être adapté à cette nouvelle politique en affectant aux différentes formes d'une fiche un identificateur propre. Afin de distinguer ces formes entre elles, le contexte est directement intégré au schéma de nommage. Ainsi, les fiches descriptives du musée ne sont plus accédées par le seul identificateur de leur œuvre mais par un couple (IDœuvre, contexte) tel que *("La Joconde", public=enfant)*. Ce mécanisme a notamment été introduit dans *Mobisaic* [12] par le biais des URLs dynamiques. Il s'agit d'URLs au sein desquelles peuvent être incluses des variables contextuelles. Une URL dynamique permet de désigner un des documents gérés par le système. Lorsque ses variables sont instanciées, l'URL ne fait plus référence qu'à une seule occurence de ce document. Le contexte est largement utilisé par les systèmes d'aide-mémoire pour l'indexation de données (*FieldNote* [27]) ou d'événements (*Forget-me-Not* [23]).

Au niveau du stockage des documents, il est de nouveau possible de dégager deux approches : la génération dynamique et le stockage exhaustif. La première approche, implantée notament par *Cooltown* et *GUIDE*, s'avère particulièrement adaptée lorsque les documents doivent être constitués en fonction du contexte perçu. Ce dernier intégrant souvent des variables définies en intention (à l'image de l'identité des utilisateurs), il n'est pas toujours possible de générer à l'avance l'ensemble des versions d'un même document. Par exemple, si la page d'accueil de notre musée reprend le nom de l'utilisateur courant, la génération préalable de l'ensemble exhaustif des pages d'accueil est, dans le cas général, impossible. Cependant, et bien que ce schéma de fonctionnement n'ait jusque-là été que peu expérimenté, il est envisageable, dans le cadre d'une architecture Web, de contourner ce type de difficultés en effectuant une partie de l'adapatation contextuelle d'un document côté client au moyen, par exemple, d'*applets* Java. La seconde approche, le stockage exhaustif, permet, lorsque possible, d'éviter les délais de composition de documents au prix de l'occupation d'un espace de stockage considérable. Malgré ce problème, c'est cette approche qui a été adoptée par la plupart des prototypes réalisés.

TAB. 1 – *Récapitulatif*

Prototype	Capture contexte		Gestion contexte		Nommage contextuel	Composition dynamique
	Infra.	Term.	Infra.	Term.		
Cyberguide	X			X	? [1]	Non
Chameleon		X		X	?	?
Stick-e Notes	X			X	Oui	Non
ParcTab	X		X		?	Non
Active Badge	X		X		* [2]	*
Metraunot	X			X	Non	Non
C-Map	X	X	X		Non	Non
CybreMinder	?		X		Oui	Non
Mobisaic	?			X	Oui	Non
Forget-me-Not	X		X		Oui	*
StartleCam	X	X			?	Non
comMotion	X		X		Oui	Non
FieldNote	X	X		X	Oui	
Hippie	X		X		Non	Non
GUIDE	X			X	Non	Oui
Cooltown	X			X	Non	Oui
Impulse	X			X	Oui	Non
SIDE Surfer		X		X	Non	Non
Proem		X		X	Non	Non
Proxy Lady		X		X	*	*

[1] indéterminté

[2] non applicable

3.3.2 Caractérisation des architectures

Le tableau 1 permet de situer chacun des prototypes étudiés (présentés par classe d'architecture) à la lumière des différents critères adoptés. Il est possible, à l'aide de ce récapitulatif, d'esquisser le profil de chacune des quatre classes d'architectures.

Réplication totale. Les applications de cette classe peuvent être caractérisées par une gestion du contexte au niveau des terminaux mobiles. Il convient de rappeler

que les seules infrastructures utilisées dans ce cadre sont celles dédiées à la diffusion de variables contextuelles (localisation, conditions météorologiques ...). Outre la réplication locale du système d'information de l'application, les terminaux doivent donc supporter l'ensemble des traitements à effectuer. Ces considérations, combinées aux contraintes énergétiques dont sont affligés les calculateurs mobiles, font de cette architecture un modèle inadapté à des réalisations conséquentes.

Infrastructures affranchies du contexte. La déportation, par rapport à la classe d'applications précédente, du système d'information des terminaux mobiles vers une infrastructure dotée de capacités de traitements se traduit, dans bon nombre de cas, par le transfert vers cette infrastructure de la gestion du contexte. Les applications exploitent ainsi les capacités de calcul de l'infrastructure qui sont tout à la fois plus importantes que celles des terminaux et non dépendantes, au point de vue énergétique, de batteries à la durée de vie limitée. Du fait du rôle prépondérant de la localisation dans les systèmes étudiés, l'acquisition du contexte repose, au moins partiellement, sur des infrastructures dédiées. Elle peut être complétée au niveau des terminaux, soit par des capteurs locaux (*StartleCam*, *FieldNote*), soit par la prise en compte des préférences des utilisateurs (*C-Map*). Enfin, de nombreux systèmes de cette classe, notamment les applications d'aide-mémoire (*Forget-me-Not*, *comMotion* ...), incorporent un mécanisme de nommage contextuel.

Infrastructures contextuellement distribuées. Ce type d'applications recourt largement aux infrastructures d'acquisition de contexte. Cependant, à la différence de la classe précédente, la gestion du contexte est désormais effectuée par les terminaux mobiles. Cette évolution est imputable à la distribution contextuelle du système d'information puiqu'une partie de la sélection contextuelle des données est opérée implicitement via la distribution de l'information. Les opérations de sélection contextuelle restant explicites sont généralement celles dépendant du contexte *utilisateur* et sont, de fait, réalisées par les terminaux mobiles. Enfin, les seules mises en œuvre de mécanismes de composition dynamique de documents sont issues des applications de cette classe (*Cooltown* et *GUIDE*).

Systèmes mobiles personnels. Cette dernière classe d'applications suppose une indépendance complète des différents nœuds du système. C'est donc sans surprise qu'acquisition et gestion du contexte sont entièrement à la charge des calculateurs mobiles. Chacun de ces calculateurs est également responsable d'une portion du système d'information (généralement celle qui a trait à l'entité physique à laquelle

est attaché le calculateur) et des traitements qui s'y rapportent. Hormis l'aspect espace de stockage (qui est ici une ressource moins critique), les applications de cette classe doivent composer avec des contraintes identiques à celles des systèmes à réplication, à savoir puissance de calcul et ressources énergétiques limitées.

4 Ubiquité numérique et informatique nomade

Nous avons, dans la partie précédente, proposé un tour d'horizon des techniques utilisées pour l'accès à l'information en ubiquité numérique. Nous allons maintenant montrer en quoi ces mécanismes se distinguent ou, dans certains cas, se rapprochent de ceux qui ont été mis en œuvre pour l'informatique nomade. Nous commençons par introduire les problèmes auxquels les applications nomades sont confrontées. Nous présentons ensuite brièvement les différentes techniques développées dans ce cadre. Enfin, nous discutons des différences et similitudes constatées entre accès à l'information en informatique nomade et en ubiquité numérique.

4.1 Problématique de l'informatique nomade

Nous l'avons évoqué dans l'introduction, l'informatique mobile connaît aujourd'hui un essor important dans les domaines des systèmes de communication entre personnes et des systèmes d'information. Les systèmes d'information nomades cherchent à mettre en œuvre un accès transparent à des données distantes, le plus souvent par le biais de communications sans fil. Ces accès peuvent être effectués par le biais de systèmes à grande échelle, à l'image du WAP [2] qui permet d'accéder au Web à partir de téléphones cellulaires. Ils peuvent également être illustrés par des systèmes au déploiement plus restreint tels qu'un système de fichiers réparti mis en œuvre dans un immeuble disposant d'un réseau sans fil. Quel que soit le type de l'application et l'échelle à laquelle elle est déployée, l'objectif est de fournir un service aussi proche que possible de celui existant en environnement fixe. Pour ce faire, les systèmes spécialisés se doivent de prendre en compte les contraintes auxquelles l'informatique nomade est soumise. On distingue, à ce niveau, trois grandes catégories.

Connectivité intermittente. Bon nombre d'applications d'informatique mobile reposent sur des infrastructures de communication de type réseau cellulaire. Les utilisateurs évoluant dans un tel cadre peuvent être amenés à se déplacer hors de la zone de couverture de l'infrastructure, ce qui a pour effet de les isoler du reste du système. Il est intéressant de remarquer que ce problème de connectivité n'est pas limité aux

seules zones non couvertes. En effet, du fait de la présence d'obstacles, il est possible de se trouver dépourvu d'accès réseau au sein d'une zone théoriquement couverte par l'infrastructure. Les utilisateurs du GSM sont ainsi coutumiers de l'absence de connectivité dans les tunnels.

Bande passante réduite. Quel que soit l'état de la connectivité, les systèmes nomades doivent composer avec une bande passante qui, malgré des avancées régulières, reste nettement en deçà de ce qui existe en environnement filaire. Ainsi, alors que le GSM [1] plafonne à 9,6kb/s, le GPRS [10] [63] permettra d'atteindre, dans un avenir proche, un débit de 38kb/s. Quant aux réseaux locaux sans fil, la norme 802.11b [46] ne permet pas de dépasser les 11Mb/s. Outre un débit limité, il est également nécessaire de prendre en compte les variations importantes auxquelles peut être soumise la bande passante, les communications sans fil étant sensibles à l'environnement dans lequel elles sont effectuées.

Ressources énergétiques limitées. Les terminaux dont disposent les utilisateurs nomades ont une autonomie limitée puiqu'ils sont tributaires de batteries au niveau énergétique. Cette autonomie est, de plus, fonction de l'utilisation qui est faite des ressources locales telles que le processeur ou la carte de communication. Pour plus d'information sur le sujet, on pourra consulter un document de synthèse consacré aux techniques de gestion et de réduction de la consommation énergétique pour les systèmes embarqués [64].

Afin de palier à ces problèmes, un certain nombre de mécanismes ont été mis en œuvre dans le cadre de l'informatique mobile. Nous les présentons dans la prochaine partie.

4.2 Mécanismes adaptés à la nomadicité

Il apparaît, à la lumière des contraintes que nous venons de présenter, que le déploiement d'applications adaptées aux terminaux mobiles ne va pas sans difficultés. De nombreux travaux ont donc été consacrés à la conception de mécanismes permettant de faciliter la mise en œuvre d'applications en environnement mobile. Ces mécanismes peuvent être regroupés en deux grandes classes : d'un côté, ceux qui cherchent à masquer la mobilité et, de l'autre, ceux qui permettent aux applications de s'y adapter. Ces deux approches à la mobilité ayant déjà fait l'objet d'un

10. General Packet Radio Service

document de synthèse [65], nous nous contenterons ici de rappeler les principes des techniques étudiées.

4.2.1 Adaptation à la mobilité

Les communications sans fil, nous venons de le souligner, sont exposées à d'importantes, et parfois brutales, variations de bande passante. Hors, il est possible que de simples variations soient à l'origine d'une perturbation, voire d'une interruption de service. Considérons l'exemple d'un utilisateur désirant visualiser une vidéo, stockée sur un serveur fixe, à partir d'un terminal mobile. Dès que la bande passante disponible n'est plus suffisante pour transmettre au terminal mobile les prochaines images, le service doit être interrompu.

Une des façons de surmonter ce type de difficulté est, lorsque c'est possible, d'adapter la qualité de l'information transférée à la bande passante disponible. La qualité d'une information peut être définie comme son degré de *fidélité* à une *copie de référence* (non dégradée). Cette technique ne peut cependant être appliquée qu'à des données structurées, c'est-à-dire des données pouvant être dégradées sans altération de sens. Les images et les vidéos peuvent être considérées comme telles.

Odyssey [66, 67] est le principal système d'adaptation pour la mobilité. Bien que se voulant générique, il est principalement axé sur les variations de bande passante. Il se présente comme une surcouche du système d'exploitation à laquelle est confiée la gestion de l'adaptation pour les différentes applications d'un terminal mobile. Dans cet objectif, *Odyssey* associe un composant, appelé *surveillant*, à chaque type de donnée dégradable. Les applications confient au surveillant adapté leur fenêtre de tolérance aux variations de la bande passante. Lorsqu'un surveillant constate que la valeur de cette dernière sort de l'une des fenêtres gérées, il avertit l'application appropriée. La politique d'adaptation est à la charge de l'application. C'est elle qui décide, en fonction des retours fournis par les surveillants, de la fidélité des données à manipuler. Elle doit également adapter ses fenêtres de tolérance aux variations de la bande passante.

4.2.2 Masquage de la mobilité

Une connectivité intermittente peut isoler temporairement un utilisateur du reste du système. Dans une telle situation, le service proposé ne peut généralement plus être assuré dans la mesure où il est impossible d'accéder à de nouvelles informations distantes. Pour palier aux "trous" de connectivité, les systèmes nomades recourent généralement à des techniques de *préchargement*. L'idée est de profiter de la présence

d'un accès réseau pour anticiper le chargement sur un terminal mobile des documents qui seront ultérieurement nécessaires à son utilisateur. De cette façon, le système peut rester opérationnel malgré l'absence de connectivité (puisque les données sont stockées localement), et les déconnexions deviennent transparentes aux applications. Par extension, un terminal mobile peut, en se contentant d'utiliser les copies locales, faire l'économie de communications même lorsque le réseau est accessible.

Toutefois, la mise en œuvre du préchargement impose de traiter les deux problèmes que sont la détermination de l'ensemble des données à précharger et, en cas d'accès concurrents, la gestion de leur consistence. La sélection des données à précharger s'effectue généralement à partir d'une combinaison d'informations explicitement fournies par l'utilisateur et d'informations acquises automatiquement (telles qu'un historique des fichiers accédés).

En ce qui concerne le problème de consistence des données, deux politiques sont envisageables. La première d'entre elles, la politique de *réplication pessimiste*, n'autorise les accès concurrents qu'en lecture seule. Les accès en écriture sont dans ce cas des accès exclusifs. Cette politique a pour avantage d'assurer à tout moment la consistence des données. En contrepartie, la disponibilité de ces dernières peut se voir réduite de façon considérable. Ainsi, une donnée verrouillée par un utilisateur s'étant par la suite déconnecté devra attendre la reconnexion de ce dernier pour être déverrouillée. Ce constat est à l'origine du développement de systèmes basés sur des politiques de *réplication optimiste*. Ce type de politique autorise les accès concurrents et permet donc à plusieurs utilisateurs de modifier simultanément leur réplicat d'une même donnée. La consistence de l'information n'est ici plus assurée et l'utilisation de mécanismes de résolution de conflits devient nécessaire.

La réplication optimiste peut provoquer deux types de conflits : les conflits *écriture/écriture* et *lecture/écriture*. Suivant le type de système considéré, les conflits *lecture/écriture* pourront ou non être détectés. Il est nécessaire, pour qu'ils le soient, que les accès en lecture soient mémorisés. La situation pour les conflits *écriture/écriture* est plus simple. Ces conflits sont en effet détectés lorsque deux réplicats d'une même donnée ayant subi des modifications différentes doivent être fusionnés. Dans une telle situation, la résolution de conflit est chargée de construire une version de la donnée conciliant les modifications effectuées de part et d'autre. Pour mener cette tâche à bien, il est nécessaire, mais pas suffisant, que les données manipulées soient structurées (comme le sont les informations stockées dans une base de données). Lorsqu'une réconciliation ne peut être effectuée automatiquement, le système est contraint de faire appel à l'utilisateur, ce qui ne manque pas de constituer une faiblesse pour un système se voulant transparent.

De nombreux systèmes ont été conçus avec le souci de rendre la mobilité transparente aux applications. L'idée a notament été appliquée aux systèmes de fichiers et bases de données répartis. Alors que ces dernières peuvent exploiter une struturation fine de l'information, les systèmes de fichiers doivent composer, dans ce domaine, avec la seule organisation arborescente des répertoires.

Systèmes de fichiers répartis. Le système de fichiers *Coda* [68] est le prototype le plus représentatif dans le domaine. Il s'agit d'un système de fichiers réparti reposant sur un serveur unique centralisant les réplicats maîtres de tous les fichiers. Initialement conçus pour des déconnexions prolongées, les clients *Coda* opèrent dans l'un des trois modes suivants : *hoarding*, *emulation* et *reintegration*. Connecté au serveur, un client *Coda* en mode *hoarding* s'occupe du préchargement de données. Pour ce faire, il exploite la liste des derniers documents accédés ainsi que la *hoard database* de l'utilisateur. Cette dernière, renseignée par l'utilisateur lui-même, regroupe l'ensemble des informations qui lui sont utiles. Après une déconnexion, le client passe en mode *emulation* dans lequel il utilise les données préalablement préchargées. Toute modification de donnée préchargée est enregistrée au sein du *replay log*. A la reconnexion, le passage en mode *reintegration* permet au client de concilier, à l'aide du *replay log*, les données présentes dans son cache avec les réplicats maîtres. Les conflits de répertoires peuvent faire l'objet d'une résolution automatique [69] alors que, dans le cas des fichiers, l'intervention d'un utilisateur est nécessaire.
Des aménagements visant à gérer le modèle de connectivité intermittente, ont été apportés au système [70]. Leur objectif est de profiter des périodes de connectivité pour poursuivre les processus de *reintegration* et de *hoarding*.
L'idée de réplication optimiste a été exploitée par d'autres prototypes de systèmes de fichiers dont *Rumor* [71], PFS[11] [72] et *Ficus* [73]. Elle a également été intégrée à des systèmes de fichiers classiques tels que AFS[12] [74] et NFS[13] (avec le prototype NFS/M [75]).

Bases de données réparties. La mobilité transparente peut également être appliquée avec profit au domaine des bases de données pour utilisateurs nomades. Le système *Bayou* [76], développé par Xerox, propose ce type de service. Il implémente, à l'image de *Coda*, une stratégie de réplication non contrainte mais, à la différence de celui-ci, il autorise la propagation de modifications entre réplicats secondaires.

11. Prayer File System
12. Andrew File System
13. Networked File System

La gestion des conflits est spécifique à chaque application [77], une procédure de conciliation (appelée *mergeproc*) étant associée à chaque opération d'écriture.

4.3 Discussion

Ubiquité numérique et informatique nomade cherchent, toutes deux, à faciliter aux utilisateurs mobiles l'accès aux services proposés par les systèmes numériques. Elles adoptent cependant des approches tout à fait différentes. En effet, l'informatique nomade s'applique à transposer aux terminaux mobiles les applications et interactions caractéristiques des stations de travail en dépit de conditions d'exécution rendues variables par la mobilité. De son côté, l'ubiquité numérique prend le parti d'une intégration transparente des environnements numériques au monde physique afin de rendre implicites les interactions que peuvent entretenir utilisateurs et systèmes d'information. Les systèmes conçus dans ce cadre se doivent d'adapter les services proposés à la situation réelle de leur utilisateur.
Cette différence dans l'approche se répercute au niveau des paramètres considérés par chacune des deux démarches. Ainsi, les systèmes nomades se focalisent sur les paramètres matériels (bande passante disponible, périphériques accessibles, ressources énergétiques restantes ...) constituant ce que Chen et Kotz [4] nomment le contexte numérique dont ils cherchent à isoler les applications des variations. Les applications d'ubiquité, quant à elles, ne peuvent se contenter d'une vision de leur environnement limitée à ce contexte numérique; une véritable représentation de l'environnement physique des utilisateurs devient nécessaire. La capture des contextes utilisateur (localisation, préférences) et physique (conditions météorologiques, niveau de bruit ...) permet d'obtenir, partiellement, une telle représentation.

Dans les deux cas, l'acquisition du contexte se veut aussi transparente que possible. Les problèmes de configuration automatique, par exemple dans le cadre des accès réseaux, ont déjà fait l'objet de nombreux travaux. Ainsi, le système Mobile IP [78], proposé par Perkins, permet à un utilisateur nomade d'accéder à Internet en conservant son adresse IP d'origine. Dans le domaine du contexte physique, ainsi que celui de la localisation, un certain nombre de mécanismes ont été proposés en ubiquité numérique qui visent à automatiser l'acquisition du contexte. Certains aspects, notamment au sein du contexte utilisateur, restent cependant bien souvent à la charge des utilisateurs. C'est notamment le cas de leurs préférences qui doivent, dans bon nombre de systèmes (tels que *C-Map* [17], *Proem* [30]), être directement renseignées par l'intéressé. Les prototypes *SIDE Surfer* [31] et *Hippie* [18] intègrent toutefois des mécanismes d'apprentissage visant à automatiser la capture de ces préférences.

D'un strict point de vue technologique, les deux approches reposent bien souvent sur le même type d'infrastructure matérielle, tant au niveau des terminaux considérés que des interfaces de communication. Par exemple, le système GSM, sur lequel s'appuie le WAP, est également utilisé par les terminaux mobiles du prototype *FieldNote* [27] afin de communiquer avec leur serveur de données. Une première distinction peut toutefois être effectuée au regard de l'exploitation qui est faite de ces technologies. Considérons, par exemple, les réseaux cellulaires de type 802.11 utilisés pour équiper certains immeubles. Ce type de matériel est également exploité en ubiquité numérique, entre autres par le prototype *GUIDE* [14], afin de couvrir/définir des zones d'intérêt dans l'espace physique. Dans le cadre d'un réseau local, les ruptures de communication sont traitées comme des erreurs, c'est-à-dire des événements anormaux. *GUIDE*, lui, tient ces ruptures pour normales, considérant simplement qu'un utilisateur quittant une cellule 802.11 n'est plus intéressé par les informations qui lui sont associées.

C'est d'ailleurs du couplage étroit, introduit en ubiquité, entre information et contexte que découlent les principales différences architecturales constatées entre les deux approches. Hormis les systèmes d'accès à Internet, la plupart des prototypes d'informatique nomade sont articulés autour d'un unique serveur de données. Les informations qu'ils manipulent, étant décorrélées des entités physiques qu'elles désignent, peuvent être consultées dans n'importe quel contexte. En ubiquité, les informations sont fortement liées aux objets (ou lieux) qu'elles représentent et n'ont de valeur qu'en présence de ce contexte. C'est donc tout naturellement que les prototypes d'ubiquité cherchent à répartir physiquement leurs données en fonction du contexte pour lequel elles sont pertinentes. La localisation s'est très vite révélée être un facteur de répartition à la fois efficace et facile à mettre en œuvre : stocker localement, par l'intermédiaire d'un serveur dédié, l'ensemble des informations associées à un lieu, ainsi qu'aux objets qui y sont présents, permet de cantonner les requêtes contextuelles à des accès locaux. Constatant que de nombreux objets, du fait de leur mobilité, ne peuvent être rattachés à un lieu, certains systèmes proposent une répartition contextuelle de l'information plus fine dans laquelle chaque entité stocke elle-même sa représentation virtuelle (cf. *Cluefinder* [52]). Cette répartition contextuelle de l'information implique des applications d'ubiquité qu'elles disposent, pour accéder aux données, de mécanismes adaptés, ces derniers pouvant être directement intégrés au schéma de nommage (*Mobisaic* [12]) ou d'indexation (*Forget-me-Not* [23]).

Dans le domaine de l'informatique mobile, l'ubiquité numérique présente une philosophie tout à fait différente de celle pronée par l'approche traditionnelle. A un service global faisant appel à des informations distantes, elle oppose une kyrielle de

services, répartis contextuellement et exploitant, lorsque c'est possible, des données locales. Les deux approches ne sont toutefois pas concurrentes, loin s'en faut. De même que la plus-value des services contextuels est indéniable, il est évident que ces derniers ne permettent pas aux utilisateurs de disposer d'un panel d'information aussi important que celui qui est, par exemple, disponible sur le Web. Il est donc fort probable que les deux approches soient, dans un avenir proche, systématiquement couplées afin de présenter une offre exhaustive, en matière de services, aux utilisateurs nomades.

5 Conclusion

Nous nous sommes intéressés, dans le présent document, au problème de l'accès à l'information au sein de l'ubiquité numérique. Après avoir introduit les grands principes et les objectifs du domaine, nous en avons présenté les principales classes d'applications. Nous avons alors procédé à une classification de ces applications s'appuyant sur les architectures, ainsi que les techniques employées pour accéder à l'information. Les différents prototypes ont ainsi été répartis en quatre grandes classes : les systèmes recourant à une réplication complète des données sur les terminaux mobiles, ceux utilisant un système d'information centralisé, ceux qui distribuent l'information en fonction de la localisation et, enfin, les systèmes qui distribuent l'information sur toutes les entités pouvant le supporter. Cette classification a permis de mettre en lumière une évolution des architectures, d'une tendance "centralisée" vers des schémas de plus en plus répartis au sein desquels chaque entité physique appartenant au système a la charge de sa propre représentation numérique. La tendance constatée s'accompagne, au niveau des communications, d'un recours aux technologies de communication de proximité en lieu et place des systèmes de communications globaux. Les systèmes issus de ces évolutions semblent donc plus à même de répondre aux contraintes de passage à l'échelle qui ne manqueront pas d'être posées aux applications d'ubiquité de demain.

Cette étude nous a également montré que, malgré d'importantes avancées technologiques, les travaux actuels sont encore loin de la vision de Weiser. En effet, les études effectuées dans le domaine s'appuient, pour la plupart, sur l'utilisation de PDAs. Ces équipements, malgré leur taille réduite, ne répondent pas à la contrainte d'intégration transparente souhaitée par Weiser, leur utilisation n'étant pas assez *naturelle* pour ne pas accaparer l'attention de leurs utilisateurs. Afin de surmonter cette difficulté, plusieurs solutions peuvent aujourd'hui être envisagées, notamment les interfaces utilisant des techniques de réalité augmentée [79] et de réalité virtuelle

[80]. L'intégration de cette nouvelle génération d'interfaces aux architectures que nous venons de présenter permettra probablement de franchir un nouveau cap vers la concrétisation de la vision de Weiser.

Références

[1] M. Rahnema. Overview Of The GSM System and Protocol Architecture. *IEEE Communications Magazine*, 31(4):92–100, April 1993.

[2] WAP Forum. WAP 2.0 Technical White Paper. Available at http://www.wapforum.org/what/whitepapers.html.

[3] M. Korkea-aho. Context-Aware Applications Survey. Available at http://www.hut.fi/ mkorkeaa/doc/context-aware.html, 2000.

[4] G. Chen and D. Kotz. A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, Department of Computer Science, Dartmouth College, 2000.

[5] P. Couderc, A-M. Kermarrec, and M. Banâtre. Approches adaptatives en mobilité : une synthèse. Technical Report 1276, IRISA, November 1999.

[6] M. Weiser. Some Computer Issues in Ubiquitous Computing. *Communications of the ACM*, 36(7):74–84, July 1993.

[7] P. H. Dana. Global positioning system overview. Available at http://www.colorado.edu/geography/gcraft/notes/gps/gps_f.html.

[8] R. Want, A. Hopper, V. Falcão, and J. Gibbons. The Active Badge Location System. *ACM Transactions on Information Systems*, 10:91–102, January 1992.

[9] R. Want, B. N. Schilit, N. I. Adams, R. Gold, K. Petersen, D. Goldberg, J. R. Ellis, and M. Weiser. An Overview of the ParcTab Ubiquitous Computing Experiment. *IEEE Personal Communications*, 2(6):28–33, December 1995.

[10] T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, B. Serra, and M. Spasojevic. People, Places, Things: Web Presence for the Real World. In *Proceedings of the Third Annual Workshop on Wireless and Mobile Computer Systems and Applications (WMCSA'2000)*, December 2000.

[11] J. Youll, J. Morris, R. Krikorian, and P. Maes. Impulse: Location-based Agent Assistance. In *Proceedings of the Fourth International Conference on Autonomous Agent*, June 2000.

[12] G. M. Voelker and B. N. Bershad. Mobisaic: An Information System for a Mobile Wireless Computing Environment. Technical Report TR-95-04-01, Department of Computer Science and Engineering, University of Washington, 1994.

[13] N. Marmasse and C. Schmandt. Location-aware information delivery with *comMotion*. In *Proceedings of the Second Symposium on Handheld and Ubiquitous Computing (HUC'2000)*, pages 157–171, September 2000.

[14] N. Davies, K. Mitchell, K. Cheverst, and G. Blair. Developing a context sensitive tourist guide. In *Proceedings of the First Workshop on Human Computer Interactions for Mobile Devices*, pages 64–68, May 1998.

[15] G. D. Abowd, C. G. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton. Cyberguide: a Mobile Context-Aware Tour Guide. *ACM Wireless Networks*, 3:421–433, October 1997.

[16] A. Smailagic and R. Martin. Metraunot: A Wearable Computer with Sensing and Global Communication Capabilities. In *Proceedings of the First International Symposium on Wearable Computers (ISWC'97)*, pages 116–122, October 1997.

[17] S. Fels, Y. Sumi, T. Etani, N. Simonet, K. Kobayashi, and K. Mase. Progress of C-Map: a Context-Aware Mobile Assistant. In *Proceedings of the AAAI Spring Symposium on Intelligent Environments*, pages 60–67, March 1998.

[18] R. Oppermann, M. Specht, and I. Jaceniak. Hippie: A Nomadic Information System. In *Proceedings of the First International Symposium on Handheld and Ubiquitous Computing (HUC'99)*, pages 330–333, September 1999.

[19] W. Chan. DealFinder: A Collaborative, Location-Aware Mobile Shopping Application.

[20] K. Nagao and J. Rekimoto. Agent Augmented Reality: A Software Agent Meets the Real World. In *Proceedings of the Second International Conference on Multiagent Systems (ICMAS'96)*, December 1996.

[21] A. Asthana, M. Cravatts, and P. Krzyzanowski. An Indoor Wireless System for Peronalized Shopping Assistance. In *Proceedings of the IEEE Worshop on Wireless Mobile Computing Systems and Applications (WMCSA'94)*, pages 69–74, December 1994.

[22] G. W. Fitzmaurice. Situated Information Spaces and Spatially Aware Palmtop Computers. *Communications of the ACM*, 36(7):39–49, July 1993.

[23] M. Lamming and M. Flynn. Forget-me-not: Intimate Computing in Support of Human Memory. In *Proceedings of FRIEND21: International Symposium on Next Generation Human Interface*, pages 125–128, February 1994.

[24] J. Healey and R. W. Picard. StartleCam: A Cybernetic Wearable Camera. In *Proceedings of the Second International Symposium on Wearable Computers*, pages 42–49, October 1998.

[25] A. K. Dey and G. D. Abowd. CybreMinder: A Context-Aware System for Supporting Reminders. In *Proceedings of the Second Symposium on Handheld and Ubiquitous Computing (HUC'2000)*, pages 172–186, September 2000.

[26] P. J. Brown. The stick-e document: a framework for creating context-aware applications. *Electronic Publishing*, 9(1):1–14, September 1996.

[27] N. Ryan and J. Pascoe. FieldNote: a Handheld Information System for the Field. In *Proceedings of the First International Workshop on TeleGeoProcessing (TeleGeo'99)*, May 1999.

[28] J. Haartsen, M. Naghshineh, J. Inouye, O. Joeressen, and W. Allen. Bluetooth: Vision, Goals, and Architecture. *Mobile Computing and Communications Review*, 2(4):38–45, October 1998.

[29] Per Dahlberg, Fredrik Ljungberg, and Johan Sanneblad. Proxy Lady: Mobile Support for Opportunistic Interaction. *Scandinavian Journal of Information Systems*, 15, 2000.

[30] G. Kortuem, Z. Segall, and T. G. Cowan Thompson. Close Encounters: Supporting Mobile Collaboration through Interchange of User Profiles. In *Proceedings of the First International Symposium on Handheld and Ubiquitous Computing (HUC'99)*, pages 171–185, September 1999.

[31] D. Touzet, J-M. Menaud, M. Banâtre, P. Couderc, and F. Weis. SIDE Surfer: Enriching Casual Meetings with Spontaneous Information Gathering. *ACM SigArch Computer Architecture Newsletter*, 29(5), December 2001.

[32] G. Kortuem, J. Schneider, J. Suruda, S. Fickas, and Z. Segall. When Cyborgs Meet: Building Communities of Cooperating Wearable Agents. In *Proceedings of the Third International Symposium of Wearable Computing (ISWC'99)*, pages 124–132, October 1999.

[33] S. Björk, J. Falk, R. Hansson, and P. Ljungstrand. Pirates! Using the Physical World as a Game Board. In *Proceedings of Interact'2001*, pages 423–430, July 2001.

[34] T. Kindberg and K. Zhang. Context authentication using constrained channels. Technical Report HPL-2001-84, Internet and Mobile Systems Laboratory, Hewlett-Packard Laboratories, 2001.

[35] S. Long, R. Kooper, G. D. Abowd, and C. G. Atkeson. Rapid Prototyping of Mobile Context-Aware Applications: The Cyberguide Case Study. In *Procee-*

dings of the Second ACM International Conference on Mobile Computing and Networking (MobiCom'98), pages 97–107, November 1996.

[36] P. J. Brown, J. D. Bovey, and X. Chen. Context-aware Applications: from the Laboratory to the Marketplace. *IEEE Personal Communications*, 4(5):58–64, October 1997.

[37] P. J. Brown. Triggering information by context. *IEEE Personal Communications*, 2(1):1–9, September 1998.

[38] B. N. Schilit, N. Adams, R. Gold, M. M. Tso, and R. Want. The ParcTab Mobile Computing System. In *Proceedings of the Fourth Workshop on Workstation Operating Systems*, pages 34–39, October 1993.

[39] B. Schilit, N. Adams, and R. Want. Context-Aware Computing Applications. In *Proceedings of the Workshop on Mobile Computing Systems and Applications*, pages 85–90, December 1994.

[40] A. K. Dey, G. D. Abowd, and D. Sabler. A Context-Based Infrastructure for Smart Environments. In *Proceedings of the First International Workshop on Managing Interactions in Smart Environments (MANSE'99)*, pages 114–128, December 1999.

[41] J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol (HTTP/1.1). Available at http://www.w3.org/Protocols/rfc2616/rfc2616.html, June 1999. Internet RFC 2616.

[42] K. Cheverst, K. Mitchell, and N. Davies. Design of an Object Model for a Context Sensitive Tourist GUIDE. *Computer and Graphics*, 23(6):883–891, December 1999.

[43] K. Cheverst, N. Davies, K. Mitchell, A. Friday, and C. Efstratiou. Developing a Context-aware Electronic Tourist Guide: Some Issues and Experiences. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI'2000)*, pages 17–24, April 2000.

[44] K. Cheverst, N. Davies, K. Mitchell, and A. Friday. The Role of Connectivity in Supporting Context-Sensitive Applications. In *Proceedings of the First Symposium on Handheld and Ubiquitous Computing (HUC'99)*, pages 193–207, September 1999.

[45] K. Cheverst, N. Davies, K. Mitchell, and A. Friday. Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project. In *Proceedings of MobiCom'2000*, pages 20–31, August 2000.

[46] B. P. Crow, I. Widjaja, J. G. Kim, and P. T. Sakai. IEEE 802.11 Wireless Local Area Network. *IEEE Communications Magazine*, 35(9):116–126, September 1997.

[47] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. Broadcast Disks: Data Management for Asymmetric Communication Environments. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 199–210, May 1995.

[48] D. Caswell and P. Debaty. Creating Web Representations for Places. In *Proceedings of the Second International Symposium on Handheld and Ubiquitous Computing (HUC'2000)*, pages 114–125, September 2000.

[49] T. Kindberg and J. Barton. A Web-Based Nomadic Computing System. *Computer Networks*, 35(4):443–456, March 2001.

[50] S. Pradhan, C. Brignone, J-H. Cui, A. McReynolds, and M. T. Smith. Websigns: Hyperlinking Physical Locations to the Web. *IEEE Computer Magazine*, 34(8):42–48, August 2001.

[51] J. Youll and R. Krikorian. Wherehoo Server: An interactive location service for software agents and intelligent systems. In *Workshop on Infrastructure for Smart Devices*, September 2000.

[52] J. Schneider and G. Kortuem. How to Host a Persuasive Game: Supporting Face-to-Face Interactions in Live-Action Roleplaying. In *Proceedings of the Workshop on Designing Ubiquitous Computing Games*, September 2001.

[53] M. Banâtre and F. Weis. SIS: a new paradigm for mobile computer systems. In *Proceedings of the Information Society Technologies Conference (IST'99)*, November 1999.

[54] M. Banâtre and F. Weis. Système d'Information Spontané (SIS) : Problématique et Premiers Eléments de Solutions. Technical Report 1222, IRISA, December 1998.

[55] A. Troël, M. Banâtre, P. Couderc, and F. Weis. Predictive Scheme for Proximate Interactions. In *Proceedings of the Twenty First IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'2001)*, pages 235–239, April 2001.

[56] D. Touzet, J-M. Menaud, M. Banâtre, P. Couderc, and F. Weis. SIDE Surfer: a Spontaneous Information Discovery and Exchange System. In *Proceedings of the Second International Workshop on Ubiquitous Computing and Communications (WUCC'2001)*, September 2001.

[57] G. Kortuem and J. Schneider. An Application Platform for Mobile Ad-hoc Networks. In *Proceedings of the Workshop on Application Models and Programming Tools for Ubiquitous Computing*, September 2001.

[58] D. L. de Ipiña. Video-Based Sensing for Wide Deployment of Sentient Spaces. In *Proceedings of the Second International Workshop on Ubiquitous Computing and Communications (WUCC'2001)*, September 2001.

[59] M. D. Addlesee, A. H. Jones, F. Livesey, and F. S. Samaria. The ORL Active Floor. *IEEE Personal Communications*, 4(5):35–41, October 1997.

[60] R. J. Orr and G. D. Abowd. The Smart Floor: A Mechanism for Natural User Identification and Tracking. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI'2000)*, April 2000.

[61] Ulf Leonhardt. *Supporting Location-Awareness in Open Distributed Systems.* PhD thesis, Department of Computing - Imperial College of Science, Technology and Medicine - University of London, May 1998.

[62] M. Rizzo, P. F. Linington, and I. A. Utting. Integration of Location Services in the Open Distributed Office. Technical Report 14-94, Computing Laboratory, University of Kent, August 1994.

[63] R. Kalden, I. Meirick, and M. Meyer. Wireless Internet Access Based on GPRS. *IEEE Personal Communications*, 7(2):8–18, April 2000.

[64] F. Parain, M. Banâtre, G. Cabillic, T. Higuera, V. Issarny, and J-P. Lesot. Techniques de réduction de la consommation dans les systèmes embarqués temps-réel. Technical Report 1332, IRISA, May 2000.

[65] J. Jing, A. Helal, and A. Elmagarmid. Client-Server Computing in Mobile Environments. *ACM Computing Surveys*, 31(2):117–157, June 1999.

[66] B. D. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn, and K. R. Walker. Agile Application-Aware Adaptation for Mobility. In *Proceedings of the Sixteenth ACM Symposium on Operating Systems Principles*, pages 276–287, October 1997.

[67] B. D. Noble and M. Satyanarayanan. Experience with Adaptative Mobile Applications in Odyssey. *Mobile Networks and Applications*, 4(4):245–254, December 1999.

[68] J. J. Kistler and M. Satyanarayanan. Disconnected Operation in the Coda File System. *ACM Transactions on Computer Systems*, 10(1):3–25, February 1992.

[69] P. Kumar and M. Satyanarayanan. Log-Based Directory Resolution in the Coda File System. In *Proceedings of the Second International Conference on Parallel and Distributed Information Systems*, pages 202–213, January 1993.

[70] L. B. Mummert, M. R. Ebling, and M. Satyanarayanan. Exploiting Weak Connectivity for Mobile File Access. In *Proceedings of the Fifteenth ACM Symposium on Operating Systems Principles*, pages 143–155, December 1995.

[71] R. Guy, P. Reilher, D. Ratner, M. Gunter, W. Ma, and G. Popek. Rumor: Mobile Data Access Through Optimistic Peer-to-Peer Replication. In *Proceedings of the ER'98 Workshop on Mobile Data Access*, pages 254–265, November 1998.

[72] D. Dwyer and V. Bharghavan. A Mobility-Aware File System for Partially Connected Operation. *ACM Operating Systems Review*, 31(1):24–30, January 1997.

[73] P. Reither, J. Heidemann, D. Ratner, G. Skinner, and G. Popek. Resolving File Conflicts in the Ficus File System. In *Proceedings of the Summer USENIX Conference*, pages 183–195, June 1994.

[74] L. B. Huston and P. Honeyman. Disconnected Operation for AFS. In *Proceedings of the USENIX Mobile and Location-Independent Computing Symposium*, pages 1–10, August 1993.

[75] J. C. S. Lui, O. K. Y. So, and T. S. Tam. NFS/M: an Open Platform Mobile File System. In *Proceedings of the Eighteen International Conference on Distributed Computing Systems (ICDCS'98)*, pages 488–495, May 1998.

[76] A. Demers, K. Petersen, M. Spreitzer, D. Terry, M. Theimer, and B. Welch. The Bayou Architecture: Support for Data Sharing among Mobile Users. In *Proceedings of the Workshop on Mobile Computing Systems and Applications*, pages 2–7, December 1994.

[77] D. Terry, M. Theimer, K. Petersen, A. Demers, M. Spreitzer, and C. Hauser. Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System. *Operating Systems Review*, 29(5):172–183, December 1995.

[78] C. E. Perkins. Mobile IP. *IEEE Communications Magazine*, 35(5):84–99, May 1997.

[79] R. T. Azuma. A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, August 1997.

[80] A. Steed. A Survey of Virtual Reality Literature. Technical Report QMW-DCS-1993-623, Department of Computer Science, Queen Mary and Westfield College, March 1993.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399