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.

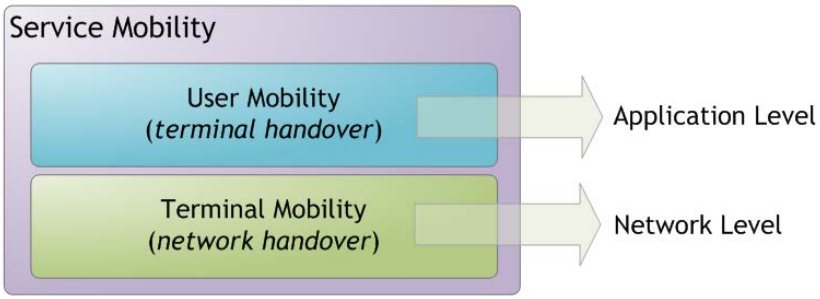

FIGURE 2.1 – Problématiques de mobilité.

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).

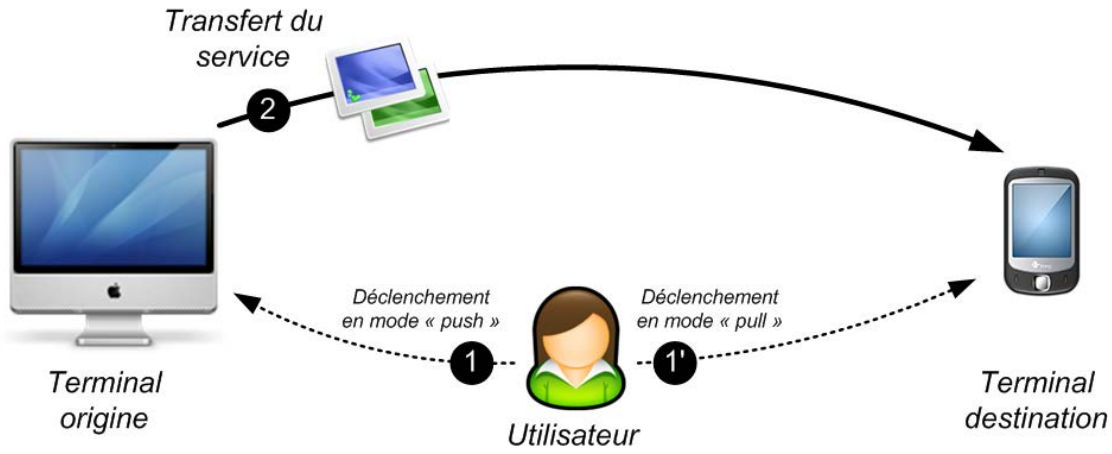

FIGURE 2.2 – Principe de transfert (*terminal handover*).

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).

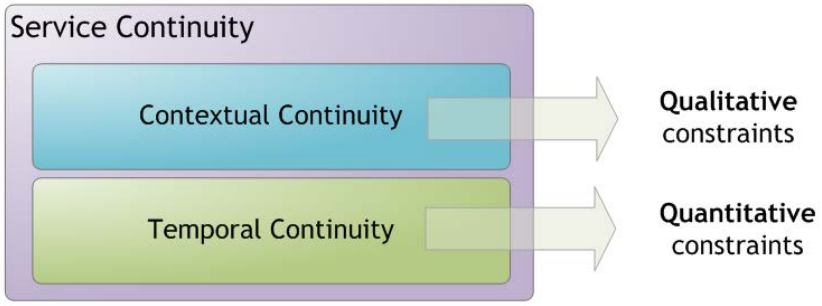

FIGURE 2.3 – Problématiques de continuité.

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.

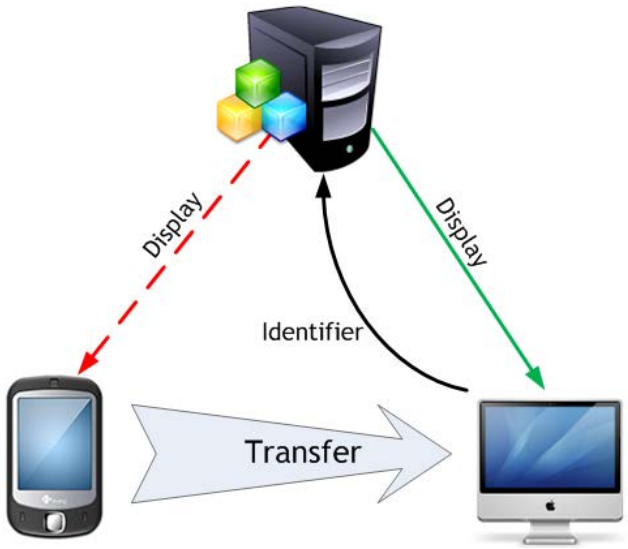

FIGURE 2.4 – Approche architecturale.

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

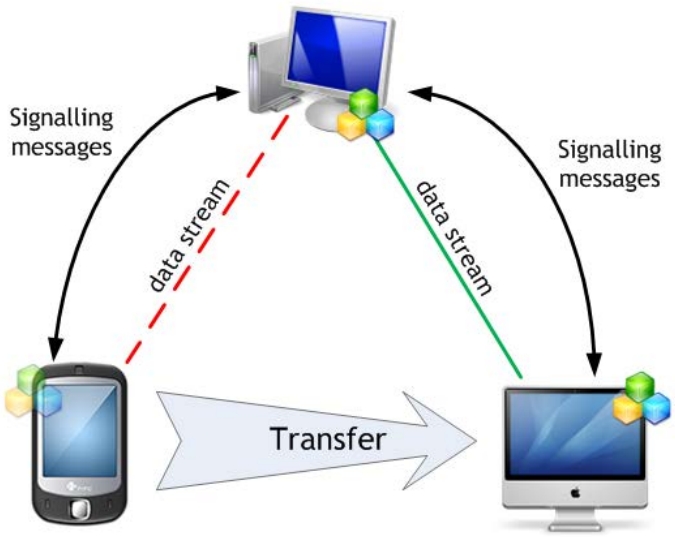

FIGURE 2.5 – Approche session.

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.

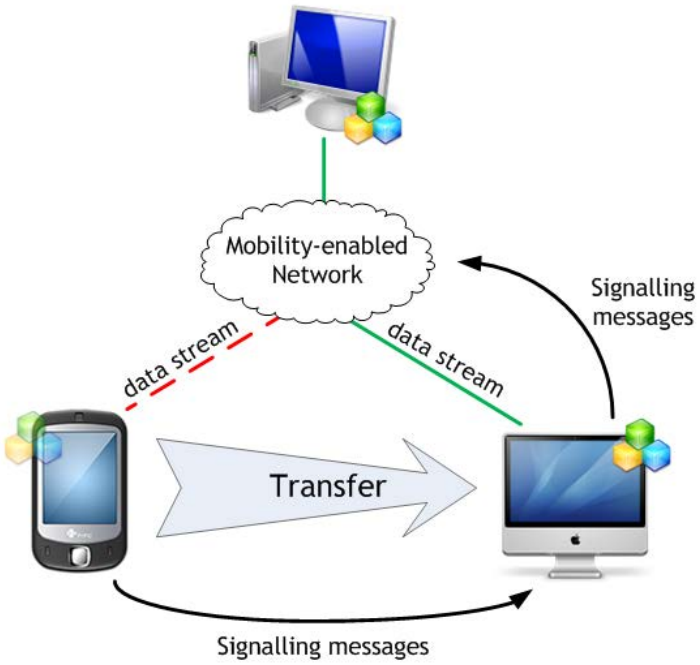

FIGURE 2.6 – Approche réseau.

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.

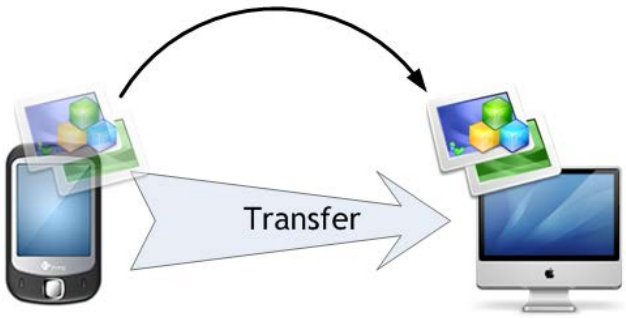

FIGURE 2.7 – Approche applicative.

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.