# Approche applicative de la continuité des services en mobilité dans un milieu hétérogène

Vincent Verdot

École Doctorale EDITE de Paris

---

Thèse présentée pour l'obtention du diplôme de
Docteur de l'Institut National des Télécommunications

*Doctorat délivré conjointement par*
*L'Institut National des Télécommunications et l'Université Pierre et Marie Curie – Paris 6*

Spécialité
Informatique, Télécommunications et Électronique

Par
**Vincent Verdot**

# Approche applicative de la continuité des services en mobilité dans un milieu hétérogène.

Thèse soutenue le 9 décembre 2009 devant le jury composé de

| | | |
|---|---|---|
| **Dr Valérie Issarny** | INRIA Paris-Rocquencourt | *Rapporteur* |
| **Pr André-Luc Beylot** | ENSEEIHT, Toulouse | *Rapporteur* |
| **Pr Guy Pujolle** | LIP6, Paris | *Examinateur* |
| **Pr Harry Perros** | NC State University, USA | *Examinateur* |
| **M. Arnaud Gonguet** | Alcatel-Lucent Bell Labs France | *Examinateur* |
| **Dr Noël Crespi** | Télécom SudParis, Évry | *Directeur de la recherche* |
| **M. Yann Gasté** | Alcatel-Lucent Bell Labs France | *Directeur de la recherche* |
| **Pr Djamal Zeghlache** | Télécom SudParis, Évry | *Directeur de la thèse* |

**Thèse n°2009TELE0019**

*À Eugénie,*
*qui a su me porter et me supporter.*

# Remerciements

Bien qu'une thèse soit avant tout une entreprise personnelle, son succès serait impensable sans le concours de nombreuses personnes pour lesquelles j'éprouve une sincère gratitude.

Tout d'abord merci au Professeur Djamal Zeghlache de m'avoir accueilli au sein de son département *Réseaux et Services Multimédia Mobiles* à Télécom SudParis et de m'avoir fait confiance en acceptant de diriger cette thèse.

Je dois également beaucoup à Dr Noël Crespi, directeur d'études à Télécom SudParis et à M. Yann Gasté, responsable de recherche à Alcatel-Lucent Bell Labs, qui ont encadré mes travaux. Je suis particulièrement reconnaissant envers eux pour leurs efforts d'accompagnement tout au long de ce doctorat.

Merci à M. Arnaud Gonguet, chercheur aux Bell Labs, pour ses conseils avisés et sa vision toujours singulière des problématiques de recherche.

Merci à Dr Bruno Aidan, directeur du domaine *Applications* et à M. Paul Labrogère, responsable du département *HybridCOM*, qui m'ont permis de travailler dans des conditions optimales au sein des Bell Labs.

Un grand merci aux membres du jury dont Dr Valérie Issarny (INRIA Paris-Rocquencourt), Pr André-Luc Beylot (ENSEEIHT Toulouse), Pr Guy Pujolle (LIP6 Paris) et Pr Harry Perros (NC State University USA), qui ont accepté de venir apprécier mes travaux.

Merci à toute ma famille pour son affection et son soutien indéfectible, notamment ma femme Eugénie à qui cette « œuvre » est dédiée.

Et enfin merci à tous mes collègues de Télécom SudParis et des Bell Labs qui m'ont précieusement aidé de par leur amitié...

*Vincent Verdot.*

# Abstract

**Title : Applicative continuity approach of mobile services in heterogeneous environments.**

People now live within an electronic sphere of heterogenous devices providing similar services. From now on they can freely choose the best interface corresponding to their needs, switching from a device to another while enjoying a same service. This freedom leverage new mobility issues that directly impact the user experience.

The state of the art introduces two services types with specific properties : applicative services commonly known as local or remote applications and telecommunication services based on standardised infrastructures. The study of the mobility constraints and the existing approaches shows that current solutions in this domain are incomplete and cannot assure an end-to-end continuity : selection, transfer, adaptation, etc.

We implemented continuity mechanisms in the IMS and Web environments for telecommunication and multimedia services in order to emphasize the lacks of the existing solutions and identify the basic features of a unified mobility management model. These works led to a more abstract vision of the service concept, required to define a generic continuity model adapted to the user's heterogeneous environment. We introduced innovative concepts and original mobility mechanisms, which we implemented in a *distributed Service Manager*.

Finally, we evaluated this model with qualitative and quantitative tests on a prototype according to a transfer scenario of text-edition service in a heterogeneous environment. Results are very satisfying, they prove the feasibility of our solution and its adequacy with the temporal and contextual constraints.

**Keywords :** service, session, application, mobility, continuity, transfer.

# Résumé

L'Homme est aujourd'hui au centre d'une sphère électronique composée de terminaux hétérogènes capables d'offrir des services similaires. Il choisit désormais l'interface adaptée à ses besoins, passant librement d'un terminal à un autre pour bénéficier d'un même service. Cette liberté induit de nouvelles problématiques de mobilité ayant un impact direct sur l'expérience utilisateur.

L'état de l'art fait apparaître deux catégories de services aux caractéristiques propres : des services applicatifs revêtant diverses formes (applications locales et distantes) et des services de télécommunications reposant sur des infrastructures standardisées. L'étude des contraintes de mobilité et des approches existantes révèle que les solutions actuelles dans ce domaine sont incomplètes et ne peuvent assurer une continuité de bout en bout : désignation, transfert, adaptation, etc.

Nous avons implémenté des mécanismes de continuité dans les environnements IMS et Web pour des services de télécommunications et multimédias afin de mettre en évidence les lacunes des solutions existantes, et identifier les fonctions élémentaires d'une gestion globale de la mobilité. Le résultat de ces travaux révèle qu'une approche plus abstraite du concept de service est nécessaire dans la définition d'un modèle générique de continuité adapté au milieu hétérogène de l'utilisateur. Nous avons introduit de nouveaux concepts et des mécanismes de mobilité originaux que nous avons ensuite implémentés dans un *distributed Service Manager*.

Enfin nous avons évalué ce modèle grâce à des tests qualitatifs et quantitatifs sur un prototype selon un scénario de transfert d'un service d'édition de texte dans un milieu hétérogène. Les résultats particulièrement satisfaisants prouvent la faisabilité de notre solution et son adéquation avec les contraintes de continuités temporelle et contextuelle.

**Mots clés :** service, session, application, mobilité, continuité, transfert.

# Table des matières

# Table des figures

# Liste des tableaux

# Liste des acronymes

**3GPP** 3rd Generation Partnership Project

**3PCC** 3rd Party Call Control

**API** Application Programming Interface

**AS** Application Server

**BSD** Berkeley Software Distribution

**CPU** Central Processing Unit

**CSCF** Call Session Control Function

**dSM** distributed Service Manager

**ETSI** European Telecommunications Standards Institute

**GPS** General Positionning system

**GSM** Global System for Mobile communications

**GUI** Graphical User Interface

**HSS** Home Subscriber Server

**HTML** Hypertext Markup Language

**HTTP** Hypertext Transport Protocol

**IAS** IMS Application Server

**I-CSCF** Interrogating - Call Session Control Function

**IETF** Internet Engineering Task Force

**iFC** initial Filter Criteria

**IIP** INT IMS Platform

**IP** Internet Protocol

**ITU** International Telecommunication Union

**ITU-T** International Telecommunication Union - Telecommunication standardization sector

**IM** Instant Messaging

**IMS** IP Multimedia Subsystem

**INSEE** Institut national de la statistique et des études économiques

**JDK** Java Development Kit

**JVM** Java Virtual Machine

**JXTA** Juxtapose

**LGPL** Lesser General Public License

**MMS** Microsoft Media Server

**MMS** Multimedia Messaging Service

**MS** Media Server

**NGN** Next Generation Network

**OMA** Open Mobile Alliance

**OSI** Open System Interconnection

**P2P** Peer-to-Peer

**PC** Personal Computer

**P-CSCF** Proxy - Call Session Control Function

**PDA** Personal Digital Assistant

**PoC** Push-to-talk over Cellular

**PSE** Personal Service Environment

**RFB** Remote Framebuffer

**RTC** Réseau Téléphonique Commuté

**RTMP** Real Time Messaging Protocol

**RTP** Real-time Transport Protocol

**RTCP** RTP Control Protocol

**RTSP** Real-Time Streaming Protocol

**SCAS** Session Continuity Application Server

**SCTP** Stream Control Transmission Protocol

**S-CSCF** Serving - Call Session Control Function

**SE** Système d'Exploitation

**SDP** Session Description Protocol

**SIP** Session Initiation Protocol

**TCP** Transmission Control Protocol

**TIC** Technologies de l'Information et de la Communication

**TISPAN** Telecommunications and Internet converged Services and Protocols for Advanced Networking

**UDP** User Datagram Protocol

**UI** User Interface

**UMTS** Universal Mobile Telecommunications System

**URI** Uniform Resource Identifier

**URL** Uniform Resource Locator

**USB** Universal Serial Bus

**VNC** Virtual Network Computing

**VoD** Video on Demand

**W3C** World Wide Web Consortium

**WMS** Windows Media Services

**WSA** Web Service Architecture

**WSD** Web Service Description

**WSDL** Web Service Description Language

**XML** eXtensible Markup Language

# Introduction générale

# 1 Contexte

Nous vivons dans un monde de services. Ceux-ci prennent une place chaque jour plus importante dans les régions industrialisées du globe, pour exemple le secteur des services représentait en 2008 près de 73% de l'emploi total en France avec une croissance d'un peu moins de 30% en 25 ans [1]. Ce succès est directement lié au modèle de vie des pays developpés basé sur la consommation ; ainsi le service en tant que bien immatériel constitue une ressource inépuisable de grande valeur.

S'il est un domaine de prédilection pour le service, où la dématérialisation est la règle, c'est bien celui des Technologies de l'Information et de la Communication. Reposant intrinsèquement sur la notion service, les TIC s'appuient sur un modèle économique exclusivement basé sur les biens immatériels. L'essor de l'informatique ces 25 dernières années a accompagné et accentué ce phénomène jusque dans notre vie de tous les jours. Et elle en fut changée.

Prenons l'exemple du iPhone d'Apple [2].

Il fut un temps, les appareils électroniques étaient conçus pour répondre à un besoin précis, leurs capacités étaient donc limitées à la réalisation d'un objectif connu a priori, chose qu'ils faisaient d'ailleurs plutôt bien. Ainsi un téléphone servait à téléphoner. Le progrès a rendu les appareils électroniques plus forts : plus de mémoire, plus de calculs, plus attirants, ... Un même appareil pouvant réaliser plusieurs fonctions, le modèle vertical qui associait une tâche à un terminal était dépassé.

Cette révolution présageait la disparition des terminaux pour une convergence de l'ensemble des fonctions existantes dans un unique « super-appareil », tellement plus pratique. Or aujourd'hui il n'en est rien ; nous sommes entourés d'ordinateurs, tous plus capables les uns que les autres : PDA, téléphone mobile, ordinateur portable, PC de bureau, navigateur GPS, home-cinéma, etc. La plupart de ces terminaux nous offrent les mêmes services, certes de manière différente mais les fonctions restent les mêmes. Mais alors quelle est la différence et surtout quel est l'intérêt ?

La réponse, nous venons de la donner : l'expérience utilisateur, la manière dont celui-ci perçoit le service. Un home-cinéma et un téléphone permettent tous deux de voir un film, si le premier offre une qualité d'image supérieure, le second

permet d'en profiter en déplacement. L'appareil électronique n'est plus qu'une interface pour l'utilisateur qui la préfèrera à une autre en fonction de ses qualités intrinsèques.

Revenons alors à notre iPhone. La stratégie de communication d'Apple illustre parfaitement ce nouvel ordre. Ce terminal n'est pas aujourd'hui vendu en tant que téléphone mais en tant qu'interface vers un univers de services, la téléphonie étant accessoirement l'une de ces fonctions. Au lieu de mettre en avant les qualités téléphoniques d'un téléphone, on vante les fonctions de l'interface : accéléromètre, GPS, affichage haute définition, caméra vidéo, appareil photo, écran tactile, ... ceci était un téléphone.

# 2 Problématique

L'utilisateur est au centre d'une sphère électronique composée de l'ensemble de ses appareils. Il est libre de choisir à chaque instant quelle interface répondra le mieux à ses besoins pour lui offrir un service (cf. Figure 1)

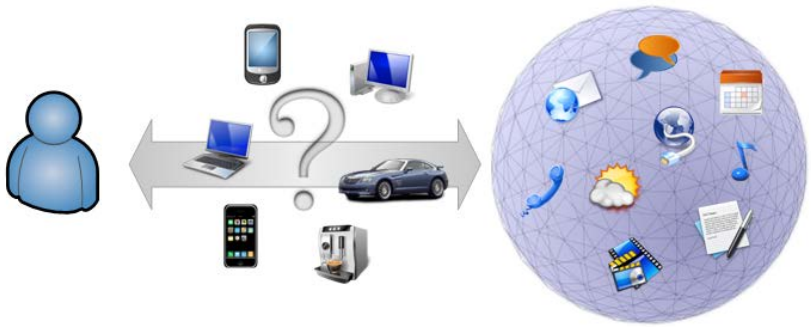


FIGURE 1 – La nouvelle liberté de l'utilisateur.

Mais les contraintes qui dictent le choix d'un terminal évoluent avec le temps, l'humeur ou la localisation de l'utilisateur. La meilleure interface, celle désirée par son propriétaire à l'instant $t$, peut alors changer et parfois même pendant la fourniture du service.

Si l'utilisateur est libre dans ce nouveau modèle orienté « services », il est toutefois limité par les systèmes existants qui ne considèrent pas sa sphère électronique comme une entité à part entière. En effet, chaque constructeur ayant aujourd'hui

un intérêt à cloisonner le marché, aucun mécanisme n'est proposé pour gérer la continuité des services entre les interfaces de l'utilisateur. Si j'écoute de la musique et que je souhaite changer d'interface, je dois nécessairement éteindre l'application en question, puis la relancer sur le nouveau terminal, rechercher l'album, sélectionner le titre que j'écoutais (si je m'en rappelle) et enfin le lire. Tout cela pour uniquement poursuivre le service à partir d'une autre interface, d'autant que j'aurai certainement à régler de nouveau le volume sonore, l'*equalizer*, etc.

Les services sont gérés localement à chaque interface, chacune manipulant une instance qui lui est dédiée. L'utilisateur ne peut disposer de manière naturelle et intuitive de ses services qui sont enfermés dans une implémentation. Il ne peut profiter pleinement de la richesse de sa sphère électronique, adapter dynamiquement ses services à l'hétérogénéité de l'environnement pour tirer avantage des propriétés de chaque interface.

Cette mobilité de service est absente dans un contexte au potentiel exceptionnel.

# 3 Objectifs

L'objectif de cette thèse est de définir un modèle générique de mobilité qui offre à l'utilisateur une expérience continue de ses services, adaptée à l'hétérogénéité de l'environnement.

Le but à atteindre étant particulièrement ambitieux, les travaux de recherche sont déclinés en sous-objectifs à atteindre successivement.

- La connaissance du *Service* est le premier d'entre eux, objet central de l'étude, il est indispensable de cerner le concept et d'en identifier les propriétés.
- Appréhender son environnement et donc celui de l'utilisateur est également nécessaire afin de dessiner les relations qui existent entre ses composantes tels que les terminaux ou les applications.
- Comprendre ensuite ce qu'est et implique la mobilité qui est la problématique de ces travaux, s'intéresser à ses déclinaisons et les différentes contraintes qui la composent, dont la continuité.

– Puis expérimenter différentes solutions de mobilité afin de dégager des mécanismes efficaces et identifier les lacunes des systèmes existants.
– Avec l'expérience ainsi acquise, proposer un modèle générique de mobilité de service qui fait abstraction des contraintes liées à l'hétérogénéité de son environnement tout en en exploitant la richesse.
– Et enfin, implémenter et évaluer ce modèle afin de prouver sa faisabilité et son adéquation avec la problématique posée.

# 4 Organisation du document

Ce mémoire de thèse est organisé en deux grandes parties qui suivront cette introduction générale.

La première partie intitulée « *État de l'art* » présentera les connaissances actuelles relatives aux concepts étudiés dans ces travaux de recherche (partie I). Divisée en deux chapitres, nous nous intéresserons d'abord à la notion de « service » (chapitre 1), ses différentes formes, les infrastructures existantes et ses relations avec les composantes de son environnement. Nous étudierons ensuite la problématique de mobilité (chapitre 2), les différentes contraintes qui la composent tel que la continuité et les solutions existantes classées par type d'approche.

La seconde partie intitulée « *Contributions* » présentera différents travaux convergeant vers la définition d'une solution de continuité de service répondant à la problématique identifiée (partie II). Divisée en six chapitres, nous nous intéresserons dans une premier temps à des mécanismes de mobilité spécifiques impliquant des architectures et des applications particulières : télécommunications IMS (chapitre 3) et multimédias (chapitre 4), Web (chapitre 5). Nous décrirons les implémentations et les propriétés ainsi identifiées dans la réalisation de cas concrets. Les trois derniers chapitres seront alors dédiés à la définition d'un modèle générique, basé sur l'état de l'art étudié et les différentes approches implémentées. Après avoir redéfini différentes notions pour une vision plus abstraite de la problématique (chapitre 6), nous décrirons les mécanismes d'un tel système (chapitre 7) ainsi que leur implémentation dans un prototype que nous évaluerons.

Nous conclurons par un bilan de l'étude réalisée et des résultats obtenus avant de présenter les travaux en cours et les perspectives à court et moyen termes. Une liste des publications est également disponible en annexe A.

# Première partie

# État de l'art

# Chapitre 1

# Qu'est ce qu'un service ?

Un service, bien que réel, est un objet difficile à définir et à représenter dans la mesure où le même terme peut désigner différents concepts selon le domaine considéré. Par définition et hors de tout contexte on peut cependant admettre que d'une manière ou d'une autre, un service apporte une fonctionnalité répondant à un besoin d'une personne réelle, même si celle-ci n'en a pas directement conscience. Ainsi, un service « offrira » généralement une fonction pratique à son bénéficiaire, celui-ci ne pouvant obtenir le même résultat par ses propres moyens, sinon de manière aussi simple et efficace (de son point de vue).

*« Un service vaut ce qu'il coûte. » Victor Hugo.*

La valeur ajoutée intrinsèque des services révèle un potentiel économique exceptionnel : ils répondent à autant de besoins qu'ils n'en créent de nouveaux et permettent de commercialiser des produits non matériels. L'Institut national de la statistique et des études économiques (INSEE) définit le service de la manière suivante [3] :

*« Une activité de service se caractérise essentiellement par la mise à disposition d'une capacité technique ou intellectuelle. ».*

Bien que relatif au secteur économique tertiaire dont il est la base, le service, de par sa qualité de production immatérielle, convient parfaitement au domaine des Technologies de l'Information et de la Communication (TIC). Ainsi, bien que de nombreux exemples de services puissent trouver résonance dans la vie de tous les jours, nous nous focaliserons sur les formes propres aux TIC, domaine exclusif de l'ensemble de ces travaux de thèse.

De nombreux exemples de services adoptant différentes formes coexistent dans les Technologies de l'Information et de la Communication, afin d'en dégager des propriétés communes (cf. 1.3) nous nous sommes intéressés à ces cas d'utilisation selon deux axes : au niveau applicatif (cf. 1.1) et dans les télécommunications (cf. 1.2).

## 1.1 Les services applicatifs

Par abus de langage, nous qualifierons d'« applicatif » l'ensemble des services informatiques, qu'ils soient délivrés localement par un système d'exploitation ou à distance via un réseau d'accès. Nous détaillons dans cette section ces deux cas qui abordent différemment la notion de service et précisons les concepts introduits par chacun d'eux.

### 1.1.1 Services locaux

Lorsque l'on s'intéresse à l'ensemble des systèmes informatiques et plus particulièrement à la raison d'être de ce que l'on appelle plus communément « ordinateurs », on constate rapidement que leur objet premier est de nous rendre des services. L'informatique étant par définition le traitement automatisé de l'information, les systèmes informatiques fournissent à leurs utilisateurs des fonctionnalités pratiques ou agréables, « *techniques et intellectuelles* » (par exemple un éditeur de texte, une base de données, un lecteur multimédia, etc).

Les caractéristiques supérieures « innées » d'un système informatique sur l'Homme en termes de vitesse de calcul et de capacité mémoire en font le serviteur par excellence pour rendre des services et alléger l'utilisateur de tâches complexes, fastidieuses ou simplement irréalisables. Ainsi, en ce sens, toute contribution d'un système informatique à l'Homme pourrait dans l'absolu être qualifiée de service.

Cependant, le terme « service » est peu employé en tant que tel dans un environnement applicatif local, on parle plus volontiers d'application, de logiciel, de programme, de tâche, de processus, ... Je ne prétends nullement que cette riche terminologie soit incongrue ou que ces mots soient synonymes, je tiens juste à souligner que l'utilisateur face à son ordinateur pense peu en terme de « service »,

confronté physiquement à sa réalisation il le nomme plus naturellement par leur représentation concrète.

Cette particularité est d'autant plus intéressante qu'elle s'applique exclusivement aux systèmes locaux. Nous le verrons par la suite, l'utilisateur qui jouit d'une fonctionnalité sans avoir conscience de son implémentation comme c'est le cas pour les services distants ou de télécommunications est plus enclin à la nommer « service ».

Le terme exact « service » est toutefois employé dans le monde applicatif pour désigner un type précis de fonctionnalité. Introduit par le système d'exploitation Windows de Microsoft, un « *service Windows* » [4] est un programme qui possède les caractéristiques suivantes [5] :

- un cycle de vie calqué sur celui du système hôte (et donc indépendant de l'utilisateur),
- pas ou peu d'interactions avec l'utilisateur (le service est souvent rendu directement au système et contrôlé par celui-ci),
- un mode d'exécution privilégié (généralement plus de droits que l'utilisateur lambda),
- une exécution « discrète » (l'utilisateur n'a souvent pas conscience du service rendu bien qu'il en jouisse directement ou indirectement).

Cette vision de « service » est unique à MS Windows, les autres principaux systèmes d'exploitation tels que UNIX (et compatibles, *i.e.* Linux et BSD) ou encore MAC OS X d'Apple préfèrent l'utilisation du terme « démon » (*daemon* [6]) pour désigner ce type de programme. Les fonctionnalités offertes par les services Windows sont diverses tel que la gestion des réseaux, l'indexation des données ou encore la synchronisation de l'heure, cf. [7] pour plus d'information sur les composants de base.

On peut également noter que Apple Inc. introduit dans MAC OS X des *« services systèmes »*. Ces services ne constituent pas un nouveau programme mais plutôt l'interface d'une application existante par laquelle celle-ci expose les fonctions basiques qu'elle peut réaliser pour un tiers (typiquement une autre application). Ainsi les services « rendus » par les applications sont échangés via une architecture spé-

cifique qui permet le transfert de données entre programmes tel que présenté dans la Figure 1.1.

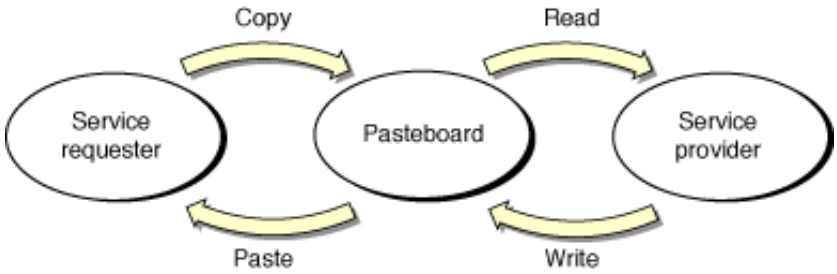



FIGURE 1.1 – Architecture de *services systèmes* dans MAC OS X.

Les services disponibles correspondant aux types de données de l'application en cours sont automatiquement listés dans un menu dédié. La Figure 1.2 présente l'exemple d'un tel menu (appelé ici « *Services* ») qui permet à un éditeur de texte (*TextEdit*) d'accéder au service fourni par un navigateur Web. L'information échangée est alors une adresse Internet (`www.rabbits.com`) résultant vraisemblablement au chargement du site Internet correspondant dans le navigateur Web offrant le service.

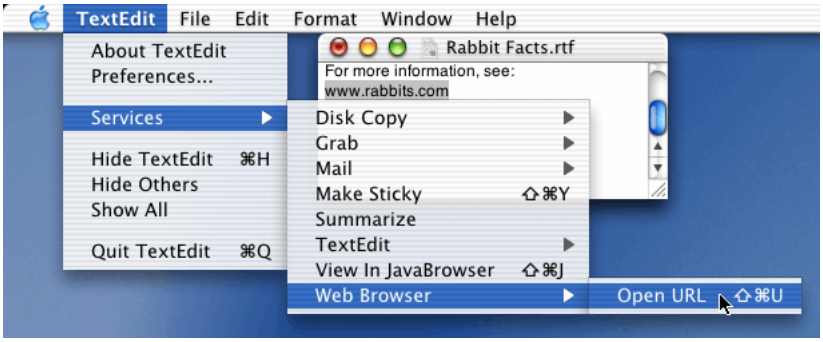



FIGURE 1.2 – Exemple de *services* MAC OS X.

Les *services systèmes* de MAC OS X et les *services Windows* présentés précédemment sont les deux seuls exemples d'introduction de la notion de service dans le domaine applicatif. De plus ces approches sont relativement différentes entre elles et ne considèrent l'utilisateur qu'indirectement : la première étant dédiée au système d'exploitation tandis que la seconde aux applications. Cependant, et nous

y reviendrons plus loin dans cette section, la majorité des fonctionnalités fournies par un environnement informatique, typiquement via des applications gérées par un système d'exploitation, sont ni plus ni moins des services pour l'utilisateur.

### 1.1.2 Services distants

Les services distants sont caractérisés par une architecture asymétrique « client-serveur » dissociant physiquement la consommation et l'implémentation de la fonctionnalité offerte. Le *serveur* comme son nom l'indique, implémente les mécanismes nécessaires à la fourniture du service au *client*, lequel à son tour le présente de manière « consommable » à l'utilisateur. Le service résulte ainsi de l'exécution d'au moins deux processus distincts, l'un du côté utilisateur et l'autre généralement plus complexe chez le fournisseur qui réalise les principaux traitements. Les exemples de tels services sont multiples, les plus communs étant peut-être sur le Web, rendus possible par la collaboration entre le navigateur de l'utilisateur et les serveurs Web des fournisseurs de sites sur le réseau Internet (cf. Figure 1.3).


FIGURE 1.3 – Modèle client/serveur, exemple d'Internet.

À la différence d'un environnement applicatif local (cf. 1.1.1), la fourniture d'un service distant requiert la collaboration de plusieurs applications, et ce via un protocole de communication prédéfini. Cette communication permet le déclenchement et le contrôle du service, menant généralement à un transfert de données. Ces données brutes sont alors traitées et interprétées par le client puis présentées à l'utilisateur de manière à rendre la fonctionnalité désirée.

**Notion de session.**

Le flux d'informations échangé entre clients et serveurs est appelé *session de communication* [8] (cf. Figure 1.4). Dans un environnement distant, les services n'existent qu'au travers de ces sessions qui contrôlent et transportent les informations essentielles à la fonctionnalité offerte. De manière générale, on peut distinguer deux types de sessions présentes simultanément entre le client et le serveur lors de la fourniture d'un service.

- Les *sessions de contrôle* qui permettent aux entités de gérer le service, par exemple le protocole HTTP (*Hypertext Transport Protocol* [9]) utilisé par le navigateur pour demander le contenu d'une page Web au serveur correspondant.
- Les *sessions de données* qui transportent les informations propres au service tel que le contenu d'une page Web : code HTML, images et divers objets, transférés du serveur vers le navigateur.

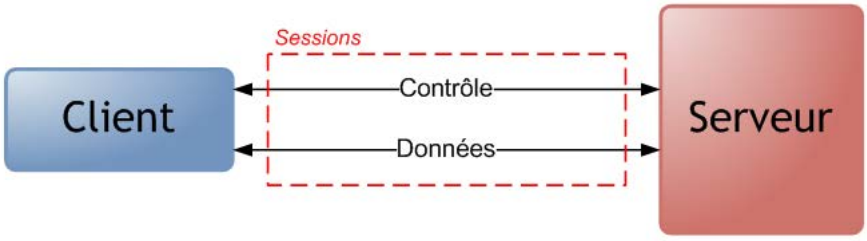



FIGURE 1.4 – Sessions d'une architecture client-serveur.

Une relation classique entre un client et un serveur implique nécessairement ces deux sessions simultanément, le format de chacune étant formellement défini afin d'assurer l'interopérabilité, soit de manière ouverte via un standard (par exemple les spécifications du W3C et de l'IETF pour HTTP [10]), soit de manière fermée via un protocole propriétaire (par exemple Adobe Flash RTMP [11]). À noter que physiquement, une session de communication peut multiplexer les informations de contrôle et les données comme c'est le cas avec HTTP ou les gérer séparément comme pour le protocole SIP (*Session Initiation Protocol* [12]). Le contrôle du service de communication étant assuré par SIP lui-même et les données multimédias acheminées via le protocole RTP (*Real-time Transport Protocol* [13]).

**Modèle pair-à-pair.**

D'un point de vue « services », les architectures pair-à-pair (P2P [14]) sont un cas particulier de modèle client/serveur dans la mesure où chaque entité, ou « pair » (*peer*) est à la fois client et serveur (cf. Figure 1.5). Les services sont ainsi distribués et rendus entre pairs, sans serveur centralisé. Cependant, les caractéristiques d'un service fourni via un tel modèle distribué restent les mêmes, la fonctionnalité étant créée à partir d'une (ou plusieurs) session de données gérée par une session de contrôle.

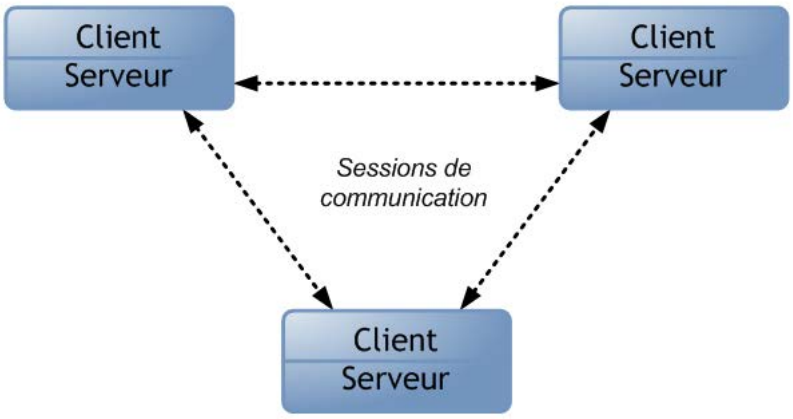


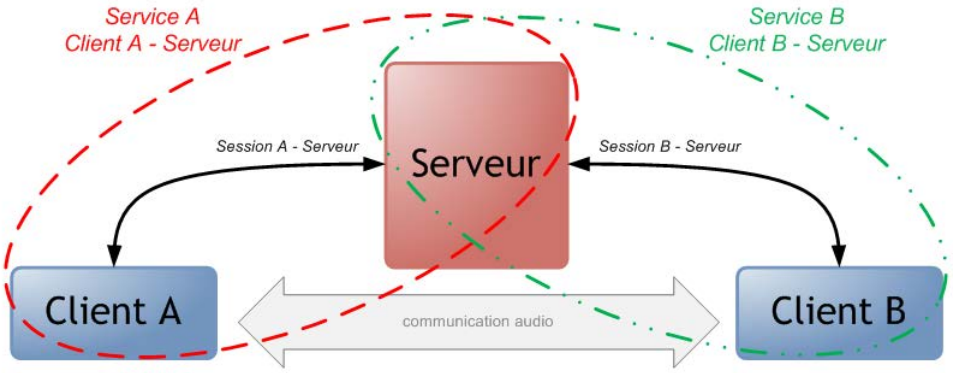

FIGURE 1.5 – Modèle pair-à-pair.

Que l'on considère une architecture client/serveur classique ou un modèle P2P, la fourniture d'un service est matérialisée par un transfert de données entre deux entités, l'une jouant le rôle de client et l'autre de serveur. Chaque session de communication (contrôle et données) correspondant à une fonctionnalité, il existe donc nécessairement une instance de service par relation client/serveur.

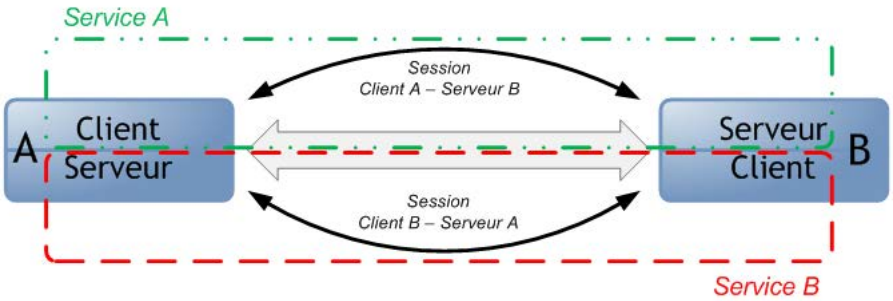
**Instances de services.**

Ainsi chaque utilisateur possède une instance de service qui lui est propre, et ce même si la fonctionnalité finale ne peut être rendue sans l'existence d'autres clients (typiquement un service de communication) ; ces derniers possèderont alors chacun leur instance de service, éventuellement fournie par le même serveur. Les Figures 1.6 illustrent les instances de services présentes lors d'une communication

audio entre deux clients dans une architecture client/serveur 1.6(a) puis selon un modèle pair-à-pair 1.6(b).




(a) Dans une architecture client/serveur.




(b) Dans un modèle pair-à-pair.

FIGURE 1.6 – Instances de services, exemple d'une communication audio.

La Figure 1.6 met en valeur la relation entre l'utilisateur et son service. Quel que soit le nombre d'entités impliquées dans le rendu d'une fonctionnalité, une instance de service est créée pour chaque utilisateur. Il appartient alors au serveur qui implémente la fonction offerte de combiner les différentes sessions afin de réaliser le traitement adéquat.

### 1.1.3 Services Web

Nous avons vu que le terme « service » était peu employé dans les environnements locaux (cf. 1.1.1), préférant par exemple la notion d'application. De plus,

lorsqu'une fonctionnalité est dissociée de son implémentation, l'utilisateur l'appelle plus naturellement « service » car il en perçoit directement les effets sans voir sa réalisation. C'est le cas des services distants (cf. 1.1.2) délivrés par un bien nommé « serveur » à un client, tous deux connectés via un (ou des) réseau. Cette section est consacrée aux *services Web*, un exemple de service distant particulièrement intéressant.

Interconnectant *à l'infini* les systèmes informatiques entre eux, le réseau Internet constitue la première infrastructure de services du point de vue de l'offre. Entièrement ouverte, elle favorise la création et le déploiement de tout type de services par n'importe quel fournisseur ; la richesse de l'offre ainsi proposée n'ayant d'égal que l'anarchie qui y règne. En effet, chaque fournisseur offrant une solution propriétaire sans considération de l'existant, l'architecture de services naturellement fondée sur le Web perd en efficacité et ne peut répondre à certains (nouveaux) besoins tels que (liste non exhaustive) :

- la découverte,
- la réutilisation,
- la composition,
- et l'interopérabilité des services.

Le but de ces fonctions est de traiter dynamiquement des services afin de créer une offre personnalisée plus riche, adaptée à l'utilisateur. Mais une opération automatisée sur un objet, fut-ce un service, implique un format standard et non une forme propriétaire, imprédictible, dépendante du fournisseur. C'est pour répondre à cette problématique que les *services Web* furent introduits par le World Wide Web Consortium (W3C [15]), constituant une première approche de formalisation en standardisant l'architecture de services Web.

La Figure 1.7, extraite de la spécification de l'*Architecture des Services Web* (WSA [16]) du W3C, illustre les interactions entre les différents acteurs dans la mise en œuvre d'un service. On y retrouve naturellement les deux entités essentielles : le client qui sollicite la fonctionnalité (ici *Requester Entity*) et le serveur qui la fournit (ici *Provider Entity*). Le schéma met également en avant la présence de l'Homme qui propose (fournisseur) et dispose (consommateur) du service via un système informatique, ici implémenté par des agents.

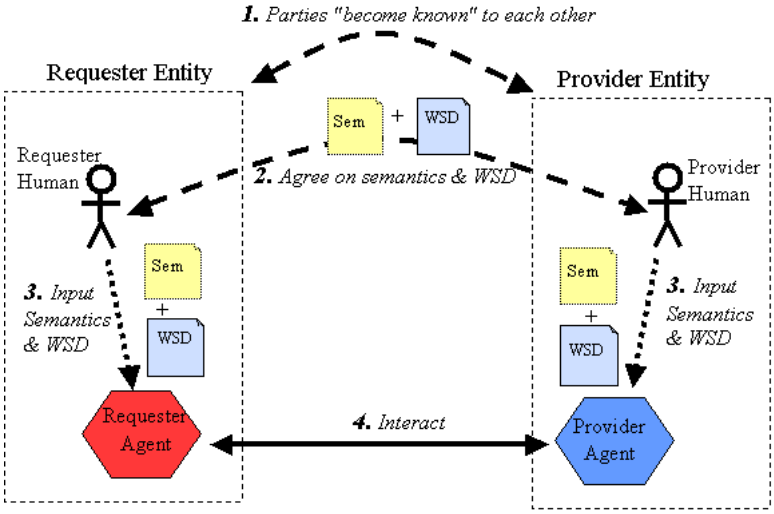



FIGURE 1.7 – Mise en œuvre d'un *service Web*.

Toujours selon la Figure 1.7, la mise en œuvre d'un service Web est décrite selon quatre étapes.

1. Phase de découverte des deux entités. C'est généralement l'utilisateur qui, selon ses besoins, recherche, sélectionne puis contacte directement un serveur. Ce dernier ayant publié préalablement les informations correspondantes à la fonctionnalité fournie, permettant aux clients potentiels de le solliciter (par exemple via un annuaire de services, un site Internet, une application, etc).
2. Le client et le serveur conviennent ensuite de la fonctionnalité à délivrer en fonction des possibilités du fournisseur et des attentes de l'utilisateur, signant une sorte de contrat numérique. Le comportement de chacune des entités y est formellement décrit à l'aide d'un langage spécifique : WSDL (*Web Services Description Language* [17]).
3. La description formelle du service Web (WSD, *Web Service Description*) est alors transmise aux agents qui implémentent les mécanismes requis pour chaque entité.
4. La fourniture proprement dite du service s'effectue alors entre le client et le serveur, matérialisée par des échanges de données entre les deux agents.

Ainsi, les services exposent des interfaces standardisées leur permettant d'être réutilisés par d'autres. On peut alors manipuler, réutiliser, composer ces *briques fonctionnelles* pour construire des services dits « riches », combinant plusieurs fonctionnalités dans une offre unique. La Figure 1.8 est un exemple de composition de services Web, elle est extraite d'un document du W3C proposant divers scénarios d'utilisation et démontrant les possibilités offertes par une telle architecture (cf. [18]).

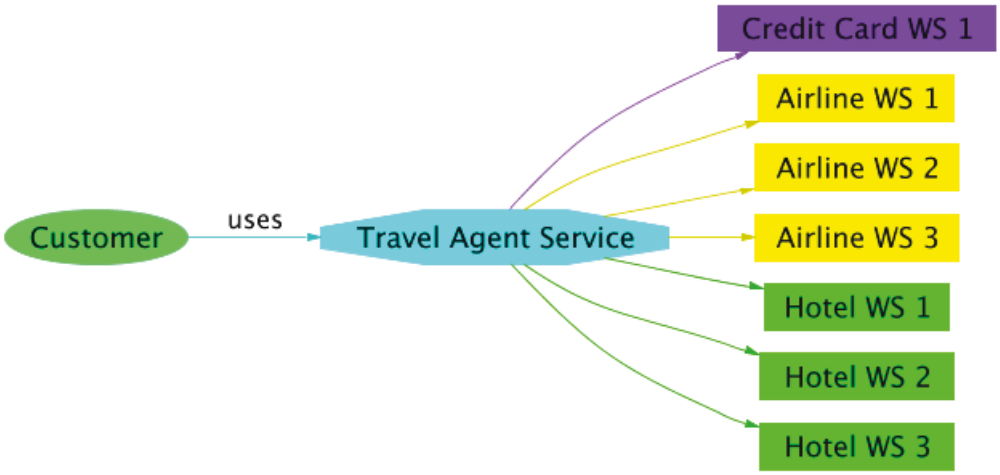



FIGURE 1.8 – Exemple de composition de services Web.

Le scénario présenté dans la Figure 1.8 est relativement basique, illustrant une utilisation assez courante de service Web. Un utilisateur souhaite réserver en ligne un séjour auprès d'une agence de voyage. Pour se conformer à la terminologie employée jusqu'alors, le *client* est donc le navigateur Internet de l'utilisateur, le *serveur* est l'application de réservation en ligne de l'agence de voyage et le *service* offert est complexe : recherche et comparaison d'hôtels et de vols à partir de critères fournis par l'utilisateur puis paiement en ligne.

La figure fait apparaître trois types d'acteurs : le client qui demande le service, le serveur qui répond à la requête et un ensemble de fournisseurs tiers qui assurent chacun une fraction de la fonctionnalité finale offerte. Le serveur sert l'utilisateur tout en jouant le rôle de client vis-à-vis de fournisseurs tiers mettant à sa disposition des briques fonctionnelles qui, une fois assemblées, constituent un service riche. Ici l'application de réservation ne possède aucune information ni contrôle sur les hôtels, les vols ou encore les mécanismes de paiement ; l'agence de voyage doit

passer par des fournisseurs tiers dont la combinaison des fonctionnalités offertes lui permet de satisfaire à la requête du client.

La spécification de ces interfaces rend possible de nombreux nouveaux usages, offrant une véritable architecture de services basée sur le Web, contrairement aux solutions locales qui restent généralement isolées, propriétaires et incompatibles (à l'exception peut-être des services systèmes de *mac os x*, cf. 1.1.1). Cette approche apporte de nombreux avantages : l'enrichissement dynamique de services par la découverte de fonctionnalités tierces, le déploiement accéléré de services complexes par la réutilisation de briques fonctionnelles existantes ou encore la personnalisation de l'offre.

Il faut cependant noter que la force de la composition basée sur la collaboration dynamique de fournisseurs de services tiers, entraîne des problèmes potentiels relatifs à l'interaction des fonctionnalités dont l'inter-fonctionnement peut générer des conflits. Cette problématique majeure appelée « interaction de services » fait l'objet de nombreux travaux de recherche, essentiellement dans le domaine des télécommunications comme nous verrons dans la section suivante (cf. 1.2.2).

## 1.2 Les services de télécommunications

À ses débuts dans les années 1870, le réseau téléphonique ne proposait qu'un seul et unique service : la télécommunication voix. Cependant, fondée plus d'un siècle avant son homologue informatique, l'infrastructure de télécommunications a su évoluer, proposant progressivement de nouvelles fonctionnalités : renvoi d'appel, présentation du numéro, renseignements, horloge parlante, messagerie vocale... Bien qu'exclusivement audio initialement, les services se sont progressivement enrichis avec du texte, des images, des données, permettant aux opérateurs de proposer une véritable offre multimédia : visiophonie, messagerie multimédia (MMS), vidéo à la demande (VoD), télévision, etc.

Mais l'évolution la plus remarquable du point de vue des services fut sans conteste l'interconnexion avec un autre monde : Internet (cf. 1.2.3). Avec 5 milliards de clients (abonnements et forfaits prépayés), le réseau téléphonique (fixe et mobile) est la première plateforme de services *en terme du nombre de clients*,

loin devant Internet et ses 1.6 milliards d'utilisateurs (ITU 2008 [19]). Cependant, bien que plus récente, la richesse de l'offre de services sur Internet est aujourd'hui nettement supérieure à celle proposée par l'infrastructure historique de télécommunications. Deux principaux facteurs expliquent ce constat inattendu.

**Des terminaux limités.**

Les appareils téléphoniques sont nettement moins performants que les ordinateurs, et ce en tous points : vitesse de calcul, capacité mémoire, interface utilisateur, connectivité, etc. C'est la conséquence des approches structurellement opposées du monde télécom (des télécommunications) et d'Internet.

Deux philosophies s'opposent. D'un côté des terminaux basiques asservis à un réseau téléphonique complexe et puissant qui assure toutes les fonctionnalités de manière centralisée (*l'intelligence au cœur*), de l'autre Internet, une simple tuyauterie qui achemine des données d'un ordinateur à un autre, ces derniers produisant et consommant les services (*l'intelligence à la périphérie* [20, 21]).

Ainsi les terminaux téléphoniques sont limités car conçus pour consommer uniquement les fonctions que le réseau dont ils dépendent est en mesure de leur offrir. Cette contrainte est double car elle limite également le réseau qui ne peut offrir de nouvelles fonctionnalités sans renouveler les appareils clients, d'où une offre de services restreinte. À l'inverse, les terminaux informatiques ne sont ni dépendants du réseau Internet (en terme de fonctionnalités), ni dédiés à un ensemble prédéfini de services, ils sont pensés pour des usages multiples et indéterminés a priori. La création de nouveaux services est alors débridée, sans considération des caractéristiques des clients, non sans quelques problèmes d'incompatibilité.

**Une infrastructure fermée.**

Autre différence structurelle entre les deux réseaux, le modèle Internet de part sa logique d'intelligence déportée à la périphérie du réseau, mélange producteurs et consommateurs de services. En effet, le réseau n'offrant aucun service en tant que tel, ce sont les éléments terminaux qui échangent directement, collaborent, se comportant comme producteur ou consommateur de fonctionnalités selon leurs besoins.

À l'opposé, le réseau téléphonique est géré par des opérateurs qui délivrent directement les fonctionnalités à leurs clients, l'offre est ainsi cloisonnée, fermée à des fournisseurs de services tiers (du moins sans un accord commercial avec l'opérateur). Ainsi, le modèle téléphonique cantonne l'utilisateur à son rôle de consommateur, ne pouvant devenir acteur comme avec Internet. Cette nouvelle contrainte limite intrinsèquement la génération de nouvelles fonctionnalités, le création de services étant réservée aux fournisseurs autorisés.

Si l'offre en terme de services peut paraître faible comparée au Web, l'infrastructure de télécommunications possède elle de nombreuses qualités : connectivité entre domaines et technologies d'accès, identification des clients (et authentification), mécanismes de paiement, gestion de la qualité de service, etc. L'infrastructure est l'atout majeur du monde télécom, sa robustesse (ou sa « lourdeur » selon ses détracteurs) est l'œuvre des efforts de standardisation qui assurent son interopérabilité, sa force.

### 1.2.1 Standardisation

L'Union Internationale des Télécommunications (ITU, créée sous le nom d'Union Internationale du Télégraphe en 1865) est le premier et principal effort de standardisation des télécommunications (infrastructure et services) jusque dans les années 1980. Alors, le développement des ordinateurs personnels a donné naissance à de nouveaux modes de communication qu'il a fallu rapidement normaliser.

L'ITU étant peu réactif, de nouveaux groupes de normalisation tels que l'Internet Engineering Task Force (IETF) ou le World Wide Web Consortium (W3C) se sont constitués afin de fournir une réponse rapide et éviter l'adoption *de facto* de solutions propriétaires. Similairement, l'évolution et l'essor des radiotélécommunications a engendré l'émergence de l'ETSI (*European Telecommunications Standards Institute*) et du 3GPP (*3rd Generation Partnership Project*). Enfin, certains groupes tels que l'OMA (*Open Mobile Alliance*) ou le *Parlay Group* se sont consacrés à la normalisation des services, assurant ainsi une certaine cohérence face à la multiplication des acteurs, inhérente à l'ouverture du modèle.

Il est intéressant de noter que les deux mondes, Web et télécom, tendant à se confondre au fil des évolutions technologiques, certains standards Internet générale-

ment plus flexibles et ouverts, sont adoptés dans les normes de télécommunications. Un exemple connu est SIP, un protocole de signalisation applicatif standardisé par l'IETF qui fut adopté par le 3GPP pour son infrastructure IMS.

Un standard de télécommunications a eu une importance toute particulière durant ces travaux de recherche : l'IP Multimedia Subsystem (IMS). Cette architecture de services très prometteuse lors de son introduction peu avant le début de la thèse propose des mécanismes de mobilité intéressants par lesquels l'étude a débuté. De plus c'est la première approche qui établisse véritablement le lien entre les mondes Web et télécom. Ainsi la section 1.2.2 présentera brièvement l'IMS et quelques-uns de ses principaux concepts.

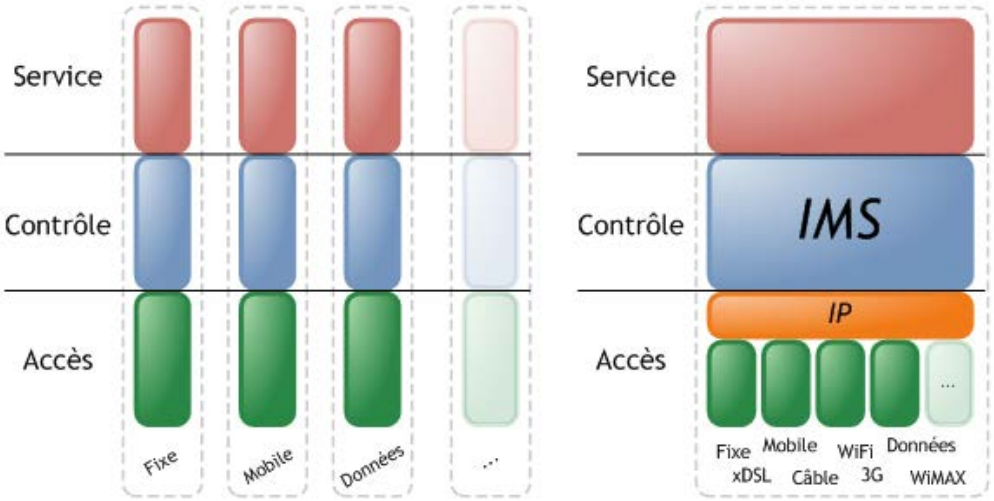
### 1.2.2 IP Multimedia Subsystem

L'IP Multimedia Subsystem est l'architecture de services de référence introduite par le 3GPP en 2003. Elle constitue la dernière évolution du cœur de réseau de télécommunications UMTS (*Universal Mobile Telecommunications System*, une norme de téléphonie mobile 3ème génération). Pour en savoir plus, [22] est un ouvrage particulièrement complet sur l'IMS.

La philosophie de l'IMS est de faire le lien entre services et utilisateurs de manière transparente, quelque soit la technologie d'accès employée, en d'autres termes assurer la convergence. Pour cela, le 3GPP propose de changer le modèle vertical classique (cf. Figure 1.9(a)) où chaque réseau possède sa couche de contrôle et son portefeuille de services dédiés (le modèle téléphonique fermé et monopoliste en vigueur jusqu'alors). Outre la situation protectionniste avantageuse pour l'opérateur *sur son réseau*, la verticalité est dommageable aux niveaux contrôle et service. Les mécanismes de contrôle sont dédiés à la technologie d'accès sous-jacente et donc non transposables à d'autres environnements, de même les services doivent être développés pour une couche de contrôle donnée et donc incompatibles avec d'autres infrastructures.

Face à ce manque d'efficacité manifeste, le 3GPP apporte une solution. Le modèle est désormais organisé en couches horizontales (cf. Figure 1.9(b)), services et réseaux d'accès sont articulés autour d'une couche de contrôle unique basée sur le protocole IP (*Internet Protocol*, [23]) et assurée par l'IP Multimedia Subsystem.

En dessous, l'interconnexion avec les différents réseaux d'accès est normalisé par le groupe *Telecoms & Internet converged Services & Protocols for Advanced Networks* (TISPAN de l'ETSI). Au dessus, les services sont standardisés par l'OMA (ou encore le Parlay Group et l'ETSI pour assurer la (rétro-)compatibilité).




(a) Modèle vertical. (b) Modèle horizontal.

FIGURE 1.9 – Modèles d'architectures de services.

**Architecture.**

L'IMS est en réalité un ensemble d'infrastructures IMS possédées par divers opérateurs télécoms et inter-connectées entre elles. Par abus de langage on appelle IMS chaque domaine autonome déployé par un opérateur, un schéma simplifié de réseau 3G comportant un cœur IMS est illustré en Figure 1.10.

Un IMS, ou plus exactement un domaine IMS est composé d'un grand nombre de composants qui collaborent afin de réaliser l'ensemble des fonctions nécessaires à une infrastructure de services (télécom) : sécurité (identification, authentification, autorisation), routage (établissement de sessions, communication inter-opérateur), déclenchement de service, facturation, etc. Le tout est assuré par les *Call Session Control Functions* (CSCF), la colonne vertébrale de l'IMS constituée de trois éléments clés (cf. [24, 25]).

– **Proxy-CSCF**. Le point d'entrée d'un domaine IMS, il assure une liaison sécurisée entre le terminal et l'infrastructure.
– **Interrogating-CSCF**. Composant médiateur, il assure le routage inter-domaine et permet en outre de masquer la topologie interne de l'infrastructure sous-jacente.
– **Serving-CSCF**. Le cerveau du domaine IMS, il réalise les opérations de base (routage, authentification) mais surtout il déclenche les services de chaque utilisateur en fonction du contexte.

Enfin, ces composants de contrôle sont connectés à une base de données, le *Home Subscriber Server* (HSS) qui détient l'ensemble des informations relatives aux clients du domaine : données d'authentification, profils de services, etc.

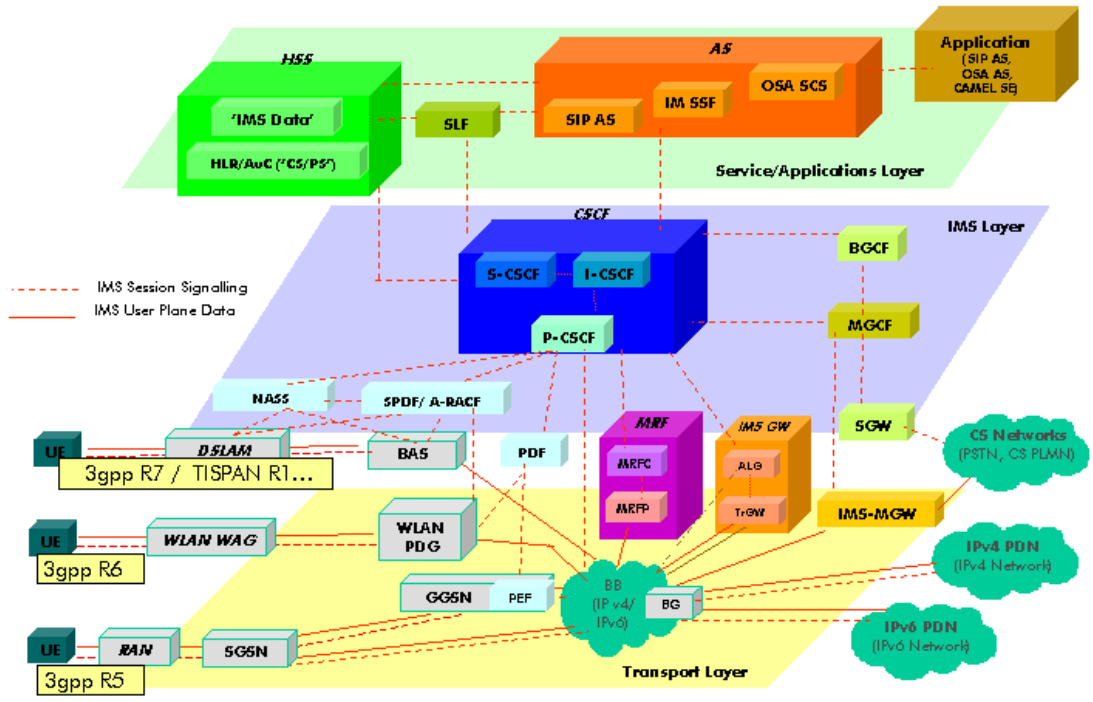



FIGURE 1.10 – Réseau 3G selon le 3GPP.

Sur la Figure 1.10 représentant l'intégralité d'un réseau 3G tel que proposé par le 3GPP, vous saurez y retrouver l'organisation structurelle horizontale vue précédemment (cf. Fig. 1.9(b)) : la couche contrôle assurée par l'IMS au milieu, la couche application/services au-dessus et la couche transport/accès au-dessous.

Au niveau IMS, les communications sont effectuées via SIP, un protocole venu du monde Internet et standardisé par l'IETF qui permet l'établissement et le contrôle de sessions multimédias. L'ensemble des services sont ainsi délivrés de bout en bout via SIP. Pour plus de détails sur ce protocole, cf. RFC 3261 [12].

Le rôle premier d'une infrastructure de services est de gérer la fourniture des services aux clients : déclenchement, composition, interactions, ... Voyons en quoi cela consiste et de quelle manière est-ce implémenté dans l'IMS.

**Déclenchement des services.**

Le déclenchement (de services) est le mécanisme qui contrôle la fourniture des fonctionnalités aux utilisateurs en fonction du contexte. En effet, les services souscrits par un client doivent être fournis au bon moment conformément à ses attentes. Ainsi, pour chaque utilisateur, des règles sont établies afin de conditionner la fourniture d'un service. Par exemple, lors de l'établissement d'une communication avec un service d'anonymat, divers mécanismes entrent en jeu afin de cacher l'identité de l'appelant tel qu'il l'a souscrit auprès de son opérateur. Le déclenchement du service est réalisé de manière transparente pour le client.

Dans l'IMS, le déclenchement est assuré par le S-CSCF qui intercepte les messages SIP initiés par les terminaux clients et les redirige temporairement vers des serveurs d'applications (AS, *Application Server*) chargés de réaliser une action précise correspondant à un service tel que masquer l'identifiant de l'appelant. Ce déclenchement s'opère au sein du S-CSCF lorsque un ensemble de conditions est satisfait, ces règles appelées iFC (*initial Filter Criteria* [26]) sont stockées dans le HSS et évaluées à chaque message SIP transitant par le S-CSCF. Ainsi, en fonction de l'état d'une session de communication entre deux clients (création, modification, terminaison, etc) les services adéquats peuvent être appliqués aux utilisateurs.

Le déclenchement de services est réalisé pour chaque intervenant de la session de communication et pour chaque service souscrit et ce par le domaine dont

l'utilisateur dépend. Les messages SIP qui permettent de gérer la session de communication sont ainsi routés entre les S-CSCF et leurs AS respectifs.

**Composition et interactions.**

Outres les fonctions de base intégrées à la couche de contrôle (à l'IMS) tels que le routage, l'authentification ou la facturation, les autres services sont proposés au-dessus, par les AS dans la couche application. Il est à noter que certains services essentiels et appelés *enablers* sont standardisés par le 3GPP ou l'OMA tels que la *Présence* [27, 28] ou le PoC (*Push-to-talk over Cellular* [29, 30]) afin d'offrir des « briques fonctionnelles » à des services plus évolués. C'est alors en composant ces *enablers* et éventuellement d'autres services de base que des fournisseurs d'applications tiers peuvent déployer à moindre coût des services riches sur l'infrastructure IMS ; du moins, c'est l'objectif affiché.

La composition reste toutefois complexe et elle est définie que très vaguement dans l'environnement IMS [31], cependant de nombreux travaux existent sur ce sujet tels que [32, 33, 34].

Un même service pouvant être composé des fonctionnalités de plusieurs AS, le nombre de déclenchements réalisés pour chaque participant à la communication peut alors devenir très important, augmentant d'autant le risque d'interférence entre les fonctionnalités offertes, c'est ce que l'on appelle « interaction de services ».

La problématique d'interaction de services n'est pas récente, de nombreux travaux ont été réalisés sur le Réseau Téléphonique Commuté (RTC) afin d'assurer une gestion cohérente des fonctionnalités offertes (cf. [35]). Le test des services avant leur déploiement (*offline*) et l'offre de packs de services compatibles permettaient aux opérateurs de contrôler a priori les interactions. Cependant, le modèle s'est ouvert et l'opérateur ne possède plus un contrôle total sur l'offre de service, des fournisseurs tiers interviennent directement. Il devient alors impossible de résoudre a priori les interactions ; de nouvelles solutions, dynamiques (*online*), doivent être adoptées. Aucun mécanisme adéquat n'est explicitement défini dans les spécifications techniques du 3GPP, cependant de nombreux travaux existent tels que [36, 37, 38] ou encore [39] (mémoire de Master recherche du même auteur).

Il est intéressant de constater à quel point les réseaux de télécommunications ont évolué, faisant une place de plus en plus grande aux services. S'ils étaient initialement centrés sur les couches basses proposant des fonctionnalités simples tel que la voix, les améliorations techniques au niveau de l'accès (technologies sans fil, débits) ont bouleversé les modèles économiques des opérateurs. Ceux-ci misent désormais sur les services et leur infrastructure qui constituent aujourd'hui la véritable valeur ajoutée : ouverture du modèle, déploiement rapide, composition automatique, détection et résolution d'interactions, ... L'IMS en est une illustration.

### 1.2.3 Une fenêtre sur le Web

Au début du 21ème siècle, les évolutions technologiques aidant, notamment en matière de radiocommunications (débit accru avec l'avènement de la téléphonie de troisième génération) et d'électronique (miniaturisation des appareils informatiques), le monde télécom s'est timidement ouvert en proposant un nouveau « super-service » à ses abonnés : pas moins que le Web, la toile de services accessibles via Internet.

Avec une fenêtre sur le Web, l'infrastructure de télécommunications efface ainsi l'un de ses principaux défauts : une offre de services réduite, rigide et peu attrayante où le client reste passif. Cependant, si l'ouverture du modèle est une révolution pour les utilisateurs, elle pose de nombreux problèmes aux opérateurs télécoms qui doivent faire face à une concurrence nouvelle, introduite au sein même de leur réseau via Internet.

En effet, non seulement les opérateurs perdent leur monopole dans la fourniture des services mais les fournisseurs tiers provenant du Web sont très nombreux, performants et présents dans tous les domaines. La communication n'échappe évidemment pas à la règle et les opérateurs sont obligés dans un premier temps de restreindre techniquement les abonnés afin de conserver un minimum de contrôle et préserver leur modèle économique le temps de trouver des solutions viables.

Et c'est là tout le paradoxe né de la rencontre de ces deux mondes, les opérateurs télécoms cherchent à vendre un service qui va à l'encontre même des principes de leur infrastructure. Pour illustration, au moment de la rédaction de ce manuscrit (2008-2009) différents mécanismes sont mis en œuvre chez certains opérateurs

afin de « canaliser » les fonctionnalités Internet offertes à leurs abonnés mobiles : bridage du débit maximal des terminaux, blocage ou surtaxe de ports réseaux spécifiques, etc.

## 1.3 Cadre conceptuel

Via Internet, les mondes applicatif et des télécommunications tendent à se rapprocher, suffisamment du moins pour établir des propriétés communes et positionner différents concepts autour de la notion de service. Mais l'état de l'art présenté dans ce premier chapitre présente des lacunes dans la définition du concept de service, essentiellement dû au fait qu'il n'est généralement appréhendé que d'un angle restreint au domaine d'étude : économie, systèmes, Web, télécommunications, etc. Afin de permettre une compréhension constante des travaux exposés dans ce mémoire, nous proposons un cadre conceptuel qui reprend les définitions existantes et les complète afin de dessiner les relations entre ces différents concepts. La carte conceptuelle présentée en Figure 1.11 offre une vue haut-niveau d'un service et des différentes relations le liant aux notions qui l'entourent définies en section 1.3.1.

### 1.3.1 Autour du service

Comme nous avons vu précédemment (cf. 1.1.1), un *Service* fournit des *Fonctionnalités* à une personne réelle directement ou indirectement. En effet, conformément à l'état de l'art nous considérons qu'un service est intrinsèquement lié à un *Utilisateur*.

Le service ainsi fourni est instancié par une *Application* qui implémente et gère la fonctionnalité correspondante. Une application peut revêtir diverses formes comme nous l'avons vu dans cette première partie d'état de l'art : application locale, client/serveur, pair-à-pair, ... elle n'est pas limitée à un seul programme ou logiciel.

Cette application est exécutée par un *Système*, lui même embarqué dans un *Terminal*. La notion de système est généralement synonyme de *système d'exploitation* (SE [40]) cependant sur certains terminaux basiques à capacités limitées, on parle plutôt de *système embarqué* (cf. [41]) où l'application se confond au sys-

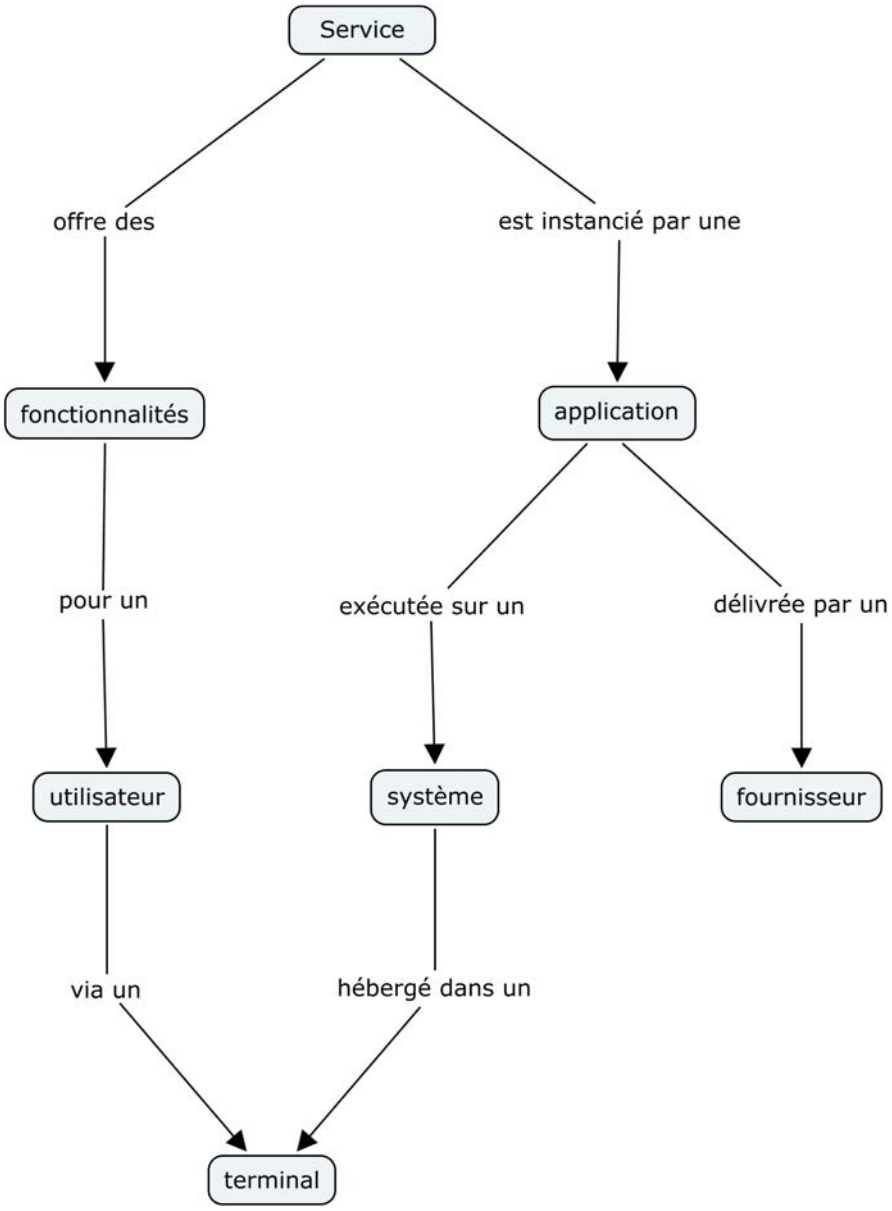



FIGURE 1.11 – Carte conceptuelle autour de la notion de *Service*.

tème dédié à un service spécifique. Les terminaux seront définis plus longuement en section 1.3.3.

Enfin, si le service est fourni à un Homme, il l'est également par un Homme. Le *Fournisseur* est celui qui offre (ou plutôt vend) le service. Dans le cadre des Technologies de l'Information et de la Communication, le fournisseur délivrera une application ou une autorisation d'accès via une licence qui offrira la fonctionnalité.

La Figure 1.11 illustre le positionnement de ces concepts et leurs relations. Les définitions proposées ici respectent et complètent l'état de l'art afin d'offrir au lecteur une meilleure compréhension des notions utilisées dans ce mémoire. Le niveau d'abstraction est volontairement très élevé afin de rester indépendant de toute implémentation, nous verrons par la suite que c'est cette abstraction même qui permet une généricité des solutions proposées. Enfin, certains de ces concepts seront développés plus loin (cf. 6.2) afin de proposer une vision plus pragmatique de la notion de service.

### 1.3.2 Notion de connexion

La notion de *connexion* apparaît souvent dans la littérature des Technologies de l'Information et de la Communication, cependant les concepts de modes « connecté » et « non-connecté » s'appliquent généralement à la couche réseau. Ainsi, le mode connecté correspond à la synchronisation entre l'émetteur et le récepteur d'une transmission de données s'effectuant sur une liaison réseau préétablie qui garantit une communication fiable. L'illustration classique de ce mode est le modèle de transfert du protocole TCP (*Transmission Control Protocol* [42]). À l'inverse, le protocole UDP (*User Datagram Protocol* [43]) offre un mode non-connecté dans le sens où la liaison émetteur-récepteur n'est pas établie a priori, la transmission n'étant alors pas garantie au niveau réseau.

La notion de connexion au niveau réseau est un indicatif de fiabilité de transmission, ce qui a peu de sens au niveau applicatif. Cependant la dépendance d'un service au réseau est une propriété importante lorsque l'on considère certaines contraintes, notamment celles de mobilité tel que nous le verrons dans la deuxième partie de cet état de l'art (cf. 2). Les notions de service *connecté* et *non-connecté* sont peu présentes dans la littérature or nous pensons qu'il serait inexact de leur

faire correspondre les concepts réseaux du même nom. En effet, nous assumons que d'un point de vue « service », la notion de connexion indique une dépendance au réseau, peu importe le mode du protocole de transmission sous-jacent.

Par exemple, une application qui consomme (ou produit) un flux de données constant basé sur UDP tel qu'une communication audio ou du streaming vidéo, offrira un service connecté bien que le protocole réseau utilisé soit en mode non-connecté car l'arrêt de la transmission interromprait immédiatement la fourniture du service. Inversement, une application basée sur un protocole connecté fournit un service non-connecté si l'interruption de la connexion réseau peut être supportée, ne serait-ce que temporairement, pendant la fourniture du service ; par exemple le protocole HTTP d'un navigateur Web basé sur TCP. Les services locaux (cf. 1.1.1) seront alors nécessairement non-connectés tandis que ceux basés sur des flux de données continus seront toujours en mode connecté.

### 1.3.3 Les terminaux

Dernier terme qu'il reste à définir, la notion de *terminal*. Les terminaux ne sont pas directement concernés par notre étude qui porte plus sur les services et leur mobilité, cependant ils sont omniprésents dans le contexte de nos travaux car ils sont l'unique objet physique, l'interface entre l'utilisateur et le(s) service(s). Le terme « terminal » est très employé dans la littérature mais aussi très peu défini de manière générique car toujours considéré dans un contexte spécifique. Dans le cadre de ce mémoire nous resterons très haut-niveau en considérant les terminaux comme des *appareils électroniques* possédant au moins les caractéristiques suivantes :

- une interface utilisateur,
- une connectivité réseau,
- une application.

Cette définition de terminal ne tient pas pour définition générale, elle indique uniquement au lecteur comment comprendre ce terme tout au long de ce mémoire.

# Chapitre 2

# Problématiques de mobilité

Nous savons maintenant ce qu'est un service, la pluralité des formes qu'il peut revêtir ainsi que ses nombreuses propriétés. Nous arrivons donc au second aspect, problématique fondamentale du sujet d'étude : la mobilité. Ce deuxième chapitre de l'état de l'art est consacré à la mobilité d'un point de vue « service », mais tout d'abord restons un peu génériques et intéressons-nous à ce que l'on entend par « mobilité ».

## 2.1 Types de contraintes

La mobilité « c'est quand ça bouge », pourrait-on penser benoîtement. Et en effet, même dans le domaine des Technologies de l'Information et de la Communication, il faut du mouvement pour parler de mobilité. Mais dans un environnement composé de terminaux, de réseaux d'accès, d'utilisateurs, d'applications, de services, de sessions, ... les combinaisons de mouvement sont infinies. Cette potentielle mobilité totale des composantes du système crée de multiples contraintes et ce à toutes les couches réseau du modèle de référence OSI [44]. Au niveau transport bien entendu mais également au niveau applicatif.

La mobilité a longtemps été exclusivement étudiée dans les couches inférieures, typiquement pour résoudre des problèmes de routage ou d'accès. Mais avec l'explosion des télécommunications mobiles ces quinze dernières années, la gestion de la mobilité au niveau applicatif a pris une importance croissante, notamment avec le développement d'architectures télécoms de nouvelle génération NGN (*Next*

*Generation Network*). Les différents types de contraintes de mobilité ont été répertoriés, formalisés notamment dans une recommandation de l'ITU-T (Q.1706 [45]) qui semble partagée par l'ensemble de la littérature. Quatre contraintes de mobilité y sont définies.

- **Mobilité de terminal** (*terminal mobility*). Permettre à un terminal en mouvement de conserver l'accès au réseau. L'ITU-T insiste en fait ici sur la capacité du réseau à gérer le mouvement de ses terminaux (problématiques de localisation, d'identification, de routage, d'itinérance ou *roaming*, etc).
- **Mobilité de réseau** (*network mobility*). Permettre à un sous-réseau de se réorganiser afin de changer de point de rattachement à son super-réseau (typiquement Internet). Nous sommes encore dans des problématiques de réseau et plus précisément de routage.
- **Mobilité personnelle** (*personal mobility*). Permettre à un utilisateur de changer de terminal tout en conservant l'accès au réseau ainsi qu'à son profil et à ses services. Les problématiques restent au niveau architecture avec la capacité du réseau à identifier les utilisateurs et gérer les profils correspondants.
- **Mobilité de service** (*service mobility*). Enfin la mobilité de service qui reste encore et toujours au niveau réseau. L'ITU-T crée ici une nouvelle catégorie qui regroupe à la fois les contraintes de *mobilité de terminal* et de *mobilité personnelle*. Donc permettre à un utilisateur, quel que soit sa localisation ou le terminal qu'il utilise d'avoir accès à ses services personnalisés.

Ces contraintes sont basées sur des considérations architecturales pour la définition de la mobilité dans les NGN, l'ITU répond ainsi à la question : quels mécanismes un réseau, dans le sens « infrastructure de service », doit implémenter pour assurer une mobilité totale de ses services télécoms. Or nous nous intéressons dans cette étude à tous les services et ce du point de vue de l'utilisateur et non pas d'un point de vue purement architectural. La classification de l'ITU est particulièrement juste et le nombre de travaux qui la réutilisent en est la preuve. Cependant, pour réellement comprendre quels sont les enjeux et la vraie problématique de mobilité appliquée aux services, nous devons analyser l'impact de chacune de ces contraintes au niveau applicatif.

La mobilité de réseau étant comme son nom l'indique une problématique de bas niveau (d'après le modèle OSI), nous ne nous y intéresserons pas, d'autant que certains de ses aspects sont traités par la mobilité de terminal (cf. 2.1.1). Il ne reste donc plus que la mobilité de service, c'est à dire les mobilités de terminal et personnelle (cf. Figure 2.1). Nous analysons ces deux contraintes d'un point de vue service dans les sections suivantes sous le nom de *terminal en mouvement* (cf. 2.1.1) et *déplacement de l'utilisateur* (cf. 2.1.2). Enfin nous nous intéresserons à une contrainte supplémentaire, la *mobilité partielle de service* qui consiste en une mobilité interne au service lui-même, d'où son absence dans la classification ITU de plus bas niveau.

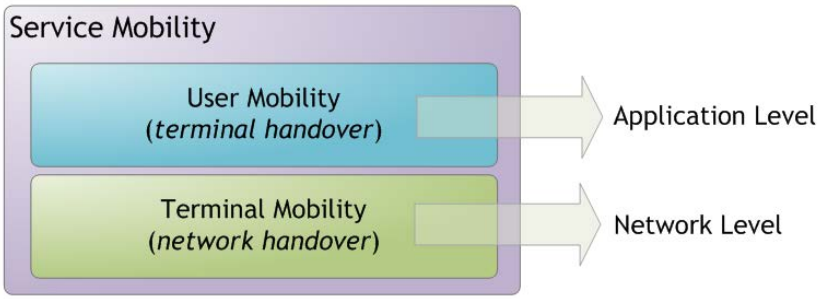



FIGURE 2.1 – Problématiques de mobilité.

### 2.1.1 Terminal en mouvement

Peut-être la contrainte de mobilité la plus commune : le terminal en mouvement. Lorsque vous passez un appel en voiture avec votre téléphone mobile (et votre kit main libre), sans le savoir votre communication audio peut être transférée plus de vingt fois... une vraie prouesse technique. C'est l'exemple même d'une mobilité de service réalisée avec succès car totalement transparente pour l'utilisateur. Ce type de contrainte intrinsèque à toute connectivité sans fil fait l'objet de nombreux travaux, on peut même dire que c'est l'effort principal des problématiques de recherche sur la mobilité.

Un terminal en déplacement, typiquement un téléphone mobile, n'est opérationnel que lorsqu'il est relié au réseau téléphonique dont il dépend. Pour être

plus exact, c'est le service de communication délivré via ce terminal qui requiert une connexion à l'infrastructure de l'opérateur téléphonique. Cette connectivité est réalisée grâce à diverses technologies de radiocommunication entre le terminal et les antennes-relais déployées par l'opérateur afin d'assurer une couverture maximale et donc une connectivité constante de ses clients. Toute la problématique consiste alors à maintenir la connexion entre le terminal et le réseau, chaque antenne ayant une portée limitée il est nécessaire de changer régulièrement de point d'accès pendant un déplacement, ce mécanisme de transfert s'appelle *handover* ou *handoff* [46].

Le *handover* est un mécanisme fondamental dans la télécommunication mobile et il se décline en deux sous-catégories : handover *horizontal* et *vertical* [45]. Le handover horizontal consiste en le passage d'un point d'accès à un autre via la même technologie radio, c'est ce qui arrive typiquement à votre téléphone mobile GSM lorsque vous vous déplacez. Mais l'émergence d'un grand nombre de nouvelles normes radio (UMTS, Wi-Fi, WiMAX, etc) a fait qu'un même terminal peut avoir simultanément le choix de connectivité entre plusieurs technologies d'accès. Le terminal se déplace alors « abstraitement » au niveau réseau (pas de mouvement géographique nécessaire), passant d'un point d'accès à un autre en changeant de technologie radio. C'est ce que l'on appelle « handover vertical ». Ce deuxième type de handover est plus difficile à mettre en œuvre, de part l'hétérogénéité des technologies et des mécanismes entrant en jeu, mais aussi des infrastructures pouvant être contrôlées par différents opérateurs. De nombreux travaux de recherche se sont intéressés à ces problèmes qui sont généralement traités par paire de technologies parmi les plus connues : GPRS, Wi-Fi, UMTS, WiMAX, ...

Que ce soit dans le cadre d'un handover horizontal ou vertical, un terminal en mouvement entraîne uniquement des conséquences de niveau réseau : interruption de la connectivité, modification de la bande passante ou du délai, etc. D'un point de vue applicatif, seuls les services connectés (cf. [45]) sont sensibles à ce type de contrainte.

### 2.1.2 Déplacement de l'utilisateur

Deuxième contrainte ayant un impact applicatif : le déplacement de l'utilisateur, et uniquement lui... En effet, si le terminal se déplace en même temps que son propriétaire, alors on revient au cas précédent où la seule mobilité est celle du terminal par rapport à son environnement (2.1.1). On suppose ici que c'est l'interface Homme/Machine qui change, l'utilisateur passe d'un terminal à un autre comme précédemment le terminal passait d'un point d'accès à un autre, on parle alors de *terminal handover*, par opposition au *(network) handover* que nous venons de voir.

Ce type de handover a un impact direct sur tous les services de l'utilisateur qui passe d'une interface, d'un environnement, à un autre a priori complètement différent. Cette contrainte de mobilité est finalement très peu étudiée alors qu'elle concerne l'ensemble des services, posant de nombreux problèmes notamment ceux de continuité et d'adaptabilité. Cependant, une nouvelle fois le problème a été approché d'un point de vue « réseau », ainsi en se basant sur les sessions (cf. 1.1.2) certains mécanismes ont rendu possible le transfert de services connectés simples (communication audio) d'un terminal à un autre. SIP par exemple permet de transférer un service de communication multimédia tout en l'adaptant à son nouvel environnement, cf. 2.3.2. Ce type de transfert est communément illustré par la redirection d'appels dans les standards téléphoniques.

Le transfert de sessions ne répond toutefois pas de manière satisfaisante au problème, un service connecté (ou non-connecté a fortiori) ne peut être réduit à une (ou des) session. Il est également intéressant de noter que d'un point de vue réseau, un terminal handover se traduit uniquement par un changement d'adressage (typiquement IP) de manière analogue à un network handover vertical (2.1.1). Ainsi une solution qui permettrait le transfert d'un service basique (constitué uniquement de sessions) entre réseaux hétérogènes pourrait dans une certaine mesure être utilisé pour réaliser un terminal handover.

### 2.1.3 Principe de transfert.

La contrainte de déplacement de l'utilisateur qui induit le mécanisme de *terminal handover* sera le type de mobilité le plus étudié durant ces travaux. Afin

d'assurer une compréhension uniforme, nous rappelons ici le principe de transfert et la terminologie employée.

Un *terminal handover* a pour principe le déplacement d'un service en cours, d'un *terminal origine* vers un *terminal destination*, cf. Figure 2.2. Le déclenchement du processus de mobilité est réalisé soit de manière automatique, soit de manière manuelle par l'utilisateur. Dans le cas manuel, on distingue deux types de déclenchement : le mode « *push* » lorsque l'utilisateur contrôle la mobilité depuis le terminal origine (et « exporte » le service vers sa destination) et le mode « *pull* » où il contrôle la mobilité depuis le terminal destination (et importe le service distant vers celui-ci).

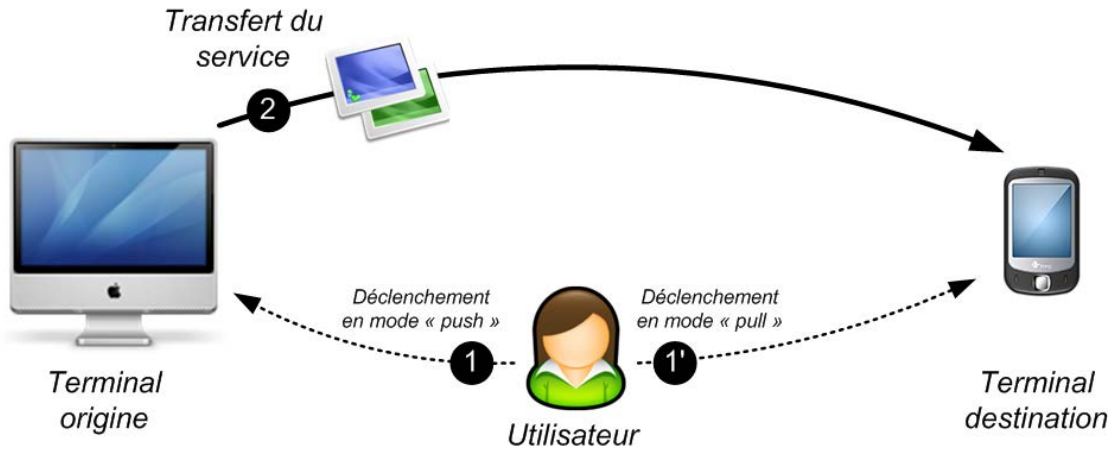



FIGURE 2.2 – Principe de transfert (*terminal handover*).

### 2.1.4 Mobilité partielle de service

Dernière contrainte de mobilité de niveau applicatif : la mobilité partielle de service. La notion de service est assez vague et dépend du domaine concerné tel que nous l'avons vu dans le première chapitre de cet état de l'art (1). Or il est parfois possible qu'un service résulte de la composition de plusieurs autres, plus simples. Dans ce cas la mobilité peut s'appliquer à ses différentes composantes afin de, par exemple, adapter la fonctionnalité globale à son environnement.

Les services connectés qui font l'objet de nombreux travaux notamment dans le domaine des télécommunications illustrent bien ce découpage, chacune des ses-

sions de ces services pouvant généralement être assimilée à un sous-service indépendant et donc mobile selon des contraintes terminal ou utilisateur (cf. 2.1.1 et 2.1.2). Par exemple, un service de visiophonie (télécommunication voix + vidéo) peut être vu comme la composition des services de communication audio et vidéo (correspondant aux deux sessions multimédias). Pour diverses raisons (coût, délai, encombrement, etc), il est théoriquement possible de séparer les composantes de ce service et de gérer leur mobilité de manière indépendante : le service audio pourrait par exemple changer de technologie d'accès tout en restant sur le même terminal tandis que le service vidéo serait transféré vers un autre périphérique. Cet « éclatement » du service en sous-services dont la mobilité est gérée indépendamment est appelé *session splitting*. Naturellement, le mécanisme inverse qui consiste à « rassembler » plusieurs fonctionnalités en un seul super-service est également réalisable et s'appelle *session merging*.

Les travaux réalisés sur cette contrainte de mobilité ne concernent aujourd'hui que les services basés sur des sessions et les approches existantes permettent uniquement le transfert de ces sessions, cf. [47, 48]. Or ce type de mobilité confère de grandes capacités d'adaptation aux super-services concernés, en effet le fractionnement des blocs fonctionnels permettent naturellement une adaptation plus fine à l'environnement.

## 2.2 Notion de continuité

Nous avons vu les concepts de service et de mobilité, nous arrivons donc à l'aspect « continuité » qui est le concept clé du sujet d'étude. À ce titre, la continuité aurait pu être traitée dans un chapitre à part entière, cependant la continuité n'est autre qu'une contrainte relative et inaliénable au concept général de mobilité. Ainsi dans cette section nous définirons le principe de continuité, puis nous présenterons ses deux aspects majeurs perçus par l'utilisateur.

### 2.2.1 Principe

La continuité est une contrainte supplémentaire à celle de mobilité. Pour être satisfaite, les processus mis en œuvre doivent assurer une fourniture continue de

la fonctionnalité en mobilité. Plus concrètement et du point de vue utilisateur, la continuité est l'assurance d'une mobilité transparente ; ainsi les aspects qui rendent la mobilité perceptible doivent être minimisés, optimisés à défaut d'être supprimés.

La mobilité de service qui est l'objet de cette étude ne peut être considérée sans la continuité, elle était d'ailleurs implicitement présente dans les définitions précédentes (cf. *handover*). En effet, l'utilisateur est en contact direct avec la couche applicative, et la mobilité des fonctionnalités offertes est directement et immédiatement ressentie. Une mobilité de service sans continuité reviendrait à offrir le même service via différents terminaux et ce sans cohérence... cela est déjà possible : je peux aujourd'hui établir une communication audio depuis un téléphone fixe, un ordinateur ou un terminal mobile. En d'autres termes, la continuité est la mobilité de l'*instance* d'un service (cf. 1.1.2).

La mobilité de service tel que l'entend l'ITU correspond à une problématique réseau qui consiste à assurer une continuité des profils utilisateur, ainsi les mêmes services *personnalisés* seront disponibles quelque soit le terminal utilisé. Comme nous nous intéressons dans ce mémoire à l'impact utilisateur, nous considérons la mobilité de service d'un point de vue applicatif qui ne peut réalisée sans continuité. Cette problématique de continuité de service présente deux aspects principaux perçus par l'utilisateur : les continuités temporelle et contextuelle (cf. Figure 2.3).

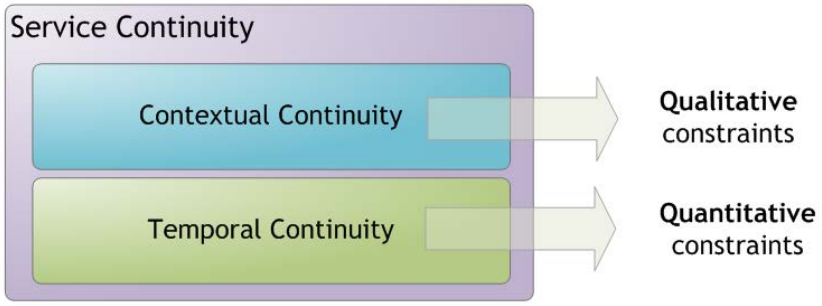



FIGURE 2.3 – Problématiques de continuité.

### 2.2.2 Continuité temporelle

La continuité temporelle impose des contraintes quantitatives aux mécanismes de mobilité. En effet, les travaux existants sur la mobilité de session, qui peuvent être vus comme une première approche de mobilité de service (cf. 2.3.2) cherchent à minimiser une valeur unique : le temps de transfert. La mobilité d'un service implique généralement une interruption liée au mécanisme de transfert (vers un nouveau réseau 2.1.1 ou terminal 2.1.2). La première des « anomalies » de continuité alors perçue par l'utilisateur est le délai d'indisponibilité de la fonctionnalité. Cette anomalie a un fort impact sur l'utilisateur car elle le prive temporairement de son service, nécessitant parfois une réinitialisation dans la cas de services connectés.

La continuité temporelle requiert ainsi des mécanismes spécifiques afin de minimiser, voire masquer, cette interruption inéluctable. Le protocole SIP par exemple, via la méthode REFER (cf. 2.3.2), implémente un mécanisme de synchronisation pour le transfert de sessions audio, le délai est ainsi minimisé et masqué par une superposition des flux audio provenant des terminaux origine et destination. Similairement, un network handover vertical peut être réalisé de manière quasi-transparente si le terminal est capable de gérer la connexion simultanée à plusieurs réseaux d'accès.

Le délai de transfert peut être optimisé afin d'offrir une continuité temporelle satisfaisante, c'est notamment le cas avec certains services connectés simples qui nécessite finalement une faible quantité de données à transférer. Mais qu'en est-il des services non-connectés ou possédant un contexte important ?

### 2.2.3 Continuité contextuelle

La continuité contextuelle ne peut être évaluée par une métrique telle que le temps, elle est appréciée qualitativement par l'utilisateur. Cette propriété abstraite quoique tout à fait réelle en fait un sujet peu étudié, cependant la continuité contextuelle est très intéressante car non seulement elle est universelle aux services mais elle est l'aspect le plus « visible » au niveau applicatif.

La continuité contextuelle a pour objectif de fournir à l'utilisateur des repères de contexte, d'expérience, afin qu'il ressente une continuité d'utilisation du service. Cela se traduit par une continuité de tout ce qui entoure le service : le contexte, les

propriétés, les profils, l'interface graphique, l'historique, etc. De plus, dans le cadre d'un terminal handover, cette continuité doit se réaliser dans un nouvel environnement potentiellement complètement différent de celui d'origine, une adaptation est alors nécessaire des points de vue matériel et logiciel. Pour un network handover, les mécanismes entrant en jeu étant généralement situés au niveau de la couche réseau, les environnements cible et destination sont quasiment identiques (excepté les nouvelles propriétés réseau : bande passante, délai, ...), la continuité contextuelle sera naturellement assurée.

## 2.3 Solutions existantes

Nous avons introduit et défini la problématique principale des travaux présentés dans ce mémoire, à savoir la mobilité de service et sa contrainte majeure : la continuité (désormais nommée par abus de langage « continuité de service »). Nous avons délibérément cherché les problématiques applicatives qui ont un impact direct sur l'*expérience utilisateur* tel que le handover de terminal. En effet, très peu de travaux existent dans ce domaine mais de nombreuses approches visant de près ou de loin des problématiques de mobilité apportent des solutions partielles mais fort intéressantes pour la suite de notre étude.

Nous avons étudié ces travaux et nous les avons classés selon quatre catégories d'approches : *architecturale*, *session*, *réseau* et enfin *applicative*. Pour chacune d'entre elles, nous donnerons le principe ainsi qu'un exemple puis nous discuterons de leurs points forts et de leurs faiblesses.

### 2.3.1 Approche architecturale

#### Principe.

L'approche architecturale est basée sur une infrastructure client-serveur complète qui, par sa structure, facilite la gestion de la mobilité. Une solution de ce type qui est certainement la plus ancienne approche de mobilité de service est l'affichage déporté (ou *display forwarding*). Les implémentations les plus connues étant le *X Window System* [49, 50] ou encore le *Virtual Network Computing* (VNC, protocole RFB [51, 52]). Bien que ces solutions n'aient pas été conçues dans le but d'assurer la continuité à proprement parler, elles présentent des propriétés intéressantes.

Dans cette approche, les applications sont hébergées par un terminal hôte dédié qui assure tous les traitements et que l'on appelle serveur. Les terminaux clients identifiés et autorisés peuvent accéder aux applications en se connectant au serveur. Le terminal client transfère alors les *inputs* (données en entrée : clavier, souris, ...) de l'utilisateur vers le serveur qui renvoie en retour l'*output* des applications (données en sortie : interface graphique, ...). Ainsi les terminaux clients sont indifféremment interchangeables dans la mesure où ils implémentent le protocole nécessaire à la connexion au serveur. L'utilisateur peut alors employer n'importe quel terminal pour accéder à ses services, son déplacement ne rompt pas la continuité tant qu'il conserve la connexion avec le serveur (cf. Figure 2.4). Il est à noter que le mécanisme d'affichage déporté est en fait lui-même un service distant tel que défini en 1.1.2.

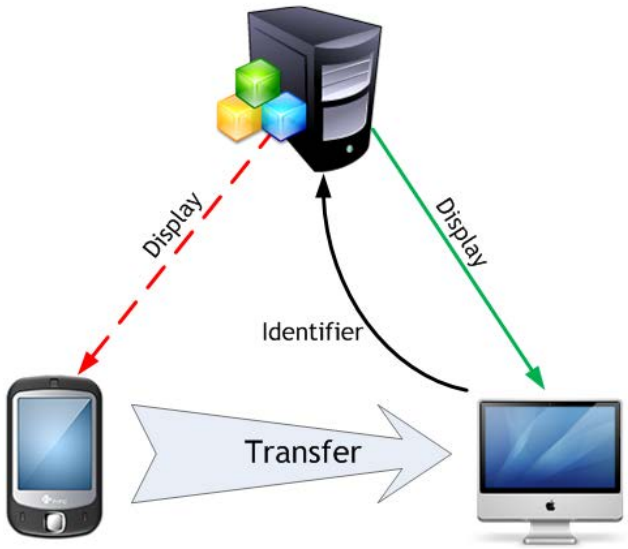



FIGURE 2.4 – Approche architecturale.

Avec l'essor des nouvelles technologies du Web « 2.0 », des solutions similaires sont apparues directement dans le navigateur Internet. Celui-ci joue alors le rôle de client léger qui permet aux utilisateurs d'accéder à des applications hébergées à

distance tel que *Google Docs* [53] ou encore des bureaux et des systèmes d'exploitation Web complets (*Web OS*) tels que `Windows4all.com` ou encore `GlideOS.com`.

**Avantages.**

Bien que cette approche soit la plus ancienne, elle est peut être la plus efficace du point de vue de sa continuité temporelle et contextuelle. Essayons d'en dégager les propriétés essentielles.

Principal avantage, le délai de transfert d'un service d'un terminal à un autre est négligeable. En effet, seules les données de sortie de l'application doivent être redirigées, ce qui consiste généralement en l'affichage uniquement, quelque soit le service concerné. La redirection de flux de données qui permet à l'utilisateur d'avoir un affichage actualisé se fait en quelques secondes, seule la phase d'authentification peut ralentir légèrement le processus surtout lors d'une première connexion (échange de clés, mot de passe ou adresse du serveur à entrer, etc) mais ces étapes ne font pas partie du transfert proprement dit.

De plus, le service n'est jamais arrêté, l'interruption ressentie par l'utilisateur est uniquement causée par son changement de terminal et les diverses étapes d'initialisation du transfert. Le terminal client quant à lui est très léger, il joue le rôle d'interface et doit seulement implémenter un protocole spécifique lui permettant de communiquer avec le serveur. N'hébergeant aucune donnée (de manière permanente), ni n'exécutant de processus, il n'existe aucun problème de compatibilité entre les clients.

Enfin, le mécanisme de transfert est indépendant du type de service et des capacités du terminal. Les mécanismes d'adaptation du service quant à eux sont inutiles car l'environnement du service ne change jamais.

**Inconvénients.**

Cette approche est très efficace, on en viendrait à oublier que ce n'est pas à l'origine une solution de continuité de service. Cependant il faut noter quelques contraintes.

Tous les services de l'utilisateur requièrent une connexion au serveur pour être accessibles. Si une coupure intervient ou si le serveur est momentanément indisponible, aucun service ne peut être délivré, le terminal client n'étant qu'une interface.

Certaines solutions Web peuvent proposer néanmoins des modes hors-ligne, basés sur des mécanismes de synchronisation et de cache qui permettent à un utilisateur de continuer à utiliser un service temporairement malgré un problème de connectivité, cf. *Google Gears*. La gestion de mode hors-ligne nécessite cependant un logiciel spécifique (plug-in navigateur par exemple) qui réduit considérablement les avantages de l'approche architecturale, l'hébergement du service redevenant local.

Cette approche pose également un problème simple et d'actualité : la protection de la confidentialité et de la vie privée. L'ensemble des données de l'utilisateur sont externalisées au niveau du serveur qui par principe gère un grand nombre de clients. La conservation et la confidentialité des données ne peut être garantie, l'utilisateur doit faire confiance au tiers qui gère le serveur, une condition parfois inacceptable dans un environnement de travail.

Le principe même de l'efficacité de la mobilité de service dans l'approche architecturale est justement le non-transfert des services. Cependant, cela implique que l'application qui instancie le service ne changera jamais, quelque soit le terminal utilisé. En fait celui-ci ne joue plus aucun rôle, il n'est qu'une interface, l'utilisateur ne pourra pas bénéficier des propriétés intrinsèques du nouveau terminal qui peut être une motivation du transfert.

Enfin, l'approche architecturale n'offre aucun mécanisme de gestion des terminaux de l'utilisateur, ce dernier doit savoir a priori quel serveur héberge un service en particulier qu'il souhaiterait utiliser depuis son terminal client. Il doit ensuite utiliser son adresse (IP ou nom d'hôte routable) depuis le client pour contacter le serveur, le transfert ne peut être initié que par le terminal destination.

### 2.3.2 Approche session

#### Principe.

L'approche session apporte une solution partielle au problème de continuité. Nous avons vu précédemment ce qu'était une session (cf. 1.1.2), nous savons qu'elle consiste en un flux de données entre deux terminaux dans le cas de services distants. Les fournisseurs de services connectés, dépendant intrinsèquement de ces sessions se sont penchés sur les problématiques de mobilité et donc de continuité de session. Les services en question étant principalement des applications multimé-

dias ou de communication, un effort de recherche conséquent des grands groupes de standardisation tel que l'IETF a permis l'introduction de mécanismes spécifiques particulièrement efficaces car directement intégrés dans les protocoles concernés (couche contrôle de la session).

La continuité de session a fait l'objet de nombreux travaux, l'un des plus aboutis car intégré à la norme est le mécanisme REFER du protocole SIP. REFER [54] est un message de contrôle qui permet le transfert de sessions multimédias initiées par SIP [55]. Cette extension au protocole de base est également accompagnée de champs additionnels tel que *Replaces* [56] ou *Referred-by* [57] qui assurent une synchronisation des transferts pour une continuité optimale. Lorsque des terminaux SIP implémentent ces extensions, il devient possible pour un utilisateur de transférer une communication multimédia entre eux de manière transparente pour les interlocuteurs.

Le mécanisme de transfert est basé sur un principe de substitution de session. Une fois le transfert déclenché (par le réseau ou l'utilisateur), une seconde session est créée entre le correspondant de la communication et le terminal destination. Une fois la nouvelle session établie, l'ancienne session est déconnectée et la communication peut continuer entre le correspondant et l'utilisateur via son nouveau terminal (cf. Figure 2.5). En maintenant deux sessions de données pendant la phase critique de transfert, la communication n'est pas interrompue.

D'autres travaux concernant des protocoles multimédias tels que RTP (*Real-time Transport Protocol* protocole de transport [13]) ou encore RTSP (*Real-Time Streaming Protocol* protocole de streaming vidéo [58]) proposent des mécanismes similaires de continuité de session, cf. [59, 60, 61].

**Avantages.**

Étant intégré au protocole, à la couche contrôle de la session, le mécanisme de transfert est particulièrement efficace, les problèmes de synchronisation, d'interopérabilité et d'authentification étant gérés nativement. Ainsi la continuité temporelle est optimisée, l'interruption de service qui est ici bien réelle à l'inverse de l'approche architecturale, est imperceptible par l'utilisateur.

L'adaptation est également un point fort de cette approche, notamment dans la cas de substitution de session (comme avec SIP). En effet, l'initialisation d'une

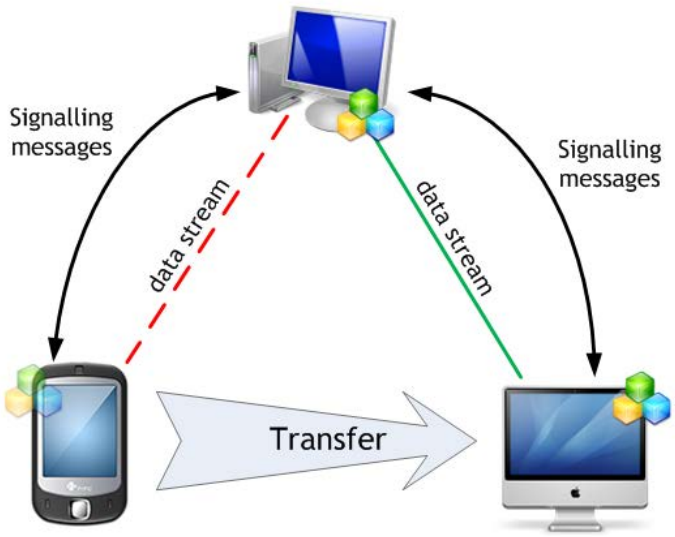



FIGURE 2.5 – Approche session.

nouvelle session permet au terminal d'en renégocier les paramètres (typiquement l'encodage utilisé), le service peut ainsi s'adapter au nouvel environnement : capacités du terminal et du réseau sous-jacent, etc.

**Inconvénients.**

Bien que la session puisse être adaptée aux propriétés du nouvel environnement, généralement résultant d'une amélioration ou d'une dégradation de la qualité des données transmises, la continuité contextuelle reste faible. En effet, si la continuité des sessions est assurée, le contexte complet du service n'est pas géré par les mécanismes de niveau protocolaire. Un service de communication basé sur SIP est généralement plus qu'une (ou des) session, d'autres éléments existent tels qu'une liste de contacts, un historique d'appel, des préférences d'affichage et de notification, ... autant de repères contextuels qui seront perdus une fois le service transféré. Conséquence directe, les services locaux qui ne possèdent aucune session seront par nature ignorés par ce type d'approche.

Cette approche impose également une contrainte applicative forte au niveau de l'utilisateur. La mobilité de session requiert une instanciation spécifique des services, en d'autres termes une communication voix par exemple ne pourra être transférée qu'entre terminaux implémentant ce service via la même application (du moins le même protocole). Cela paraît évident car le protocole est partie intégrante du service dans ce cas mais cela pose un vrai problème dans la mesure où la mobilité ne peut ici être gérée au niveau service mais uniquement au niveau applicatif/instance. Enfin, le protocole utilisé par les terminaux origine et destination doivent être strictement identiques, implémentant l'ensemble des normes et des extensions requises.

Enfin, comme pour l'approche architecturale, les mécanismes existants ne traitent que la phase de transfert. Or la mobilité de service ne se résume pas uniquement au transfert, l'identification des terminaux par exemple est nécessaire tout comme le déclenchement du processus. Ici l'identification se fait vraisemblablement via l'adresse de contact du terminal (tel qu'un SIP URI ou un numéro de téléphone) que l'utilisateur doit connaître. De plus, le déclenchement des mécanismes de transfert ne peuvent être effectués que depuis le terminal origine (mode *push* cf. 2.1.3) ou par le réseau (mode 3PCC pour SIP [62]), il n'existe pas de mode *pull* car les services (ici les sessions) ne sont connus que des terminaux qui les hébergent.

L'approche sous-entend également que le terminal destination est connecté et prêt à recevoir une requête de connexion entrante via le protocole en question... ce qui est concevable pour un terminal dédié à ce service (téléphone mobile SIP par exemple) mais déjà moins dans le cas d'un terminal multitâches tel qu'un ordinateur de bureau.

### 2.3.3 Approche réseau

#### Principe

L'approche session de la continuité est basée sur des mécanismes applicatifs, d'autres travaux de plus bas niveau (relatif au modèle OSI) s'intéressent au même problème de mobilité mais au niveau réseau. Le protocole Mobile IP [63] est certainement la solution la plus complète dans ce domaine. Ce protocole réseau standardisé par l'IETF assure la continuité d'un transfert de données en garantissant une adresse IP fixe assignée au terminal et ce quelque soit sa mobilité. Les *network*

*handovers* sont ainsi réalisés de manière totalement transparente pour les couches supérieures (au niveau applicatif notamment). Les mécanismes mis en œuvre ont fait l'objet de nombreux travaux de recherche, tels que [64] ou [65].

Le principe de base repose sur des composants spécifiques disséminés dans l'infrastructure réseau qui assurent l'acheminement des données vers le terminal en mobilité (cf. Figure 2.6). Avec Mobile IP, un *home agent* situé dans le réseau d'origine du terminal mobile collecte les données destinées à ce dernier pendant son absence, ces données sont ensuite redirigées vers la position courante du terminal mobile. Hors de son réseau d'origine, le terminal mobile se connecte aux *foreign agents* afin d'acquérir une adresse locale valide et notifier le *home agent* de sa nouvelle position. Le terminal mobile reste ainsi disponible via une adresse unique, le *home agent* étant chargé de faire suivre les flux de données.

D'autres solutions ont également été proposées pour gérer la continuité de session à bas niveau (couche transport en l'occurrence). Ainsi on peut mentionner le protocole SCTP (*Stream Control Transmission Protocol* [66]) et l'une de ses variantes mobile-SCTP [67] ou encore des études plus rares portant sur TCP [68]. À noter enfin les travaux de recherche présentés dans la thèse [69] particulièrement intéressants pour une lecture plus approfondie dans ce domaine.

**Avantages.**

Le point fort de l'approche réseau (qui se révèle être également son point faible) est son indépendance avec la couche applicative. En effet, les problématiques de connectivité sont entièrement gérées au niveau réseau de manière totalement transparente pour les services qui conservent une adresse IP fixe. Les handovers sont transparents et la continuité des sessions est garantie.

Cette indépendance entre les différentes couches entraîne une autre propriété importante : le mécanisme de mobilité n'est pas intégré au protocole. La continuité étant assurée uniquement au niveau réseau, il n'y a pas comme dans l'approche précédente une contrainte applicative forte au niveau utilisateur. En d'autres termes, si pour l'approche session l'instance de service doit être de même type dans tous les terminaux, ici les sessions peuvent être transférées entre applications différentes. Cependant cela reste vrai d'un point de vue uniquement théorique car en pratique la couche contrôle de la session correspond généralement à un protocole spécifique.

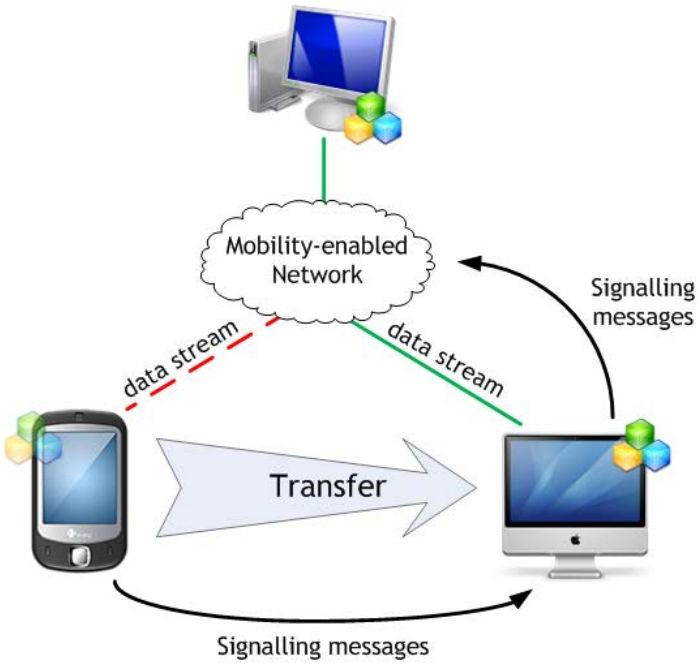



FIGURE 2.6 – Approche réseau.

Enfin, étant donné que les mécanismes de mobilité liés à cette approche restent au niveau réseau, le transfert consiste simplement à une redirection des flux de données. L'ensemble des mécanismes applicatifs présents dans l'approche session tels que la synchronisation, l'authentification, l'interopérabilité ou l'adaptation ne sont pas implémentés ici (laissé à la discrétion des couches plus hautes), le traitement est donc simplifié, voire plus rapide.

**Inconvénients.**

Si la continuité temporelle est assurée, la continuité contextuelle n'est simplement pas considérée, plutôt logique étant donné que le contexte est propre au niveau applicatif. L'approche réseau peut difficilement convenir à la continuité de service car elle se désintéresse intrinsèquement des problématiques applicatives, celles ayant justement un impact fort sur l'utilisateur. C'est la raison pour laquelle nous ne nous attarderons pas sur ces approches de bas niveau, cependant de telles

solutions peuvent se révéler complémentaires dans la gestion de la mobilité de session.

Encore une fois, cette approche ne prend pas en compte les services locaux ni les services qui se composent d'autre chose que de sessions. On peut également constater que comme dans les approches précédentes et peut-être même encore plus dans celle-ci, seule la réalisation du transfert est considérée, il n'existe en effet aucun mécanisme de déclenchement ou de découverte des services (ici des sessions) ou des terminaux ; l'adressage (ici de niveau réseau) est également problématique.

Enfin, cette approche nécessite une infrastructure réseau adaptée, des composants doivent y être déployés afin d'implémenter les mécanismes de mobilité. Ceci est une contrainte architecturale considérable, une telle solution ne dépend pas seulement des terminaux et des utilisateurs mais également de l'environnement.

### 2.3.4 Approche applicative

#### Principe.

Quatrième et dernière approche, l'approche applicative, plus encline a priori à gérer des problématiques de haut niveau. Les solutions applicatives, comme celles architecturales sont originellement prévues à un dessein autre que la continuité de service, cependant encore une fois l'efficacité des mécanismes utilisés mérite une attention particulière.

Cette approche est basée sur le principe de virtualisation [70, 71]. Pour faire simple, la virtualisation consiste à émuler un environnement informatique matériel et logiciel dans un système hôte. Ce processus permet de créer et d'isoler des environnements opérationnels complets, les possibilités de manipulation sont alors innombrables. Une opération classique consiste à créer une image du système émulé, appelé aussi *checkpointing*. Cette image contient l'ensemble des données de l'environnement émulé (mémoire, registre, CPU, etc), il devient alors aisé, depuis le système hôte de transférer cette image vers un autre hôte et de reprendre l'exécution à partir de l'image dans un nouvel environnement émulé identique (cf. Figure 2.7).

Des travaux de recherche se sont appuyés sur ces mécanismes afin de proposer des solutions de *mobilité d'applications*. La principale motivation de ces travaux est d'assurer une persistance applicative, en effet certaines applications (typiquement

des serveurs) nécessitent d'être opérationnelles en permanence. Or pour diverses raisons : maintenance du matériel, mise à jour logicielle, incident d'alimentation, ces applications doivent parfois être déplacées. Or la perte des données « volatiles » du système en cours d'exécution ainsi que les temps importants d'initialisation en cas de redémarrage de l'environnement et des services associés ne sont pas acceptables. La problématique de mobilité est ici le transfert d'applications entre plusieurs hôtes afin de conserver l'environnement d'exécution. Les travaux [72] notamment présentent des solutions intéressantes de mobilité d'applications basées sur la virtualisation et le *checkpointing* pour le système d'exploitation Linux.

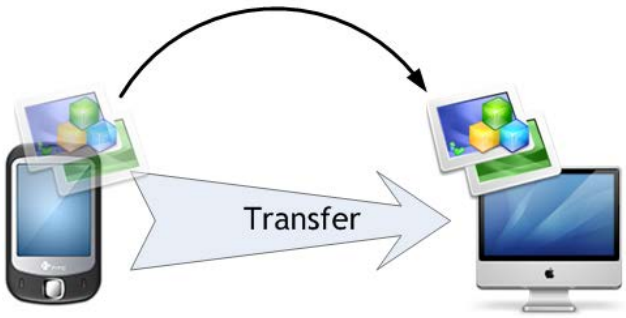



FIGURE 2.7 – Approche applicative.

**Avantages.**

Le point fort de cette approche est une continuité contextuelle parfaite : l'environnement applicatif est transféré dans son ensemble, la perception finale de l'utilisateur est donc identique à l'originale. On pourrait d'ailleurs parler plus d'une « copie » que d'un transfert, l'instance de service étant dupliquée contrairement à l'approche architecturale où l'instance est unique.

Aucune adaptation n'est nécessaire, le système étant transféré avec l'application, la seule contrainte est que le terminal destination soit en mesure de supporter l'émulation de l'image reçue, autant d'un point de vue matériel que logiciel. Du coup, comme pour l'approche architecturale, les capacités du terminal destination

sont ignorées et le transfert n'apportera à l'utilisateur aucun bénéfice particulier si ce n'est la délocalisation de l'application.

**Inconvénients.**

La continuité contextuelle ainsi offerte a un prix : un grand nombre de données à transférer. En effet, si une application est volumineuse, la transférer avec le système sous-jacent est difficilement réalisable. La quantité de données à transférer pourrait se compter en plusieurs dizaines de gigaoctets au regard des systèmes d'exploitation actuels, un tel transfert n'est pas envisageable via un réseau si l'on souhaite garantir un minimum de continuité temporelle. Le principal problème est le manque de granularité des informations à transférer, certains travaux cherchent à isoler des sous-ensembles, plus proches de l'application mais cela implique un environnement strictement identique sur les terminaux origine et destination.

Les contraintes sont également très fortes au niveau du terminal qui doit non seulement supporter l'application de virtualisation chargée d'émuler l'image reçue mais également exécuter l'environnement et l'ensemble de ses programmes. Seuls certains terminaux définis a priori et préparés pourront être la destination d'un tel transfert, aucune adaptation n'étant possible.

Enfin, l'environnement est figé puis dupliqué sur un autre système hôte lors de chaque transfert, on peut imaginer que des problèmes de synchronisation apparaissent si l'image de l'environnement utilisée n'est pas la plus récente. En effet, comment savoir si le terminal actuel possède le dernier état de l'application... n'existant de mécanisme de synchronisation il faudrait supprimer les images systèmes une fois celles-ci transférées. Ce type d'approche ne peut convenir que pour des transferts peu fréquents et limités : sauvegarde pour maintenance par exemple.

## 2.4 Conclusion

Dans ce chapitre, nous avons introduit le concept de mobilité, les différents types de contraintes ainsi que la notion de continuité, le tout d'un point de vue applicatif correspondant à l'orientation de nos travaux de recherche. Nous nous sommes ensuite intéressés à l'état de l'art concernant la problématique de continuité de service. Nous avons présenté différentes approches qui proposent des so-

lutions spécifiques considérant les services à des niveaux d'abstraction différents, architecture, application, session et réseau. Vous pourrez également trouver des travaux de recherche particulièrement intéressants qui échappent à ce classement tels que [73, 74, 75, 76] ainsi que la thèse [77].

Cependant aucune approche n'est pleinement satisfaisante du point de vue de l'utilisateur et de son expérience d'usage : facilité d'utilisation, adaptation à l'environnement, liberté de mobilité, continuités temporelle et contextuelle...

Les approches session 2.3.2 et réseau 2.3.3 implémentent des mécanismes très efficaces, optimisés et standardisés qui assurent une continuité temporelle remarquable. Néanmoins, décorrélées des problématiques applicatives, elles ne peuvent gérer correctement la mobilité des services plus complexes qu'un simple ensemble de sessions.

L'approche applicative 2.3.4 possède le défaut inverse, si la continuité contextuelle est assurée par la gestion du système dans sa totalité (et tous ses services avec), l'utilisateur se retrouve pénalisé par une continuité temporelle irréalisable.

L'approche architecturale 2.3.1 est de loin celle qui apporte les meilleures réponses. Les mécanismes présents offrent une continuité à la fois temporelle et contextuelle. Cependant, la solution architecturale, au-delà de ne pas être conçue pour gérer la mobilité de service, présente un problème de taille : elle n'est pas adaptée aux usages actuels. En effet, le temps du terminal $X$ est révolu, l'utilisateur vit aujourd'hui au milieu d'une sphère électronique riche, hétérogène, il possède de nombreux terminaux relativement performants, chacun possédant des caractéristiques propres qui en feront l'interface adéquate au moment opportun (cf. *Introduction Générale*). Les continuités temporelle et contextuelle doivent être assurées par des mécanismes qui respectent et améliorent les usages de l'utilisateur, en l'occurrence ils doivent exploiter l'hétérogénéité de l'environnement en permettant aux services de s'y adapter de manière transparente pour l'utilisateur.

Enfin, l'ensemble des approches présentées qui couvrent l'état de l'art existant dans le domaine de la continuité de service présentent un point commun : elles traitent exclusivement les problématiques liées à la phase de transfert du service. Or la problématique de continuité va bien au-delà du simple transfert, différents processus entrent en jeu avant (identification) et après (adaptation). Il s'avère

qu'aucune des solutions étudiées ne s'y intéresse et que même les mécanismes de déclenchement du transfert ne sont pas prévus.

# Deuxième partie

# Contributions

# Chapitre 3

# Continuité des services de télécommunications dans l'IMS

## 3.1 Motivation

On ne peut aujourd'hui étudier les services de télécommunications sans considérer l'IP Multimedia Subsystem (IMS cf. 1.2.2). Cette architecture nouvelle génération a donc été le point de départ de notre étude sur la continuité et ce pour plusieurs raisons : une orientation « télécoms » qui correspondait à l'angle d'attaque choisi, une infrastructure complète, standardisée et prometteuse s'imposant comme la nouvelle architecture de référence et enfin la disponibilité de plateformes expérimentales IMS permettant d'évaluer nos contributions au sein même de nos laboratoires à TELECOM SudParis et Alcatel-Lucent Bell Labs France.

L'infrastructure IMS est le cœur fonctionnel de la téléphonie mobile nouvelle génération (3G), il comporte ainsi un grand nombre de mécanismes participant à la gestion de la mobilité : routage, itinérance (*roaming*), profils de service, identification, etc. De plus, l'un des objectifs majeurs étant d'assurer la convergence des réseaux, l'interopérabilité des différentes technologies d'accès est réalisée autour d'un noyau IP commun, l'IMS, sur lequel repose les services. La couche accès étant dissociée de la couche service dans ce modèle horizontal, un plus grand nombre de terminaux et donc de clients potentiels apparaissent ; les services doivent alors

être délivrés de manière cohérente entre ces appareils électroniques de plus en plus nombreux, puissants mais surtout aux capacités hétérogènes.

Avec plus de réseaux d'accès, plus de terminaux et plus de clients, le besoin en mobilité se renforce. Si les mécanismes IMS basés sur le protocole SIP (cf. 2.3.2) assurent les différents types de mobilité tels que définis par l'ITU (cf. 2.1), la mobilité de service au sens propre, qui inclut la contrainte de continuité, n'est pas assurée. Les problématiques classiques de continuité : network et terminal handover, ne sont pas prises en compte.

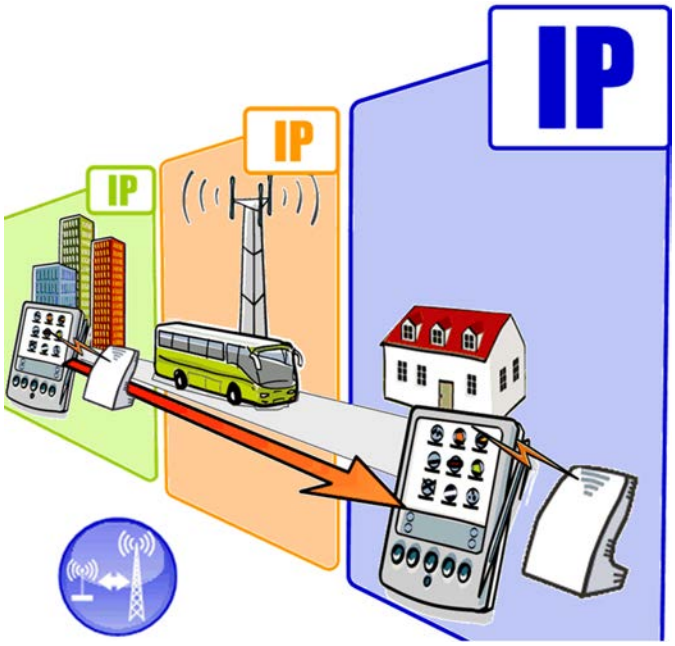
Le network handover est partiellement réalisé uniquement. Si les handovers horizontaux sont gérés de manière transparente pour l'IMS au niveau de l'accès, les handovers verticaux nécessitant un changement d'adresse ont pour incidence d'interrompre les sessions de communications. Or l'IMS, reposant sur SIP, ne délivre intrinsèquement que des services basés sur des sessions. Ce type de continuité requiert des mécanismes spécifiques qui font actuellement défaut dans l'IMS.

Le terminal handover, quant à lui, n'est simplement pas réalisable en l'état actuel de l'infrastructure. Le protocole SIP peut théoriquement effectuer un transfert de sessions entre plusieurs terminaux, mais d'un point de vue architectural l'IMS n'implémente pas les mécanismes requis pour mettre en œuvre ces solutions.

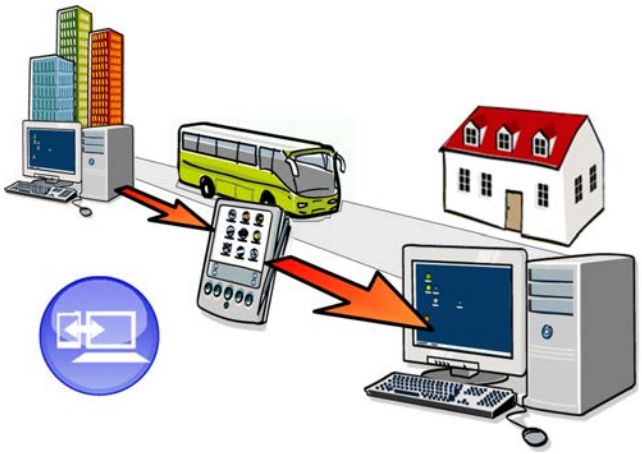
Face à un besoin croissant en mobilité et une infrastructure de services 3G telle que l'IMS qui ne peut assurer une continuité et donc une expérience utilisateur satisfaisante, nous essayons de répondre à cette problématique et ses deux principales contraintes : le network handover (mobilité de terminal) et le terminal handover (mobilité d'utilisateur). L'objectif est de permettre à un utilisateur de conserver ses services en mobilité lors de la transition entre différents milieux : au travail, à la maison, dans un lieu public... transition qui se traduit généralement par divers transferts (inter- terminaux et technologies cf. Figure 3.1). Nous étudions ces différents scénarios dans le cadre de l'IMS, donc d'un point de vue applicatif, et proposons des solutions compatibles avec les spécifications actuelles.

## 3.2 Problématiques et approche

L'IMS est une infrastructure fortement standardisée par de nombreux acteurs de la couche réseau à la couche applicative, assurant ainsi son interopérabilité,




(a) Network handover.



(b) Terminal handover.

FIGURE 3.1 – Scénarios de continuité.

son point fort. Afin d'offrir une solution applicable sans redéfinition de la norme, nous avons adopté une approche applicative qui essaie de résoudre les problèmes de continuité « d'en haut », via un serveur d'application (AS). Les AS sont des composants situés dans la couche service, contrôlés par l'IMS et qui offrent des fonctionnalités additionnelles (de celles de base tel que la communication audio) aux clients. Nous avons ainsi défini une application IMS supplémentaire, le *Session Continuity AS* (SCAS), dont le rôle est d'offrir la fonction de continuité de service. Cette nouvelle fonction est manipulée par l'IMS comme n'importe quel service et cible uniquement les services de télécommunications gérés nativement via SIP par l'infrastructure.

### 3.2.1 Network handover

Comme précisé en section 2.1.1, seul le cas du network handover *vertical* requiert un comportement applicatif spécifique pour assurer la continuité de service, le reste de cette section sera dédiée aux problématiques propres à ce cas précis dans la définition du *Session Continuity AS*.

#### Déclenchement.

Comment déclencher un transfert entre différents réseaux ? Les motivations d'un changement de technologie d'accès pendant la fourniture d'un service sont multiples : coût, encombrement du réseau, perte de signal, qualité de service requise, ... De manière générale, ces motivations résultent d'une décision automatique, basée sur des règles, et effectuée au niveau du terminal ou de l'infrastructure selon des critères réseau. L'utilisateur est rarement l'initiateur d'un tel transfert, alors la question est moins de savoir comment mais quand effectuer ce transfert ?

Selon notre approche AS et avec un terminal standard, la décision ne peut s'effectuer qu'au niveau infrastructure et ce d'après les messages de signalisation SIP qui transitent par le cœur IMS. Un changement de réseau d'accès pourra alors se détecter de deux manières en fonction des capacités du terminal et de la couverture des technologies réseaux. Soit le terminal peut se connecter simultanément aux deux réseaux (on parle de *hard handover*), soit il devra se déconnecter au réseau d'origine avant d'initier la seconde connexion (on parle alors de *soft handover*) cf. [78]. On peut intuitivement supposer qu'une déconnexion préalable sera moins

optimale qu'une connexion simultanée d'un point de vue continuité (temporelle) du service.

Pour pouvoir se connecter simultanément à deux accès, deux conditions doivent être remplies au moment du transfert : le terminal doit être couvert par les deux technologies (cf. Figure 3.2) et il doit gérer les deux interfaces simultanément (au niveau réseau, propriété appelée *multihoming* [79] mais aussi au niveau radio, propriété appelée *dual mode*). Si les deux conditions sont remplies, le terminal restera connecté en continu avec l'infrastructure (via l'une ou les deux interfaces), un processus de transfert sans interruption est alors envisageable, sinon le terminal passera obligatoirement par une phase déconnectée (cf. Figure 3.3), on pourra alors seulement offrir une solution de secours ou reconnexion.

***Soft handover.***

Le *soft handover* est un transfert sans interruption réalisable si le terminal est connecté simultanément à l'IMS via deux interfaces différentes. Il est alors identifié doublement au niveau de l'infrastructure qui le voit comme deux entités distinctes, chacune possédant une adresse réseau différente. La problématique de network handover revient ainsi à celle de terminal handover, en transférant le service d'une adresse réseau à une autre le mécanisme de continuité réalisera en fait un changement d'interface sur le même terminal. Les détails du mécanisme en question sera présenté dans la section 3.3.3.

Il est à noter que l'infrastructure IMS doit supporter l'enregistrement multiple des clients. En effet, l'adresse SIP-IMS (appelée SIP URI) doit pouvoir être associée à de multiples adresses réseaux. Cette fonctionnalité aussi appelée *multi-registration* incluse dans la norme est parfois non supportées par les solutions IMS. Le cas échéant, le terminal devra utiliser un SIP URI différent pour chaque interface et non par terminal, le mécanisme de transfert sera lui indique.

***Hard handover.***

Lorsque la déconnexion est inévitable, une solution alternative de (relative) continuité consiste à rétablir le service en initiant une nouvelle connexion. En enregistrant les messages de signalisation, le SCAS peut sauvegarder les différentes caractéristiques des services de communications établis dans l'IMS. L'interruption

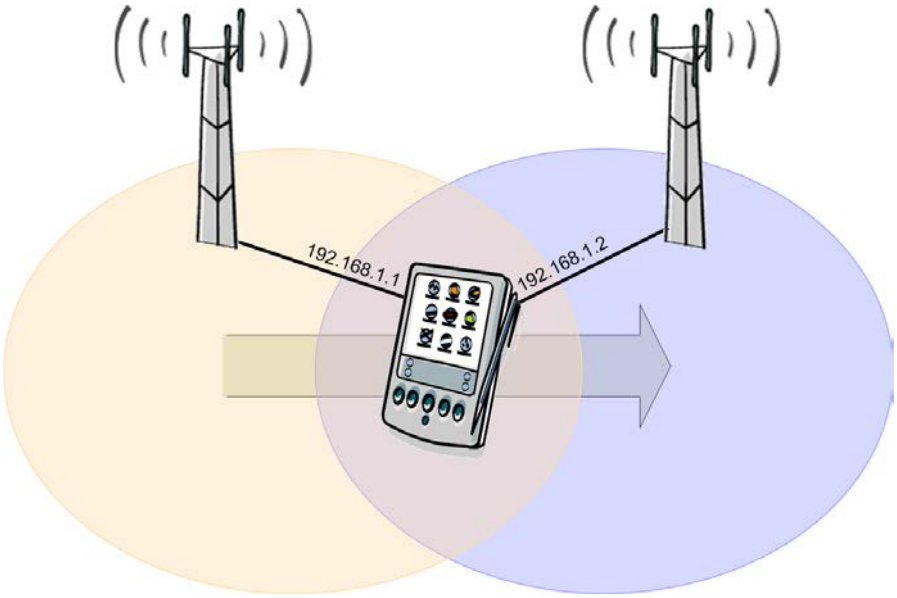



FIGURE 3.2 – *Soft handover.*

causée par le changement de réseau d'un terminal va terminer la (ou les) session correspondante de manière brutale, la laissant dans un état non-terminal. En effet, si la session n'est pas terminée « proprement » par un message de type BYE [12], le SCAS pourra déterminer qu'un incident de connectivité s'est produit. La reconnexion du terminal à l'IMS une fois le changement d'interface effectué, le SCAS possèdera suffisamment d'informations pour rétablir la communication en établissant une nouvelle session entre les interlocuteurs.

Cette solution peut être implémentée de deux manières différentes, soit via un contrôle de la session par une tierce partie (3PCC [62]) soit via le mécanisme SIP REFER (cf. [54]). Quel que soit la méthode choisie, il est nécessaire de respecter la logique de paiement de l'appel initialement établi. En effet, il doit y avoir une cohérence entre les services mis en œuvre (et donc facturés) dans l'appel initial interrompu et la communication nouvellement établie qui n'est que la continuité de ce premier. Par exemple, si $A$ appelle $B$ puis est interrompu, après reconnexion $A$ doit être facturé (et non $B$) comme pour un unique appel et conformément à la durée réelle de connexion.

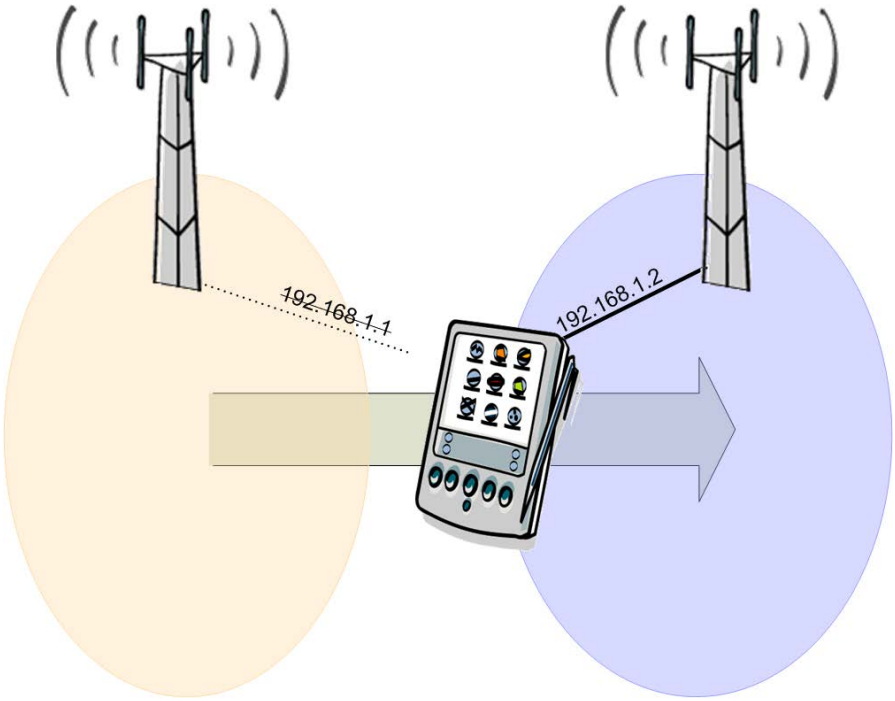



FIGURE 3.3 – *Hard handover.*

### 3.2.2 Terminal handover

Les mécanismes de terminal handover sont absents dans l'IMS bien que le protocole sous-jacent (SIP) comporte des fonctionnalités qui rendent cette mobilité possible. Nous proposons une solution conforme aux spécifications IMS qui repose sur ces fonctions. Tout d'abord intéressons-nous aux différentes problématiques qui interviennent avant et après le transfert proprement dit.

#### Déclenchement.

Contrairement au network handover, le changement de terminal est généralement déclenché manuellement. Bien que des critères réseau ou terminal puissent avoir un sens dans un terminal handover (batterie faible par exemple), changer automatiquement l'interface du service est particulièrement intrusif et potentiellement gênant pour l'utilisateur. Ainsi nous considérons le cas le plus vraisemblable, le déclenchement du transfert initié par l'utilisateur.

Une fois le transfert décidé, un grand nombre de choix se présentent à l'utilisateur : quel service transférer ? depuis quel terminal ? vers quel autre ? Un besoin fort d'identification apparaît alors, comment désigner (description et identification) un service, un terminal, un utilisateur. Ces éléments sont connus par l'IMS mais ces données destinées à un tout autre dessein ne sont pas formatées pour l'Homme... Comment choisir entre les sessions `784A5E@ims.fr?id=3263` et `D45E78@ims.fr?id=9652` ?

L'objectif est de définir une solution simple d'utilisation adaptée à un environnement télécom où les interactions entre l'utilisateur et un terminal généralement mobile sont relativement faibles. Nous proposons une approche mixte qui automatise et simplifie au maximum les mécanismes de déclenchement tout en laissant l'utilisateur maître du transfert. Les mécanismes d'identification et de désignation sont ainsi masqués. Une implémentation de la solution est détaillée en section 3.3, celle-ci permet des transferts de type « pull » et « push » (cf. 2.1.3). Dans tous les cas nous étudions le transfert entre deux terminaux du même utilisateur et non un transfert d'appel qui serait néanmoins similaire d'un point de vue technique.

## 3.3 Implémentation

Comme indiqué en introduction de ce chapitre, nous avons pu mener nos recherches sur deux plateformes IMS, l'IIP (INT IMS Platform) un projet open source développé au sein même du laboratoire RS2M de TELECOM SudParis et la solution Alcatel-Lucent 5400 IAS (IMS Application Server). Le *Session Continuity Application Server* que nous présentons ici a été développé et déployé dans les laboratoires d'Alcatel-Lucent, l'IAS étant plus proche de la norme. Ces travaux ont été publiés dans [80].

Le SCAS est donc un serveur d'applications qui communique avec le cœur IMS (la couche de contrôle), il a été développé en Java et est embarqué dans la solution 5400 IAS qui connecte les services fournis par les AS aux fonctions de contrôle réalisées par les CSCF (cf. 1.2.2). Le principe d'un AS est de réaliser des traitements spécifiques à partir des messages SIP qui transitent par le cœur IMS lors de l'établissement, le maintient et la terminaison de sessions multimédias (cf. 1.2.2). Le SCAS est chargé d'orchestrer les handovers afin d'assurer la continuité, pour cela

il doit déclencher et réaliser les transferts de sessions entre les terminaux IMS (cf. Figure 3.4).

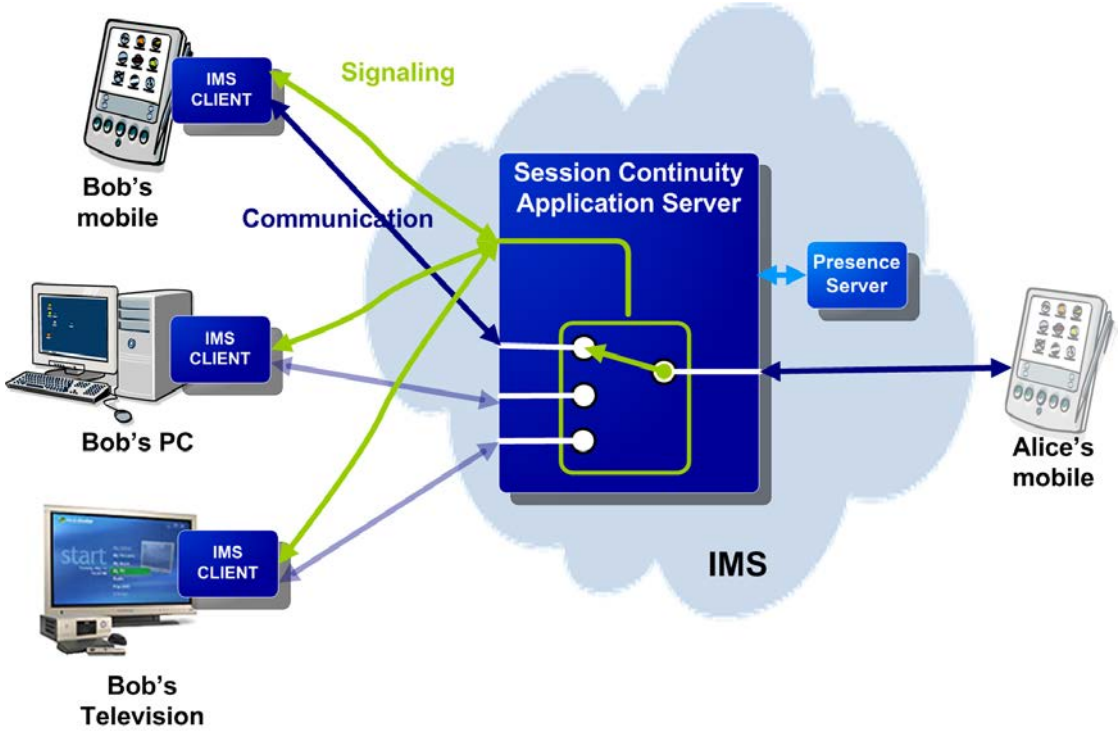



FIGURE 3.4 – Rôle du SCAS.

L'implémentation du SCAS a été réalisée uniquement dans le cadre d'un terminal handover. En effet, le mécanisme de transfert étant similaire dans l'IMS pour un network handover (*soft*) comme pour un terminal handover (cf. 3.2.1), et face à l'absence de terminaux mobiles IMS *dual-mode* supportant le *multihoming*, nous nous sommes focalisés sur le transfert inter-terminal.

### 3.3.1 Scénario

Nous avons développé deux variantes d'un même scénario qui correspondent à deux types de déclenchements, tous deux semi-automatiques requérant l'interaction de l'utilisateur pour réaliser le transfert de session en mode *push* ou *pull*.

Le scénario de base est simple : Alice est abonnée à un opérateur mobile doté d'un réseau IMS et elle a souscrit au service « *Session Continuity* » (délivré par le SCAS). Alors qu'elle est en communication avec Bob sur son ordinateur (téléphone

logiciel SIP-IMS), elle doit quitter son bureau pour aller en rendez-vous. Au lieu d'interrompre l'appel elle préfère transférer la communication sur son téléphone mobile (même abonnement IMS) pour poursuivre la discussion avec Bob pendant son déplacement. La question est comment Alice initie le transfert ?

Nous avons étudié deux modes de déclenchement : un mode *push* où Alice transfère explicitement sa session depuis le terminal origine et un mode *pull* où elle récupère la session à distance, depuis le terminal destination (cf. Figure 3.5). En mode *push*, plus manuel, c'est l'utilisateur qui déclenche le transfert via une interface spécifique sur le terminal d'origine, une confirmation peut éventuellement être requise au niveau du terminal destination. En mode *pull* le transfert est déclenché en fonction de l'état de présence des terminaux de l'utilisateur. Le SCAS récupère les informations de présence de chacun des terminaux des utilisateurs (ayant souscrit au service). En fonction de règles prédéterminées (nouveau terminal disponible, état de présence spécifique, etc) le SCAS déclenche le transfert, une confirmation peut éventuellement être requise ici aussi, le but étant de ne pas nuire à l'expérience utilisateur.

### 3.3.2 Architecture

Le scénario de transfert implique de nombreux acteurs :
- le terminal origine de Bob (un ordinateur doté d'un téléphone logiciel IMS),
- le terminal d'Alice (interlocutrice de Bob),
- le terminal destination de Bob (un téléphone mobile IMS),
- l'infrastructure IMS qui gère les sessions de communications,
- le serveur de présence qui indique l'état de chacun des terminaux,
- le SCAS chargé d'assurer la continuité.

Nous allons nous intéresser aux relations entre ces éléments et plus précisément aux blocs fonctionnels du SCAS durant les étapes successives d'un transfert entre terminaux.

#### Souscription au service de continuité.

Le SCAS fournit un service de continuité de service aux abonnés IMS qui ont explicitement souscrit à cette offre. Cette souscription est contractée à l'avance


FIGURE 3.5 – Scénario de transfert (mode *push*).

(hors connexion), entre l'opérateur télécom qui gère le domaine IMS et l'utilisateur. Au niveau de l'infrastructure, cette souscription se traduit par l'ajout de règles de déclenchement (*initial Filter Criteria* cf. 1.2.2) dans le HSS qui seront traitées par le S-CSCF, le composant IMS par lequel transitent l'ensemble des messages de signalisation. Chaque nouvel abonné est ainsi associé à un iFC.

**Traçage des sessions.**

Les iFCs ajoutés lors de la souscription indiquent au S-CSCF quels messages de signalisation rediriger vers le SCAS, permettant ainsi au serveur d'écouter et d'enregistrer les informations relatives à ses abonnés (terminaux, sessions de communications, etc). C'est ce que l'on appelle « traçage ».

Les messages d'enregistrement (et d'authentification) des clients IMS via la méthode REGISTER du protocole SIP sont les premiers à être interceptés. Ils permettent de connaître l'état de connexion à l'IMS d'un utilisateur, ainsi que l'adresse réseau, l'URI SIP et éventuellement le type de terminal. Les messages sont reçus et analysés par le *Session Logger* qui en extrait les informations utiles et les stocke dans une base de données locale au SCAS (*Logs* sur la Figure 3.6).

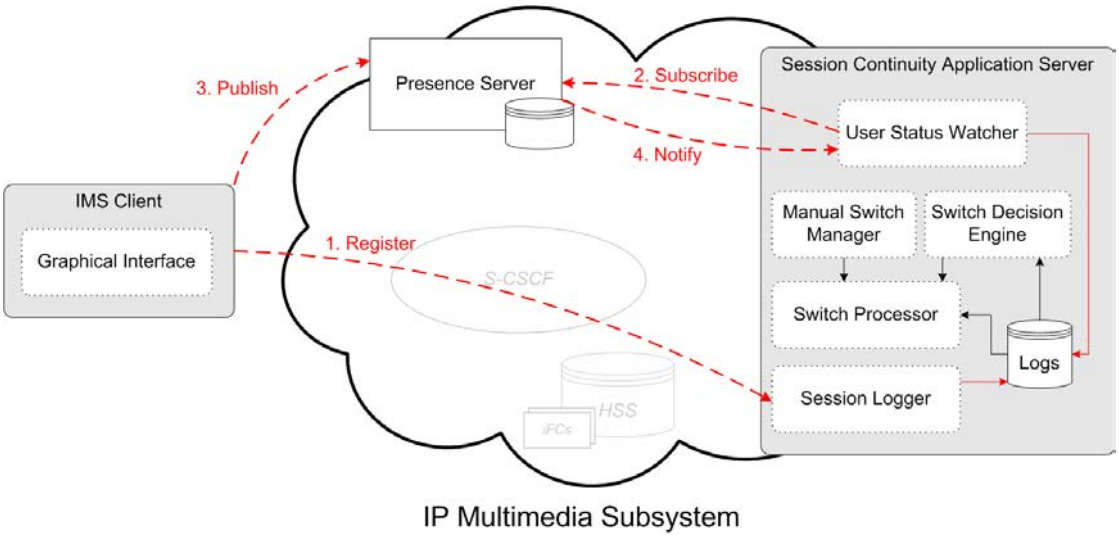



FIGURE 3.6 – SCAS durant l'enregistrement d'un client.

De plus, le processus d'enregistrement active les mécanismes SCAS relatifs à l'utilisateur en question, tel que la souscription aux informations de présence dans

le cas du mode *pull* (cf. 3.3.1) dont le déclenchement est basé sur ces informations d'état. Le composant *User Status Watcher* va alors souscrire aux informations de présence de l'utilisateur, ainsi à chaque modification opérée par le client le SCAS sera notifié, les informations seront alors stockées dans les *Logs* et pourront intervenir dans le processus de déclenchement du transfert.

Enfin, lorsque l'utilisateur créera ou sera invité à une session de communication, l'ensemble des échanges protocolaires seront analysés par le *Session Logger* qui stockera, comme pour l'enregistrement, les informations nécessaires en cas de transfert, cf. Figure 3.7. Il est à noter que le traçage des messages par le SCAS est réalisé de manière transparente pour les clients ; les échanges de requêtes SIP lors de l'enregistrement et de l'établissement des sessions de communications restent échangés vis à vis de la norme.

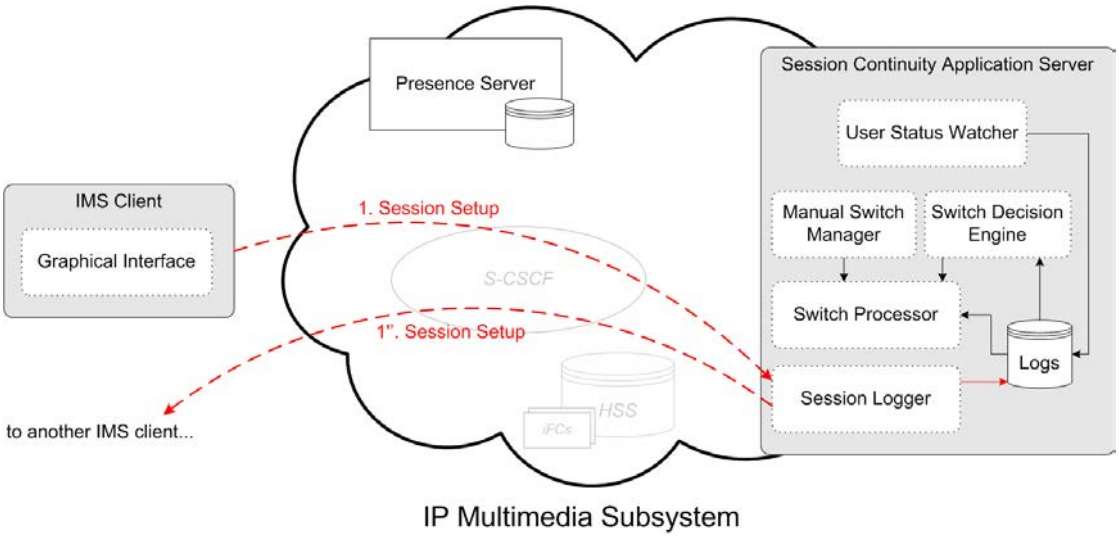



FIGURE 3.7 – SCAS durant l'établissement d'une session.

**Déclenchement.**

Nous avons implémenté deux types de déclenchements : un mode *push* où l'utilisateur requiert le transfert explicitement depuis le terminal d'origine et un mode *pull* où le transfert dépend de règles, l'utilisateur ne faisant que confirmer depuis le terminal destination (cf. 2.1.3). La Figure 3.8 présente l'interaction entre les différents composants dans ces deux cas. Il faudra alors considérer le client

IMS représenté comme le terminal origine pour le premier cas (mode PUSH) et comme terminal destination dans le second. Dans les deux cas, le transfert est initié depuis le terminal client, de manière plus ou moins directe et transparente pour l'utilisateur.

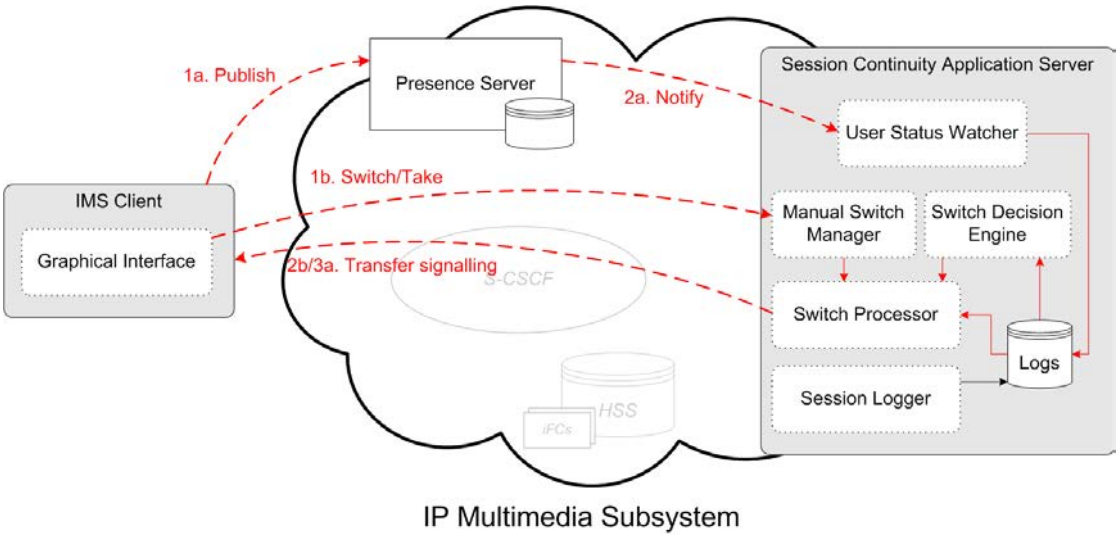



FIGURE 3.8 – SCAS durant le transfert d'une session.

En mode *push*, un ordre spécifique est envoyé au SCAS et traité par le *Manual Switch Manager* qui, en fonction de la commande *Switch* ou *Take*, va préparer et décider ou pas de déclencher le handover. La commande *Switch* marque (au niveau du SCAS) les sessions en cours sur le terminal comme « transférables », cela correspond donc à une volonté explicite du client de transférer le service de communication, cf. Figure 3.5. Ce mécanisme déclenche alors sur les autres terminaux de l'utilisateur une proposition de transfert, l'acceptation résulte de l'envoi de la commande *Take* qui identifie le terminal destination, le *Manual Switch Manager* va alors transmettre l'ordre de transfert effectif au *Switch Processor*.

Ce mode requiert la création d'un canal de communication spécifique, donc non standard, entre le client et le SCAS. Plusieurs approches sont possibles, nous avons choisi de détourner les méthodes SIP de type MESSAGE prévus pour l'envoi de messages textes courts (exception de contenu dans la signalisation SIP). Ainsi, en formatant de manière propriétaire ces messages textes, nous transmettons des informations de contrôle (ici *Switch*/*Take*) de manière transparente via le cœur IMS.

La force de l'approche *push* basée sur les commandes *Switch* et *Take* est sa double simplicité. Simplicité tout d'abord pour l'utilisateur qui n'a que deux interactions à réaliser en tout pour effectuer un transfert, point très important dans le cadre de services de télécommunications où le transfert mais également son déclenchement doit être simple et rapide. Cette approche est également simple d'un point de vue implémentation car celle-ci est allégée des considérations de désignation (sessions et terminaux) particulièrement complexes, les modalités du transfert sont directement déduites de l'usage de l'utilisateur : l'interface par laquelle il confirme le handover correspond au terminal destination.

Il faut cependant noter que ce mode de désignation et le mode *push* de manière générale car plus « manuel », requiert une interface spécifique, non standard, au niveau du client IMS. Nous avons travaillé avec des produits Alcatel-Lucent auxquels nous avions accès au code source, ainsi il nous a été possible de développer une interface minimaliste comportant les actions *Switch* et *Take*. Des captures d'écran correspondant au mode *push* sont présentées sur la Figure 3.5.

Le mode *pull* est légèrement plus simple mais aussi plus respectueux du standard. En effet, il n'existe pas ici de communication directe SCAS-client, bien que ce dernier soit l'initiateur du transfert. Il existe de nombreux critères de déclenchement automatique, nous avons choisi l'état de présence. Pendant une communication, tout nouveau terminal disponible (d'un point de vue présence) appartenant à l'utilisateur devient la destination du transfert. Seule une confirmation est requise au niveau de l'interface destination afin de s'assurer que le handover est désiré. Le *Switch* est ici induit par un changement d'état de présence et le *Take* correspond à la confirmation de la destination qui ne fait qu'accepter le transfert au niveau protocolaire (aucun message spécifique n'est envoyé).

Seul le message de confirmation de transfert pourrait paraître non-standard, cependant la spécification du mécanisme de transfert basé sur le message REFER ([54], cf. 3.3.3 pour plus de détails) suggère fortement que le handover soit acquitté au niveau terminal.

Enfin le *Switch Processor* va mettre œuvre les mécanismes de handover basés sur les informations précédemment collectées et stockées dans les *Logs*. Ce composant va communiquer avec les différents terminaux en SIP afin d'orchestrer le transfert, ces mécanismes sont décrits en détail dans la section 3.3.3.

Pour un mode de transfert manuel, néanmoins peu adapté aux services de communications : sélection individuelle de la session et des terminaux origine et destination, de nombreuses approches basées sur un canal de communication propriétaire client-SCAS sont envisageables. Il est toutefois intéressant de remarquer que le serveur de présence notifie les clients des états des autres terminaux et que le modèle de données, bien que standard, offre de nombreuses possibilités. Un mécanisme basé sur l'état de présence d'une session, d'un service ou encore un état spécifique indiquant la volonté ou la disponibilité pour un transfert pourrait être aisément implémenté. Nous verrons dans d'autres cas d'utilisation, notamment avec les services multimédias (cf. 5), ce genre d'approche.

### 3.3.3 Mécanisme de transfert

La mise en œuvre du transfert repose sur des mécanismes du protocole SIP, nous allons détailler ce processus dans cette section. Comme décrit précédemment, le SCAS trace et enregistre les messages de signalisation des utilisateurs ayant souscrit au service de continuité. Ainsi le *Switch Processor*, composant fonctionnel responsable de la négociation du transfert dispose de l'ensemble des données nécessaires.

#### Principe.

Au niveau SIP, un transfert de session peut être réalisé grâce à un champ spécifique nommé *Replaces* [56], une extension de l'IETF. Ce champ est utilisé dans les messages INVITE lors de l'initiation d'une nouvelle session. Lorsqu'un *Replaces* est présent dans un tel message, le récepteur de l'INVITE va alors créer la nouvelle session (s'il accepte) puis terminer la session identifiée par le contenu du champ *Replaces*. De cette manière on peut simplement substituer une session par une autre, donc un participant à une communication par un autre.

Cependant, utilisé de la sorte, seul l'un des participants à la communication peut générer un tel message car il lui faut connaître les identifiants de la session. De plus, la nouvelle session est nécessairement créée avec l'initiateur du message contenant un *Replaces*. Or nous souhaitons que le transfert soit orchestré par un tiers, le SCAS en l'occurrence. Si les informations privées de la session sont connues grâce au traçage des messages de signalisation, il réside le problème que la nouvelle

session ne doit pas être créée entre le SCAS et Bob ou Alice mais directement entre eux.

On utilise alors le message REFER [54], une autre extension du protocole SIP. Un message REFER permet d'indiquer à son destinataire une action à réaliser, typiquement initier une communication avec un autre client. De plus, des paramètres peuvent être ajoutés afin de rendre l'action réalisée par procuration plus précise encore. Dans notre cas, il suffit alors d'envoyer un message REFER indiquant au destinataire d'émettre une requête INVITE comportant un champ *Replaces* spécifique.

La Figure 3.9 illustre une exemple basique d'utilisation du message REFER permettant la substitution de la session Bob2-Alice par la session Bob1-Alice. On remarque sur ce diagramme de séquence des messages SIP que la session initiale n'est terminée qu'après l'établissement de la nouvelle, il y a donc une superposition des deux communications pendant un court laps de temps permettant un transfert plus fluide, moins perceptible par l'utilisateur. Cependant ce mécanisme suppose que le terminal d'Alice maintienne deux communications simultanément, imposant une contrainte forte à la fois logicielle (multiples sessions) et matérielle (gestion du microphone, occupation mémoire, etc).

**Concrètement.**

Plus concrètement, appliqué à notre SCAS, le mécanisme de transfert REFER présenté ci-dessus est géré par le *Switch Processor* grâce aux informations tracées et stockées dans les *Logs*. Les diagrammes de séquences 3.10 et 3.11 détaillent les messages échangés entre les différentes composantes du système en mode *push* (communication directe entre le client et le SCAS) et en mode *pull* via le serveur de présence.

## 3.4 Conclusion

Cette première approche de la continuité de service dans un cadre concret, l'IMS, a permis d'identifier des problématiques insoupçonnées jusqu'alors. En effet, bien que nous ayons restreint notre étude à un type de service particulier (télécoms), dont le protocole est clairement spécifié et qui de plus possède des mécanismes de

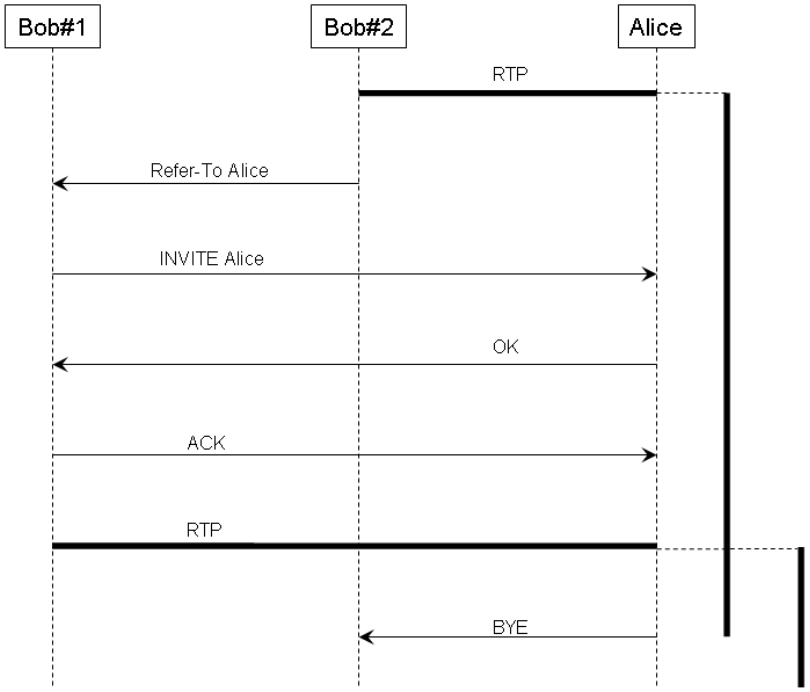



FIGURE 3.9 – Principe de transfert via le message REFER.

continuité standards, la mise en œuvre dans la vie réelle de la mobilité n'est pas encore satisfaisante d'un point de vue utilisateur.

Si l'on considère uniquement les qualités des procédés de transfert proposés, le bilan est positif conformément à l'état de l'art, cf. 2.3.2. La continuité d'une session est réalisée de manière efficace, rapide et synchronisée. L'implémentation d'un prototype a cependant mis en avant que le mécanisme de transfert seul était insuffisant, l'ensemble du processus de continuité qui commence et termine par l'utilisateur requiert de nombreuses fonctions permettant d'identifier les sessions, les terminaux, les utilisateurs, les désigner et enfin orchestrer la mobilité. Le déclenchement et l'orchestration ne consistent pas uniquement à exécuter les algorithmes de transfert mais également à contrôler les applications et la manière dont elles fournissent le service avant et après son transfert. Quelques exemples significatifs : il n'existe aujourd'hui aucune interface prévue pour déclencher un transfert depuis

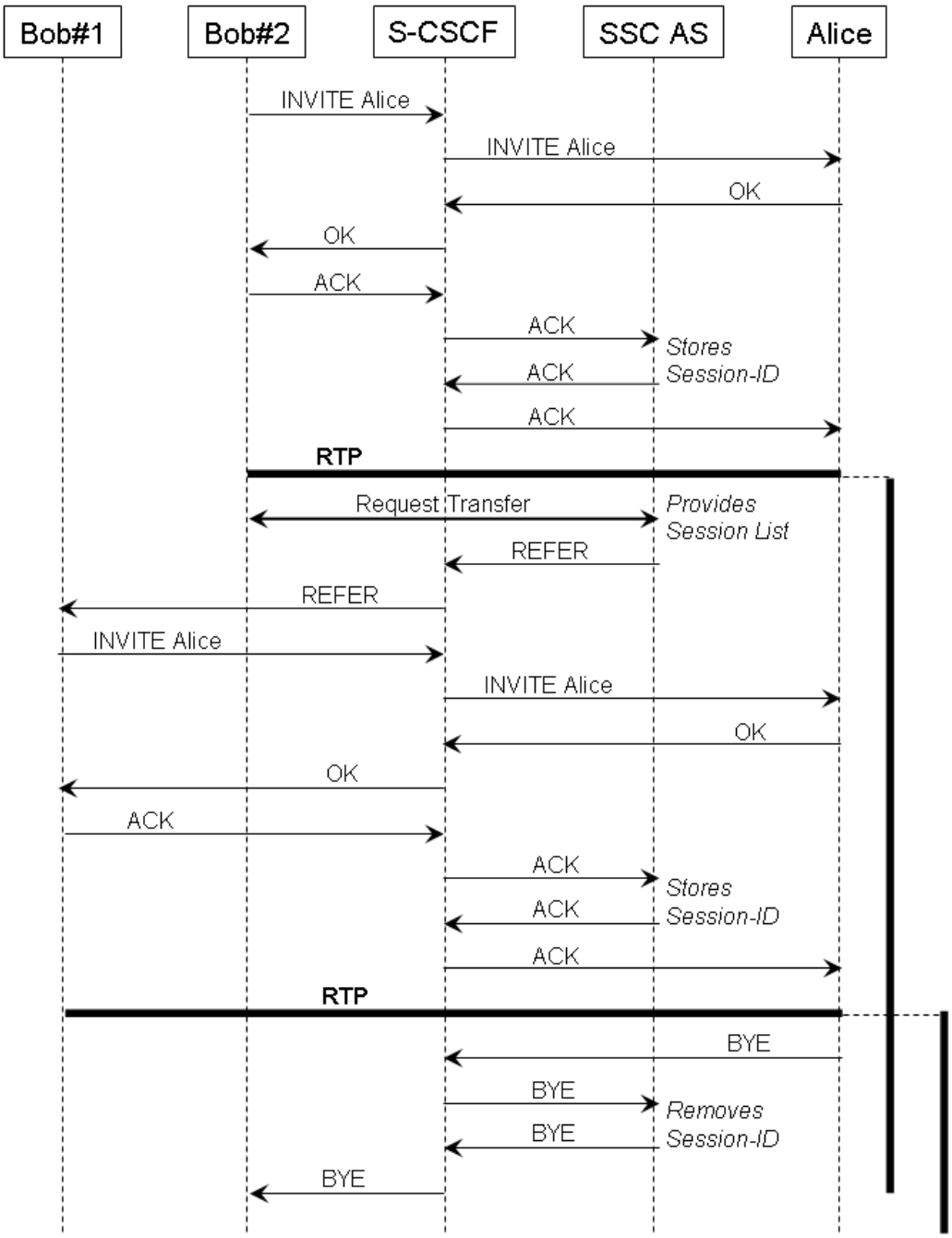



FIGURE 3.10 – Échanges SIP d'un transfert en mode *push*.

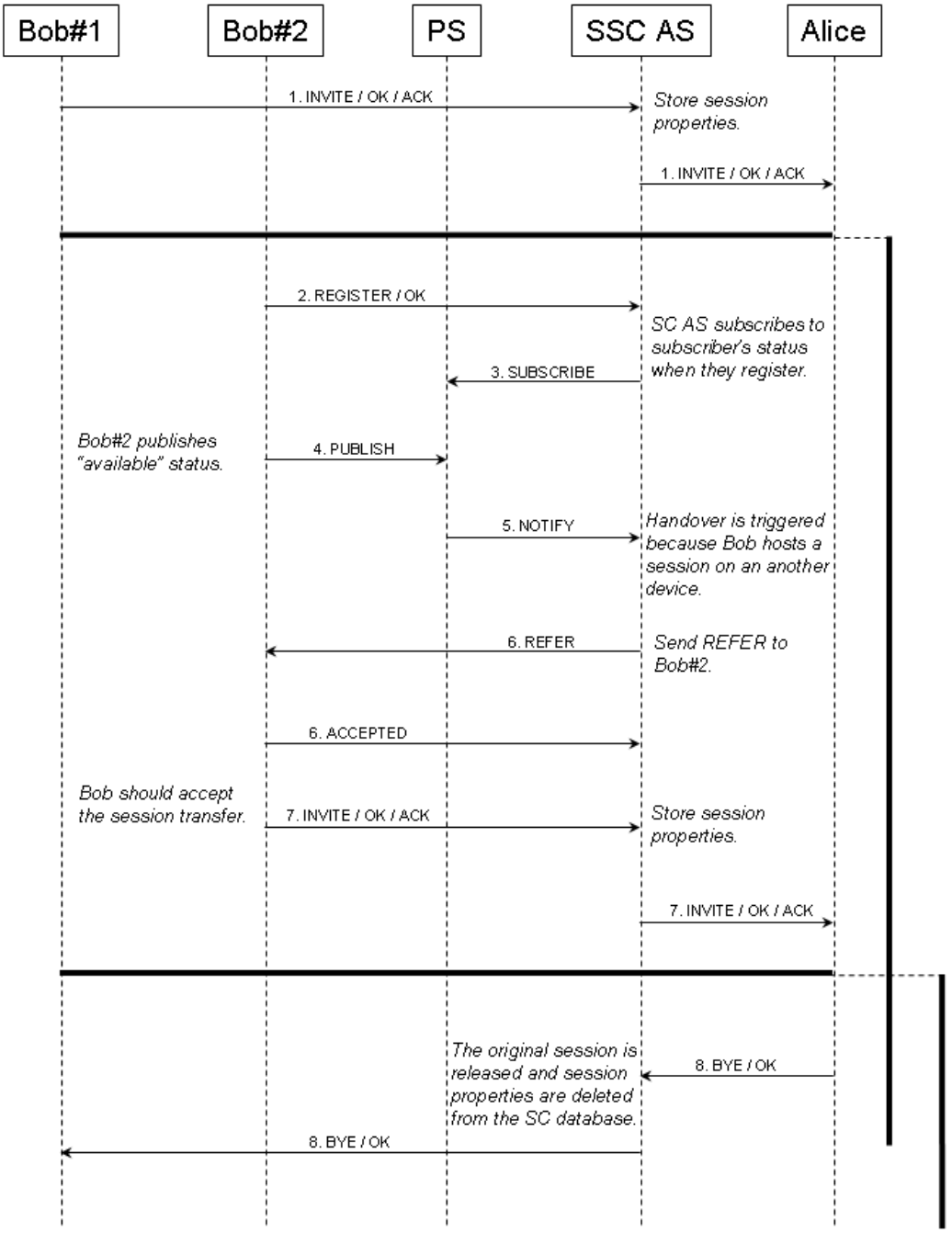



FIGURE 3.11 – Échanges SIP d'un transfert en mode *pull*.

un client IMS (nous avons dû l'implémenter), aussi un terminal ne peut accepter un transfert que si le client IMS est lancé et à l'écoute (aucun contrôle applicatif), enfin un utilisateur ne possède aucun moyen de désigner explicitement un de ses terminaux comme destinataire du transfert.

Il est relativement facile d'introduire de la mobilité dans un tel environnement homogène, l'IMS est une architecture standardisée du terminal à l'infrastructure, en passant par les services de base (*enablers*) et les protocoles. Il y a peu de surprises entre le terminal d'origine et celui de destination : capacités équivalentes, application et protocole identiques, l'adaptation du service est extrêmement faible et limitée à un rare changement d'encodage des flux de données.

Dans ce cadre rigide, la continuité temporelle est parfaite, cependant la continuité contextuelle manque cruellement à l'expérience utilisateur. Le service de télécommunication n'est pas qu'un simple flux de données, de multiples éléments sont perçus par l'utilisateur : configuration (réglage de la sonnerie, renvoi d'appel), liste de contacts, historique d'appels, avatar, etc. Si les approches basées sur une architecture de services forte telle que l'IMS permettent la continuité de certains de ces éléments et les déplaçant du terminal vers l'infrastructure (nécessitant autant de nouveaux standards, cf. présence), les caractéristiques des applications permettant d'offrir des services plus ou moins riches pour une expérience utilisateur optimale sont simplement ignorées.

# Chapitre 4

# Continuité des services multimédias dans l'IMS

## 4.1 Motivation

Après les services de télécommunications, nous nous sommes intéressés aux services multimédias qui se présentent généralement sous la forme d'applications de streaming audio/vidéo. Ces services distants, connectés et basés sur des sessions, sont intrinsèquement sensibles aux déconnexions et donc aux problématiques de mobilités utilisateur et de terminal que nous avons identifiées précédemment. Nous avons choisi le cadre IMS pour commencer à considérer la continuité de ce type de service et ce pour deux raisons : premièrement la convergence des mondes Web et télécom font une place plus importante aux services multimédias qui deviennent un enjeu stratégique pour les opérateurs, or la continuité doit être assurée dans ces milieux mobiles ; deuxièmement, la problématique de continuité dépasse largement le mécanisme de transfert tel que nous l'avons identifié dans le chapitre précédent (cf. 3), l'identification, la désignation, le déclenchement,... de nombreuses fonctions entrent en jeu or elles sont en partie gérées dans l'IMS ou via notre *Session Continuity Application Server*.

Lors de cette première approche des services multimédias via l'IMS, nous avons cherché à étendre les mécanismes de continuité offerts par le SCAS à un tout autre type de service. Ainsi, nous apportons une solution globale et cohérente de mobilité

de service aux utilisateurs IMS, leur permettant une continuité de l'ensemble de leurs services télécoms et multimédias. Cependant, si les mécanismes de transfert des sessions de communications IMS, basées sur SIP sont relativement bien définis car standardisés et donc imposés par l'infrastructure, il en est tout autre des applications de streaming issues de l'ouverture de l'IMS vers Internet. Contrairement aux services de télécommunications dont l'ensemble de la signalisation est gérée par l'architecture IMS, une application de streaming est un service distant qui implique un client (ici IMS) et un serveur de médias (*Media Server*, MS), il est donc important de s'intéresser aux nouveaux protocoles et mécanismes mis en jeu.

Notre approche a été influencée par des considérations stratégiques. En effet, Alcatel-Lucent ne se positionne pas en tant qu'acteur dans le monde des services multimédias, il a donc été nécessaire d'adopter une approche transparente pour les lecteurs et les serveurs existants, aucune modification de ces programmes n'étant envisageable. Dans ce chapitre nous présentons les modifications nécessaires au SCAS pour la réalisation d'une solution de mobilité gérant à la fois la communication et les médias et ce à toutes les étapes d'un transfert de service multimédia entre deux terminaux. Pour les mêmes raisons que le chapitre précédent, nous ne nous intéresserons pas au cas de *network handover* qui est géré au niveau accès de l'IMS, nous nous focalisons uniquement sur l'aspect applicatif. De plus les incidences d'un changement de réseau sont gérées lors de l'enregistrement de l'utilisateur (message SIP INVITE), le transfert étant alors transparent et similaire à un *terminal handover*.

## 4.2 Problématiques et approche

L'approche est identique à celle du chapitre précédent : une solution applicative basée sur le SCAS qui respecte les différents standards en présence (SIP, SDP, IMS, etc) garantissant la préservation de l'interopérabilité des composants existants. Cependant nous nous intéressons maintenant à un type de service différent qui, chose importante, est extérieur à l'IMS. En effet, les services multimédias qui se présentent généralement sous la forme de diffusion de flux audio/vidéo en direct (télévision) ou à la demande (*Video on Demand*, VoD) offerts aux clients IMS via un portail de l'opérateur ou directement via Internet, n'impliquent en aucune manière les composants IMS (cf. 1.2.2) dédiés à la communication. Le terminal

du client gère le service directement avec le MS via une application (*un lecteur*) multimédia, l'enjeu est de mettre en relation les mécanismes existants du SCAS (identification, déclenchement manuel et automatique) avec ceux qui fournissent le service multimédia à transférer.

### 4.2.1 Protocoles multimédias

SIP étant dédié à la communication, de nouveaux protocoles sont nécessaires pour délivrer les services multimédias. De nombreux existent, les principales solutions étant listées ci-dessous.

- RTSP (Real-Time Streaming Protocol [58]). Un protocole standardisé par l'IETF qui présente l'avantage majeur d'être ouvert et bien documenté.
- MMS (Microsoft Media Server [81]). Un protocole propriétaire (aujourd'hui ouvert) spécifié par Microsoft qui était l'un des plus utilisés lors du début de ces travaux, la majorité des solutions serveurs étant des produits Microsoft (*Windows Media Services* WMS [82]). Cependant il a laissé sa place progressivement à RTSP, la version 2008 de WMS l'a totalement remplacé.
- HTTP (Hypertext Transfer Protocol [10]). Un protocole initialement destiné à la navigation sur le Web, il permet en outre la lecture de fichiers multimédias en cours de téléchargement (technique appelée « téléchargement progressif », légèrement différente du streaming pur [83]). Cette solution beaucoup utilisée historiquement a laissé sa place à des approches de streaming, basées sur des flux, mieux adaptées aux terminaux mobiles à faible capacité mémoire et plus sûres en terme de droit d'auteur pour les diffuseurs de contenus ; le média transmis est effacé à mesure qu'il est consommé par le client.

Il existe de nombreuses autres approches telles que les solutions Web pures basées sur HTTP et les technologies Flash d'Adobe [84]. Un très grand nombre de sites tels que *Youtube* et autres *Dailymotion* ont émergé et leur forte popularité liée à leur simplicité d'utilisation ont vite rendu les protocoles de streaming « obsolètes » : client constitué d'un simple navigateur Web, pas de configuration nécessaire (ports réseaux, pare-feu, etc). Cependant ces solutions Web ne sont pas adaptées au monde télécom et bien que nous adressons les services multimédias, nous nous situons dans un environnement IMS servant des terminaux mobiles dits

« légers », peu capables d'un point de vue pratique à permettre une navigation sur le Web. Les services multimédias provenant certes d'Internet seront plus vraisemblablement offerts par l'opérateur IMS via une interface spécifique adaptée au terminal mobile.

Évidemment, si cet constat était vérifié au début de ces travaux, il n'en est rien aujourd'hui. L'offre des opérateurs et les terminaux se sont adaptés à ces nouveaux besoins et revenus potentiels, permettant aux utilisateurs de consommer ces services multimédias Web sur des clients mobiles tel que le démontre aujourd'hui le succès de l'iPhone d'Apple. Nous nous intéresserons à ce type de services multimédias dans le chapitre suivant qui se libère du cadre de l'IMS, cependant ce chapitre se focalise sur les contraintes des terminaux mobiles dans un cadre IMS avec des clients légers, les solutions Web pures ou HTTP sont donc pour l'instant écartées.

### 4.2.2 Déclenchement

Si l'on revient au SCAS en tant que service de continuité dans l'IMS puisqu'il est question de l'étendre, les problématiques liées au transfert d'un service de communication sont les mêmes pour un service multimédia. Et le premier des mécanismes à intervenir est le déclenchement. Ici aussi deux modes sont possibles : un mode automatique où le transfert des flux médias est déclenché en fonction de règles prédéfinies ou un mode manuel qui requiert une désignation de la part de l'utilisateur. Comme pour la communication, un flux média sera vraisemblablement unique pour un utilisateur donné : communiquer avec plusieurs interlocuteurs (hormis le mode conférence) ou regarder plusieurs médias à la fois est plutôt rare. Cette unicité est propice au mode automatique ou aux modes manuels simples qui nécessitent peu de paramètres de transfert.

Ainsi nous avons repris les mêmes mécanismes de déclenchement que pour les services de communications : un mode automatique basé sur les informations de présence (les services sont transférés sur le terminal actif) et un mode manuel que l'utilisateur déclenche à la demande depuis le terminal origine. Ces modes sont décrits dans le chapitre précédent en tant que modes *pull* et *push*. Cependant une différence de taille se présente ici, si le déclenchement est effectué depuis le client IMS, le transfert effectif est lui réalisé sur le client média. Le terminal est le même

mais les applications sont non seulement distinctes mais elles ne possèdent aucun point commun au niveau de l'infrastructure. Une relation doit être établie.

### 4.2.3 Identification

Le service multimédia doit être associé à l'utilisateur IMS. Cette condition est nécessaire pour que les mécanismes de continuité du SCAS puissent être mis en œuvre, le service du SCAS étant délivré uniquement à ceux qui y ont souscrit. Si l'identification est une fonction de base dans l'infrastructure IMS, et donc réutilisable par le SCAS, le service multimédia issu de la relation directe entre le lecteur et le serveur de médias échappe à toute identification IMS.

Il faut recréer ce couplage fort entre l'utilisateur et ses services, pour cela nous nous intéressons au contenu des échanges entre le lecteur et le serveur de médias. Ce traçage des messages relatifs au service multimédia permettra de l'associer à un utilisateur IMS connu par l'infrastructure et ainsi de mettre en œuvre les mécanismes adéquats. Pour que l'association soit possible, un élément de référence est nécessaire : soit l'identifiant de l'utilisateur ou du terminal ajouté de manière transparente dans la signalisation du service média (si possible), soit des paramètres réseaux tel que l'adresse IP qui est commune aux clients média et IMS.

### 4.2.4 Contrôle

Autre problème directement lié, comme l'identification, à la séparation physique entre le service multimédia et l'infrastructure IMS où se situe le SCAS : le contrôle du transfert. Si la mobilité de session fait partie intégrante des mécanismes natifs de SIP, protocole interne de l'IMS, ici nous devons considérer une infrastructure et un protocole autre qu'il faut manipuler. Le *terminal handover* consiste en un certain nombre d'étapes qui permettent le transfert du service, nous verrons plus tard comment celui-ci est réalisé effectivement. Néanmoins le transfert requiert une intervention dans la fourniture du service, que ce soit au niveau du lecteur média, comme de la signalisation ou encore du flux de données. Les mécanismes du SCAS situés dans l'IMS doivent pouvoir agir sur ces éléments extérieurs, nécessitant éventuellement une modification de la topologie réseau.

Le traçage des messages de signalisation par exemple que nous avons vu au chapitre précédent mais également nécessaire aux fonctions d'identification 4.2.3

est une étape indispensable pour collecter les informations nécessaires à la manipulation de sessions de données par un tiers (ici le SCAS). Si dans le cas des sessions de communications IMS, l'infrastructure (et donc le SCAS) se positionne naturellement et avantageusement en tant qu'intermédiaire, il est nécessaire de mettre également le SCAS en tant qu'intermédiaire pour les services multimédias. Un composant devra donc jouer le rôle de PROXY entre le lecteur et le serveur de médias.

## 4.3 Implémentation

Cette fois encore, un prototype a été implémenté sur la plateforme IMS d'Alcatel-Lucent (solution 5400 IAS). Nous avions également à notre disposition des clients IMS pour terminaux fixes et mobiles que nous pouvions modifier afin d'adapter les interfaces graphiques pour déclencher les mécanismes de transfert. Cependant l'objectif reste de proposer des solutions intégrables dans des environnements définis et non modifiables (pour des raisons d'interopérabilité), que ce soit l'infrastructure, les clients ou les protocoles impliqués. C'est pour cela que nous préfèrerons toujours les approches compatibles ou à défaut celles qui ont le moins d'impact sur les environnements existants.

Les travaux ont porté spécifiquement sur les protocoles MMS et RTSP, l'ensemble des mécanismes décrits ici étant indifféremment applicables (sauf mention contraire). Le protocole HTTP quant à lui s'est révélé peu intéressant pour cette étude de par son aspect non standard, nous aurons cependant l'occasion de nous y intéresser dans le chapitre suivant. Le serveur de média utilisé est *Windows Media Server* et le lecteur multimédia est *Windows Media Player*, l'objectif conforme à notre stratégie (cf. 4.1) étant de proposer une solution transparente pour les clients comme pour les serveurs multimédias existants.

Nous considérons dans cette section le SCAS dans sa globalité (continuité des communications et des flux multimédias) mais plus particulièrement les relations entre les différents composants pour offrir la continuité des services multimédias dans l'IMS. Le SCAS tient le même rôle que précédemment, c'est à dire une application IMS assurant la continuité des services de communications mais aussi maintenant celle des services multimédias et ce via un *proxy média* (cf. Figure 4.1)

. Nous commencerons par un scénario qui illustre le transfert et sera la trame de nos explications, puis nous détaillerons l'architecture en place et enfin le mécanisme de transfert. Ces travaux ont été publiés dans [85].

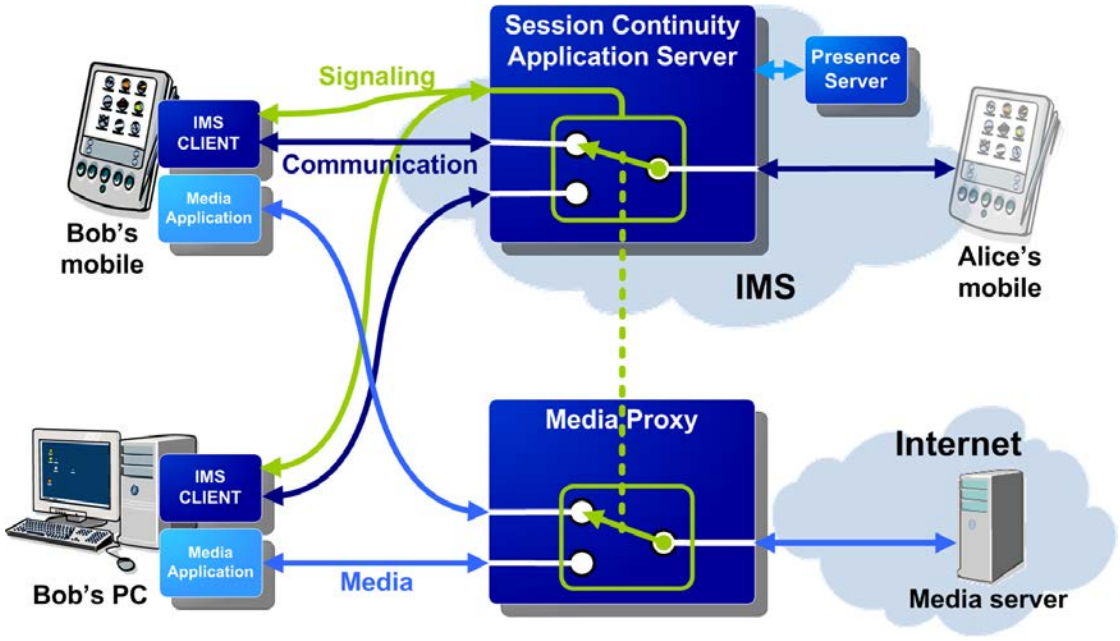



FIGURE 4.1 – Positionnement du SCAS et du proxy média.

### 4.3.1 Scénario

Le scénario est assez simple et n'implique maintenant plus qu'un seul utilisateur. Bob regarde une bande annonce de film sur son téléphone mobile IMS doté d'un lecteur multimédia. Une fois arrivé chez lui, il préfèrerait continuer de regarder le média sur son ordinateur de bureau doté d'un écran bien plus large. Il déclenche alors le transfert (manuel ou automatique), une confirmation est requise sur le terminal destination puis la vidéo reprend à l'endroit même où il l'avait laissée (cf. Figure 4.2).

Il est important de noter que les terminaux de Bob, que ce soit son mobile ou son ordinateur possèdent chacun un client IMS, comme pour un transfert de service de communication.


FIGURE 4.2 – Scénario de transfert de service multimédia.

### 4.3.2 Architecture

Les acteurs du scénario sont les suivants :

- le terminal origine de Bob (téléphone mobile IMS doté d'une lecteur multimédia),
- le terminal destination de Bob (ordinateur de bureau avec client IMS et lecteur multimédia),
- infrastructure IMS + serveur de présence (pour le mode automatique),
- un serveur de médias qui délivre les flux vidéos,
- le SCAS doté d'un proxy média.

Nous allons nous intéresser maintenant aux mécanismes qui entrent en jeu dans cet environnement et plus particulièrement au SCAS et au proxy média lors de la mise en œuvre d'un transfert de service multimédia entre deux terminaux IMS.

Le service offert par le SCAS doit être a priori souscrit par Bob auprès de son opérateur IMS afin que les mécanismes de continuité soient correctement mis en œuvre, exactement comme pour la continuité des services de communications (cf. 3.3.2).

**Traçage des sessions.**

Le SCAS trace et stocke les informations relatives aux abonnés du service de continuité. Comme pour les services de communications (cf. 3), ce mécanisme qui traite les informations des messages SIP (sessions et états de présence) est initié lors de l'enregistrement du client IMS auprès de son domaine, cf. Figure 3.6.

Cependant les informations enregistrées dans les *Logs* ne se limitent pas à l'IMS, le *Media Proxy* intervient également. C'est la « patte » du SCAS qui permet de tracer les informations relatives aux sessions multimédias échangées entre le lecteur et le serveur de médias. Ce composant doit obligatoirement jouer le rôle de proxy auprès du lecteur multimédia de l'utilisateur IMS pour pouvoir intercepter les messages de signalisation. En effet, le *Media Proxy* doit être positionné entre le lecteur et le serveur de médias, cela peut se traduire par un composant réseau supplémentaire nécessitant une configuration du terminal (possible aujourd'hui avec la plupart des applications de ce type) ou une fonction spécifique intégrée au client. Cette deuxième approche est cependant plus complexe car elle requiert une

modification du client et l'instauration d'un interface supplémentaire entre celui-ci et l'infrastructure (le SCAS).

Une fois en place en position de proxy, le *Media Proxy* joue plusieurs rôles. En premier lieu il reçoit l'ensemble des requêtes d'initiation des sessions multimédias provenant des terminaux utilisateurs abonnés au service SCAS (messages SETUP RTSP par exemple), cf. Figure 4.3. Un contrôle d'accès via le composant *Access Control* est alors réalisé afin de limiter l'utilisation du service aux seuls abonnés. En effet, si l'IMS garantit ce contrôle, nous nous trouvons ici à l'extérieur de l'infrastructure, il est donc nécessaire d'implémenter un mécanisme similaire. Comme peu d'informations sont disponibles à partir des messages de signalisation du média et que le client IMS est décorrélé du lecteur, nous avons choisi d'effectuer un contrôle d'accès par adresse IP, cf. 4.2.3. Lors de l'enregistrement IMS de l'abonné, l'adresse réseau est collectée par le *Session Logger* du SCAS (cf. 3.3.2) puis transmise au *Media Proxy* pour être autorisée. Inversement, lorsque le client IMS se désenregistre, l'adresse IP n'est plus autorisée et le lecteur média correspondant ne peut plus bénéficier des mécanismes de continuité du SCAS (il peut cependant lire des flux médias dans la mesure où il n'utilise plus le SCAS en tant que proxy.

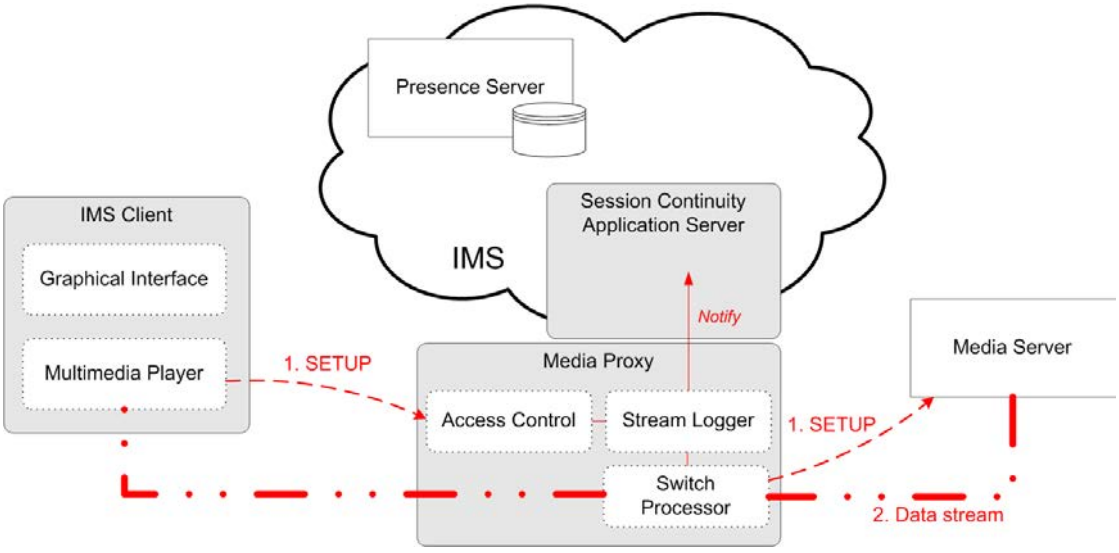



FIGURE 4.3 – SCAS durant l'établissement d'une session multimédia.

Une fois autorisés, les messages de signalisation média sont analysés par le *Stream Logger* qui extrait diverses informations tel que l'adresse source du flux

(URL), les identifiants de connexion, etc. Les informations collectées sont stockées dans les *Logs* du SCAS, cf. 3.3.2. Enfin, le *Switch Processor* est le dernier maillon de cette chaîne de fonctions, et il se comporte en véritable mandataire (*proxy*). Il reste pendant toute la durée de la session entre le lecteur et le serveur en transmettant, dans les deux sens, les messages de signalisation et les flux de données. Le *Switch Processor*, de par sa position stratégique, peut enregistrer localement un grand nombre d'informations essentielles contenues dans la signalisation mais aussi dans le flux de données, ce qui permettra par la suite de réaliser le transfert.

**Déclenchement.**

Nous ne rentrerons pas dans le détail du déclenchement qui est tout à fait comparable à celui décrit dans le chapitre précédent (cf. 3.3.2) étant donné qu'il est réalisé depuis le client IMS. Nous noterons simplement la difficulté de la désignation de la session multimédia. En effet si une session de communication est caractérisée par deux interlocuteurs clairement identifiés, ici ce n'est pas le cas : le lecteur n'est pas identifié et le serveur correspond généralement à une adresse Web (URL), particulièrement illisible pour l'Homme. On pourra alors simplement supposer que l'association réalisée à partir de l'adresse IP permettant d'identifier l'utilisateur (cf. 4.2.3) est suffisante dans le mesure où il est fort probable qu'il ne regarde qu'un flux média à la fois.

Le déclenchement est donc réalisé depuis la couche IMS et aura pour conséquence d'initier le mécanisme de transfert directement par le *Switch Processor* du *Media Proxy*, cf. Figure 4.4. Des informations sont également envoyées vers le client IMS qui déclenchera les mécanismes de handover relatifs au terminal (cf. section 4.3.3). Comme vu dans le chapitre précédent, des messages non standards doivent être transmis au client IMS, ceci peut s'effectuer via des informations de présence spécifiques ou des requêtes SIP de type MESSAGE, dans tous les cas le client devra être capable de les interpréter.

### 4.3.3 Mécanisme de transfert

Un lecteur média, contrairement à un client SIP-IMS ne possède pas de mécanisme prévu au transfert de sessions et ce quel que soit le protocole en question. Le mécanisme de transfert doit donc être réalisé à un autre niveau que le terminal.

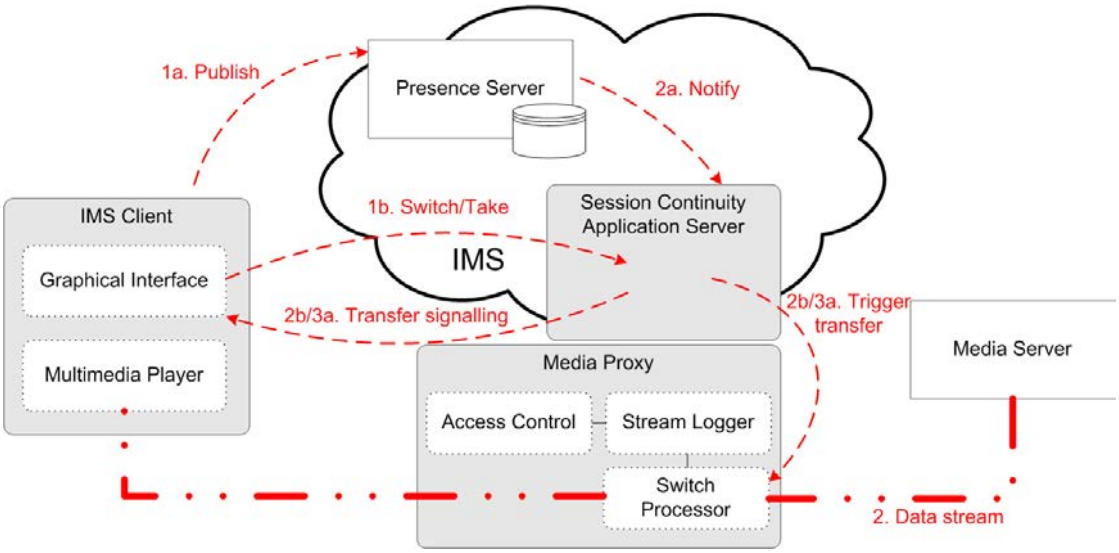



FIGURE 4.4 – Déclenchement du transfert d'un service multimédia.

Pour qu'un transfert de médias entre terminaux soit réalisé avec succès il faut une synchronisation entre l'arrêt du flux sur le premier lecteur et le démarrage du même média sur le second. Ce mécanisme qui ne peut être mis en œuvre au niveau client doit alors être assuré par l'infrastructure, c'est à dire ici par le SCAS.

On peut identifier deux types de sessions multimédias, les flux « en direct » comme la télévision et les flux à la demande (VoD). De manière simple, la différence principale est qu'un flux direct ne peut être manipulé, les commandes qui permettent de contrôler la lecture (play, pause, etc) ne sont pas applicables et ignorées par le serveur. Le mécanisme de transfert par substitution que nous avons vu auparavant (cf. 2.3.2) est la solution classique à ce genre de problématique, basée sur la synchronisation de deux sessions. Mais cette approche nécessite la connexion simultanée de deux clients à un même serveur. Or le modèle commercial du fournisseur de contenu limite par nature l'accès au média, le transfert par substitution sera considéré par le serveur comme deux transactions distinctes, requérant deux paiements...

Il existe des solutions à ce problème, basées sur l'enregistrement du flux par un tiers et qui permet la « pause du direct », cependant nous resterons ici dans un cas simple et ne considérerons que le transfert de flux à la demande. Cependant,

ce type de solution serait tout à fait applicable au *Media Proxy* et aux mécanismes de transfert décrits ici.

Comme nous l'avons indiqué précédemment, des informations contenues dans les messages de signalisation et les flux de données sont collectées par le *Media Proxy* grâce à sa position d'intermédiaire. Les informations essentielles minimales pour rétablir une session multimédia sont l'URL et la position courante du lecteur dans le flux. L'URL est capturée sans difficulté, la position dans le flux est plus complexe à déterminer.

**Position du lecteur dans le flux.**

Deux difficultés se présentent pour déterminer la position du flux : le protocole et en particulier son implémentation au niveau du serveur qui peut ne pas fournir cette information, et la mémoire tampon du lecteur. Au niveau du protocole, nous avons travaillé avec MMS et RTSP. RTSP étant standard et ouvert, une simple requête (PLAY par exemple) envoyée par le *Switch Processor* au serveur permet de déterminer la position actuelle du flux (cf. [58]).

MMS par contre était au début de cette étude un protocole fermé, il était donc difficile d'obtenir cette information. Nous avons donc dû analyser les paquets de données RTP qui acheminent le flux média vers le lecteur. Ces paquets comportent des repères temporels (*timestamps*) qui permettent de déduire la position dans le flux. Cependant cette position n'est qu'approximative, en effet les paquets de données émis par le serveur ne sont pas consommés immédiatement par le lecteur. Afin de compenser les fluctuations du réseau, notamment en terme de jigue, le lecteur remplit une réserve de données, appelée « tampon » (*buffer*) avant de jouer effectivement le flux. Il se crée alors un décalage entre la réception des données et leur lecture effective par le client média. Notre solution étant basée sur une approche infrastructurelle, nous ne pouvons agir sur ce paramètre ni l'estimer précisément, chaque lecteur utilisant un tampon potentiellement différent.

Ainsi, en traçant également le flux de données, nous estimons approximativement la position de lecture du média et ce légèrement en avance par rapport à la lecture réelle. L'utilisateur ne perdra pas alors de données, au contraire le flux se répètera pendant quelques secondes sur le terminal destination, lui laissant plus de temps pour achever son déplacement physique d'un terminal à l'autre.

Enfin, si pour une raison quelconque (protocole de transport non supporté ou chiffré) il est impossible d'analyser le flux de données, le *Media Proxy* peut également évaluer le temps de lecture par rapport à l'initiation de la session multimédia et aux différents messages de contrôle (lecture, pause, avance rapide, etc) qui sont tracés par le proxy. Le temps ainsi calculé sera relativement précis, cependant si l'utilisateur modifie manuellement la position de lecture du flux dans les limites du tampon accumulé, aucun message de signalisation ne sera émis et le proxy ne pourra ajuster son décompte.

**Réalisation du handover.**

Considérons alors que la position du lecteur dans le média et l'URL soient correctement collectés (cas classique implémenté avec MMS et RTSP). Une fois le transfert déclenché, le client IMS du terminal destination reçoit l'URL du média à lire. Il exécute alors le lecteur multimédia du terminal avec cette adresse généralement préfixée avec le nom de protocole utilisé (par exemple `rtsp://` ou `mms://`). Ce préfixe permet au SE d'associer un type d'URL à une application spécifique, évitant toute configuration manuelle. Le lecteur charge alors directement cette adresse et lit le flux correspondant, une requête d'initiation (SETUP avec RTSP par exemple) est envoyée au serveur média, via le *Media Proxy*, cf. Figure 4.5.

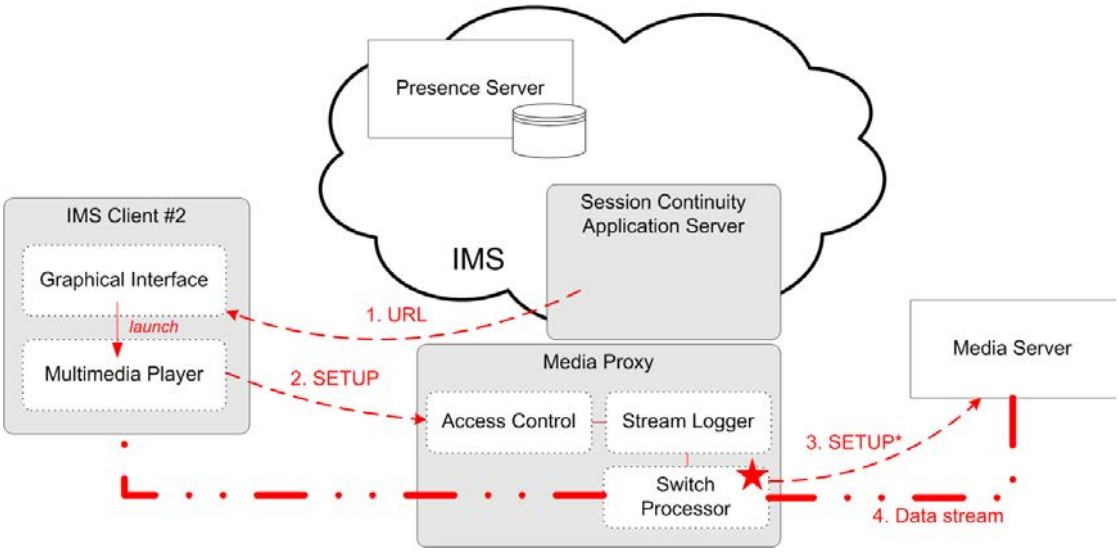



FIGURE 4.5 – Transfert d'une session multimédia.

Le *Media Proxy* qui a été notifié par le SCAS qu'une demande de transfert a été requise entre les deux adresses IP correspondant aux deux clients IMS en question se prépare à recevoir les nouvelles requêtes. Lorsque le message de lecture du flux (message PLAY par exemple pour RTSP) est reçu par le *Switch Processor*, il modifie la signalisation en y ajoutant le *timestamp* de la session précédente (champ *range* cf. Figure 4.6) et transmet le message au serveur. Le serveur de médias établit donc cette nouvelle session avec le second lecteur à la position indiquée, lorsque le *Media Proxy* constate l'établissement de la deuxième session en traçant les messages de signalisation, il interrompt le flux initial et le service multimédia est bien transféré de manière synchronisée.

```
C->S: PLAY rtsp://audio.example.com/twister.en RTSP/1.0
      CSeq: 833
      Session: 12345678
      Range: smpte=0:10:20-;time=19970123T153600Z

S->C: RTSP/1.0 200 OK
      CSeq: 833
      Date: 23 Jan 1997 15:35:06 GMT
      Range: smpte=0:10:22-;time=19970123T153600Z
```

FIGURE 4.6 – Ajout du *timestamp* dans le message RTSP PLAY.

## 4.4 Conclusion

Cette seconde approche, moins restrictive que l'approche purement IMS offre une continuité contextuelle plus aboutie. Une certaine liberté est accordée aux terminaux : le prototype pouvant gérer de manière transparente plusieurs protocoles, le lecteur multimédia adéquat est sélectionné en fonction du protocole du service transféré. Le service multimédia est ainsi dissocié de son implémentation offrant une capacité d'adaptation plus importante. Cependant, seul le flux et et la position de lecture sont ici continués, comme vu dans le chapitre précédent, d'autres informations propres à l'application et enrichissant le service sont ignorées dans le transfert : configuration (paramètres de langue/sous-titres), historique de lecture, signets, etc.

D'un autre côté, la continuité temporelle n'est pas parfaite, la solution proposée au niveau infrastructure (en tant que proxy) et l'absence de mécanisme

standard synchrone à l'instar des services de télécommunications, entraînent une synchronisation non garantie au niveau du terminal. En fonction du protocole, les manipulations du flux au niveau terminal (mémoire tampon, déplacement dans le flux, pause) ne sont pas toujours perçues par le proxy, un transfert éventuel positionnerait le flux à un emplacement approximatif. Une solution plus proche du terminal utilisateur résoudrait ce problème, encore faut-il avoir le contrôle du client, ce qui n'est pas le cas avec l'IMS sans se démarquer de la norme.

Il est à noter que l'approximation de la continuité temporelle est ici moins grave que dans le cadre de services de télécommunications dans la mesure où l'utilisateur est ici unique (pas d'interlocuteur) et que le mouvement physique de ce dernier accorde une délai suffisant pour gérer la synchronisation. Dans le cas de flux multimédias, quelques secondes de superposition des médias sur le terminal source et destination est acceptable par l'utilisateur.

L'implémentation du prototype a également mis en valeur une difficulté particulière : la nécessité d'implémenter des mécanismes spécifiques pour chaque protocole à supporter (ici RTSP et MMS). Ce problème est important car il limite le champ d'application de la solution qui est dédiée à quelques implémentations du service multimédia.

Enfin, comme pour l'approche IMS, de nombreux mécanismes outre celui de transfert ont dû être implémentés pour assurer l'orchestration de la mobilité, on peut toutefois noter que le lecteur média, lancé par le client IMS, n'a pas besoin d'être actif pour accepter un transfert. Cependant les clients IMS origine et destination doivent être actifs avant le transfert et le restent a fortiori après, tout comme le lecteur du terminal origine qui continuera d'ailleurs de jouer le média sauf si le proxy coupe intentionnellement le flux.

# Chapitre 5

# Continuité des services multimédias dans le Web

## 5.1 Motivation

Nous avons souhaité continuer notre étude sur les services multimédias mais en élargissant nos recherches au-delà du cadre IMS en nous intéressant en particulier au Web où les fournisseurs de contenus multimédias connaissent un très grand succès. L'architecture de télécommunications était un point de départ intéressant pour cette étude avec une cadre standardisé et de nombreux mécanismes assurant des fonctions essentielles telles que l'identification, le déclenchement (automatique notamment) ou encore la communication entre le client (IMS) et l'infrastructure. Cependant elle imposait aussi de nombreuses contraintes, du point de vue de la norme stricte mais également au niveau des terminaux mobiles à faible capacité et des clients peu flexibles d'où une réelle difficulté dans la mise en œuvre de modes manuels de transfert, plus proches de l'utilisateur.

De très nombreux sites proposant la diffusion de flux multimédias (*streaming*) ont émergé depuis le début de ces travaux, prouvant l'engouement des utilisateurs pour ce type de services. L'environnement Web offre en effet de nombreuses opportunités : ouverture totale, aucune limitation au niveau client, richesse des services, ... avec en contrepartie la coexistence d'un grand nombre de solutions propriétaires présentant des difficultés particulières d'interopérabilité pour la conception d'une solution globale. Cette approche Web nous permet cependant d'aborder

la mobilité sous un autre angle, d'étudier plus en détail le mode manuel mais aussi un autre genre de continuité offrant une meilleure expérience utilisateur.

## 5.2 Problématiques et approche

Le cadre de travail change radicalement par rapport aux deux chapitres précédents, nous nous dégageons totalement des contraintes du monde télécom et de l'IMS en particulier. Les limitations de performances intrinsèques aux terminaux mobiles ne sont plus applicables, nous nous intéresserons d'ailleurs uniquement à des terminaux de type « ordinateur » dotés, en comparaison, de ressources illimitées. En conséquence, nous nous focaliserons sur le cas de *terminal handover*, le changement de réseau étant peu probable dans ces conditions ; c'est ici l'utilisateur qui crée la mobilité en décidant de changer d'interface (de terminal) pour son service multimédia.

L'infrastructure IMS assure de nombreuses fonctions qui font maintenant défaut au niveau de l'identification, du déclenchement et des interfaces client-serveur ; il est nécessaire de les repenser dans le cadre Web. Cependant, le *Media Proxy* introduit dans le chapitre précédent (cf. 4.3.2) est externe à l'infrastructure télécom, nous l'avons donc naturellement réutilisé dans le cadre de cette étude. En effet, nous avons défini plusieurs fonctions de traçage, d'identification et de transfert applicables également aux services multimédias Web. Ce composant, en tant que proxy entre le lecteur média et le serveur, est capable de collecter les informations clés des messages de signalisation et des flux qui le traversent tel que l'URL, l'adresse IP ou encore la position courante du média. Il est cependant nécessaire de l'adapter au monde Web constitué de nombreuses solutions propriétaires. Les traceurs RTSP et MMS déjà définis doivent être complétés par des analyseurs spécifiques pour chaque acteur Web de services multimédias. Une gestion exhaustive étant impossible, nous avons adressé les principaux acteurs du moment, à savoir : *Youtube*, *Dailymotion* et *Google Video*. À noter que le support des protocoles MMS et RTSP permet déjà de gérer un grand nombre de sites tels que ceux des principales chaînes de télévision qui diffusent généralement en RTSP.

### 5.2.1 Déclenchement

Le déclenchement ne peut plus être réalisé via une communication standard entre le client et l'infrastructure comme dans les approches précédentes. De plus, l'absence de mécanismes de présence et d'identification tels qu'ils existent dans l'IMS limite les scénarios automatiques où le transfert était initié en fonction de l'état de présence (ou d'enregistrement) des terminaux. Ici les terminaux sont simplement équipés d'un client HTTP (ou navigateur Web) et d'un lecteur multimédia. Ce dernier peut prendre diverses formes : logiciel dédié, plug-in du navigateur ou encore une application Flash téléchargée à la demande.

L'infrastructure est ici le Web et le client est le navigateur, il n'existe évidemment aucun mécanisme spécifique prévu au transfert et à la continuité des services multimédias accessibles sur le Web. Il est donc nécessaire de créer une nouvelle interface entre le Web et l'utilisateur pour orchestrer le transfert de ces services. De plus, l'absence de mécanisme d'enregistrement et de présence nous oriente plus facilement vers un mode de déclenchement manuel, entière géré par l'utilisateur, ce qui soulève plusieurs interrogations.

### 5.2.2 Identification

Comment identifier au niveau Web/proxy le propriétaire d'une session multimédia, en d'autres termes comment associer un utilisateur à un flux ? En effet, outre les mécanismes d'authentification nécessaires pour préserver l'utilisateur et ses services, il faut également identifier toutes les composantes du système pour permettre un mode manuel : terminaux, utilisateurs et services. Si le service peut être identifié par l'URL et le terminal par son adresse IP, il reste encore l'utilisateur. Enfin, le service n'est pas la même chose que la session (le flux) qui en est une instance liée à un terminal (ou à l'utilisateur). Nous verrons par la suite que nous adoptons une approche orientée Web qui ne nécessite pas d'identifier ni de désigner les sessions.

Une autre question se pose : comment l'utilisateur désigne une session, celle-ci n'existant qu'au travers d'une URL qui est généralement particulièrement incompréhensible pour l'utilisateur. Au niveau signalisation, seule l'URL est perçue, les informations contenues dans les pages Web n'étant pas exploitables de par leur incomplétude et leur volatilité ; seul l'utilisateur peut donner du sens à ces médias.

Bien qu'il ne regarde généralement qu'un média à la fois et que la désignation peut sembler inutile, nous verrons dans la section suivante qu'il devient nécessaire d'apporter du sens aux noms de sessions dans un environnement Web. Si ces problèmes existaient déjà dans les approches précédentes elles étaient résolues en partie grâce à des mécanismes de déclenchement automatiques ou semi-manuel.

### 5.2.3 Style Web 2.0

Cette étude porte sur un nouvel environnement et il est nécessaire de concevoir des solutions de mobilité qui s'adaptent parfaitement à ce nouveau cadre pour une expérience utilisateur optimale. Le Web s'est récemment approché d'un modèle très interactif, collaboratif, faisant de chaque consommateur de contenu un producteur potentiel, l'ensemble des technologies et l'état d'esprit même de cette approche sont usuellement qualifiés de « Web 2.0 » [86]. Nous avons cherché à insuffler aux mécanismes de continuité définis dans les deux chapitres précédents, relativement « rigides », cet esprit 2.0 en favorisant la communication, la collaboration et l'intégration dans l'environnement Web.

Nous avons recherché un mode de continuité différent, qui privilégie la continuité contextuelle à la continuité temporelle, étudier par exemple une mobilité asynchrone des services multimédias, tel un livre que l'on arrête de lire puis que l'on reprend plus tard au même endroit grâce à un marque-page. De nouvelles possibilités s'offrent alors, « marquer » les flux médias, les enrichir avec des métadonnées (*metadata*), les partager, ouvrir à d'autres applications, etc.

Ainsi est née la notion de *media bookmark*. Un *media bookmark* est un repère dans un service multimédia qui permet à l'utilisateur d'arrêter la lecture et de le reprendre au même endroit dans le flux depuis n'importe quel terminal, et pourquoi pas sur celui d'un autre utilisateur. Ce mode change radicalement de ceux plus classiques présentés jusqu'alors, il respecte pourtant la notion de continuité, de mobilité (*terminal handover*), seule la synchronisation n'est pas assurée ni recherchée d'ailleurs par l'utilisateur.

## 5.3 Implémentation

Nous avons implémenté un prototype composé en plusieurs parties. Le client est réalisé sur *Google desktop* [87], un environnement logiciel où peuvent être déployés des plug-ins (*Google Gadgets*) : des applications qui s'intègrent ainsi au bureau du système d'exploitation du terminal utilisateur. Cet environnement a été choisi car il présente une solution orientée Web qui offre un cadre permettant le développement et l'implémentation rapide d'applications. Le client que nous avons développé ici s'appelle *Google Gadget Media* (GGMedia), supposé toujours actif sur le terminal, le *Google desktop* étant généralement exécuté automatiquement au démarrage du SE. Le client se connecte à un serveur Web, le *Bookmark Server*, lui même associé au *Media Proxy* que nous ne présentons plus (cf. 4.3.2) très légèrement amélioré.

### 5.3.1 Scénario

Les scénarios sont multiples, voici un exemple d'utilisation orienté Web de notre solution, nous détaillerons l'architecture et les mécanismes par la suite.

Lionel navigue sur le Web et décide de regarder le résumé des évènements sportifs de la semaine retransmis sur le site officiel d'une grande chaîne de télévision. Il clique sur le lien du flux et le lecteur multimédia embarqué dans son navigateur joue la vidéo. Une notification provenant de son GGMedia apparaît alors, lui indiquant qu'une session a été détectée. Il doit cependant se déconnecter mais souhaite mémoriser la position de lecture courante, il affiche alors son GGMedia (localisé dans la barre des tâches *Google desktop*) et double-clique sur la session qui s'ajoute à sa liste de *bookmarks*.

Plus tard, il rallume son ordinateur, affiche le GGMedia qui liste l'ensemble des ses bookmarks multimédias dont celui du résumé sportif. Il double-clique dessus et son lecteur multimédia démarre, affichant le flux vidéo à l'endroit même où il l'avait laissé.

Enfin, tellement enthousiasmé par l'exploit d'un coureur français, il décide de donner au bookmark un nom plus explicite (« Djhone qualifié ! ») et y ajoute un petit commentaire « *La course fabuleuse.* », enfin il rend le bookmark public (via son GGMedia) afin de le partager avec ses amis. Ces derniers verront alors le nou-

veau bookmark de Lionel et pourront le contacter d'un simple clic, les informations de contact étant incluse dans le bookmark partagé.

La figure 5.1 est une capture d'écran du bureau de Lionel pendant qu'il regarde une vidéo sur le Web. Au milieu, la fenêtre du navigateur est ouverte sur un flux média, on peut remarquer sur la droite la barre *Google desktop* contenant le client GGMedia. Le client affiche la session en cours mais également deux bookmarks que Lionel a précédemment créés et déjà enrichis avec de métadonnées.

Les quelques actions présentées dans ce scénario ne sont qu'un aperçu des possibilités d'une telle solution, qui, grâce aux fonctions offertes par le Web, sont illimitées.

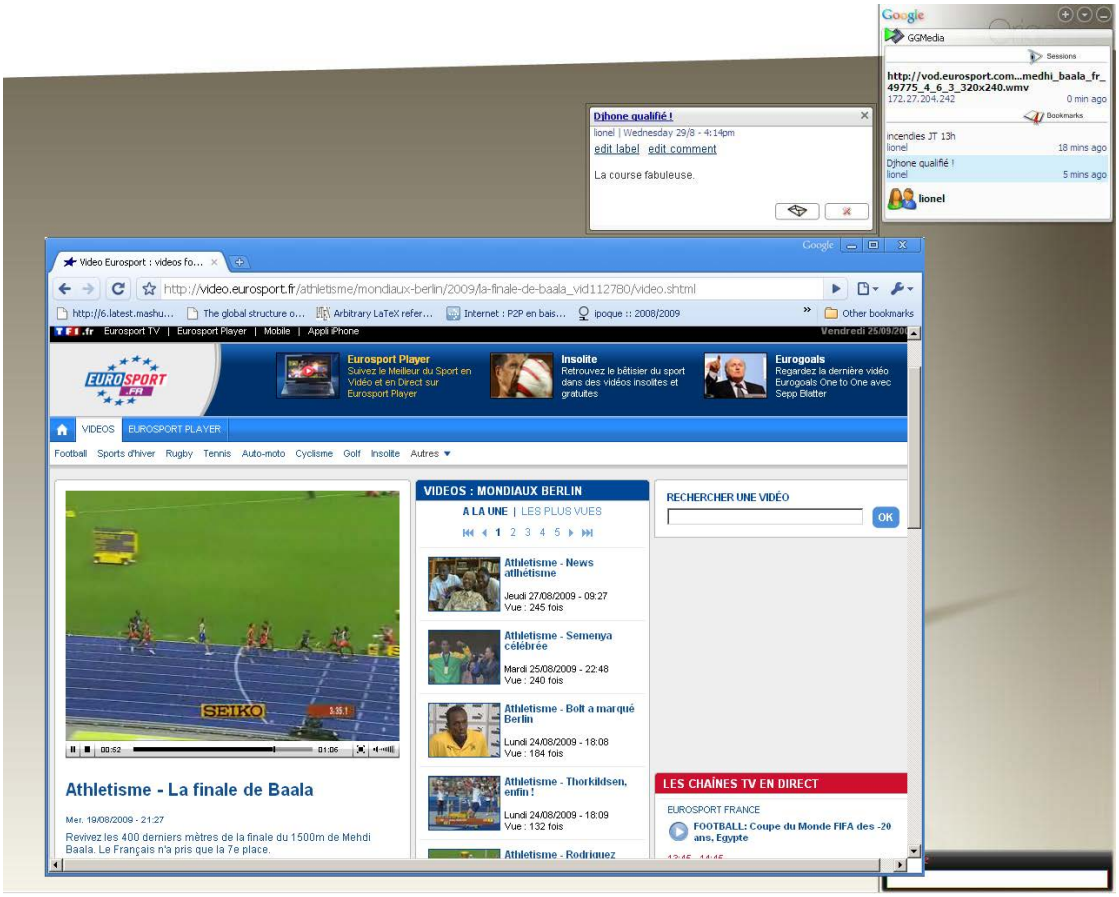



FIGURE 5.1 – Illustration du scénario.

### 5.3.2 Architecture

Les différents acteurs de ce scénario sont les suivants :

- le terminal de Lionel doté de :
  - un navigateur Web,
  - un lecteur multimédia (application dédiée ou plug-ins),
  - le client GGMedia (dans son environnement *Google desktop*) ;
- le site du fournisseur Web de contenu multimédia (joue le rôle de serveur média),
- le *Media Proxy* positionné entre le client et le site Web,
- le *Bookmark Server*, serveur Web associé au *Media Proxy*.

Nous allons nous intéresser à présent aux mécanismes clés qui entrent en jeu parmi ces acteurs représentés de manière schématique en Figure 5.2.

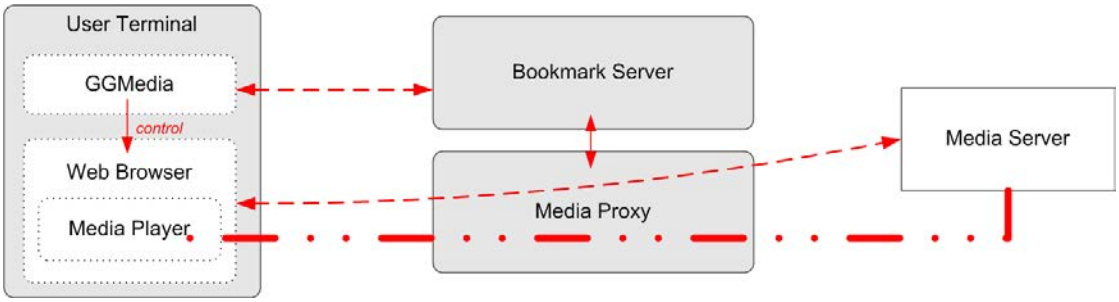



FIGURE 5.2 – Vue globale des différentes composantes du système.

**Identification.**

Le service de continuité est ici aussi uniquement fourni aux utilisateurs qui y ont souscrit, il est donc nécessaire d'identifier les clients pour assurer un contrôle d'accès permettant la commercialisation du service, la protection de la vie privée ou encore la mise en œuvre de mécanismes adéquats. L'identification est réalisée entre le client GGMedia et le *Bookmark Server* via un secret partagé délivré à la souscription. Des échanges via le protocole HTTP transportant des messages de signalisation propriétaires (nous contrôlons le client et le serveur) permettent au client de communiquer l'adresse IP du terminal qui est alors stockée par le *User Manager* (dans la base de données utilisateurs *User data*) et autorisée au niveau

du proxy, cf. Figure 5.3. Le *Media Proxy* pourra par la suite associer les sessions multimédias destinées à cette adresse IP à l'utilisateur en question.

Une fois le client identifié, l'interface GGMedia-*Bookmark Server* permet le transfert de différentes informations concernant les sessions ou les bookmarks.

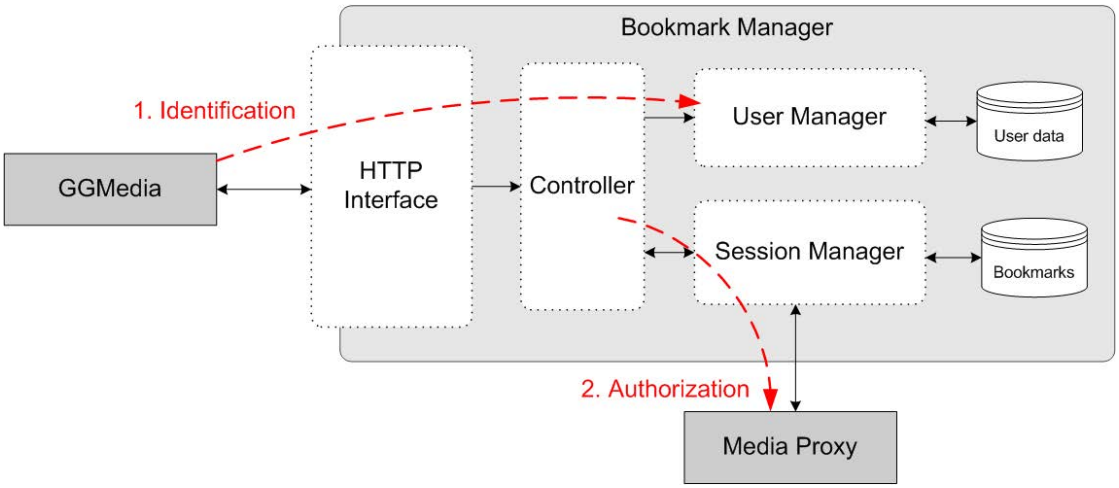



FIGURE 5.3 – Identification du GGMedia dans le *Bookmark Manager*.

**Traçage des sessions.**

Le *Media Proxy* joue toujours son rôle de proxy transparent entre le client et le serveur de médias. Or ici le client est double, il est à la fois le navigateur Web mais aussi le lecteur multimédia. Il doit donc savoir analyser plusieurs protocoles : HTTP, mais aussi RTSP, MMS, les messages de signalisation propriétaires (lecteurs basés sur la technologie Flash) et enfin les flux médias (RTP). Lorsqu'un flux média est requis par le navigateur Web, les requêtes correspondantes doivent passer par le proxy, il est donc nécessaire que le navigateur soit configuré de manière appropriée (disponible sur toutes les applications de ce type).

Le *Stream Logger* et le *Switch Processor* présentés précédemment (cf. 4.3.3) doivent assurer différentes fonctions spécifiques à chaque protocole afin de capturer les informations permettant un transfert futur. Devant la multitude de solutions multimédias propriétaires présentes dans le Web, la continuité ne pourra être réalisée que pour un nombre limité de protocoles dont les mécanismes sont connus a priori. Nous avons étudié et implémenté les mécanismes de transfert pour les prin-

cipales solutions RTSP, MMS, *Youtube*, *Google Video* et *Dailymotion* afin de couvrir la large majorité des services existants, cependant l'implémentation du proxy a requis une attention particulière afin qu'il se comporte ne manière transparente vis à vis des protocoles non supportés. En effet, si la continuité n'est pas réalisable, il est indispensable que les mécanismes de mobilité restent imperceptibles pour l'expérience utilisateur.

L'étude des protocoles propriétaires de Youtube, Dailymotion et Google Video a requis l'analyse des échanges HTTP entre le client et le serveur. Ainsi c'est par un premier travail de *reverse engineering* qu'il a été possible de retrouver dans les échanges protocolaires la position courante d'un flux ou la manière de démarrer la lecture à une position précise. Maîtriser ces paramètres identifiés dans le chapitre précédent (URL, position, etc) est la condition d'un transfert de session réalisable dans le cas des services multimédias. L'objectif final étant l'évaluation de la faisabilité de la continuité dans un environnement où coexistent de nombreux acteurs hétérogènes (en terme de mécanismes et de signalisation).

**Déclenchement.**

Le déclenchement du transfert est manuel, asynchrone et réalisé en deux temps. Tout d'abord l'utilisateur marque une session à une position spécifique en créant un bookmark depuis l'interface graphique du GGMedia. Il a alors la possibilité d'ajouter des informations (métadonnées) qu'il considère pertinentes : un titre, des commentaires, des tags, une image ... dans un pur style Web 2.0. À noter que les métadonnées peuvent également être ajoutées par le *Bookmark Manager*, cf. 5.3.2 ; la Figure 5.4 présente un exemple de bookmark enrichi.

La demande de création du marque-page média et les métadonnées sont envoyées vers le *Bookmark Manager* qui récupère l'URL de la session et le *timestamp* correspondant à la position actuelle via le *Media Proxy*, cf. Figure 5.6. Le *Session Manager* stocke ensuite le bookmark ainsi créé et le communique à tous les clients GGMedia afin qu'il soit affiché à l'utilisateur, la Figure 5.5 présente un exemple de message envoyé aux clients décrivant les sessions et les bookmarks de l'utilisateur. Cette première étape marque la session comme « continuable » un peu comme la commande *Switch* dans le chapitre 3, mais cette fois-ci la position de reprise du service est définie a priori. La seconde partie du déclenchement n'est pas néces-

sairement immédiate, c'est l'utilisateur qui décide de l'initier quand il le souhaite. Il peut alors réaliser d'autres actions : marquer de nouveau la session et créer un bookmark supplémentaire, partager le bookmark avec des amis via divers canaux de communication (courrier électronique, réseau social, blog, *Bookmark Manager*, etc). Le bookmark permet ainsi de préparer le transfert mais est également un prétexte d'échange entre utilisateurs.

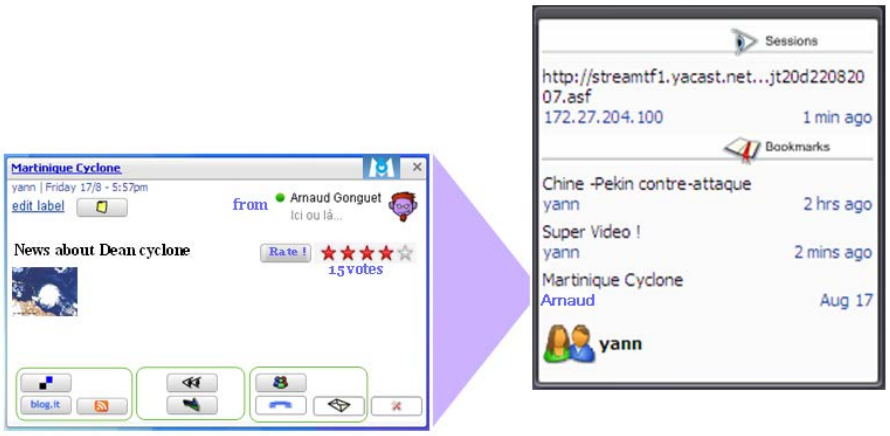



FIGURE 5.4 – Exemple de *media bookmark*

L'utilisateur active la deuxième phase du déclenchement en exécutant un bookmark depuis l'interface graphique de son client GGMedia (cf. Figure 5.7). Le bookmark peut lui appartenir ou avoir été créé et partagé par un tiers. Quoi qu'il en soit, le *Session Controller* exécute l'application correspondant au type d'URL (connu grâce à son préfixe associé par le SE à une application spécifique). Le lecteur multimédia ou le navigateur Web charge directement l'URL pour rétablir le média à la position indiquée dans le bookmark.

**Désignation.**

Cependant le déclenchement en mode manuel pose ici un problème de désignation des sessions. en effet, celles-ci sont identifiées au niveau du système par leur URL, unique pour un média donné. Mais cette information n'est pas satisfaisante pour l'utilisateur : trop longue et inintelligible. La nomination des bookmarks par l'utilisateur lors de la première phase du déclenchement est une action contraignante mais acceptable dans le cadre des bookmarks qui sont créés occasionnelle-

```
<?xml version="1.0" encoding="utf-8"?>
<userSessions username="Toto">
     <host ip="192.168.1.1">
          <session sid="452EB9856">
               <label>foo-bar</label>
               <description>TF1;Journal Télévisé;20h</description>
               <url proto="none">http://www.foo-bar.com/stream1</url>
               <created>5</created>
          </session>
          <session sid="45FG369">
               <label>foo-bar</label>
               <description>France 2;Stade 2</description>
               <url proto="http">http://www.foo-bar.com/stream2</url>
               <created>20</created
          </session>
     </host>
     <host ip="192.168.1.2">
          <session sid="89T54I">
               <label>foo-bar</label>
               <description>M6;Popstar</description>
               <url proto="mms">http://www.foo-bar.com/stream3</url>
               <created>2</created
          </session>
     </host>
     <bookmark bid="454H85A">
          <author>Bob</author>
          <label>foo-bar</label>
          <url proto="rtsp">http://www.foo-bar.com/stream1</url>
          <begin>458</begin>
          <created>85</created>
```

FIGURE 5.5 – Bookmarks (et sessions) dans un message de signalisation.

ment et ont un intérêt particulier pour l'utilisateur. Les sessions ne peuvent être gérées de la même manière, l'utilisateur serait sollicité à chaque établissement d'un flux, nuisant considérablement à son expérience du service offrant tout sauf de la continuité...

Nous avons proposé un mécanisme qui permet de résoudre ce problème grâce à l'aspect collaboratif de la nomination des bookmarks. Tous les utilisateurs peuvent enrichir leurs bookmarks avec des métadonnées, une manière de les reconnaître et de leur apporter du sens dans la liste qu'ils constituent sur leur GGMedia. Ces informations sont spécifiques à un flux, lui-même identifié par une URL. L'URL correspond donc à un (ou plusieurs) sujet/thème qui apparaîtra naturellement dans les métadonnées des utilisateurs qui marqueront ce flux. Or il se trouve que les URL des médias d'un fournisseur de contenus présentent des parties similaires, correspondant par exemple au diffuseur ou au type de programme. Ainsi, il est possible de donner du sens à ces parties d'URL en détectant les zones communes qui génèrent les mêmes métédonnées, cf. Figure 5.1.

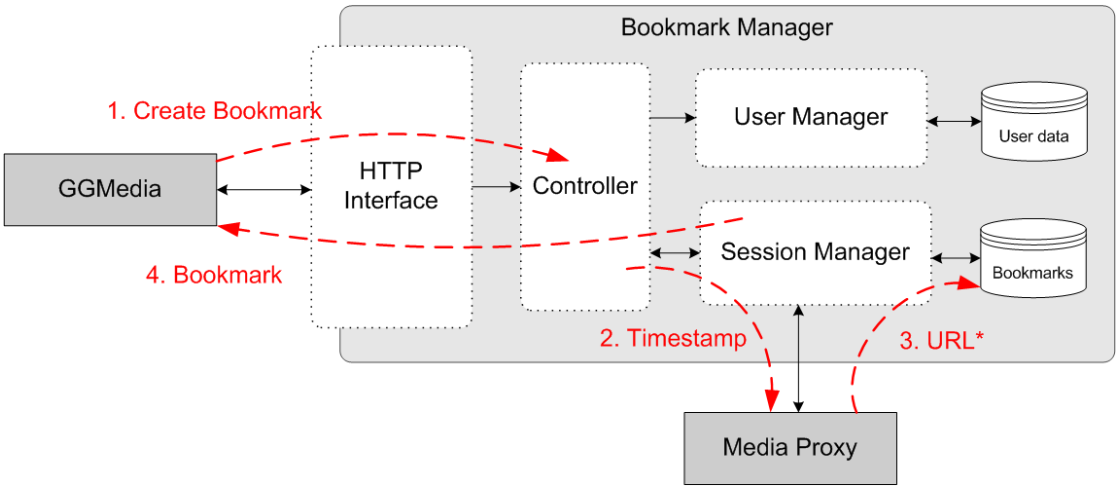



FIGURE 5.6 – Création d'un bookmark.

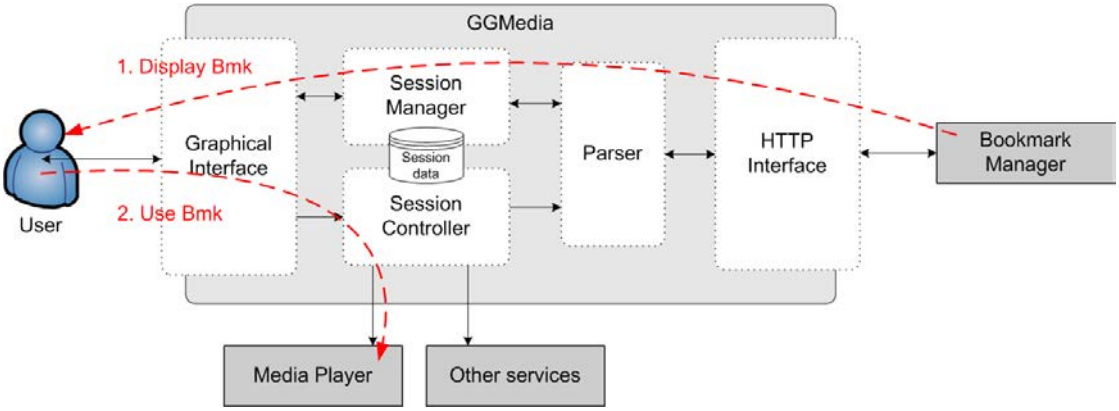



FIGURE 5.7 – Affichage et exécution d'un bookmark.

Ce mécanisme nécessite évidemment une collaboration importante afin de posséder d'une base de données significative qui permette l'identification d'un grand nombre de « blocs » (*patterns*) dans les URL et générer ainsi automatiquement des métadonnées pour les nouvelles sessions.

### 5.3.3 Mécanisme de transfert

Le mécanisme de handover est très similaire à celui présenté dans le chapitre précédent, le *Media Proxy* qui l'assurait étant également présent ici. Une différence existe cependant pour les nouveaux protocoles propriétaires supportés (Youtube,

TABLE 5.1 – Inférence de métadonnées à partir d'une URL.

| URL | Most relevant keywords |
|---|---|
| rtsp://sdmefr.media.com/jvh/23112006.wmv | TF1, journal, 20h, innondations |
| rtsp://sdmefr2.media.com/clm/21062007.wmv | TF1, clip, Gorillaz, music |
| rtsp://sdmefr.media.com/tla/matin0607.wmv | TF1, télé-achat, matin |
| rtsp://sdmefr3.media.com/jvh/11092007.wmv | TF1, journal, new york |
| **=> New URL** | **Inferred keywords** |
| rtsp://sdmefr3.media.com/jvh/02092007.wmv | TF1, journal |

Dailymotion, Google Video, etc). En effet, dans ces cas le *Media Proxy* ne modifie plus les messages de signalisation à la volée car le contrôle de ces flux se fait directement via HTTP, l'URL d'initialisation de la session (contenue dans le bookmark) peut inclure les informations de reprise, positionnant la lecture directement au bon endroit dans le flux.

Le transfert est donc préparé en amont, lors de la création du bookmark, le *Media Proxy* construit alors l'URL de reprise avec la position courante du flux (cf. 5.3.2) et l'URL de base. Le bookmark ensuite communiqué à tous les clients de l'utilisateur permet à ce dernier de reprendre le flux à tout moment simplement en chargeant l'URL du bookmark dans son lecteur multimédia sur le terminal désiré.

Cette solution est préférée à l'approche « proxy » dans la mesure où l'on contrôle le client. En effet, la modification à la volée des messages de signalisation peut avoir des effets néfastes sur le client (affichage erroné de la barre de défilement, incohérence d'état du lecteur, désynchronisation), celui-ci s'attendant à recevoir le flux de données correspondant au début du média tel qu'il l'a demandé. Cette approche n'est cependant envisageable que pour les protocoles basés sur HTTP qui permettent une lecture immédiate et directe de l'URL, seule information transmise au lecteur. Dans le cas des protocoles étudiés au chapitre précédent, l'URL ne fait que déclencher une séquence de messages de signalisation, la requête effective de lecture étant différente de l'URL initiale, rendant la transmission des informations de reprise par cette URL impossible.

## 5.4 Conclusion

Cette troisième approche dans un cadre très différent, le Web, change beaucoup des précédentes de par la gestion asynchrone de la continuité temporelle. La

synchronisation qui existait jusqu'alors entre la fin de l'instance du service sur le terminal origine et le début de la nouvelle instance sur le terminal destination est ici inexistante. Cependant la continuité existe bel et bien.

La continuité temporelle est assurée dans l'exécution du service et non dans sa fourniture, ici le flux multimédia sera repris par l'utilisateur à l'endroit exact où il l'aura laissé. L'utilisateur est l'orchestrateur du transfert, il n'existe donc plus de contrainte temporelle de cet ordre au niveau infrastructure, il déclenchera le transfert quand il le souhaitera via l'utilisation de ses bookmarks, offrant de nombreuses opportunités. On peut noter que la solution réutilisant le *Media Proxy* pour estimer la position du lecteur dans le flux entraînera les mêmes approximations que dans le chapitre précédent, lié à la mauvaise perception du contrôle utilisateur de la lecture au niveau proxy.

L'approche choisie ici se rapproche du mode Web 2.0 avec une participation et une collaboration des utilisateurs qui enrichissent le modèle de continuité matérialisé par les bookmarks. Le mode déclenchement est alors purement manuel, basé sur l'utilisateur contrairement aux deux approches précédentes, le service multimédia se prêtant particulièrement à cet exercice par l'absence de tout correspondant ou de communication nécessitant une interaction forte. Les opportunités offertes par les bookmarks et le mode Web sont illimitées et le rapprochement avec des services de communications se fait naturellement, le modèle de continuité devenant une motivation d'échange.

La continuité contextuelle est beaucoup plus marquée ici que dans les approches précédentes dans la mesure où nous nous sommes libérés totalement des contraintes de l'IMS. L'utilisateur collabore dans un mode Web 2.0 et c'est lui-même qui enrichit le service via le client GGMedia. Les commentaires, tags, titres, images, etc qui sont générés sont omniprésents d'un terminal à l'autre sans nécessiter de transfert (mêmes informations sur toutes les interfaces), on ne parlera pas de mobilité mais le contexte reste le même, à condition que l'utilisateur utilise un GGMedia (principe du profil de service cf. 2.3.1).

Si le lecteur et le client GGMedia sont relativement indépendants du protocole sous-jacent des services multimédias, il en est tout autre du *Media Proxy* qui doit implémenter des mécanismes de transfert et de traçage spécifiques pour chaque protocole supporté ; encore une fois, cette contrainte limite la mobilité des services

multimédias à quelques implémentations connues a priori. L'utilisateur ne pourra avoir une expérience uniforme pour un même service en fonction des caractéristiques du fournisseur de contenus choisi.

Enfin, quel que soit le service étudié, le mécanisme de transfert ne représente qu'un maillon du processus de mobilité, de nombreuses étapes sont indispensables pour offrir une solution complète de continuité tel que nous l'avons expérimenté via les trois prototypes présentés ici.

# Chapitre 6

# Vers un modèle générique de mobilité

## 6.1 Motivations

La continuité des services, que ce soit des points de vue temporel et contextuel ne peut aujourd'hui être assurée de manière générique dans un environnement hétérogène. Aucune des différentes approches présentées dans l'état de l'art de ce mémoire (cf. 2.3) n'ont permis de dégager de solution qui adresse cette problématique globale. L'étude de plusieurs cas, dans l'IMS puis dans le Web, avec des services de types distincts met en évidence l'absence de *modèle générique* qui traite le *Service* en tant que tel, indépendamment de son implémentation. L'absence également d'un *modèle complet* assurant la gestion de la mobilité jusqu'au bout, c'est à dire jusqu'à l'utilisateur.

Les différentes approches que nous avons étudiées ou même implémentées jusqu'à présent sont intéressantes à plusieurs titres : elles proposent des mécanismes efficaces dont nous chercherons à nous inspirer mais aussi elles mettent en évidence, dans leur réalisation, les fonctions qui manquent à une gestion complète de la continuité. S'il est un dénominateur commun entre la quasi-totalité de ces solutions, c'est certainement une mobilité appliquée à un très bas niveau, le *Service* étant délaissé au détriment de son implémentation : l'application, le protocole, la session. Le nombre important de mécanismes de mobilité purement réseau comparé aux quelques approches applicatives présentes dans la littérature est révélateur.

Certes, des mécanismes assurent aujourd'hui la mobilité de données spécifiques dans des conditions particulières mais le monde réel dans lequel vit l'utilisateur est hétérogène, composé de terminaux, de réseaux, d'applications différents, chacun possédant des caractéristiques propres qu'il est nécessaire de considérer et d'exploiter.

Ce double constat nous mène alors à considérer le *Service* d'un point de vue différent, plus haut niveau, plus abstrait, afin de dégager des propriétés générales qui puissent être exploitées par un modèle générique de gestion de la mobilité. C'est grâce à un niveau d'abstraction suffisant des concepts connus jusqu'alors qu'il sera possible de définir un tel modèle et de proposer (ou écarter) une solution de mobilité unifiée.

Ce chapitre sera dédié à la redéfinition de la notion de service, conformément à l'état de l'art présenté en première partie de ce mémoire mais vu sous un angle différent, plus haut niveau, afin de dégager des propriétés générales. Ensuite, d'après les travaux et les recherches déjà réalisés nous définirons les fonctions clés d'un modèle générique et complet de mobilité capable d'assurer la continuité des services et une expérience utilisateur optimale.

## 6.2 Une nouvelle vision du *Service*

### 6.2.1 Définitions

Il est nécessaire de reconsidérer le concept de *Service*, sans pour autant remettre en cause les définitions et les propriétés présentes dans l'état de l'art. Le terme de « service », comme nous l'avons vu, peut être employé dans de nombreux contextes et prendre des sens relativement différents. Dans le cadre de nos travaux de recherche qui se limitent au domaine des Technologies de l'Information et de la Communication, nous proposons une définition formelle du *service*, cohérente par rapport à la littérature tout en posant une base claire pour les réflexions à venir.

Cette définition doit être suffisamment précise et pragmatique pour permettre la définition d'un ensemble consistant de mécanismes tout en restant assez abstraite pour conserver une approche générique. La voici.

*Un service est un algorithme (la logique) qui traite un ensemble de ressources (le contexte) via une application, et qui délivre une fonctionnalité à part entière à au moins un utilisateur.*

Cette définition attribue au *Service* un grand nombre de propriétés dont certains nouveaux concepts que nous allons détailler.

Ainsi, un service est constitué de deux éléments : la *logique* qui est la raison d'être du service et le *contexte*, un ensemble de ressources (des données) produites et consommées par la logique. Cette représentation est indépendante de toute implémentation ; l'instanciation du service, nécessaire pour pouvoir être délivré à un utilisateur, est réalisée via une application qui implémente cette logique et gère la contexte (en totalité ou en partie), cf Figure 6.1.

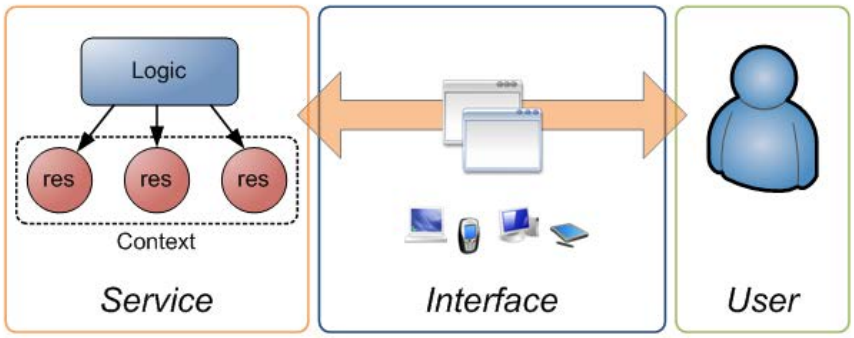



FIGURE 6.1 – Le concept de *Service*.

L'application qui peut être assimilée au système ou même au terminal hôte est qualifiée de manière simplifiée d'« interface ». Il est important de comprendre, et c'est la force de la vision du concept de service proposée, que l'interface n'est qu'une instance du service qui implémente la logique *à sa manière*. Le même service peut ainsi être perçu différemment par l'utilisateur en fonction de l'interface et de ses capacités matérielles, logicielles et environnementales, bien que la logique de service soit la même elle sera interprétée différemment. Cette propriété confère plus de souplesse et d'adaptabilité au service, l'utilisateur sera plus libre dans sa sélection d'interface qu'il ne fera plus par nécessité mais par choix. La Table 6.1 présente quelques exemples de services.

TABLE 6.1 – Exemple de services.

| Service | Instance | Host | Resources |
|---|---|---|---|
| Text-edition | MS Word, Vim, Notepad | PC, PDA, Laptop | Raw text, formating, typing history |
| Telecommunication | H.323/SIP client, Skype | Cellular, PC, car | Audio stream, call history, address book |
| Video | WMP, Flash, VLC | PDA, Set-top box, PC | Data stream, playback position, bookmarks |
| Internet browsing | IE, Firefox, Opera | PC, PDA, Cellular | URL, bookmarks, history |
| Weather forecasts | Desktop/Web widget, mobile service | PC, cellular, coffee maker | Location, language, alarms |

La logique d'un service correspond à son type, l'application *MS Word* par exemple, implémente un service d'« édition de texte ». Nous emploierons régulièrement l'exemple du service d'édition de texte car il est simple, connu par tous et s'écarte du service « cliché » réduit à des applications distantes, connectées et basées sur des sessions.

La logique du service d'édition de texte est comme son nom l'indique : éditer, formater, imprimer, ... du texte, rien de définitif, simplement les fonctions que l'utilisateur attend de ce type de service. Le contexte, pour rester dans l'exemple de l'édition de texte pourra être composé de diverses ressources : corps du texte, historique d'édition, styles, dictionnaire, etc.

Un service doit offrir une fonctionnalité « à part entière », cela signifie que la logique isolée doit avoir un intérêt pour l'utilisateur. Un programme qui consisterait par exemple à augmenter la taille de police d'un texte sera utile dans le cadre d'une application plus complexe d'édition de texte mais n'aura aucun sens de manière isolée pour l'utilisateur, il ne pourra être considéré comme un service. Il existe donc une granularité fonctionnelle minimale pour la notion de service.

Enfin, la fonctionnalité produite doit bénéficier directement à « au moins un utilisateur ». En effet, conformément à notre approche depuis le début de nos travaux, nous nous intéressons aux services au travers de l'expérience utilisateur. Certaines fonctionnalités rendent effectivement un service à l'utilisateur de manière indirecte : défragmentation de la mémoire, antivirus, pare-feu, backup, etc. Cependant ces programmes ne servent pas directement l'utilisateur, ils sont dédiés à des éléments spécifiques logiciels ou matériels, souvent dans un but de maintenance ou d'administration du SE, ils ne peuvent être qualifiés de service utilisateur. À noter que la mobilité de ce type de fonctionnalité a peu de sens, transférer une recherche de virus ou l'optimisation du disque dur d'un terminal vers un autre ?

La Figure 6.2 reprend la carte conceptuelle proposée en section 1.3.1, enrichie des nouvelles notions introduites ici.

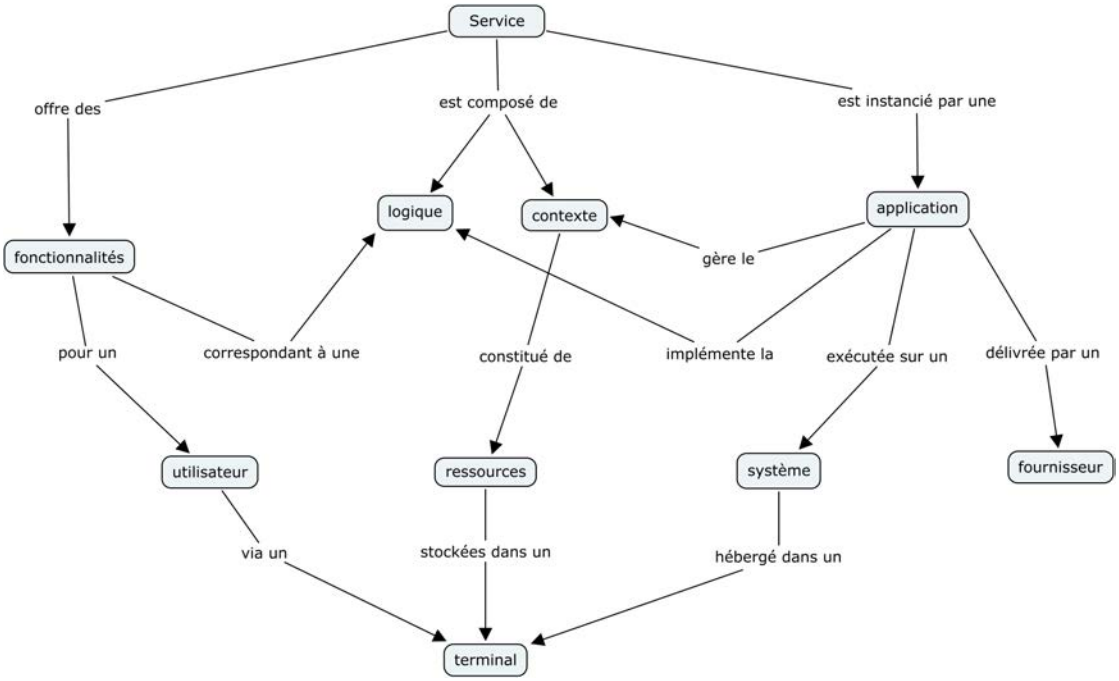



FIGURE 6.2 – Carte conceptuelle actualisée autour de la notion de *Service*.

### 6.2.2 Les ressources

La logique étant abstraite (implémentée par l'interface), le contexte qui est un ensemble de ressources est l'unique élément matériel du service. C'est donc la partie « encombrante » du service (en termes d'octets) qu'il sera question de transférer durant un handover.

Une ressource est une donnée (une suite de `1` et de `0`) stockée quelque part dans le terminal (mémoire vive, registre, disque dur, ...) et nécessaire à la logique du service en question. L'ensemble des données du service constitue le contexte. Sa taille est quelconque, du bit à la taille infinie pour le cas d'un flux continu de données. La taille d'un contexte rendra un service plus ou moins mobile, les ressources de grande taille ou de type flux poseront naturellement des problèmes particuliers qu'il faudra gérer. Les applications basées sur des protocoles de signalisation com-

plets tel que SIP disposent par exemple de mécanismes spécifiques permettant de rediriger les flux tel que nous l'avons vu dans en section 3.3.3.

**Granularité.**

L'implémentation de la logique étant laissée à la discrétion de l'application, le contexte ne sera pas toujours exploité à 100% après un transfert. En effet, une ressource peut être inutile si l'application destination ne propose la fonction correspondante. Un exemple simple pour illustrer ce cas : supposons qu'un service d'édition de texte soit transféré d'un terminal qui l'implémente via MS Word vers un terminal doté uniquement de MS Notepad pour gérer l'édition. MS Notepad est un éditeur très simple qui ne supporte que les fonctions de base, le contexte transféré provenant de MS Word, un éditeur plutôt riche, comportera un grand nombre de ressources non gérées par Notepad : styles, images, etc, cf. Figure6.3.

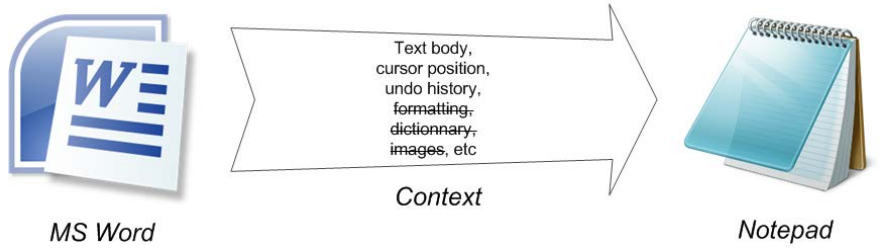



FIGURE 6.3 – Transfert de contexte entre applications hétérogènes.

Le contexte représente une quantité importante de données, l'approche applicative présentée en section 2.3.4 consistait justement à transférer un contexte maximal (à l'échelle du système) afin d'assurer une continuité contextuelle parfaite au détriment de la continuité temporelle. Avec l'exemple du transfert MS Word vers MS Notepad, on se rend rapidement compte de deux choses : premièrement le transfert du contexte peut être optimisé (des ressources sont transférées inutilement), deuxièmement la granularité de découpage des ressources dans le contexte est un facteur de cette optimisation.

Reprenons l'exemple ci-dessus, imaginons que le contexte du service d'édition de texte provenant de l'application MS Word soit le plus gros possible : une seule ressource, un fichier *.doc* (propriétarire MS Word) contenant le document complet.

La réutilisabilité d'un tel contexte par une autre application est très faible, rendant le service inadaptable. Si par contre, toutes les données du document sont séparées dans le contexte, chaque ressource sera plus petite et plus facilement assimilable par une autre application ; statistiquement une part plus importante du contexte sera adaptable à chaque transfert. On peut imaginer que lors du transfert MS Word vers MS Notepad, les ressources correspondant au contenu du texte brut et la position du curseur ont pu être gérées par le bloc notes

Un contexte détaillé, comportant des ressources minimalistes est la condition d'un transfert optimisé, une adaptation plus importante et donc une expérience utilisateur améliorée.

**Organisation.**

Nous avons vu au-dessus que par définition un contexte n'était pas forcément réutilisé dans son intégralité en fonction des capacités de l'application. Le contexte étant composé d'un grand nombre de ressources, il est nécessaire qu'il reste cohérent et consistant, on parle alors d'état « stable ». Cet état doit être atteint lorsque le contexte est transféré sinon la logique ne peut délivrer la fonctionnalité de manière correcte.

La consistance du contexte est assurée lorsque l'ensemble des ressources requises par la logique sont bien présentes et en nombre correct. La cardinalité de chaque ressource doit être respectée, par exemple un service d'édition de texte aura probablement une et uniquement une ressource correspondant au corps du document (texte brut) et un nombre indéterminé d'actions historiques (permettant d'annuler une édition précédente). De plus, il peut exister des relations entre les ressources, la présence de certaines en impliquant d'autres ; les ressources représentant l'historique d'édition ou le formatage du texte n'auraient aucun sens sans le corps du texte.

Ces contraintes du contexte de service pourraient être représentées sous forme de graphe permettant d'assurer la consistance du contexte mais aussi de vérifier qu'une application est apte à implémenter un service d'après son contexte, cf. Figure 6.4.

La consistance du contexte n'est pas suffisante pour garantir sa stabilité, il est également nécessaire de s'assurer de sa cohérence en considérant l'état de chacune

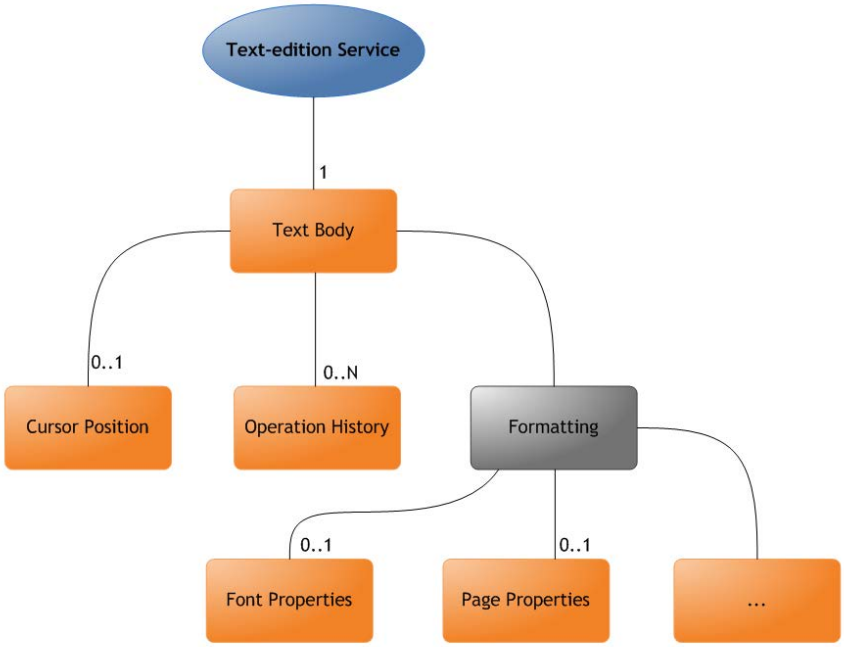



FIGURE 6.4 – Contexte d'un service d'édition de texte.

de ses ressources. En effet, pendant la fourniture d'un service le contexte est utilisé par la logique en lecture mais aussi en écriture. Les ressources étant potentiellement modifiées en continu, il peut arriver que des données soient périmées ou dans un état temporaire lors du transfert du contexte.

Dans le premier cas, le contexte est transféré alors qu'une action vient de modifier seulement une partie des ressources. Par exemple, si un transfert intervient pendant l'édition d'un texte et que la dernière action de l'utilisateur n'a pas encore été enregistrée par l'historique d'édition, les deux ressources ne seront pas cohérentes. Le contexte ainsi transféré entraînera des comportements non désirés, ici la touche d'annulation ne corrigeant pas la dernière action réalisée (mais probablement la précédente).

Dans le second cas, une ressource est en cours de modification lorsque le contexte est transféré. La donnée n'étant pas prête, deux solutions se présentent. Soit l'action en cours est annulée et la ressource (voire tout le contexte) est restaurée à sa valeur précédente avant que le contexte soit transféré ; cette approche nécessite que l'état du contexte soit sauvegardé avant d'effectuer l'action (pour

permettre l'annulation). Soit l'action poursuit jusqu'à son terme avant le transfert du contexte ; la durée de l'opération n'étant pas nécessairement connue a priori, le transfert peut être longuement retardé.

Cette nouvelle approche du concept de service offre une vision plus intuitive et de nouvelles perspectives. Il devient possible de réfléchir à un modèle de continuité tout en faisant abstraction de l'environnement hétérogène des services et essayer de dessiner des mécanismes de mobilité génériques.

## 6.3 Un modèle unifié de mobilité

D'après l'étude des solutions de mobilité présentées dans l'état de l'art et grâce à l'expérience acquise en développant des prototypes de continuité dans divers environnements, nous avons identifié les forces et les faiblesses de chacune des approches afin de dessiner les contours d'un modèle générique et complet de mobilité de service. Nous avons ainsi dégagé cinq problématiques correspondant à cinq mécanismes clés qui constituent les différentes étapes successives nécessaires à une solution globale de continuité de service. Les voici.

- **Découverte**. Identification des différents acteurs : utilisateurs, terminaux, services, etc.
- **Déclenchement**. Désignation et orchestration du transfert.
- **Capture**. Collecte des différentes informations, préparation des données au transfert.
- **Transfert**. Mobilité effective du contexte et des informations de contrôle.
- **Reprise**. Continuation du service, adaptation à l'environnement.

En nous basant sur le concept de service tel que défini dans la section précédente, nous allons détailler chacun de ces mécanismes clés. Mais avant de penser à mettre en œuvre le moindre mécanisme de mobilité, il est nécessaire de connaître l'environnement. Les solutions que nous avons successivement étudiées étaient basées sur un environnement « intelligent » avec des structures spécifiques qui offraient un certain nombre de fonctions de base : traçage, identification, déclenchement, etc. Dans notre approche générique, il n'est plus possible d'imposer des contraintes d'infrastructure qui limitent son application ; les mécanismes doivent

être indépendants et autonomes. Cependant l'environnement existe, il nous a donc fallu choisir un cadre le plus neutre possible qui fasse abstraction de toute fonction extérieure.

Cet environnement basique sera constitué uniquement de terminaux connectés entre eux de manière ad-hoc via leurs interfaces réseaux. Nous appellerons cet ensemble de terminaux connectés appartenant à un utilisateur et formant un réseau *overlay* (ou surcouche), un « environnement personnel de services » (ou *Personal Service Environment*, PSE). Le PSE est en quelque sorte matérialisé par la sphère électronique de l'utilisateur que nous avions mentionnée en introduction de ce mémoire. Les mécanismes génériques que nous allons définir s'appliquent dans le cadre d'un PSE pour un utilisateur donné, cf. Figure 6.5.

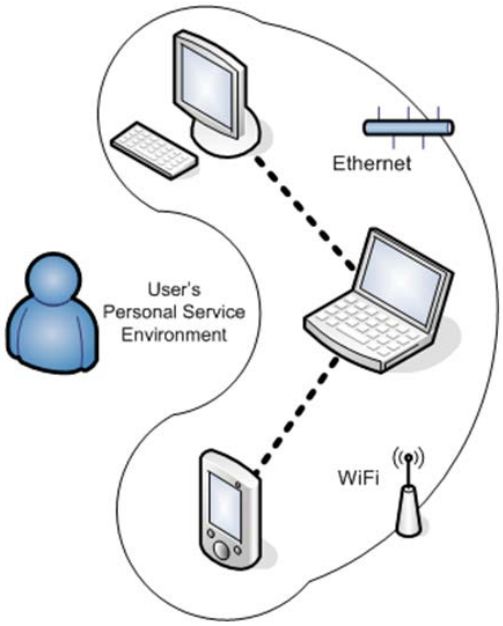



FIGURE 6.5 – Un environnement personnel de service (PSE).

### 6.3.1 Découverte

Avant de penser à mettre en œuvre le moindre mécanisme de mobilité, il est nécessaire d'identifier les acteurs en présence : terminaux et services. L'environnement étant centré sur l'utilisateur, le PSE doit être construit de manière ad-hoc

autour de lui en gérant les terminaux qui s'y ajoutent et s'en retirent. Chaque composant doit être identifié et authentifié afin d'assurer la protection des services de l'utilisateur.

Les services instanciés par les terminaux du PSE doivent également être connus par chaque interface, ainsi le PSE doit se comporter tel un système complet disposant d'autant d'interfaces qu'il est constitué de terminaux. Les interfaces doivent permettre à l'utilisateur d'accéder de manière transparente à ses services, chaque terminal doit donc avoir une vision globale du système.

La collecte d'un maximum d'informations sur ces éléments découverts est primordiale pour les étapes suivantes. Les caractéristiques détaillées des interfaces et des services permettront de guider les mécanismes de déclenchement, de transfert et d'adaptation.

### 6.3.2 Déclenchement

La seconde phase est le déclenchement du transfert. Comme nous avons vu dans les approches existantes et dans les prototypes implémentés, le déclenchement est un mécanisme particulièrement important et complexe. Il peut s'effectuer de manière manuelle, les éléments découverts à l'étape précédente seront alors désignés par l'utilisateur à partir d'une interface du PSE afin de mettre en œuvre la suite du transfert. Les modes *push* et *pull* vus précédemment (cf. 2.1.3) doivent être applicables, permettant à l'utilisateur de transférer un service précis vers l'interface locale ou un vers terminal distant.

Un mode automatique est également possible mais uniquement basé sur les informations disponibles au niveau de l'interface, il n'existe aucun composant d'infrastructure qui pourrait induire le déclenchement tel que le *Presence Server* dans l'IMS. Le détail des informations collectées dans la phase de découverte révèlent ici toute leur importance, une connaissance précise de chaque interface du PSE permettra des déclenchements efficaces tenant compte de l'environnement : faible batterie, meilleur terminal disponible, etc.

### 6.3.3 Capture

Une fois qu'un service a été choisi et que son transfert a été requis, le PSE via le terminal correspondant doit identifier l'ensemble des ressources du contexte

en question et le capturer dans un état stable. Cette capture peut se visualiser comme un « cliché » du contexte (*service snapshot*) juste avant son transfert effectif (à l'image du *bookmark média* présenté dans le chapitre 5). Ce cliché qui fige le contexte permet à l'utilisateur de retrouver des points de référence une fois le transfert réalisé, offrant à l'utilisateur une expérience similaire du service sur l'interface d'origine comme sur celle de destination.

Le terminal destination ne peut accepter qu'un contexte stable, la capture des différentes ressources doit donc être réalisée de manière efficace, cohérente et consistante (cf. 6.2.2) ce qui est parfois difficile considérant l'hétérogénéité des ressources.

### 6.3.4 Transfert

Vient alors le mécanisme de transfert qui consiste à envoyer l'ensemble des données nécessaires à la reprise du service d'une interface du PSE à une autre conformément à la désignation réalisée pendant la phase de déclenchement. La Figure 6.6 illustre le principe de transfert basé sur la définition de service haut niveau que nous avons proposée en début de ce chapitre. La logique de service est dissociée de son implémentation, seul le contexte est déplacé de l'application origine vers l'application destination qui peut être différente mais implémente nécessairement la même logique.

Le contexte à transférer pouvant être de taille importante ou comporter des ressources de type flux continu de données, des mécanismes spécifiques et optimisés doivent être mis en œuvre pour masquer la complexité et l'hétérogénéité des ressources. La conception même du contexte décrite en section 6.2.2 apporte déjà quelques pistes que nous verrons en détails dans le chapitre suivant 7 qui précise leur implémentation.

### 6.3.5 Reprise

La reprise du service est certainement la phase la plus critique mais également la plus importante du mécanisme de transfert. Ce mécanisme est généralement négligé par les solutions que nous avons étudiées pour la simple raison que la continuité contextuelle est peu adressée et que les environnements d'origine et de

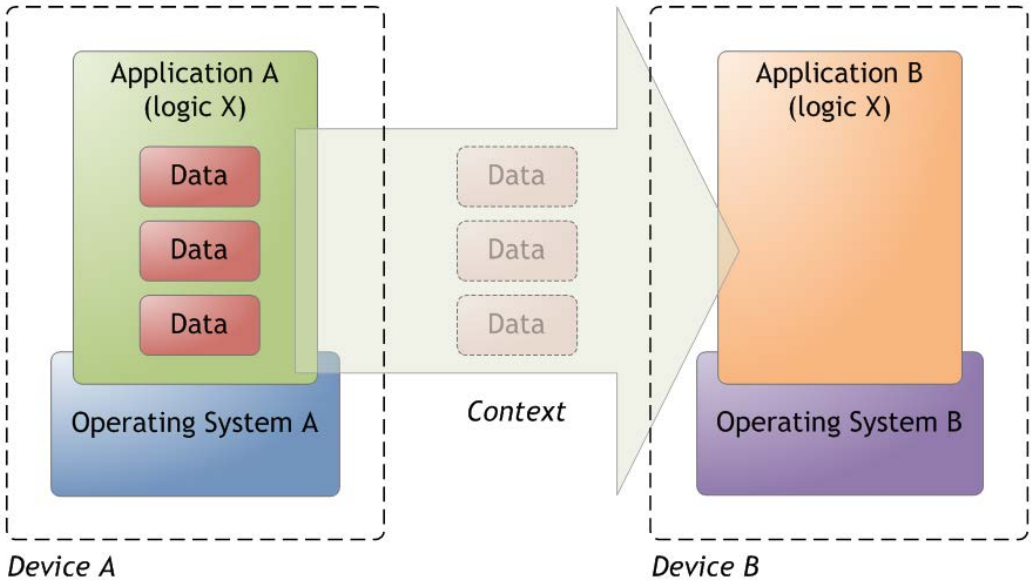



FIGURE 6.6 – Principe de transfert.

destination sont généralement identiques. Avec l'objectif d'une expérience utilisateur optimale, la continuité contextuelle est pour nous une priorité, de plus le PSE est intrinsèquement hétérogène et ce à tous les niveaux. Cette phase étant la dernière dans le chaîne de fonctions du processus de mobilité, celle qui a un impact direct sur l'utilisateur.

Alors en quoi consiste la reprise du service (ou *service resume*) ? Tout d'abord, le contexte qui a été transféré lors de la phase précédente doit être transmis à l'application qui gère la logique en question. La logique connue depuis la phase découverte ayant permis l'éligibilité et la désignation de cette interface comme destinataire du transfert. L'application destination est identifiée et automatiquement exécutée afin de continuer le service en mobilité.

Les mécanismes qui entrent en jeu ici assurent l'adaptation du service aux propriétés de la nouvelle interface, en d'autres termes ils permettent à la nouvelle application d'exploiter le contexte transféré et de restituer à l'utilisateur une expérience comparable à celle offerte par l'interface d'origine. L'efficacité de cette étape dépend directement de la présentation du contexte : granularité et organisation des ressources, cf. 6.2.2.

## 6.4 Conclusion

Ce premier travail nous confère une vision à la fois suffisamment abstraite et pragmatique de la notion de *Service*. Il nous a permis d'appréhender avec précision les relations qui existent entre l'utilisateur, ses terminaux, ses applications et ses services. Nous avons analysé de manière conceptuelle et abstraite, indépendamment de leur type, les notions de logique et de ressource afin d'identifier un certain nombre de propriétés qui se révèlent essentielles dans la gestion de la mobilité. Nous avons présenté les fonctions (haut-niveau) clés d'un modèle de continuité générique adapté à un environnement hétérogène.

Nous avons également introduit une vision neutre de l'environnement, le PSE, qui présente les services de l'utilisateur au centre d'un système accessible via ses terminaux qui ne sont finalement que des interfaces. On y retrouve l'idée de « sphère électronique » qui entoure l'utilisateur (cf. *Introduction générale*). L'objectif de la continuité dans ce cadre est de permettre à l'utilisateur d'accéder librement et de manière transparente à ses services mais de manière adaptée à l'interface choisie. L'adaptation est essentielle, elle offre des points de référence à l'utilisateur dans sa relation avec le service mais surtout elle exploite l'hétérogénéité de l'environnement afin de justifier la mobilité de service.

Le travail de définition et de positionnement des concepts présentés dans ce chapitre nous a permis de prendre de la distance par rapport à l'état de l'art qui n'apportait pas de solution satisfaisante tout en tirant des enseignements des mécanismes existants. Le modèle générique proposé ici est ambitieux et n'est pas sans poser de nombreux problèmes, nous avons donc entrepris une implémentation et un prototypage afin d'évaluer sa faisabilité et d'identifier les défis techniques qui se présentent. Ce sera l'objet des deux prochains chapitres.

# Chapitre 7

# Le *distributed Service Manager*

## 7.1 Motivations

Nous avons défini un modèle générique basé sur une nouvelle approche du concept de service (cf. chapitre 6). De nombreuses propriétés et concepts ont été dégagés de cette étude, nous permettant de dessiner les mécanismes clés d'un telle solution globale capable d'assurer la continuité dans un environnement hétérogène. Cependant nous n'avons pour l'instant proposé qu'une approche conceptuelle, de haut niveau, qui adresse les problématiques rencontrées jusqu'à présent de manière théorique.

Ce chapitre sera consacré à la définition concrète d'une solution directement issue de ce modèle, en précisant les mécanismes nécessaires aux différentes phases identifiées. Nous proposerons une architecture et détaillerons les fonctions de mobilité correspondantes tout en restant focalisé sur une contrainte : une expérience utilisateur optimale.

## 7.2 Principe

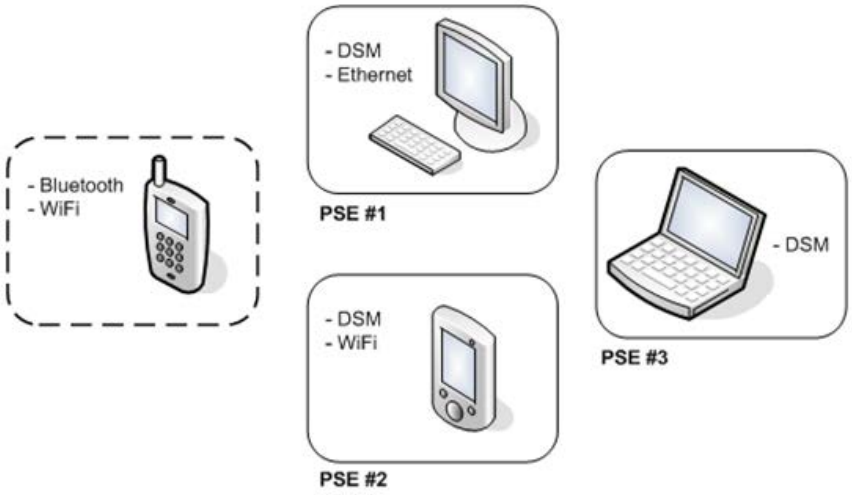
L'implémentation du modèle que nous proposons est appelée Gestionnaire de Service Distribué (noté dSM pour *distributed Service Manager*). Comme son nom l'indique, et logiquement par rapport à l'absence d'infrastructure dans le PSE, nous adoptons une approche distribuée. La logique de mobilité décrite de manière abstraite dans le chapitre précédent et qui existe au sein d'un PSE est implémentée

de manière collaborative, distribuée, par chaque interface qui le composent. Ces travaux ont été publiés dans [88].

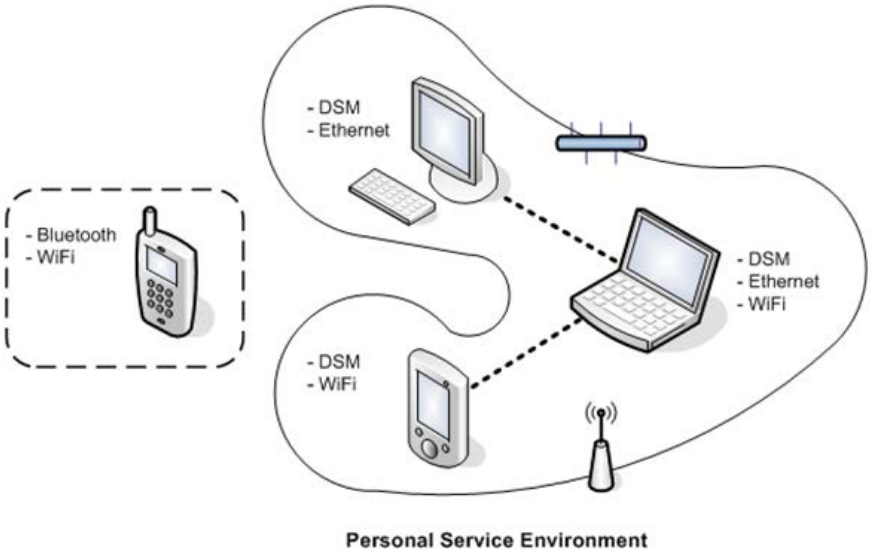
Le dSM est donc une application distribuée présente dans tous les terminaux du PSE qui collaborent entre eux pour réaliser les mécanismes de mobilité. Chaque instance est en charge du contrôle des applications et des ressources locales, elle apparaît donc comme une fonction supplémentaire au système d'exploitation de l'hôte. Un réseau pair-à-pair est ainsi établi grâce au dSM, incluant l'ensemble des interfaces du PSE. Ce réseau est géré par un protocole réseau ad-hoc capable d'assurer les fonctions de base : ajout et suppression de nouveaux pairs (terminaux de l'utilisateur), identification et authentification (pour protéger l'utilisateur et ses services), routage des messages de signalisation entre les interfaces (le PSE pouvant couvrir plusieurs réseaux physiques IP), etc. Les Figures 7.1 illustrent comment les environnements personnels de services d'un utilisateur sont gérés en fonction de la connectivité des dSM embarqués dans chaque interface : la Figure 7.1(a) montre plusieurs dSM déconnectés qui forment autant de PSE puis la Figure 7.1(b) montre les même terminaux dont la connectivité a été établie via une technologie d'accès commune (hôte de droite) et forme alors un unique PSE. On peut noter également que l'unique terminal dépourvu de dSM est exclu de l'environnement et ne pourra bénéficier des mécanismes de mobilité.

Afin de garantir la généricité, une partie des mécanismes est assurée par les applications, imposant une contrainte sur celles-ci qui doivent être compatible avec le dSM pour que les services qu'elles délivrent soient transférables. Cette approche est inévitable et ce pour deux raisons. Premièrement nous devons gérer un environnement hétérogène, il est donc difficile de connaître a priori le comportement et les propriétés de chacun de ses composants. Il est plus évident d'adresser les problématiques de l'intérieur, l'application par exemple connaît la logique de ses processus internes, il est donc judicieux de lui conférer une partie du mécanisme de mobilité. Deuxièmement, nous apportons une nouvelle fonction de mobilité à chaque terminal, constituant une sorte d'amélioration du système existant, exploitable par les applications qui le souhaitent ; la contrainte majeure étant de garantir une transparence totale pour les anciennes applications non compatibles (qui seront simplement ignorées du PSE).

L'impact sur ces applications compatibles est cependant minimal, nous détaillerons ces contraintes dans la section 7.2.2 tandis que le chapitre suivant 8 apportera




(a) Terminaux déconnectés formant plusieurs PSE.




(b) Terminaux connectés formant un PSE unique.

FIGURE 7.1 – Gestion des environnements personnels de service.

des données précises sur cet impact grâce à un prototype. Mais tout d'abord, un exemple simple du dSM en action.

### 7.2.1 Exemple

Afin d'illustrer l'utilisation du dSM dans un cas concret et avant d'entrer dans les détails techniques, nous allons présenter un exemple simple qui reprend le cas du service d'édition de texte.

Bob est dans le bus. Il profite du trajet pour rédiger un courrier sur son PDA (*Personal Digital Assitant*) grâce à une application très basique qui ne gère que du texte brut. Bob arrive enfin à son bureau, il met le PDA dans sa poche et allume son ordinateur. Une fois démarré, le PC active le dSM qui détecte aussitôt le PDA également doté d'un dSM. Via la connectivité Wi-Fi de l'environnement de travail, Bob vient d'établir un *pse* constitué de deux interfaces : le PC et le PDA. Dès lors les applications compatibles exposent leur service au PSE conférant à Bob une liberté totale de mouvement.

Bob est notifié sur son ordinateur qu'un service d'édition de texte appelé « *lettre au boss* » a été découvert dans le PSE. Sachant pertinemment qu'il s'agit du courrier qu'il a commencé à rédiger plus tôt dans le bus, il double-clique directement sur la notification et MS Word se lance, et une seconde plus tard la lettre en cours de rédaction s'affiche à l'écran. Il peut alors faire quelques ajustements, impossibles depuis le PDA : mise en page, correction orthographique, et enfin imprimer le document prêt à poster !

### 7.2.2 Les applications

Les applications doivent donc être compatibles avec le dSM pour que les services fournis soient considérés par le PSE, et donc que les mécanismes de continuité soient applicables. En effet, le dSM doit être capable de manipuler les applications et les ressources durant tout le processus de mobilité : notifier de l'état des services aux interfaces distantes, mettre en pause un service et récupérer le contexte, le transférer, lancer ou arrêter une application, reprendre l'exécution d'un service, etc. La plupart de ces fonctions peuvent être réalisées sans collaboration spécifique (démarrer ou arrêter un programme par exemple), cependant les actions portant

sur les propriétés du service et les ressources, particulièrement proches de l'implémentation requièrent la participation de l'application qui est la mieux placée pour apporter des informations correctes.

Des interfaces sont donc définies afin d'encadrer les échanges entre le dSM et l'application, précisant de manière indépendante au type de service, la nature des fonctions requises dans une application compatible. Parmi ces fonctions, deux sont essentielles : la fonctions *pause* et *resume*.

**Fonction *pause*.**

La fonction *pause* correspond en partie au mécanisme de *capture* défini dans le modèle générique (cf. 6.3.3).

Lorsqu'un transfert est déclenché, le dSM du terminal d'origine appelle la fonction pause de l'application qui instancie le service en question. Celle-ci interrompt alors la fourniture du service dès que possible et fournit un contexte stable au dSM (que ce dernier transfèrera au dSM du terminal destination).

L'application est uniquement interrompue (en pause), l'utilisateur ne peut plus bénéficier du service (ou de manière très limitée), cependant l'application n'est pas nécessairement arrêtée car le contexte doit rester intact et disponible en lecture pour le dSM. La mort des processus de l'application pourrait entraîner la fermeture de flux de données d'où une perte de ressource dommageable. De plus, si le transfert échoue pour une raison quelconque, l'application doit reprendre son cours sur le terminal origine avec le contexte préservé. Pendant la pause et jusqu'à la reprise du service à distance (signant le succès du transfert), le contexte ne peut pas être modifié, ce qui entraînerait le cas échéant une incohérence entre les états du service sur les terminaux origine et destination

Il est à noter que l'interruption causée par la pause de l'application est généralement très courte (directement liée au contexte) et masquée par le déplacement physique de l'utilisateur entre les deux interfaces désignées pour le transfert.

**Fonction *resume*.**

La fonction *resume* assure en partie le mécanisme de *reprise* défini dans le modèle générique (cf. 6.3.5).

La *pause* de l'application du terminal origine correspond au *resume* du terminal destination. Une fois le transfert du contexte effectué d'une interface à l'autre, l'application censée instancier le service en mobilité sur le terminal destination est automatiquement lancée. Elle est exécutée dans un mode spécial (reprise) qui prend en paramètre le contexte. L'application doit alors exploiter au maximum les ressources fournies, dans la limite de ses capacités (et de celles du terminal hôte), afin de restituer à l'utilisateur les références contextuelles du service adaptées à son nouvel environnement.

On constate ici que le dSM conserve une correspondance entre les types de services et les applications locales. Cette association peut être effectuée lors de l'installation de l'application ou lors de chaque utilisation, via le système d'exploitation ou directement auprès du dSM.

**Autres fonctions.**

Outre les mécanismes *pause* et *resume* qui sont les deux principales fonctions que doit implémenter les applications compatibles et qui réalisent en partie les phases de mobilité définies dans le modèle générique, d'autres fonctions sont nécessaires permettant notamment au dSM et à l'application d'échanger des messages de signalisation (orchestration du transfert, collecte d'informations, etc).

Durant la phase de découverte par exemple, de nombreuses informations concernant les services existants (nom, type, version, identifiant, etc) sont transférées entre les terminaux afin de notifier l'utilisateur et d'initier éventuellement des déclenchements. On reviendra également dans la section suivante sur certaines fonctions nécessaires à la gestion des ressources au niveau applicatif.

### 7.2.3 Gestion des ressources

La gestion des ressources est la problématique centrale de la mobilité de service tel qu'identifié dans le chapitre précédent 6. La continuité nécessitant le transfert du contexte, un mécanisme spécifique a été imaginé afin de régler les questions d'optimisation et d'adaptation. Nous avons mentionné à plusieurs reprises, par souci de simplification, que le contexte transitait d'un terminal à un autre. La réalité est plus nuancée. Lors d'un transfert, le contexte est d'abord figé dans un état stable, empêchant toute modification ultérieure des ressources (fonction *pause*

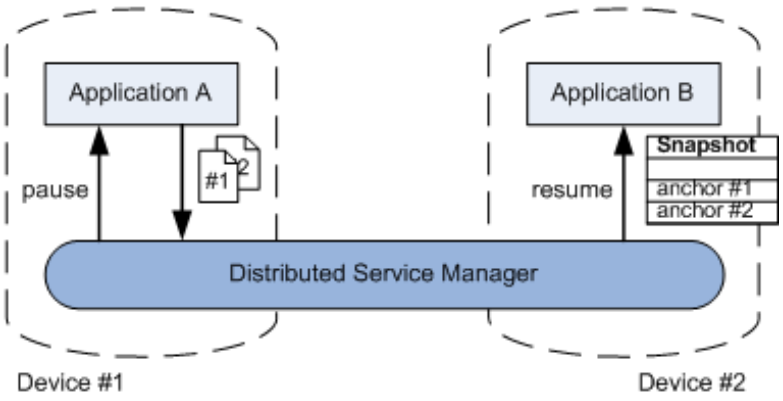
7.2.2). Chaque ressource est alors associée à un identifiant, un pointeur (ou *anchor*) unique pour une instance dSM donnée. Un cliché (*snapshot*) du contexte comportant uniquement la liste des pointeurs est alors réalisé et c'est ce *snapshot* qui est envoyé au dSM du terminal destination. D'où la nécessité également de conserver, jusqu'à la fin complète du transfert, les ressources intactes sur le terminal origine.

L'application destination est alors exécutée en mode *resume* avec comme paramètre le *snapshot* contenant les pointeurs de chaque ressource du contexte (cf. 7.2.2). En fonction de ses capacités, l'application demande au dSM local les ressources prioritaires qu'elle désire. Les dSM vont alors collaborer pour transmettre, à la demande, toutes les ressources requises et correspondant aux besoins du terminal destination. Enfin, les ressources qui n'auront pas été transférées car non supportées ou optionnelles le seront de manière automatique et asynchrone en « tâche de fond », de façon totalement transparente, afin de libérer progressivement le terminal origine. La Figure 7.2 illustre le transfert de ressources via les *anchors* entre deux terminaux lors des fonctions *pause* 7.2(a) et *resume* 7.2(b).

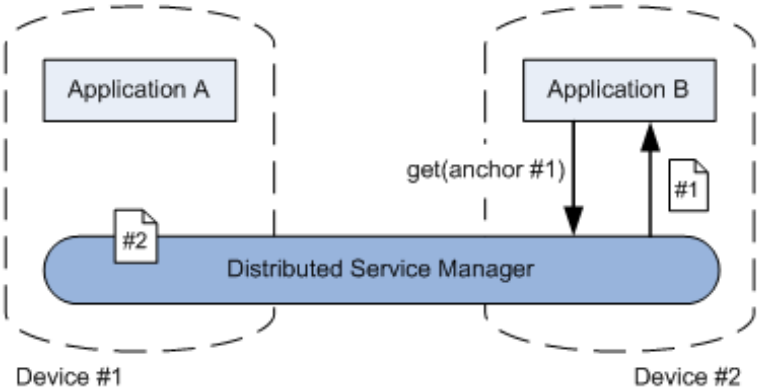
Cette gestion originale des ressources offre de nombreuses possibilités : adaptation fine du contexte à l'application, optimisation du transfert, transitivité des ressources permettant une adaptation positive, etc. Ainsi les pointeurs (*anchors*) permettent de masquer l'hétérogénéité des ressources, les flux de données par exemple étant des ressources non transférables seront simplement redirigées (*tunneling*) du dSM origine au dSM destination de manière transparente pour l'application, évitant ainsi toute coupure du flux.

### 7.2.4 Interface graphique

À de nombreuses reprises nous avons fait référence à l'interaction entre l'utilisateur et le dSM et plus particulièrement lors de la phase de déclenchement. Nous ne rentrerons pas dans les détails de la conception d'une telle interface qui sort du cadre de ces travaux, cependant de nombreuses problématiques émergent de cette question. En effet, nous nous sommes volontairement positionnés dans une approche de déclenchement manuel où l'utilisateur désigne explicitement les services et les terminaux concernés par un transfert. L'interface du dSM doit notifier l'utilisateur de l'existence des éléments distants tels que les terminaux et les services




(a) Fonction *pause*.




(b) Fonction *resume*.

FIGURE 7.2 – Principe de transfert du contexte.

tout en lui permettant d'agir directement sur son PSE en fonction de ses besoins de mobilité.

Une approche automatique qui limiterait l'importance de l'interface graphique est aussi réalisable, aucun déclenchement de ce type n'est intégré nativement au système, néanmoins le DSM prévoit un certain nombre d'interfaces (de programmation) permettant à un tiers d'implémenter un mécanisme de gestion automatique de la mobilité et d'agir sur le PSE.

## 7.3 Cas des services connectés

Certains services tels que ceux de télécommunications ou de streaming multimédia que nous avons étudiés au début de cette partie présentent une contrainte commune qui est une dépendance forte aux sessions de données. Si le flux correspondant est interrompu, le service n'est plus délivré. La nouvelle approche du concept de service et le modèle générique de mobilité qui en a découlé s'appliquent indifféremment à ces services connectés. Cependant, l'application des fonctions de continuité dans ce cas précis mérite quelques explications.

Dans le cas des services connectés, le contexte contient au moins une ressource particulière qui est la session, donc une donnée de taille illimitée, intransférable. Pour traiter ce type de ressource, deux possibilités de mobilité se présentent : soit le flux est substitué (cas des solutions étudiées jusqu'alors), soit le flux est redirigé. Nous avons déjà défini dans l'état de l'art ce qu'est une session (cf. 1.1.2), nous avions identifié qu'elle était composée de deux couches : contrôle et données. Les deux solutions de mobilité, substitution et redirection, correspondent respectivement aux approches de niveau contrôle et données.

Nous allons décrire les approches contrôle et données dans les sections suivantes. Il est important de noter qu'il n'est pas question ici du mécanisme de mobilité de service qui reste inchangé, nous ne faisons qu'expliciter la méthode de transfert des ressources de type « flux » qui est caractéristique des services connectés. Ces travaux ont été publiés dans [89].

### 7.3.1 Approche contrôle

L'approche « contrôle » est la plus classique, elle est d'ailleurs employée dans l'ensemble des mécanismes de mobilité de session, cf. 2.3.2. Dans le cadre du PSE, lors de la demande de la ressource par le terminal destination, les propriétés de la session sont envoyées par le DSM origine au terminal destination, ce dernier va alors mettre en œuvre des mécanismes spécifiques au protocole afin d'exécuter le transfert du flux de données ; on peut imaginer une application SIP qui utilise le mécanisme REFER tel que nous l'avons vu au chapitre 3. La Figure 7.3 illustre cette approche.

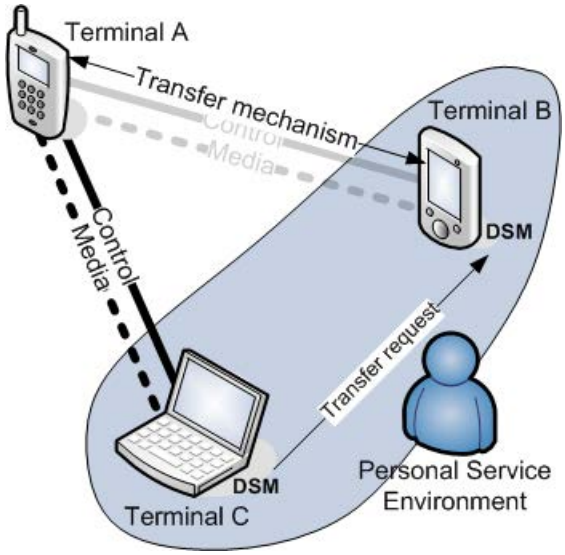



FIGURE 7.3 – Approche orientée « contrôle » de transfert des flux.

Cette solution est purement protocolaire et dépend directement de l'existence d'un mécanisme au niveau de la couche contrôle de la session. Le problème de la dépendance entre le mécanisme de mobilité et le service se pose sérieusement. En effet, le modèle générique que nous avons introduit pose comme principe la séparation des fonctions de mobilité de l'implémentation des services. Une telle solution n'est pas compatible avec l'hétérogénéité de l'environnement, son application sera restreinte à des implémentations, des protocoles spécifiques, limitant les transferts aux interfaces similaires.

Enfin, cette solution n'est applicable que si le terminal destination (A) naturellement connecté au terminal origine (B) peut également se connecter au terminal

correspondant (C), l'autre extrémité de la session. En effet, si les connectivités A-B et B-C sont évidentes, la connexion A-C n'est pas garantie or c'est là que la nouvelle session doit prendre place.

### 7.3.2 Approche données

L'approche « données » applique simplement le mécanisme de pointeurs présenté précédemment (cf. 7.2.3). Le flux de données n'est finalement qu'une ressource parmi les autres qui peut être désignée par un identifiant (*anchor*). Lorsque l'application destination souhaite obtenir cette ressource, elle fait une requête au dSM sous-jacent en indiquant ce pointeur. Le dSM origine redirige alors le flux reçu vers le dSM destination qui transmet les données directement et de manière transparente à l'application. La figure 7.4 illustre cette solution.

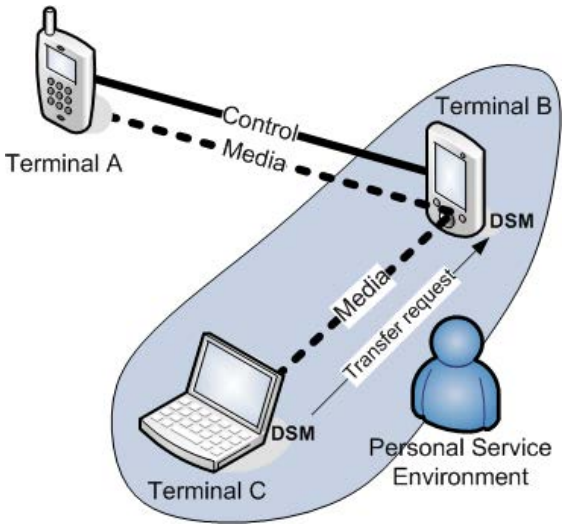



FIGURE 7.4 – Approche orientée « données » de transfert des flux.

Cette approche pose cependant un problème car la couche de contrôle qui est dépendante de l'implémentation est intrinsèquement perdue. Le flux pourra éventuellement persister car contrôlé par le terminal origine mais l'application destination ne pourra en aucun cas le manipuler. En conséquence, le flux ne pourra pas non plus être renégocié d'où l'incapacité d'adapter ses propriétés à la destination tel que l'encodage utilisé.

### 7.3.3 Approche hybride

Les deux approches contrôle et données n'étant pas satisfaisantes individuellement, elles se révèlent cependant complémentaires. À y regarder de plus près, l'ensemble des informations de session requises aux deux approches sont présentes dans le contexte du service : les propriétés de la session, son état au moment du *snapshot*, et le flux lui même. Les mécanismes de substitution de session sont eux implémentés par les applications.

En d'autres termes, le PSE est non seulement parfaitement compatible avec les deux approches mais elles se complètent : si l'application de destination supporte les ressources relatives à la couche contrôle, alors elle les utilisera pour réaliser le transfert directement, sinon le flux sera redirigé via la méthode générique classique.

## 7.4 Conclusion

Le *distributed Service Manager* est une tentative d'implémentation du modèle générique de mobilité défini dans le chapitre précédent. Comme prévu, cette entreprise a permis d'identifier de nombreuses problématiques qui ont suscité des mécanismes spécifiques. La généricité a été conservée, garante d'une continuité dans des environnements hétérogènes ; des fonctions telles que la gestion des ressources, les *snapshots* ou encore les *anchors* confèrent une grande souplesse à la solution proposée.

De nombreuses questions restent cependant en suspens : quel modèle de représentation des ressources, comment les organiser au sein du contexte (notion de groupe), faut-il normaliser les *snapshots* (représentation du contexte), comment optimiser le *tunneling* lors de la redirection d'un flux, comment réaliser la mobilité partielle des services (cf. 2.1.4) ?

Mais la question la plus immédiate est certainement celle de la faisabilité d'une telle solution. L'étape suivante dans ces travaux de recherche est le développement d'un prototype du DSM afin de prouver la possible réalisation d'un tel modèle et d'évaluer son impact sur l'utilisateur et son environnement.

# Chapitre 8

# Prototype et expérimentations

## 8.1 Motivations

Nos recherches sur la mobilité de service ont requis une étude approfondie de l'état de l'art afin d'analyser les approches existantes, et identifier les problématiques de continuité dans des environnements hétérogènes. Un long travail de redéfinition des nombreux concepts sous un angle nouveau, plus éloigné de l'implémentation des services, a rendu possible l'esquisse d'un modèle générique qui réponde à ces problématiques. Le *distributed Service Manager* est une implémentation de ce modèle ; avec le dSM nous apportons une solution complète, basée sur de nombreuses fonctions originales qui assurent une mobilité de bout en bout.

Nous arrivons ici à la dernière étape de cette démarche qui consiste à évaluer le modèle proposé, estimer si les contraintes établies ont été respectées, si les objectifs fixés ont été atteints et si la solution proposée est satisfaisante. Nous avons pour cela développé un prototype simple, qui assure l'essentiel des fonctions présentées dans les chapitres précédents. Cette preuve de concept permettra d'estimer l'impact de notre modèle sur son environnement d'un point de vue qualitatif et quantitatif.

## 8.2 Description du prototype

Le prototype du distributed Service Manager (dSM) a été développé pour prouver la faisabilité du modèle défini dans le chapitre 6, et réaliser des tests de per-

formance et d'utilisabilité. Le système central (dSM *core*), une interface utilisateur (UI), une interface de programmation (API) et enfin deux applications de test ont en tout été développés et publiés sous forme de projet *open source* LGPL sur le site Launchpad [90] (le projet s'appelle dSM, nom de code « *Chameleon* » rapport à sa propriété d'adaptation).

Le projet est entièrement écrit en Java et organisé en trois parties de la manière suivante.

- **Un système central**. Les principaux mécanismes du dSM font partie du système central (*core*) qui gère entre autres le PSE, le réseau *overlay* sous-jacent (découverte, communication et transferts de données entre terminaux via un protocole pair-à-pair implémenté par la bibliothèque Java JXTA [91][92]), les capacités du terminal, les services, les ressources ainsi que les messages entrant et sortant.
- **Une interface utilisateur**. Une interface graphique minimale (GUI) qui permet à l'utilisateur d'interagir avec le système et qui peut (et devrait) être remplacée par une interface tierce plus évoluée grâce aux évènements générés par le *core*. À noter qu'une console est également disponible.
- **Une interface de programmation**. Cette API est essentielle, intégrée aux applications elle leur permet d'interagir avec le dSM, facilitant considérablement le travail des développeurs de solutions compatibles.

En plus de ces trois grands axes de développement, deux applications tierces ont été modifiées afin de les rendre compatibles avec le dSM dans le cadre de nos tests. Pour cela nous avons intégré l'API du dSM à des éditeurs de texte open source Java relativement basiques. Ces applications sont également disponibles dans le cadre du projet open source, plus de détails seront fournis en section 8.3.2. La Figure 8.1 est une représentation fonctionnelle de l'architecture du distributed Service Manager qui positionne les différents blocs du prototype, l'annexe B propose également un diagramme de classes du système, enfin vous trouverez plus d'informations sur le site du projet [90]. Ces travaux ont été publiés dans [93].

Le système complet a été conçu afin d'être indépendant du SE sous-jacent. Le choix de Java comme langage de programmation a été fait grâce à ses qualités de portabilité mais également pour des questions de vitesse de développement. De plus, l'ensemble des communications entre le dSM et les applications se font via

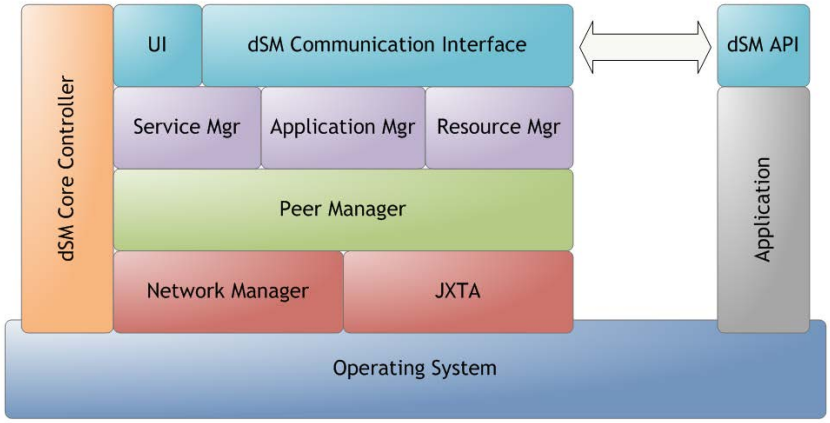



FIGURE 8.1 – Architecture fonctionnelle du prototype dSM.

des ports réseaux (le dSM utilise par exemple le port UDP 20100) pour les mêmes raisons d'indépendance avec l'environnement.

L'interface graphique n'est pas un axe actif du projet, en contrepartie un mécanisme d'évènements est fourni afin de permettre le développement d'interfaces plus évoluées sans avoir à modifier le système central. Une interface minimale de type « console » a également été développée pour les tests, permettant un accès rapide et exhaustif aux fonctions du dSM tout en limitant les mesures aux performances intrinsèques du modèle (dont le GUI ne fait pas partie).

Le fichier exécutable du distributed Service Manager (.jar) représente moins de 100ko (sans compter les bibliothèques externes et l'interface graphique). L'API, elle, fait moins de 40ko, offrant aux applications tous les mécanismes requis pour interagir avec le dSM. Il reste à l'application compatible d'initialiser l'API et d'assurer les fonctions *pause* et *resume* (cf. 7.2.2) qui ne représentent que quelques lignes de code par ressource à gérer.

Pour exemple, l'implémentation de ces fonctions dans les éditeurs de texte que nous avons modifiés a requis environ 50 lignes de code supplémentaires, soit moins de 2 à 5% de la taille totale de ces applications. Nous avons estimé que l'ajout d'une ressource supplémentaire ne coûterait qu'approximativement 5 lignes de code en plus.

## 8.3 Expérimentations

Évaluer la valeur ajoutée apportée par un modèle générique de mobilité de service n'est pas une simple tâche. En effet, nous ne proposons pas une solution qui accélère les applications ou augmente le débit d'un réseau, en réalité nous cherchons à permettre à l'utilisateur d'interagir de manière plus intuitive et plus libre avec ses terminaux, rendant la consommation des services en mobilité plus naturelle dans des environnements hétérogènes. Or il est difficile d'évaluer ces valeurs abstraites qui ont pourtant un impact bien concret au niveau de l'expérience utilisateur. Comment mesurer l'utilisabilité et la praticité ? D'un autre côté, le modèle que nous proposons se doit de respecter certaines contraintes pour prouver sa faisabilité, typiquement en terme de délai de transfert comparé aux approches existantes.

L'approche originale de mobilité de service que nous proposons doit convaincre sur deux plans afin de garantir une expérience satisfaisante : offrir à l'utilisateur une interaction intuitive avec ses services via des mécanismes de continuités temporelle et contextuelle performants. Nous avons ainsi réalisé deux expérimentations pour évaluer les deux aspects de notre modèle : utilisabilité et efficacité via une approche qualitative et quantitative de notre solution.

Dans la première expérimentation, nous avons étudié la relation des utilisateurs avec leurs services dans des situations de mobilité. Nous avons interrogé un échantillon d'utilisateurs afin de répertorier et analyser les méthodes de mobilité existantes. Dans la seconde expérience, nous avons mesuré la performance de notre prototype durant un transfert et analysé l'impact sur son environnement selon différentes métriques.

### 8.3.1 Évaluation qualitative

Dans cette première expérimentation nous avons voulu étudier le comportement des utilisateurs exposés à des contraintes de mobilité et plus particulièrement comment ils gèrent leurs services dans une telle situation. L'objectif est d'identifier les méthodes de transfert dans un cas de mobilité de service simple et réaliste. Cette première approche est une étape clé dans notre étude dans la mesure où l'expérience utilisateur est la métrique la plus importante dans l'évaluation de la mobilité de service. Ainsi, comprendre qu'est ce qui gêne ou empêche les utilisateurs de gérer

de manière simple, naturelle et intuitive leurs services parmi plusieurs terminaux nous aidera a identifier la réelle problématique.

**Méthode.**

Lors de cette expérience, nous avons demandé à des utilisateurs de répondre à des questions concernant une situation simple et leur expérience passée (questionnaire disponible en annexe C). Nous avons sélectionné un échantillon de personnes qui possèdent un ordinateur et l'utilisent au moins une fois par jour. Ainsi nous avons supposé que ces personnes avaient, de manière consciente ou non été confrontées au moins une fois à une situation de mobilité de service. Les participants étaient 33 personnes de différents âges (de 12 à 55 ans) et possédant des connaissances informatiques variées. Nous les avons classés en deux catégories selon leur niveau dans le domaine des Technologies de l'Information et de la Communication : les *spécialistes* (ingénieurs, chercheurs, étudiants) représentant 80% de l'échantillon et les *non-spécialistes* représentant les 20% restants. L'hétérogénéité de l'échantillon était recherché et nécessaire pour garantir des résultats de qualité et une vision consistante de la méthodologie utilisateur adoptée.

Nous avons présenté à chaque participant une situation simple d'utilisation d'un service informatique, puis nous avons introduit une contrainte de mobilité et nous leur avons demandé de décrire leur solution selon leur expérience. La situation proposée était basique : un utilisateur qui édite un document sur son ordinateur doit subitement interrompre son travail et le poursuivre sur un autre terminal. Les participants étaient alors invités à décrire en détails comment ils ont (ou auraient) réalisé un tel transfert.

Nous avions sciemment fourni peu de détails dans la description de la situation afin de faciliter l'adoption du cas par chaque participant. Des questions additionnelles étaient ensuite posées afin d'obtenir des informations plus précises en rapport avec leur expérience de la mobilité de service : besoins, praticité, simplicité, sentiment d'interruption, etc. Les données recueillies ont ensuite été analysées et rapprochées afin d'extraire des informations et des tendances sur les utilisateurs et leur perception des contraintes de mobilité. Les résultats sont présentés dans la section suivante.

**Résultats.**

L'objectif premier de cette expérience était d'analyser la méthodologie de mobilité de service des utilisateurs. Les participants ont donc été invités à exprimer toutes les approches qu'ils ont expérimentées ou auraient pu utiliser. Les résultats collectés sont présentés dans la Figure 8.2 qui résume les différentes méthodes de transfert par catégorie (spécialistes et non-spécialistes).

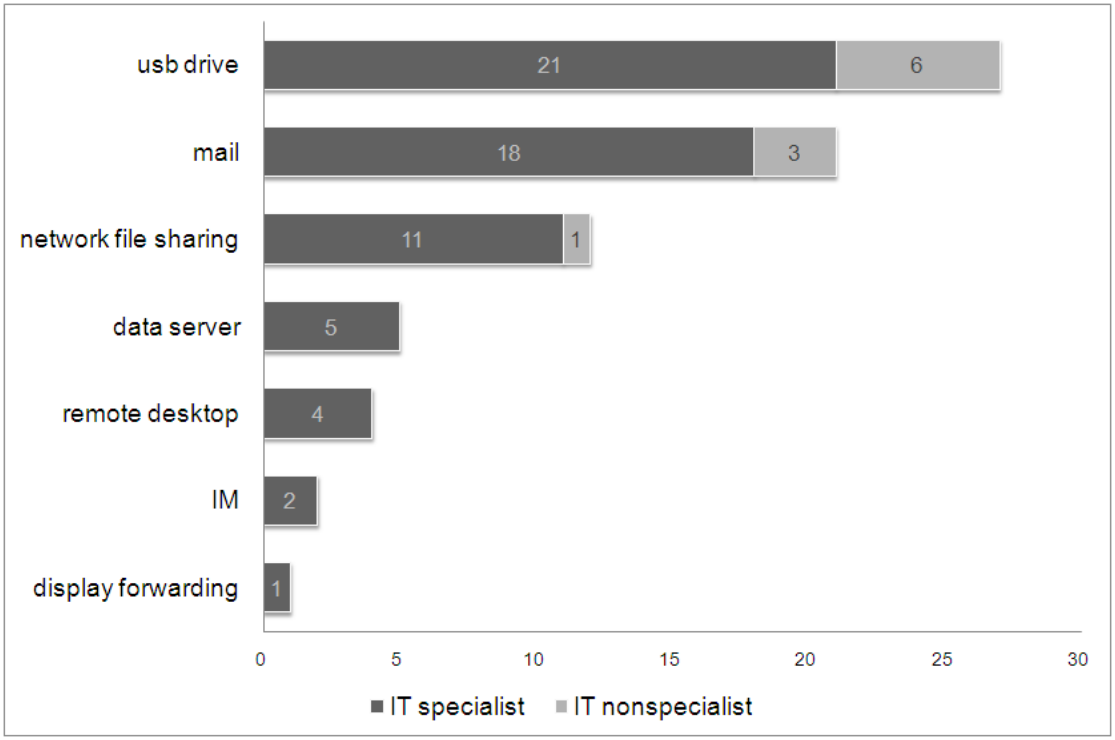



FIGURE 8.2 – Méthodes de mobilité de service expérimentées.

Le premier constat est le nombre de solutions différentes pour répondre à une même problème et ce uniquement sur un échantillon de 33 personnes (7 méthodes distinctes pour les spécialistes, 3 pour les non-spécialistes). Il n'y a pas d'approche commune, un modèle générique pour gérer une contrainte de mobilité dans un cas relativement basique. De plus, les trois approches les plus utilisées : stockage USB, courrier électronique et lecteur réseau sont partagées par les deux catégories de l'échantillon, ce qui indique que c'est bien une problématique globale et qu'il n'existe pas de solution miracle réservée aux initiés.

On peut également remarquer que ces solutions sont simples : USB et *email* sont manipulés presque quotidiennement et le partage réseau est intégré nativement aux systèmes d'exploitations, souvent prêt à l'emploi dans le milieu de travail des utilisateurs. La popularité de ces solutions familières est liée à leur simplicité d'utilisation qui en font un choix naturel bien que non prévues a priori à la mobilité de service. Cependant, simplicité n'est pas synonyme de praticité.

Nous avons également demandé aux participants d'évaluer la simplicité et la praticité de leur propres solutions, nous avons ensuite traduit leurs réponses en valeurs numériques (1, 2 et 3) selon le barème suivant :

1. **non** simple/pratique,
2. simple/pratique,
3. **très** simple/pratique.

Finalement non avons calculé la moyenne pour chacune des propriétés et de manière finalement peu surprenante, leurs méthodes sont évaluées comme assez simples (2.58) mais tout juste pratiques (1.91). En fait, par manque de solution dédiée, les approches proposées sont des alternatives « bricolées » basées sur des applications connues, maîtrisées par l'utilisateur et donc simples mais qui ne peuvent en aucun cas offrir des mécanismes efficaces et pratiques de continuité.

Pour illustrer ce manque d'efficacité, 66% des participants prétendent se sentir interrompus dans leur tâche en employant leur solution de mobilité. De plus, ils ont décrit les problèmes et les contraintes liées à leur solution, celles-ci sont résumées, par approche, ci-dessous.

- Stockage USB. Contraintes matérielles et logicielles (interface et lecteur USB requis, compatibilité du système d'exploitation, système de fichier, etc).
- Courrier électronique. accès à Internet nécessaire, limitations sur le taille et le type des données transférées (problèmes de sécurité). Délai de transfert important.
- Partage réseau. Problèmes de compatibilité avec le système d'exploitation, le protocole utilisé et le système de fichier. Solution moins triviale généralement pour les spécialistes.

Néanmoins, les utilisateurs sondés ont recours en moyenne à un transfert de service 2.73 fois par semaine, spécialistes et non-spécialistes confondus, ce qui

est particulièrement important considérant le manque de praticité des méthodes utilisées et l'absence d'un mécanisme dédié à la continuité de service. Cela prouve qu'un réel besoin existe et pas seulement pour les « geeks ». La table 8.1 résume les résultats obtenus dans le cadre de cette expérimentation.

TABLE 8.1 – Résultats de l'évaluation qualitative.

| **Metric** | **Result** | |
|---|---|---|
| **Sample** | 33 people | |
| | 80% IT spec. | 20% nonspec. |
| **Service Mobility Frequency** | 2.73 per week | |
| **Ease of Use** | 2.58 | |
| **Convenience** | 1.91 | |
| **Interruption Feeling** | 66% yes | |

Enfin, nous avons proposé aux participants une autre situation, plus complexe cette fois-ci. Nous avons supposé qu'ils regardaient une vidéo sur Internet (service multimédia tel que proposé par *Youtube*) et qu'ils devaient cette fois encore changer de terminal. Nous leur avons demandé si leur approche de mobilité initiale était applicable à cette nouvelle situation de mobilité, et si non d'en proposer une. Bien entendu, aucune de leurs approches n'a pu être appliquée à ce service différent excepté l'approche d'*affichage déporté* proposée par un spécialiste et qui est en fait une solution relativement performante de continuité de service que nous avions identifiée dans l'état de l'art (cf. 2.3.1).

76% des participants ont proposé une méthode *manuelle* de mobilité qui consiste à transférer le service multimédia en plusieurs étapes (l'URL, le nom de la vidéo, la position de lecture, etc) via des mécanismes de communication classiques reliant les terminaux origine et destination (tels que ceux listé en Figure 8.2) ou simplement en les mémorisant mentalement (le cerveau reliant effectivement les deux interfaces). Il est intéressant de constater qu'inconsciemment les participants ont identifié les éléments importants du service et transmis ces ressources manuellement afin de rétablir le contexte sur le nouveau terminal ; exactement le principe du mécanisme de transfert du distributed Service Manager. De plus, le processus

d'adaptation a également été pensé, seules les ressources compatibles aux capacités de l'application destination ont été transférées. Uniquement le contexte a été considéré, le service a été déplacé et non son instance. Il est à noter que 12% de l'échantillon n'a pu trouver de solution pour cette nouvelle situation.

**Discussion.**

Avec de plus en plus de terminaux autour de l'utilisateur, celui-ci tend à être plus mobile. Cette première expérience démontre que le besoin de mobilité est bien réel et que pourtant il n'existe pas de modèle de continuité générique, ni d'ailleurs de solution standard pour les situations simples de la vie de tous les jours. Lorsque confrontés à des contraintes de mobilité, les utilisateurs bricolent un semblant de continuité à partir d'applications simples mais prévues à un tout autre dessein. Les approches utilisateurs imposent des contraintes de compatibilité logicielles et matérielles fortes, sans mécanisme de sécurité, ni d'adaptation, offrant une expérience pauvre (sentiment d'interruption, perte de contexte, etc) et dédiées à des scénarios spécifiques. Quant aux scénarios plus complexes tels que le streaming vidéo, ils ne sont simplement pas supportés.

Ces solutions alternatives « fait-maison » induisent intrinsèquement de nouvelles problématiques, de plus elles ne peuvent assurer le processus mobilité de bout en bout comme défini dans le modèle générique : « *découverte, capture, déclenchement, transfert et reprise* » (cf. 6). En effet, au mieux, seule l'étape de transfert est réalisée, les autres phases sont généralement assurées par l'utilisateur, dans la mesure du possible.

Il n'y a pas d'identification des services et des terminaux, en conséquence ils ne peuvent être transférés directement d'une interface à une autre, l'utilisateur doit alors passer par une étape intermédiaire : lecteur USB, boîte de courrier électronique, disque partagé, etc.

Les services « simples » tel que l'édition de texte proposé dans ce scénario sont gérés au niveau application via une fichier complet (généralement propriétaire) qui laisse peu de manœuvre à l'adaptation, totalement incompatible avec un environnement hétérogène. Lorsque l'on considère des services plus complexes (en termes de mobilité) tel que les services multimédias, les phases de *capture* et de *reprise* sont effectuées manuellement (et inconsciemment) par l'utilisateur.

Les solutions alternatives proposées par les participants et listées en Figure 8.2 permettent de transférer des données. Si le service est facilement identifiable en termes de ressources et permet à l'utilisateur de capturer une partie de son contexte (via un fichier par exemple), alors il peut essayer de bricoler une pseudo-mobilité de l'application. Mais un modèle de mobilité de service et plus que le transfert de données, il doit considérer l'environnement dans son ensemble, avec son hétérogénéité de terminaux, d'applications et de services. De plus, il doit fournir une solution unique, indépendante du type de service pour une simplicité et un praticité totale.

### 8.3.2 Évaluation quantitative

Les résultats de l'expérience précédente démontrent qu'une meilleure solution aurait certainement sa place pour offrir aux utilisateurs un moyen pratique et efficace de gérer leurs services dans un environnement hétérogène. Cependant, si notre système offre déjà de nouveaux usages, nous devons garantir qu'il est techniquement réalisable et qu'il répond de manière satisfaisante aux contraintes de continuité, notamment dans la perception du transfert.

L'objectif de cette expérimentation est d'analyser l'impact de notre modèle sur son environnement : terminal hôte, réseau, mais aussi application du point de vue de la continuité temporelle du transfert. Les mesures ont été effectuées sur le dSM, un prototype expérimental relativement instable et non optimisé. De plus nous avons dû modifier des applications pour les rendre compatibles, les tests seront donc uniquement basés sur ces services d'édition de texte, la performance d'autres types ne pouvant qu'être estimée théoriquement. En conséquence, les résultats obtenus n'offriront qu'une approximation mais qui apporteront déjà un aperçu des performances de notre modèle.

#### Méthode.

Pendant ces tests, nous avons réalisé un transfert entre deux terminaux. Nous avons choisi un service de type édition de texte qui est commun, simple, et dont l'implémentation n'est pas basée sur des sessions. Choisir un service « simple » était important car nous avons dû intégrer l'API dSM à deux applications de ce type en modifiant leur code source. De plus, un service local (sans session) était

préférable afin d'étudier un cas qui s'écarte des nombreuses solutions existantes basées sur des mécanismes de continuité de session (cf. 2.3.2).

Le transfert a été réalisé dans un environnement hétérogène, entre un ordinateur de bureau (PC) et un ordinateur portable (*Laptop*), chacun possédant des caractéristiques logicielles et matérielles différentes : CPU, mémoire, interface réseau, système d'exploitation (MS Windows et Linux) et application d'édition de texte (Java Notepad et TerpWord). Les applications origine et destination sont donc différentes : d'un côté (Java) Notepad, un éditeur basique (de type bloc notes) distribué comme exemple avec l'environnement de développement Java (*Java Development Kit*, JDK [94]) ; de l'autre TerWord, un éditeur plus riche, également open source et écrit en Java, qui fait partie de la suite bureautique TerpOffice développée par l'Université du Maryland [95]. Les deux applications ont été choisies pour leur simplicité et l'ouverture de leur code Java permettant une intégration aisée de notre API. Les caractéristiques des terminaux de test sont résumées dans la Table 8.2.

TABLE 8.2 – Environnement de test de l'évaluation quantitative.

| | **Laptop** | **Personal Computer** |
|---|---|---|
| **Operating System** | MS Windows | Linux (kernel 2.6) |
| **Text application** | TerpWord | (Java) Notepad |
| **Processor** | Dual Core 1.2GHz | Single Core 2.5Ghz |
| **Memory** | 1.5GB | 2.5GB |
| **Network** | WiFi 11Mb/s | Ethernet 100Mb/s |

Le scénario de l'expérience est le suivant (cf. Figures 8.3 et 8.4). Initialement le PC et le *Laptop* sont connectés via un réseau local par leurs interfaces respectives. Il est à noter que le réseau est capable de gérer des paquets en IP Multicast ([96]). Une instance du prototype, le DSM est lancée sur chaque terminal. Une fois que les deux terminaux se sont mutuellement découverts via le DSM (1), l'application TerpWord est démarrée sur le *Laptop* (2), résultant d'une notification immédiate de « nouveau service » sur le PC et affichée sur l'interface DSM de ce terminal.

Un texte quelconque est alors entré dans l'éditeur et le curseur est positionné à un endroit précis afin de vérifier la continuité après le transfert (a). Alors, depuis l'interface dSM du PC, nous sélectionnons l'unique service affiché, c'est à dire celui fourni par l'application TerpWord, et nous déclenchons le transfert vers le terminal courant, le PC (3). TerpWord entre alors en mode *pause*.

L'application locale du PC associée au service d'édition de texte est alors automatiquement exécutée en mode *resume* par le dSM (4). Le (Java) Notepad est alors lancé avec comme paramètre le contexte du service distant (un *snapshot* plus exactement, cf. 7.2.3) et il demande immédiatement au dSM les ressources qu'il désire (5). Ces ressources sont transférées par le distributed Service Manager du terminal origine jusqu'au Notepad qui restitue le service d'édition de texte de l'utilisateur (6). Finalement, l'application TerpWord est arrêtée automatiquement, le service étant maintenant disponible sur le PC, via l'application Notepad (b).

Durant ce scénario, nous avons relevé diverses informations. Pour estimer l'impact du dSM sur son hôte, nous avons mesuré les consommations CPU et mémoire grâce à un outil système, le *Reliability and Performance Monitor* [97], une application Microsoft qui fournit des informations précises sur les processus (ici le dSM). Nous nous sommes également intéressés aux performances du transfert, nous avons donc mesuré l'impact de notre solution sur le réseau en termes de délai et d'*overhead* engendrés par les mécanismes de mobilité. Ces mesures ont été réalisées à partir de l'analyse des paquets de données (grâce à l'outil *Wireshark* [98]) et les logs générés par les dSM des deux terminaux (les horloges internes ayant été synchronisées préalablement).

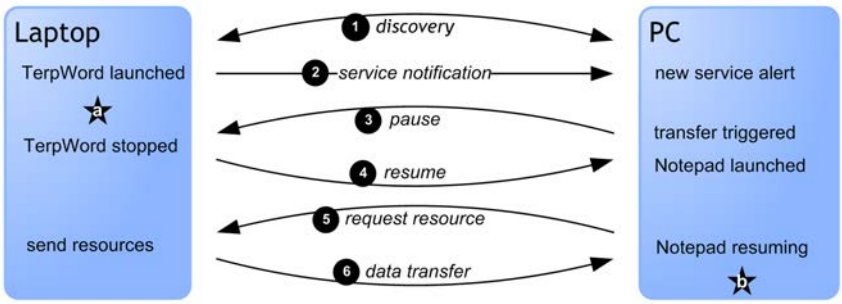



FIGURE 8.3 – Le scénario expérimental.

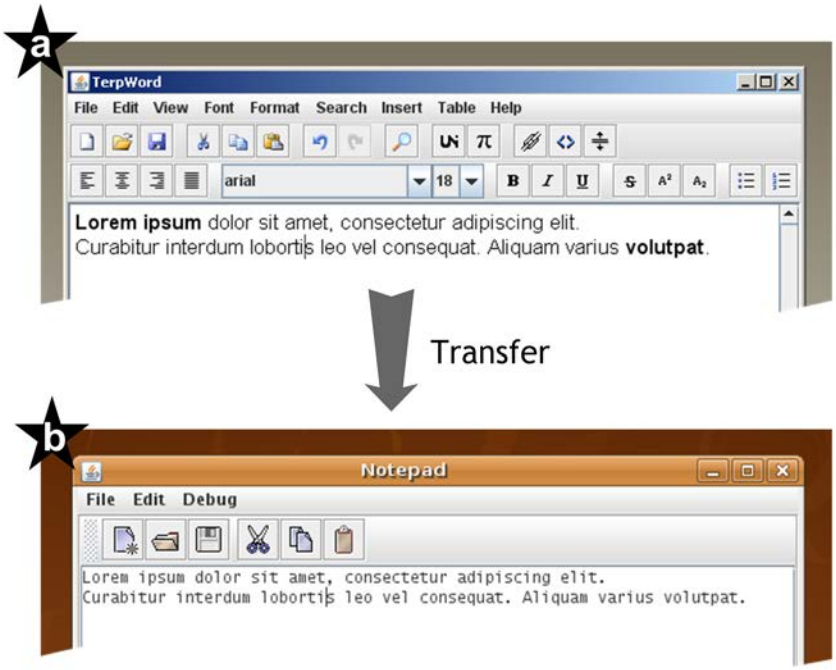



FIGURE 8.4 – Le service avant et après le transfert.

**Résultats.**

Nous avons mesuré l'impact du DSM sur son hôte pendant toute la durée du scénario décrit ci-dessus et selon trois métriques : processeur, mémoire et réseau. Les données acquises sont reportées dans la Figure 8.5 qui montre l'utilisation du CPU (pourcentage de la capacité totale du terminal), la quantité de mémoire allouée (en Mégaoctets) et l'activité réseau (en octets) pendant environ 50 secondes. Essayons d'analyser ces résultats.

D'après le graphique présenté en Figure 8.5, les consommations CPU et mémoire augmentent très fortement dans les premières secondes (33% de CPU et 26Mo de mémoire), correspondant logiquement au démarrage et à l'initialisation du DSM. On notera par la même occasion que le prototype est instantannément opérationnel, la charge CPU chutant très rapidement.

Après 12 secondes (noté T12), un autre pic d'activité CPU (15%) et réseau (700o) est observable, accompagné d'une légère augmentation de la consommation

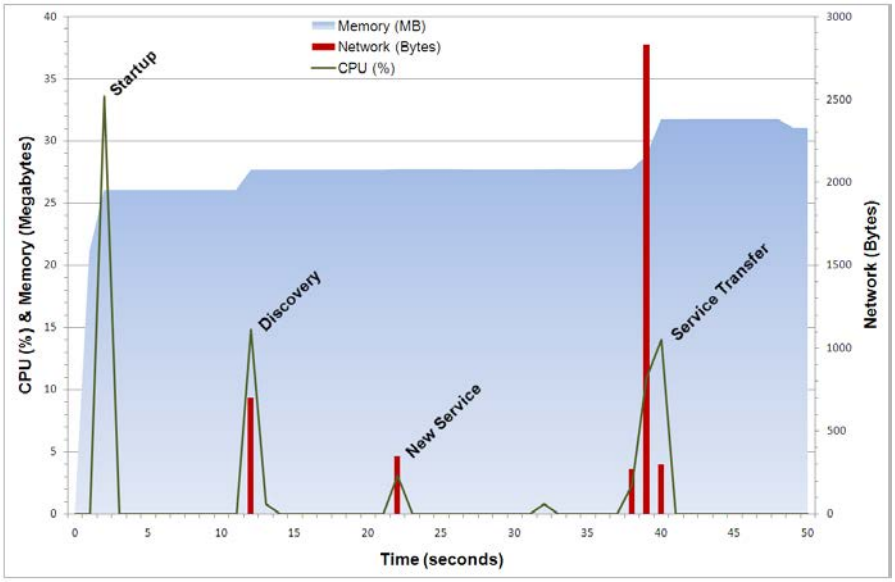



FIGURE 8.5 – Performance du DSM pendant un transfert.

mémoire (+2Mo). Cela correspond à la découverte d'un nouveau terminal, le DSM du PC distant ayant été démarré à T11, un échange de messages « hello » permettant l'identification mutuelle des terminaux a créé cette activité (cf. étape (1) dans la Figure 8.3).

L'application TerpWord est démarrée sur le *Laptop* à T21, en conséquence de nouveaux messages de signalisation sont échangés entre le DSM et l'application (processus d'enregistrement) et entre les DSM des deux terminaux (processus de notification). On observe un léger pic d'activité CPU (3%) et réseau (348o) à cette occasion à T22 (cf. étape (2) dans la Figure 8.3).

Enfin, l'activité principale observée de T37 à T41 correspond au mécanisme de transfert (cf. étapes (3-6) dans la Figure 8.3) déclenché depuis le PC à T36. Plusieurs actions sont réalisées ici : l'application est mise en *pause* et quelques messages (*snapshot*, notifications, etc) sont envoyés pour initier la reprise du service sur le terminal distant. Ensuite, chaque ressource demandée est transférée par le DSM local. Pendant le transfert, l'activité CPU atteint brièvement 15%, l'allocation mémoire est augmentée de 4Mo (pour une total de 32Mo) dû à la gestion des ressources par le DSM et l'activité globale de l'interface réseau (trafic entrant

et sortant) est d'approximativement 3,4ko (à noter que le trafic interprocessus dSM⇔Terpword est inclus dans cette mesure, cf. 8.3.2).

Le service est effectivement transféré et repris sur le PC à T41, pour un délai total de *terminal handover* égal à 5 secondes. Un résultat tout à fait satisfaisant pour ce type de service. De plus, il faut prendre en considération que l'utilisateur ne percevra pas un délai de 5 secondes, l'application destination étant lancée instantanément lors du déclenchement, avant même que le contexte ne soit effectivement transféré et traité par la fonction *resume*. La Figure 8.4 présente deux captures d'écran du même service avant et après le transfert : d'abord instancié par l'application TerpWord (a), puis par l'application Java Notepad (b). Vous noterez que le service a été adapté à l'application destination, le curseur et le texte ont été correctement restitués, par contre le formatage de certains caractères (en gras) a été ignoré car non supporté par le Java Notepad. Si un transfert supplémentaire est réalisé depuis cette instance du service et que l'application destination supporte cette ressource (formatage), alors les caractères en gras seront de nouveau restitués (principe de transitivité, cf. 7.2.3).

**Discussion.**

Cette deuxième expérimentation nous a permis de quantifier l'impact du dSM sur son environnement et d'évaluer la performance du transfert. Or ces valeurs sont approximatives car basées sur un prototype, des précisions complémentaires sont nécessaires pour une compréhension correcte des résultats.

Nous pensons que les métriques les plus importantes dans l'évaluation quantitative de la mobilité sont l'activité réseau et le délai de transfert car elles ont un impact direct sur l'environnement et la perception de continuité de l'utilisateur. Les mesures que nous avons réalisées pendant le transfert (de T37 à T41 cf. 8.5) ont donné des valeurs très satisfaisantes, cependant elles pourraient être considérablement optimisées. En effet, durant cette période, l'activité réseau totale est estimée à 3.4ko, or la bande passante mesurée ici inclut deux interfaces, dont une qui n'a pas d'impact sur l'environnement car seulement locale : dSM⇔dSM (distant) et dSM⇔application (locale au terminal). De plus, notre prototype étant développé en Java pour des raisons de rapidité d'implémentation et de déploiement, l'ensemble des mesures effectuées incluent la Machine Virtuelle Java (JVM

[99]) qui exécute l'application dSM. Cette JVM permet une portabilité de notre système mais elle le pénalise largement en termes de performances d'exécution (CPU et mémoire notamment).

La JVM n'est pas le seul facteur limitant les performances de notre système, la gestion du réseau P2P sous-jacent est assurée par la bibliothèque Java JXTA (actuellement la seule API ouverte de ce type) qui induit également une charge supplémentaire au niveau CPU et réseau (cf. initialisation à T12 dans la Figure 8.5). De plus, JXTA implémente de nombreux mécanismes de contrôle basé sur TCP et inutiles dans le cadre de notre système (UDP aurait été préféré), un *overhead* excessif est produit par ces fonctions. Dans la mesure du possible, nous avons essayé de ne pas prendre compte cet *overhead* dans nos mesures car il fausse l'évaluation des mécanismes de mobilité. Pour information, l'activité réseau incluant cet *overhead* durant la phase de transfert s'élève à 180ko... pour seulement 3,4ko de données utiles.

Nous avons indiqué au début de cette section que l'activité réseau mesurée correspond à la quantité de données échangées aux interfaces du dSM. Or le dSM possède des interfaces ouvertes vers l'extérieur qui ont un impact direct sur l'environnement mais également des interfaces internes permettant de communiquer avec les applications locales. Divers messages tels que des notifications pour l'interface utilisateur, qui n'interviennent pas dans la réalisation du transfert ont également été mesurés et surévaluent l'activité réseau inhérente au modèle. Pour être précis, seuls quatre types de messages sont échangés durant cette période : la requête PAUSE (déclenchement du transfert), la requête *Service Snapshot* (le contexte), les *Ressource Request* (demandes de ressource) et finalement les données (ressources). Ainsi, l'activité réseau engendrée par notre système pour la réalisation d'un transfert de service peut être exprimée par la formule :

$$\sum_{i=1}^{n} (size_i + \sigma_{res}) + \sigma_{tot} \tag{8.1}$$

Où $n$ est le nombre de ressources effectivement *transférées* (uniquement celles supportées par l'application destination), $size_i$ la taille de la ressource $i$, $\sigma_{res}$ l'*overhead* par ressource et $\sigma_{tot}$ l'*overhead* global généré par l'ensemble des messages de signalisation. D'après notre scénario et les mesures effectuées, $n = 2$ : le

corps du texte ($size_1 = 128$ octets) et la position du curseur ($size_2 = 2$ octets), l'*overhead* par ressource $\sigma_{res} = 90$ octets et l'*overhead* global $\sigma_{tot} = 990$ octets. Ainsi, appliqué à notre scénario de mobilité d'un service d'édition de texte, l'activité réseau exacte liée au transfert est de 1,3ko seulement, comparé aux 3,4ko donnés par une mesure globale.

Notons que cette valeur pourrait encore être optimisée, en effet les messages de signalisation du dSM sont en XML, un format de présentation de données particulièrement verbeux [100]. Or un algorithme de compression pourrait être appliqué pour réduire la quantité de données transmises en contrepartie d'une activité CPU revue à la hausse. Pour exemple, un ratio de compression de 1,4 peut être facilement atteint sur ce type de données grâce à l'outil *gzip* [101].

D'après la formule précédente (cf. Eq. (8.1)), nous pouvons à présent estimer l'impact réseau de notre système appliqué à tout type de service. Cependant, nous ne pouvons estimer la performance de transfert a priori sans connaître les détails d'implémentation du service pour une application donnée ; l'efficacité des mécanismes de continuité dépendant directement de cette implémentation : nombre, type et taille des ressources, complexité et durée des processus internes, volatilité du contexte, etc. Néanmoins, notre modèle assure la mobilité de tout service transférable grâce à un mécanisme original de gestion du contexte (*snapshots*, *anchors*) qui optimise le processus de continuité :

- priorités de transfert des ressources pour une adaptation et une reprise efficace du service,
- gestion de toutes les ressources, même les intransférables par redirection de flux,
- faible impact sur l'environnement, moins de 0.01% d'*overhead* par exemple pour un service constitué de trois ressources de 5Mo chacune.

Si la transférabilité de *tous* les services est fortement améliorée grâce au distributed Service Manager considérant l'expérience utilisateur ou d'un point de vue purement technique, la mobilité reste (et restera) dépendante des capacités du réseau sous-jacent. En d'autres termes, transférer un service qui manipule un contexte de 200Mo via un accès 3G (limité à 2Mb/s) sera réalisable si et seulement si il est possible d'extraire une quantité de données compatible aux capacités de

l'environnement (notamment les propriétés du réseau) et suffisante à l'application destination pour assurer un service minimal.

## 8.4 Conclusion

Dans ce chapitre, nous avons prouvé la faisabilité de notre modèle générique grâce à l'implémentation d'un prototype du distributed Service Manager. Nous avons essayé d'évaluer ce système dans un environnement hétérogène en s'intéressant aux contraintes de continuité contextuelle et temporelle.

Ainsi nous l'avons évalué d'un point de vue qualitatif afin de positionner notre solution par rapport aux besoins utilisateurs et à leur expérience de la mobilité. Nous avons identifié un réel besoin, des solutions inexistantes et une approche intuitive de mobilité qui se calque sur les mécanismes que nous avions conçus.

Ensuite nous avons évalué notre prototype d'un point de vue performance dans un scénario de mobilité classique. Nous avons mesuré l'impact de notre solution sur son environnement : l'application, le terminal hôte, le réseau. Nous avons estimé la performance des mécanismes de transfert, les délais et l'*overhead* induits. Enfin nous avons formulé de manière précise l'impact réel sur le réseau de notre système en fonction des caractéristiques d'un service à transférer ; cette formule permet de calculer la quantité de données effectivement transférées à partir d'un contexte donné. Elle pourra être appliquée à d'autre types de services que nous n'avons pas eu le temps de traiter ici.

Le bilan des tests réalisés sur notre prototype est satisfaisant. Tout d'abord, nous avons pu démontrer la faisabilité de notre modèle dans un environnement hétérogène. Le système a répondu avec succès aux contraintes quantitatives ; les transferts sont rapides, optimisés et l'impact du DSM sur son environnement est faible : *overhead* minimal, terminal peu affecté, implémentation triviale sur les applications compatibles... Les contraintes qualitatives sont également considérées ; la gestion de bout en bout de la mobilité de manière générique et transparente apporte la solution qui manque aujourd'hui à l'utilisateur afin qu'il gère de manière intuitive, simple et pratique ses services dans son environnement hétérogène.

# Conclusions

# 1 Bilan

Le bilan de ces travaux de recherche est particulièrement satisfaisant. L'objectif ambitieux fixé initialement a été atteint dans un domaine, « la continuité de service », peu appréhendé d'un point de vue applicatif. Nous avons donc emprunté des sentiers peu fréquentés, adopté des approches inédites, nécessitant la définition de nouveaux concepts.

L'étude de cas concrets de mobilité de service qui a abouti à l'implémentation de solutions spécifiques tel que dans l'infrastructure IMS s'est révélée riche d'enseignements, dévoilant les points forts et les lacunes des systèmes existants. Ces travaux de recherche applicative ont constitué le point de départ indispensable pour la définition d'une solution globale. La deuxième phase plus théorique a pu ainsi se reposer sur une expérience concrète de la problématique de mobilité, gage de son succès.

Le modèle générique de mobilité proposé, basé sur de nombreux mécanismes originaux, permet de s'abstraire de l'hétérogénéité de l'environnement tout en adaptant les services de l'utilisateur à ses capacités. Le travail de développement considérable d'un prototype a été récompensé par des résultats concluants : un système qui réalise les fonctions définies théoriquement par notre modèle et qui assure une expérience utilisateur optimale grâce à une continuité contextuelle et temporelle.

# 2 Travaux en cours

Les efforts de prototypage réalisés au cours de cette étude, continuent dans le cadre d'un projet *open source* [90]. Le *distributed Service Manager*, encore en phase expérimentale, peut ainsi bénéficier d'améliorations constantes et surtout d'une capacité de développement et d'évolution considérablement supérieure. De nouvelles fonctions sont ainsi imaginées, toujours dans une optique d'amélioration de l'expérience utilisateur, le détail de ces nouveaux mécanismes est décrit dans la section suivante.

# 3 Perspectives

Les travaux de cette thèse ont permis d'établir un modèle innovant, adressant d'un point de vue applicatif les problématiques de mobilité de service. Or, l'étendue du domaine est telle que les possibilités autour du *Service* sont infinies, posant toujours de nouvelles contraintes et problématiques liées à l'expérience utilisateur. Notre modèle pose cependant les bases d'une approche applicative à fort potentiel et de nombreuses nouvelles fonctionnalités sont désormais réalisables à partir des mécanismes et des concepts introduits et présentés dans ce mémoire.

On peut mentionner ici quelques nouveaux axes d'étude qui se sont imposés d'eux-mêmes pendant nos recherches mais que nous n'avons pas eu le temps d'étudier.

La gestion des applications par exemple, non d'un point vue instance ou protocolaire mais fonctionnel, permettant à des terminaux d'échanger leurs capacités logicielles, et ainsi d'augmenter l'adaptabilité des services entre interfaces hétérogènes. Cet axe de recherche nécessite de prolonger la réflexion sur le cloisonnement des marchés par les constructeurs mentionné en introduction. Un modèle économique innovant est nécessaire afin de faciliter les échanges entre les interfaces de l'utilisateur et la mise en œuvre de mécanismes de mobilité tels ceux présentés dans ce mémoire sur un large éventail de services. Cette ouverture serait bénéfique pour les différents acteurs des Technologies de l'Information et de la Communication (constructeurs, opérateurs, fournisseurs de contenus), et ce pour un coût minimal, le système ayant un impact négligeable sur son environnement tout en enrichissant l'offre et l'expérience utilisateur.

L'optimisation des mécanismes de transfert des services connectés afin de conserver la couche contrôle en toutes circonstances. Cette problématique a été explicitement identifiée en section 7.3 et la question reste ouverte, n'ayant pu y répondre de manière définitive dans le cadre des travaux présentés dans ce mémoire.

La mobilité partielle des services qui se traduit par une composition ou une dissociation de fonctions 2.1.4. Les mécanismes de gestion de ressources définis dans notre modèle semblent particulièrement adaptés à ce type de mobilité. Il faut cependant implémenter et intégrer les fonctions correspondantes au système.

La normalisation du contexte et plus particulièrement des *snapshots* (cf. 6.2.2) est également un axe d'étude intéressant, nous n'avons pas détaillé la manière d'organiser, de décrire et de présenter de manière standard les ressources. Cette problématique est particulièrement importante car elle assure interopérabilité des interfaces.

Nous avons aussi volontairement ignoré les problématiques de représentation graphique du transfert et plus précisément de déclenchement. La nouvelle approche de mobilité introduite par notre modèle requiert une étude spécifique afin que le contrôle conféré à l'utilisateur soit également perçu de manière naturelle, simple et intuitive.

Enfin, d'autres modes de transfert de services pourraient être étudiés tel que la mobilité au sein d'une même interface (changement d'instance) ou encore la notion de *bookmark* permettant aux utilisateurs de stocker et partager leurs services. Ces fonctions étant supportées par la gestion de ressources de notre modèle, permettant d'ouvrir la sphère électronique de l'utilisateur au Web 2.0, en tant que nouvelle interface ?

# Références


[1] Université de Sherbrooke. Perspective Monde : Emploi dans le secteur des services en France. http://perspective.usherbrooke.ca/bilan/tend/FRA/fr/SL.SRV.EMPL.ZS.html, 2005.

[2] Apple. iphone. http://www.apple.com/iphone/, August 2009.

[3] Institut national de la statistique et des études économiques. Définitions et méthodes : Services. http://www.insee.fr/fr/methodes/default.asp?page=definitions/services.htm, 2008.

[4] Microsoft Developer Network. Introduction to windows service applications. http://msdn.microsoft.com/en-us/library/d56de412(VS.80).aspx, Juin 2008.

[5] Wikipedia. Windows service. http://en.wikipedia.org/wiki/Windows_service, Décembre 2008.

[6] Wikipedia. Daemon (computer software). http://en.wikipedia.org/wiki/Daemon_(computer_software), Décembre 2008.

[7] Wikipedia. List of microsoft windows components : Services. http://en.wikipedia.org/wiki/List_of_Microsoft_Windows_components, Décembre 2008.

[8] Wikipedia. Session (computer science). http://en.wikipedia.org/wiki/Session_(computer_science), Décembre 2008.

[9] Wikipedia. Hypertext transfer protocol. http://en.wikipedia.org/wiki/Http, Décembre 2008.

[10] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. Updated by RFC 2817.

[11] Wikipedia. Real time messaging protocol. http://en.wikipedia.org/wiki/Real_Time_Messaging_Protocol, Décembre 2008.

[12] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP : Session Initiation Protocol. RFC 3261 (Proposed Standard), June 2002. Updated by RFCs 3265, 3853, 4320, 4916, 5393.

[13] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP : A Transport Protocol for Real-Time Applications. RFC 3550 (Standard), July 2003.

[14] Wikipedia. Peer to peer (p2p). http://en.wikipedia.org/wiki/Peer_to_peer, December 2008.

[15] Wikipedia. World Wide Web Consortium. http://en.wikipedia.org/wiki/World_Wide_Web_Consortium, Décembre 2008.

[16] World Wide Web Consortium. Web services architecture. http://www.w3.org/TR/ws-arch/, Février 2004.

[17] World Wide Web Consortium. Web services description language (wsdl) version 2.0. http://www.w3.org/TR/wsdl20/, June 2007.

[18] World Wide Web Consortium. Web services architecture usage scenarios. http://www.w3.org/TR/ws-arch-scenarios/, February 2004.

[19] International Telecommunication Union. Ict statistics database (2008). http://www.itu.int/ITU-D/ICTEYE/Indicators/Indicators.aspx, August 2009.

[20] B. Carpenter. Architectural Principles of the Internet. RFC 1958 (Informational), June 1996. Updated by RFC 3439.

[21] J. Kempf, R. Austein, and IAB. The Rise of the Middle and the Future of End-to-End : Reflections on the Evolution of the Internet Architecture. RFC 3724 (Informational), March 2004.

[22] Miikka Poikselka, Aki Niemi, Hisham Khartabil, and Georg Mayer. *The IMS : IP Multimedia Concepts and Services*. John Wiley & Sons, 2006.

[23] J. Postel. Internet Protocol. RFC 791 (Standard), September 1981. Updated by RFC 1349.

[24] 3GPP. IP Multimedia Subsystem (IMS) ; Stage 2. TS 23.228, 3rd Generation Partnership Project (3GPP), September 2008.

[25] 3GPP. Signalling flows for the IP multimedia call control based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP) ; Stage 3. TS 24.228, 3rd Generation Partnership Project (3GPP), October 2006.

[26] 3GPP. IP Multimedia (IM) Subsystem Cx and Dx Interfaces ; Signalling flows and message contents. TS 29.228, 3rd Generation Partnership Project (3GPP), September 2008.

[27] 3GPP. Presence service ; Architecture and functional description ; Stage 2. TS 23.141, 3rd Generation Partnership Project (3GPP), June 2008.

[28] Open Mobile Alliance. Presence simple specification. Approved Version 1.1, Open Mobile Alliance, 2008.

[29] Open Mobile Alliance. Oma poc control plane. Approved Version 1.0.2, Open Mobile Alliance, 2007.

[30] 3GPP. 3GPP enablers for Open Mobile Alliance (OMA) Push-to-talk over Cellular (PoC) services ; Stage 2. TR 23.979, 3rd Generation Partnership Project (3GPP), June 2007.

[31] 3GPP. Network architecture. TS 23.002, 3rd Generation Partnership Project (3GPP), September 2008.

[32] Yuan Yuan, Jia Jia Wen, Wei Li, and Bing Bing Zhang. A comparison of three programming models for telecom service composition. *Advanced International Conference on Telecommunications*, 0 :1, 2007.

[33] Anahita Gouya, Noël Crespi, and Emmanuel Bertin. SCIM (Service Capability Interaction Manager) Implementation Issues in IMS Service Architecture. *IEEE International Conference on Communications*, 4 :1748–1753, 2006.

[34] A. Maaradji, C. Huang, and N. Crespi. Towards personalized services composition on ims : A basic approach. *International Conference on Advanced Infocomm Technology*, 2009.

[35] Dirk O. Keck and Paul J. Kuehn. The feature and service interaction problem in telecommunications systems : A survey. *IEEE Transactions on Software Engineering*, 24(10) :779–796, 1998.

[36] Anahita Gouya and Noël Crespi. Detection and resolution of feature interactions in IP Multimedia Subsystem. *International Journal of Network Management*, 19(4) :315–33, 2009.

[37] Z. Chentouf, S. Cherkaoui, and A Khoumsi. Implementing online feature interaction detection in SIP environment : early results. *International Conference on Telecommunications*, 1 :515–521, 2003.

[38] Anahita Gouya and Noël Crespi. Service broker for managing feature interactions in IP Multimedia Subsystem. *International Conference on Networking*, 1 :54, 2007.

[39] Vincent Verdot. Architectures de services dans l'IMS. Technical report, Mémoire de master, Université Paris 6 – Télécom SudParis, 2005.

[40] Wikipedia. Operating system. `http://en.wikipedia.org/wiki/Operating_system`, August 2009.

[41] Wikipedia. Embedded system. `http://en.wikipedia.org/wiki/Embedded_system`, August 2009.

[42] J. Postel. Transmission Control Protocol. RFC 793 (Standard), September 1981. Updated by RFCs 1122, 3168.

[43] J. Postel. User Datagram Protocol. RFC 768 (Standard), August 1980.

[44] H. Zimmermann. Osi reference model–the iso model of architecture for open systems interconnection. *IEEE Transactions on Communications*, 28(4) :425–432, 1980.

[45] ITU-T. Mobility management requirements for NGN, ITU-T recommendation Q.1706/Y.2801. Technical report, International Telecommunication Union, 2006.

[46] 3GPP. Handover procedures. TS 23.009, 3rd Generation Partnership Project (3GPP), March 2007.

[47] R. Shacham, S. Schulzrinne, S. Thakolsri, and W. Kellerer. The virtual device : expanding wireless communication services through service discovery and session mobility. In *Wireless And Mobile Computing, Networking And Communications, 2005. (WiMob'2005), IEEE International Conference on*, volume 4, pages 73–81 Vol. 4, Aug. 2005.

[48] Min-Xiou Chen, Chen-Jui Peng, and Ren-Hung Hwang. Ssip : Split a sip session over multiple devices. *Comput. Stand. Interfaces*, 29(5) :531–545, 2007.

[49] Robert W. Scheifler and James Gettys. *X Window System-Core and Extension Protocols*. Digital Press, 1997.

[50] R.W. Scheifler. X Window System Protocol, version 11 : Alpha update April 1987. RFC 1013, June 1987.

[51] T. Richardson. The rfb protocol, October 2006.

[52] Tristan Richardson, Quentin Stafford-fraser, Kenneth R. Wood, and Andy Hopper. Virtual network computing. *IEEE Internet Computing*, 2 :33–38, 1998.

[53] Google Inc. Google docs. `http://docs.google.com`, 2006.

[54] R. Sparks. The Session Initiation Protocol (SIP) Refer Method. RFC 3515 (Proposed Standard), April 2003.

[55] Henning Schulzrinne and Elin Wedlund. Application-layer mobility using sip. *SIGMOBILE Mob. Comput. Commun. Rev.*, 4(3) :47–57, 2000.

[56] R. Mahy, B. Biggs, and R. Dean. The Session Initiation Protocol (SIP) "Replaces" Header. RFC 3891 (Proposed Standard), September 2004.

[57] R. Sparks. The Session Initiation Protocol (SIP) Referred-By Mechanism. RFC 3892 (Proposed Standard), September 2004.

[58] H. Schulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP). RFC 2326 (Proposed Standard), April 1998.

[59] Henning Schulzrinne and Elin Wedlund. Application-layer mobility using sip. *SIGMOBILE Mob. Comput. Commun. Rev.*, 4 :47–57, July 2000.

[60] M. Boutabia, L.R. Cardenas, and H. Afifi. Client driven qos adaptation for multimedia session mobility. *Global Information Infrastructure Symposium, 2007. GIIS 2007. First International*, pages 141–145, July 2007.

[61] Jong-Min Lee, Myung-Ju Yu, Seong-Gon Choi, and Bo-Seok Seo. Proxy-based multimedia signaling scheme using rtsp for seamless service mobility in home network. *Consumer Electronics, IEEE Transactions on*, 54(2) :481–486, May 2008.

[62] J. Rosenberg, J. Peterson, H. Schulzrinne, and G. Camarillo. Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP). RFC 3725 (Best Current Practice), April 2004.

[63] C. Perkins. IP Mobility Support for IPv4. RFC 3344 (Proposed Standard), August 2002. Updated by RFC 4721.

[64] Ignas Niemegeers Weidong Lu, Anthony Lo. Session mobility support for personal networks using mobile ipv6 and vnat. *5th IEEE Workshop on Applications and Services in Wireless Networks*, 2005.

[65] A. Sanmateu, F. Paint, L. Morand, S. Tessier, P. Fouquart, A. Sollund, and E. Bustos. Seamless mobility across ip networks using mobile ip. *Computer Networks*, 40 :181 – 190, 2002.

[66] R. Stewart. Stream Control Transmission Protocol. RFC 4960 (Proposed Standard), September 2007.

[67] M. Riegel and M Tuexen. Mobile sctp, ietf draft. Technical report, March 2006.

[68] F. Sultan, K. Srinivasan, D. Iyer, and L. Iftode. Migratory TCP : connection migration for service continuity in the Internet. *22nd International Conference on Distributed Computing Systems. Proceedings.*, pages 469–470, 2002.

[69] Mark Alexander Connell Snoeren. *A session-based architecture for Internet mobility.* PhD thesis, Massachusetts Institute of Technology, 2003.

[70] J.E. Smith and Ravi Nair. The architecture of virtual machines. *Computer*, 38(5) :32–38, May 2005.

[71] Iain Craig. *Virtual Machines.* Springer, Berlin, 2005.

[72] C. Le Goater, D. Lezcano, C. Calmels, D. Hansen, S.E. Hallyn, and H. Franke. Making applications mobile under Linux. *Proceedings of the Linux Symposium*, 1 :347–368, July 2006.

[73] I. Jørstad, D. van Thanh, and S. Dustdar. A service continuity layer for mobile services. *Wireless Communications and Networking Conference*, 4 :2300–2305, March 2005.

[74] Teemu Koponen, Andrei Gurtov, and Pekka Nikander. Application mobility with Host Identity Protocol. Technical report, Helsinki Institute for Infor-

mation Technology and Ericsson (Network and Distributed System Security Symposium ), 2005.

[75] Yu Zhou, Jiannong Cao, Vaskar Raychoudhury, Joanna Siebert, and Jian Lu. A Middleware Support for Agent-Based Application Mobility in Pervasive Environments. *International Conference on Distributed Computing Systems Workshops*, 2007.

[76] Marc Barisch, Jochen Kögel, and Sebastian Meier. A Flexible Framework for Complete Session Mobility and Its Implementation. *EUNICE '09 : Proceedings of the 15th Open European Summer School and IFIP TC6.6 Workshop on The Internet of the Future*, pages 188–198, 2009.

[77] Lars Erik Liljebäck. *User and Session mobility in a Plug-and-Play architecture*. PhD thesis, Norwegian University of Science and Technology, 2002.

[78] 3GPP. Radio Resource Control (RRC) ; Protocol specification. TS 25.331, 3rd Generation Partnership Project (3GPP), September 2008.

[79] Wikipedia. Multihoming. `http://en.wikipedia.org/wiki/Multihoming`, August 2009.

[80] Vincent Verdot, Arnaud Gonguet, and Yann Gasté. IMS et services émergents : exemple de la continuité de service. *Techniques de l'Ingénieur*, 2006.

[81] Wikipedia. Microsoft Media Server Protocol. `http://en.wikipedia.org/wiki/Microsoft_Media_Services`, August 2008.

[82] Microsoft. Windows Media Services. `http://www.microsoft.com/windows/windowsmedia/forpros/server/server.aspx`, 2008.

[83] Wikipedia. Progressive download. `http://en.wikipedia.org/wiki/Progressive_download`, 2007.

[84] Wikipedia. Adobe Flash. `http://en.wikipedia.org/wiki/Adobe_Flash`, August 2009.

[85] A. Gonguet, G. K. Klungsøyr, V. Verdot, I. Pellejero, and D. Belaïd. Session continuity : How to make a wish come true ? *Seamless Service MObility workshop (SSMO'07) at the IEEE Global Information Infrastructure Symposium*, 2007.

[86] Tim O'Reilly. What is Web 2.0. `http://oreilly.com/web2/archive/what-is-web-20.html`, September 2005.

[87] Google. Google desktop. http://desktop.google.com/features.html, 2008.

[88] Vincent Verdot, Noël Crespi, and Yann Gasté. A distributed Service Manager for seamless service continuity. *IEEE International Conference on Networking and Services (ICNS'07)*, 2007.

[89] Vincent Verdot, Noël Crespi, and Yann Gasté. Study of service-level continuity for communication applications. *Seamless Service MObility workshop (SSMO'07) at the IEEE Global Information Infrastructure Symposium*, 2007.

[90] Vincent Verdot. Launchpad distributed Service Manager Project. https://launchpad.net/dsm, September 2008.

[91] N. Maibaum and T. Mundt. Jxta : a technology facilitating mobile peer-to-peer networks. In *Mobility and Wireless Access Workshop. MobiWac. International*, pages 7–13, 2002.

[92] Sun Microsystems. JXTA community project. https://jxta.dev.java.net/, 2007.

[93] Vincent Verdot and Noël Crespi. dSM : distributed Service Manager for seamless mobility. *Presented at the IEEE International Conference INFOCOM'09*, 2009.

[94] Sun Developer Network. Java Development Kit. http://java.sun.com/javase/downloads/index.jsp, 2008.

[95] University of Maryland. TerpOffice 5.0. http://www.cs.umd.edu/~atif/TerpOfficeWeb/TerpOfficeV5.0/index.html, 2006.

[96] B. Quinn and K. Almeroth. IP Multicast Applications : Challenges and Solutions. RFC 3170 (Informational), September 2001.

[97] Microsoft. Windows Vista Performance and Reliability Monitoring Step-by-Step Guide. http://technet.microsoft.com/en-us/library/cc722173(WS.10).aspx, 2008.

[98] Wireshark Foundation. Wireshark. www.wireshark.org, 2008.

[99] Tim Lindholm and Frank Yellin. *The Java Virtual Machine Specification*. Prentice Hall PTR ; 2 edition, 1999.

[100] W3C - Ubiquitous Web domain. Extensible Markup Language (XML). http://www.w3.org/XML/, 2008.

[101] Jean-Loup Gailly and Mark Adler. gzip home page. http://www.gzip.org/, 2007.

# Annexes

# Annexe A

# Liste des publications

## 1 Conférences et journaux

V. Verdot. *Architectures de Services dans l'IMS* (in french), Master's thesis, Université Paris 6 – Télécom SudParis, september 2005.

V. Verdot, A. Gonguet and Y. Gasté. *IMS et services émergents : exemple de Continuité de Service* (in french), published in Techniques de l'Ingénieur, september 2006.

V. Verdot, N. Crespi, and Y. Gasté. *A Distributed Service Manager for Seamless Service Continuity* in IEEE International Conference on Networking and Services, ICNS'07, Greece, Athens, June 2007.

V. Verdot, N. Crespi and Y. Gasté. *Study of Service-Level Continuity for Communication Applications*, Seamless Service MObility workshop, SSMO'07, at the IEEE Global Information Infrastructure Symposium, Marrakech, Morocco, July 2007.

A. Gonguet, G. K. Klungsøyr, V. Verdot, I. Pellejero and D. Belaïd. *Session Continuity : How to Make a Wish Come True ?*, Seamless Service MObility workshop, SSMO'07, at the IEEE Global Information Infrastructure Symposium, Marrakech, Morocco, July 2007.

V. Verdot and N. Crespi. *dSM : distributed Service Manager for Seamless Mobility* presented in IEEE INFOCOM 2009, Rio de Janeiro, Brazil, April 2009.

V. Verdot, M. Boussard, N. Bouché, S. Shanmugalingam, and L. Fournigault. *The Bridging of Two Worlds : A Web-IMS Communication Solution*, Internet and Multimedia Systems and Applications, EuroIMSA'09, Cambridge, UK, July 2009.

V. Verdot, F. Guern, N. Crespi and Y. Gasté. *dSM, Toward a Unified Service Mobility Model*, submitted to IEEE Transactions on Mobile Computing.

V. Toubiana, V. Verdot, G. Burnside and E. Joubert. *R2M : Reputation Model for Mashups*, IEEE Consumer Communications & Networking Conference, CCNC'10, Las Vegas, USA, January 2010.

# 2 Brevets

A. Gonguet and V. Verdot. Patent 0754738 : *Automated session continuity in IMS*, 2006.

R. Daniellou and V. Verdot. Patent 07291542.4 : *RTSP field for foreseen network unavailability*, 2007.

A. Gonguet and V. Verdot. Patent 0704525 : *Media session keeper*, 2007.

R. Daniellou and V. Verdot. Patent 0704123 : *Using transparent proxies with MMS*, 2007.

V. Verdot and Y. Gasté. Patent 0801689 : *URL-based media information inference*, 2008.

V. Verdot and F. Poussière. Patent 0804331 : *IMS media community audience*, 2008.

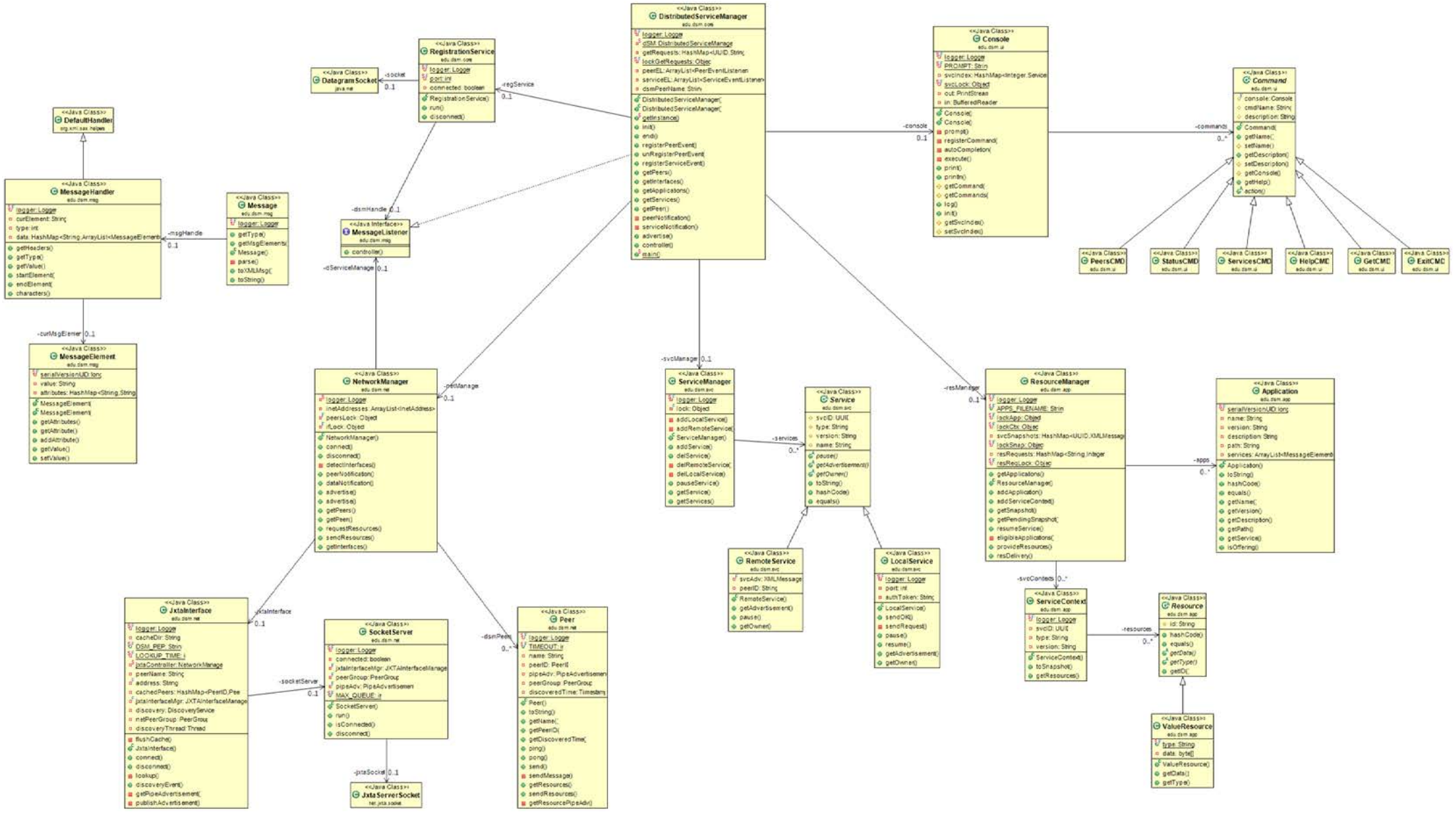
V. Verdot and B. Christophe. Patent 0903207 : *Interactive Web Agora*, 2009.

V. Toubiana, V. Verdot and B. Christophe. Patent 0953217 : *3D shapes simulating glove*, 2009.

# Annexe B

# Diagramme de classes du prototype *dSM*

*(cf. verso)*


DefaultHandler
org.xml.sax.helpers
MessageHandler
edu.dsm.msg
logger: Logger
curElement: String
type: int
data: HashMap<String,ArrayList<MessageElement>>
getHeaders()
getType()
getValue()
startElement()
endElement()
characters()
-msgHandle 0..1
Message
edu.dsm.msg
logger: Logger
getType()
getMsgElements()
Message()
parse()
toXMLMsg()
toString()
-curMsgElement 0..1
MessageElement
edu.dsm.msg
serialVersionUID: long
value: String
attributes: HashMap<String,String>
MessageElement()
MessageElement()
getAttributes()
getAttribute()
addAttribute()
getValue()
setValue()
DatagramSocket
java.net
-socket 0..1
RegistrationService
edu.dsm.core
logger: Logger
port: int
connected: boolean
RegistrationService()
run()
disconnect()
-regService 0..1
-dsmHandle 0..1
MessageListener
edu.dsm.msg
controller()
-dsServiceManager 0..1
DistributedServiceManager
edu.dsm.core
logger: Logger
dSM: DistributedServiceManage
getRequests: HashMap<UUID,Strin
lockGetRequests: Objec
peerEL: ArrayList<PeerEventListener
serviceEL: ArrayList<ServiceEventListener
dsmPeerName: Strin
DistributedServiceManager(
DistributedServiceManager(
getInstance()
init()
end()
registerPeerEvent()
unRegisterPeerEvent(
registerServiceEvent()
getPeers()
getInterfaces()
getApplications()
getServices()
getPeer()
peerNotification()
serviceNotification()
advertise()
controller()
main()
-console 0..1
Console
edu.dsm.ui
logger: Logger
PROMPT: Strin
svcIndex: HashMap<Integer,Service
svcLock: Object
out: PrintStream
in: BufferedReader
Console()
Console()
prompt()
registerCommand(
autoCompletion(
execute()
print()
println()
getCommand(
getCommands(
log()
init()
getSvcIndex()
setSvcIndex()
-commands 0..*
Command
edu.dsm.ui
console: Console
cmdName: String
description: String
Command()
getName()
setName()
getDescription()
setDescription()
getConsole()
getHelp()
action()
PeersCMD
edu.dsm.ui
StatusCMD
edu.dsm.ui
ServicesCMD
edu.dsm.ui
HelpCMD
edu.dsm.ui
GetCMD
edu.dsm.ui
ExitCMD
edu.dsm.ui
-svcManager 0..1
ServiceManager
edu.dsm.svc
logger: Logger
lock: Object
addLocalService()
addRemoteService()
ServiceManager()
addService()
delService()
delRemoteService()
delLocalService()
pauseService()
getService()
getServices()
-services 0..*
Service
edu.dsm.svc
svcID: UUID
type: String
version: String
name: String
pause()
getAdvertisement()
getOwner()
toString()
hashCode()
equals()
RemoteService
svcAdv: XMLMessage
peerID: String
RemoteService()
getAdvertisement()
pause()
getOwner()
LocalService
logger: Logger
port: int
authToken: String
LocalService()
sendOK()
sendRequest()
pause()
resume()
getAdvertisement()
getOwner()
-resManager 0..1
ResourceManager
edu.dsm.app
logger: Logger
APPS_FILENAME: Strin
lockApp: Object
lockCtx: Objec
svcSnapshots: HashMap<UUID,XMLMessage
lockSnap: Objec
resRequests: HashMap<String,Integer
resReqLock: Objec
getApplications()
ResourceManager()
addApplication()
addServiceContext()
getSnapshot()
getPendingSnapshot()
resumeService()
eligibleApplications()
provideResources()
resDelivery()
-apps 0..*
Application
edu.dsm.app
serialVersionUID: long
name: String
version: String
description: String
path: String
services: ArrayList<MessageElement>
Application()
toString()
hashCode()
equals()
getName()
getVersion()
getDescription()
getPath()
getService()
isOffering()
-svcContexts 0..*
ServiceContext
edu.dsm.app
logger: Logger
svcID: UUI
type: String
version: String
ServiceContext()
toSnapshot()
getResources()
-resources 0..*
Resource
edu.dsm.app
id: String
hashCode()
equals()
getData()
getType()
getID()
ValueResource
edu.dsm.app
type: String
data: byte[]
ValueResource()
getData()
getType()
NetworkManager
edu.dsm.net
logger: Logger
inetAddresses: ArrayList<InetAddress>
peersLock: Object
rfLock: Object
NetworkManager()
connect()
disconnect()
detectInterfaces()
peerNotification()
dataNotification()
advertise()
advertise()
getPeers()
getPeer()
requestResources()
sendResources()
getInterfaces()
-nwManage 0..1
-jxtaInterface 0..1
JxtaInterface
edu.dsm.net
logger: Logger
cacheDir: String
DSM_PEP: Strin
LOOKUP_TIME: i
jxtaController: NetworkManage
peerName: Strin
address: String
cachedPeers: HashMap<PeerID,Pee
jxtaInterfaceMgr: JXTAInterfaceManage
discovery: DiscoveryService
netPeerGroup: PeerGroup
discoveryThread: Thread
flushCache()
JxtaInterface()
connect()
disconnect()
lookup()
discoveryEvent()
getPipeAdvertisement()
publishAdvertisement()
-socketServer 0..1
SocketServer
edu.dsm.net
logger: Logger
connected: boolean
jxtaInterfaceMgr: JXTAInterfaceManage
peerGroup: PeerGroup
pipeAdv: PipeAdvertisemen
MAX_QUEUE: i
SocketServer()
run()
isConnected()
disconnect()
-jxtaSocket 0..1
JxtaServerSocket
net.jxta.socket
-dsmPeer 0..*
Peer
edu.dsm.net
logger: Logger
TIMEOUT: i
name: String
peerID: PeerI
pipeAdv: PipeAdvertisemen
peerGroup: PeerGroup
discoveredTime: Timestamp
Peer()
toString()
getName()
getPeerID()
getDiscoveredTime()
ping()
pong()
send()
sendMessage()
getResources()
sendResources()
getResourcePipeAdv()

# Annexe C

# Questionnaire du test qualitatif

Extrait de la version française du questionnaire envoyé aux participants du test qualitatif décrit en section 8.3.1, visant à étudier le comportement des utilisateurs face à des contraintes de mobilité.

### Début du test.

#### Situation.

Imaginons que vous possédez deux ordinateurs.

Vous rédigez un document sur le premier ordinateur. Après un certain laps de temps vous devez interrompre votre travail et le poursuivre sur un deuxième ordinateur. Votre but étant de reprendre rapidement et simplement votre tâche, comment procèderiez-vous ?

(Exprimez votre solution le plus simplement possible en donnant si nécessaire des détails tirés de votre expérience, précisez ce que vous savez faire ou ce que vous imaginez possible. Si vous n'avez aucune idée, inventez une méthode).

Votre solution (racontez pas-à-pas votre dernière expérience) : ...

#### Questions.

Considérant la solution proposée ci-dessus, répondez à ces cinq questions en fournissant autant de détails que vous jugez utile. Passez les questions que vous ne comprenez pas.

**Question 1** : Avez-vous souvent l'occasion d'employer votre solution (précisez la fréquence) ? ...

**Question 2** : Selon vous, votre solution est-elle pratique et simple à mettre en œuvre, justifier ? ...

**Question 3** : Quelles sont les difficultés éventuelles liées à votre solution (système d'exploitation, applications, connexion réseau, matériel, etc) ? ...

**Question 4** : Avec votre solution, avez-vous l'impression d'être interrompu ou ralenti dans votre tâche ? Cela vous gêne-t-il, pourquoi ? ...

**Question 5** : Si au lieu d'éditer un document, vous regardiez une vidéo sur Internet, votre solution permettrait-elle de transférer de la même façon cette tâche d'un ordinateur à l'autre ? Si non, comment vous y prendriez-vous ? ...

### Commentaires.

Si vous avez des remarques concernant ce test ou des précisions à apporter à vos réponses, indiquez-les ici.

Vos remarques : ...

## Fin du test.