N° D'ORDRE 05/2005-M/IN

REBUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE D'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE

« HOUARI BOUMEDIENE »

FACULTE D'ELECTRONIQUE ET D'INFORMATIQUE

MEMOIRE

Présenté pour l'obtention du diplôme de MAGISTER

En : INFORMATIQUE

Spécialité : Programmation et Systèmes

Par : BOUABACHE Fatiha

SUJET

Le Context-aware Computing

Conception d'une Architecture d'Adaptation pour une Application Multimédia Context-Aware

Soutenu Le 05/06/2005 , devant le jury composé de :

Mr. BADACHE.N	Professeur, USTHB,	Président
Mme. NOUALI N.	Chargée de recherche, CERIST ,	Directeur de thèse
Mr. AHMED NACER M.	Professeur, USTHB,	Examinateur
Mme. BOUKALA M.	Maître de conférence,USTHB	Examinateur
Mme. MOUSSAOUI S.	Chargé de cours, USTHB,	Invité

Dédicaces

A la mémoire de mon père,

A ma très chère mère,

A mon mari,

A tous les membres de ma famille,

A tous mes amis,

Je dédie ce travail.

Remerciements

Je tiens à remercier le tout puissant de m'avoir donné le courage et la patience jusqu'à l'achèvement de ce travail.

J'exprime ma profonde reconnaissance et mes vifs remerciements à ma directrice de thèse Mme. N. NOUALI, chef de la Division de l'Informatique Mobile (CERIST), pour m'avoir fait confiance en me proposant ce sujet. Je la remercie surtout pour son soutien moral, et également pour ses lectures attentives de mes rapports, dont les critiques et suggestions ont été d'un grand apport pour la finalité de ce travail.

Je remercie Mr. N. BADACHE, Mme M. BOUKALA, Mr. M. AHMED NACER et Mme. S. MOUSSAOUI d'avoir accepté de juger ce travail.

Mes remerciements s'adressent aussi à ma famille, mes amis et tous ceux qui m'ont aidé et contribué de prés ou de loin par une quelconque façon, que ce soit par leur amitié, leurs conseils ou leur soutient moral ou matériel.

Sommaire :

Introduction Générale

Les progrès réalisés en matière de communication (réseaux haut débit, normalisation des protocoles et des architectures à objets distribués, explosion d'Internet, les communications non filaires…) conjugués à la prolifération des calculateurs ultralégers connectables à l'Internet (agendas électroniques, téléphones cellulaires, cartes à puce,…) doivent permettre à un utilisateur d'accéder à des données et d'exécuter des traitements n'importe où, n'importe quand et à partir de n'importe quel terminal. Les applications de ce type sont multiples. Il peut s'agir d'applications personnelles dans lesquelles un utilisateur veut pouvoir accéder à tout moment à des données privées de diverses natures (données bancaires, agenda, carnet d'adresse, dossier médical, book marks,…). Il peut également s'agir d'applications professionnelles dans lesquelles des employés doivent accéder et partager en permanence et où qu'ils se trouvent des informations relatives à leur entreprise.

Contrairement aux applications traditionnelles qui s'exécutaient sur des environnements fixes caractérisés par contexte statique, les applications informatiques dans l'environnement mobile ont comme support des réseaux hétérogènes comportant des unités fixes ou mobiles ayant des capacités variables et connectées via une liaison filaire ou sans fil avec un débit également variable. Les applications subissent pendant leur exécution des changements très fréquents dans leur contexte d'exécution (ressources disponibles, la location, serveur d'attachement, qualité de la bande passante, ajout de nouveau service, etc.). Cette nature très dynamique de changement dans les conditions d'exécution engendrées par les différentes contraintes du réseau mobile a induit de nouveaux besoins qui sont pris en charge dans ce qui est appelé *l'adaptation dynamique et le context-aware computing*.

De nombreux travaux ont été menés pour adapter les environnements informatiques aux contraintes de la mobilité. Les approches existantes s'étendent du niveau système masquant la mobilité à l'utilisateur consistant ainsi en une émulation des protocoles appliqués dans les systèmes statiques, jusqu'au niveau applicatif, où l'application est appelée à faire partie du processus d'adaptation en étant sensible à l'environnement d'exécution et capables de choisir une qualité de service selon le besoin et les conditions d'exécution. Dans ce mémoire nous nous sommes essentiellement intéressés aux applications multimédias. En effet, d'une part, le flux multimédia est très sensible aux variations dynamiques engendrées par les différentes contraintes de l'environnement mobiles, et d'autre part, la tendance sans cesse croissante est d'inclure ce type de flux dans la plupart des applications mobiles. Notre contribution dans ce mémoire s'articule autour de la proposition d'une architecture d'adaptation aux variations dynamiques de contexte pour une application multimédia. Pour cela nous avons organisé notre thèse en deux grandes parties incluant chacune deux chapitres :

La première partie (état de l'art) est présentée dans l'objectif de familiariser le lecteur avec le problème d'adaptation et le context-aware computing. Le chapitre 1 introduit l'environnement mobile en présentant les différentes caractéristiques d'un système mobile

ainsi que ces différentes contraintes et les problèmes qu'il a engendrés. L'accent est mis sur le besoin d'adaptation et du context-aware computing.

Le chapitre 2, présente une étude des différents travaux liés à l'adaptation et au context-aware computing. Sont présentés l'évolution historique et les besoins de ces deux processus. Il commence d'abord par donner les différentes définitions du concept d'adaptation, de contexte, et de context-aware computing. Nous passons en revue les différentes solutions proposées en donnant une taxonomie et une étude comparative des différentes solutions.

La deuxième partie va présenter le domaine d'application choisi et la solution proposée ; elle sera structurée en deux chapitres. Le chapitre 3 est consacré au domaine d'application choisi qui est le multimédia. Il commence par présenter les différentes caractéristiques du flux multimédia, les différents types d'application multimédias, et les protocoles les plus utilisés dans ce genre d'applications. Il met principalement l'accent, sur l'effet des contraintes de l'environnement mobile sur la qualité de service, et le besoin d'une adaptation dynamique. Ce chapitre fini par donner un aperçu sur les travaux d'adaptation (techniques et systèmes) liés au domaine.

Le chapitre4, quant à lui présente notre contribution qui se résume en une architecture d'adaptation d'une application de vidéo à la demande et plus précisément d'une application de vidéo formation. L'objectif de cette architecture est de garantir le bon fonctionnement de l'application, et donc un certain niveau de qualité de service perçu par l'utilisateur. La principale caractéristique de l'architecture proposée réside dans sa capacité à offrir un framework permettant à l'application non seulement de s'adapter aux variations du contexte mais aussi de le faire de façon dynamique en se basant sur des données du contexte obtenues elles aussi dynamiquement.

Première partie : Etat de l'art

L'objectif de cette partie est de familiariser le lecteur avec le problème d'adaptation et le context-aware computing. Nous allons commencer par présenter les différentes contraintes d'un environnement mobile et mettre l'accent sur le besoin de ces deux processus. Ensuite, nous allons présenter les différentes catégories d'applications adaptables et context-aware. Et enfin, par présenter les applications multimédias comme domaine d'application (les problèmes, et les différentes solutions).

Chapitre I :
Un Système Distribué Mobile :

I.1 Introduction :

Les progrès réalisés en matière de communication (réseaux haut débit, normalisation des protocoles et des architectures à objets distribués, explosion d'Internet, les communications non filaires…) conjugués à la prolifération des calculateurs ultralégers connectables à l'Internet (agendas électroniques, téléphones cellulaires, cartes à puce,…) doivent permettre à un utilisateur d'accéder à des données et d'exécuter des traitements n'importe où, n'importe quand et à partir de n'importe quel terminal, donnant naissance ainsi à un nouvel environnement, appelé environnement mobile ou nomade. Ce nouvel environnement n'astreint plus l'usager à une localisation fixe, mais lui permet une libre mobilité tout en restant connecté au réseau, introduisant ainsi de nouveaux problèmes et de nouveaux besoins

I.2 Architecture d'un système distribué avec sites mobiles :

Comme le montre la figure suivante (Figure1.1), un environnement mobile est composé de deux ensembles d'entités : les sites fixes d'un réseau de communication filaire classique et les sites mobiles. Certains sites fixes, appelés stations de base (SB) ou stations de support mobile (Mobile Support Station) sont munis d'une interface de communication sans fil pour la communication directe avec les sites ou unités mobiles (UM), localisés dans une zone géographique limitée, appelée cellule. A chaque SB correspond une cellule à partir de laquelle des unités mobiles peuvent émettre et recevoir des messages. Alors que les sites fixes sont interconnectés entre eux à travers un réseau de communication filaire, généralement fiable et d'un débit élevé, les liaisons sans fil ont une bande passante limitée qui réduit sévèrement le volume des informations échangées. Une unité mobile ne peut être, à un instant donné, directement connectée qu'à une seule station de base. Elle peut communiquer avec les autres sites à travers la station à laquelle elle est directement rattachée. L'autonomie réduite de sa source d'énergie, lui occasionne de fréquentes déconnexions du réseau ; sa reconnexion peut alors se faire dans un nouvel environnement et donc dans une nouvelle localisation [Bad95].

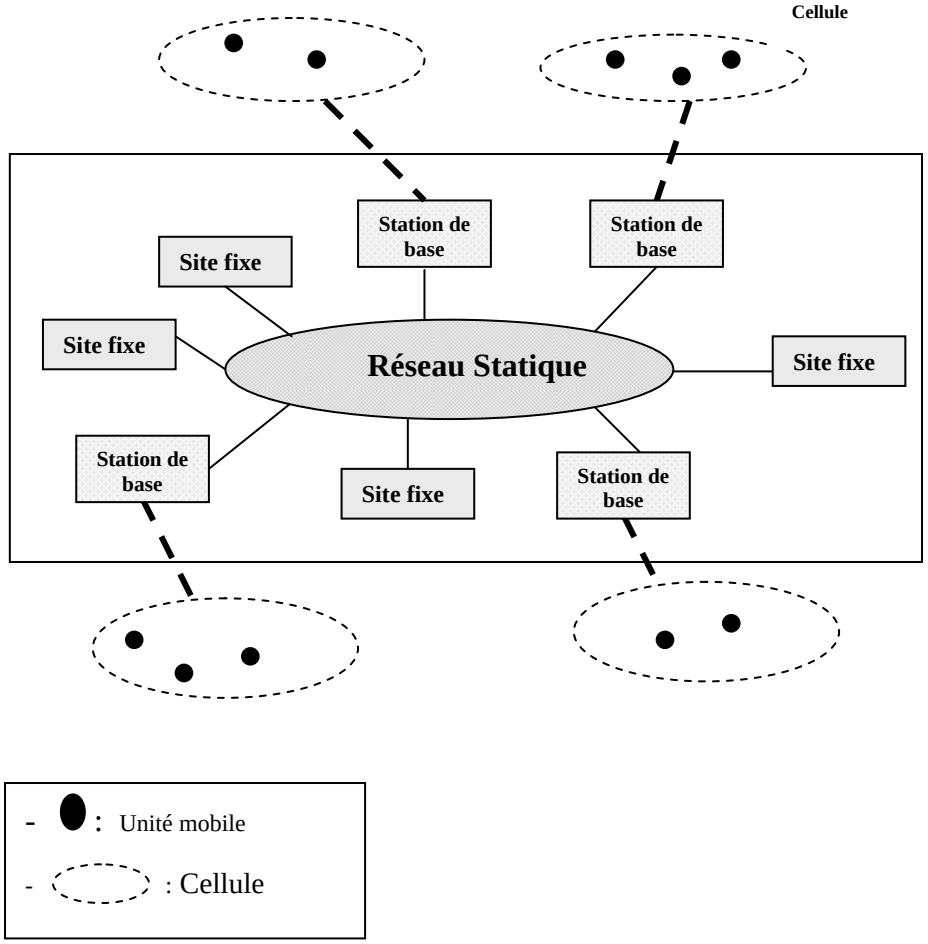

Figure1.1. *Architecture avec des sites mobiles*

I.3 Caractéristiques des réseaux mobiles sans fil :

Un système distribué est constitué d'un ensemble de composants distribués sur plusieurs ordinateurs et reliés à travers un réseau informatique. Ces composants interagissent pour échanger des données ou accéder à des services d'un autre site [CLW02].

Bien que cette définition s'applique aux systèmes fixes et mobiles, ces deux types de systèmes ont des caractéristiques différentes au niveau des dispositifs, de la connexion réseau et du contexte d'exécution (Figure1.2.) [CLW02].

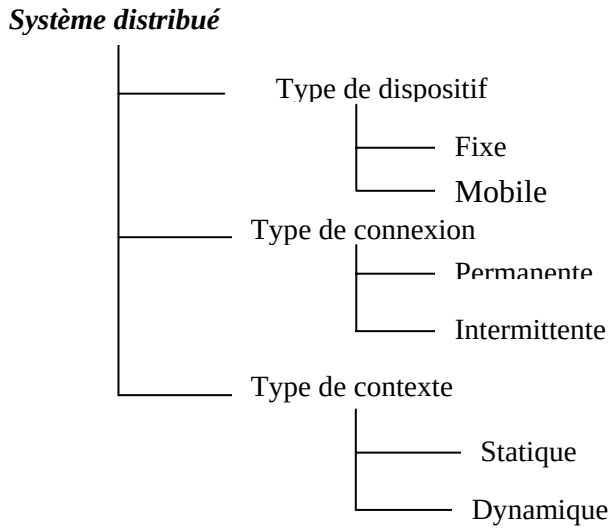

Figure1.2. *Caractéristiques des systèmes distribués mobiles*

I.3.1 Les Unités mobiles :

Les unités mobiles peuvent être de diverses configurations : avec ou sans disque, de capacité de mémorisation et de calcul plus au moins modeste et alimentées par des sources d'énergie autonomes (batteries). Elles sont équipées d'interfaces de communication sans fil pour l'accès aux réseaux d'information [Bad98].

Ces configurations diverses permettent d'envisager deux cas extrêmes d'utilisation des unités mobiles dans les systèmes distribués [Bad95] [Nar94] : Soit que les unités mobiles sont utilisées comme de simples terminaux dépourvus de capacité de calcul et dépendant d'un site fixe. Dans ce cas l'exécution des applications se fait sur le réseau fixe. Soit qu'elles sont utilisées comme de véritables stations de travail avec suffisamment de ressources. On parle dans ce cas, d'un modèle avec cache.

I.3.2 Mode de fonctionnement d'une unité mobile

Contrairement aux systèmes distribués classiques, où un site est soit connecté, soit totalement déconnecté du réseau, dans un environnement mobile, il peut y avoir quatre modes de connexion : fortement connecté, partiellement connecté, veille et déconnecté. Ces modes de fonctionnement dépendent des ressources disponibles, entre autres la largeur de la bande passante [Bag96][Bad95].

Le mode fortement connecté : dans ce cas l'unité mobile dispose d'une connexion normale assimilée à une connexion LAN, le mobile peut basculer de ce mode d'exécution vers le mode veille pour des considérations d'énergie ou vers le mode partiellement connecté dans le cas d'affaiblissement de la bande passante.

Le mode veille : dans ce cas la vitesse du processeur est réduite et aucun calcul n'est effectué. L'unité mobile est alors en attente de réception d'un message pour reprendre son fonctionnement normal. Ce mode peut être utilisé pour économiser de l'énergie.

Le mode partiellement connecté : Durant ce mode de fonctionnement, toutes les communications avec le réseau doivent être limitées. Une unité mobile peut basculer à cet état en exécutant un protocole spécifique.

Le mode déconnecté : Alors que dans un environnement distribué classique, les déconnexions sont imprévisibles, dans un environnement mobile, un site doit pouvoir anticiper une possible déconnexion et donc exécuter un protocole de déconnexion avant qu'elle ne se détache physiquement du réseau. Ceci permet d'assurer que l'unité a téléchargé suffisamment de données et les informations d'état nécessaires, lui permettant de continuer à fonctionner indépendamment des autres sites.

Finalement, il faut noter qu'au cours de son fonctionnement et dans n'importe quel mode, une unité mobile peut sortir des limites de sa cellule courante pour entrer dans une nouvelle cellule. Un protocole dit protocole "handoff" est exécuté et permet le transfert des informations d'état relatives au calcul mobile en cours à la station de base de la nouvelle cellule.

I.3.3 Les types de réseaux sans fil :

L'évolution des réseaux sans fil tend aujourd'hui à l'établissement de chaînes de mobilité capables de répondre au besoin croissant de connexion permanente des individus et des organisations dans l'ensemble de leurs déplacements. Pour les prochaines années, se dessine un scénario de complémentarité/concurrence où différentes technologies mobiles coexisteront au profit d'usages pluriels. Une complémentarité peut être envisagée entre :

- Les WWAN (Wireless Wide Area Network) : ce sont des technologies associées à des services de pleine mobilité, qui offriront une couverture quasi-universelle et une continuité de communication grâce à leur fonction de roaming. Elles peuvent être classées en deux catégories : les technologies de la téléphonie mobile telles que (HSCSD, GPRS, EDGE, UMTS, et CDMA), et les technologies satellitaires telles que Globalstar, Ellipso, Intelsat, et thuraya [Had03]
- Les WLAN (Wireless Local Area Network) : un WLAN est un système de communication de données pouvant être implémenté comme l'extension ou comme l'alternative à un réseau câblé dans un bâtiment. Il utilise comme vecteur de transmission la *lumière infrarouge* ou les *fréquences radios*. Les fréquences radio sont les plus utilisées en raison d'une portée plus longue, d'une bande passante supérieure et d'une couverture plus large. Généralement la portée d'un WLAN peut aller de dix à quelques centaines de mètres. Parmi les technologies utilisées citons la Wi-Fi IEEE 802.11 et l'hiperLAN).
- Et les WPAN (Wireless Personal Area Network) : ce sont des réseaux sans fil individuels de faible portée (cette portée peut aller de 10 jusqu'à 30 mètre), mais qu'avec le temps devraient augmenter leurs débits et leurs portées pour devenir de véritables concurrents des Wlan. Bluetooth est une des technologies les plus utilisées pour mettre en œuvre ce type de réseau.

Il faut noter qu'une même unité mobile, pendant son déplacement, peut utiliser différents types de technologie à différents moments. Selon son emplacement et ses besoins.

I.4 Contraintes de l'environnement mobile :

Un réseau mobile fait face aux trois contraintes suivantes :

I.4.1 La portabilité :

L'apparition des réseaux mobiles a été encouragée par les avancées technologiques dans la conception des unités mobiles. Cependant, la portabilité et les considérations de poids, de puissance, de taille et d'ergonomie pénalisent les ressources de calcul telles que la vitesse

du processeur, la taille mémoire et la capacité du disque, engendrant ainsi beaucoup de problèmes :

I.4.1.1 Une capacité de stockage réduite :

L'espace de stockage dans une unité portable est limité par la taille et les exigences d'énergie.

Le problème de stockage de données se posait déjà dans les réseaux filaires où des solutions à base de compression, d'accès distant et de partage de données ont été proposées. Cependant, ces solutions sont peu commodes pour des unités mobiles qui sont sujettes à de fréquentes déconnexions.

I.4.1.2 Une petite interface utilisateur :

Les unités mobiles doivent être de petites dimensions afin d'être facilement transportables. La contrainte de taille exige une petite interface utilisateur. Cette restriction sur la taille de l'interface pose le problème d'affichage et de saisie, ainsi que l'adaptation de l'affichage aux différentes tailles qui varient d'un appareil à un autre.

I.4.1.3 Source d'énergie limitée :

Les contraintes de mobilité et de portabilité ont fait que les terminaux sans fil soient alimentés par des sources d'énergie autonomes et donc limitées (batteries).

Malgré les progrès technologiques, le problème de conservation de la source d'énergie le plus longtemps possible reste l'un des défis scientifiques qui ne cesse d'exister. Ce problème doit être pris en compte à tous les niveaux matériels et logiciels.

I.4.1.4 La vulnérabilité :

La portabilité des équipements augmente le risque des accidents de vol et de perte. Ce qui rend les unités mobiles moins fiables et moins sécurisées que les unités statiques.

I.4.2 La mobilité :

Dans l'environnement mobile, la mobilité est une contrainte de base qui doit être prise en compte.

En informatique mobile, trois types de mobilités sont étudiés : la mobilité des applications, la mobilité du matériel, et la mobilité des utilisateurs [Bag96] :

- La mobilité des applications : cette mobilité permet à l'utilisateur d'avoir ses applications actives quelque soit sa localisation.
- La mobilité du matériel : dans ce type de mobilité, un appareil peut se déplacer tout en étant connecté au réseau. Ce qui lui permet de changer d'adresse dynamiquement.
- La mobilité des utilisateurs : ce type de mobilité permet à un utilisateur de se déplacer et d'accéder aux mêmes services depuis différents terminaux. Dans ce cas, les adresses des machines restent fixes.

La mobilité des utilisateurs conjointe à la mobilité du matériel constitue un des cas les plus étudiés dans l'informatique mobile et engendrent plusieurs problèmes:

I.4.2.1 La migration d'adresse et la gestion de localisation :

Contrairement aux réseaux fixes où chaque station a une adresse fixe par laquelle elle se connecte au réseau. Les unités mobiles durant leurs déplacements auront à se connecter via différents points d'accès et donc différentes adresses. Ce qui a un effet direct sur :

1. la conception d'algorithmes distribués qui ne peut être basé sur une topologie fixe du réseau,
2. les protocoles de routage existants qui ne peuvent être utilisés dans les réseaux à grande mobilité,
3. et le coût de communication qui est augmenté par le coût de localisation.

Cette migration dans le réseau doit respecter les points suivants :

- la migration dans un nouvel environnement doit se faire de façon transparente à l'utilisateur et à ses correspondants,
- les mêmes services réseaux doivent être conservés quel que soit le point de rattachement,
- disposer d'une continuité de service lors d'un déplacement en cours de communication.

I.4.2.2 Les informations de localisation :

Dans les réseaux traditionnels où les ordinateurs sont fixes, les informations qui dépendent de leurs localisations telles que les imprimantes disponibles, le serveur de nom (DNS), la zone horaire, …etc., sont configurées de façon statique.

L'un des problèmes de l'environnement mobile est de détecter les changements dans la localisation, et de fournir des mécanismes pour obtenir les informations de configuration appropriées à l'endroit courant. Par exemple, un utilisateur rentrant dans un espace géographique, pourra savoir quels sont les services disponibles, détecter les équipements voisins et les périphériques présents : imprimantes, fax, etc.

En plus du problème de configuration dynamique, dans certains cas, les unités mobiles nécessitent l'accès à plus d'informations relatives à leurs localisation et cela pour leur servir de guide.

Le défi majeur de l'environnement mobile dans ce contexte est de garantir un accès plus flexible à ses informations sans violer l'intimité des autres utilisateurs[1]

I.4.2.3 Hétérogénéité :

A cause de la mobilité, la connectivité dans les réseaux mobiles varie sensiblement en terme de performance et de fiabilité.

Pendant son déplacement, une unité mobile peut communiquer via différentes interfaces réseaux hétérogènes et différents protocoles d'accès. Par exemple : un déplacement de l'intérieur vers l'extérieur peut forcer l'unité de passer d'un réseau WLAN à base de la technologie infrarouge à un réseau cellulaire à base des fréquences radio.

Cette hétérogénéité rend la gestion des réseaux et des applications mobiles plus complexe que la gestion des réseaux et des applications traditionnels.

[1] L'intimité peut être assurée en ne permettant pas à un utilisateur de connaître l'endroit où se trouve une autre personne, même si elle est proche de lui.

I.4.2.4 Les déconnexions :

Pendant son déplacement, une unité mobile peut rentrer dans une zone non couverte ce qui cause sa déconnexion. De plus, la migration des unités mobiles à une certaine cellule peut causer la surcharge de cette dernière ce qui conduit à une faiblesse de la bande passante et peut être à une déconnexion.

I.4.2.5 La sécurité :

En plus de tous ces problèmes, la contrainte de mobilité soulève des problèmes très importants liés à la sécurité, tel que le problème d'authentification, d'anonymat, …etc.

I.4.3 Communication sans fil :

Bien que les unités mobiles puissent parfois se connecter aux réseaux filaires pour avoir une meilleure connexion et ceci quand ils restent stationnaires, un réseau mobile nécessite un accès sans fil pour permettre une libre mobilité aux utilisateurs.

Une communication sans fil où l'environnement interagit avec le signal, bloquant ainsi son chemin et introduisant du bruit et des échos, est beaucoup plus difficile à établir qu'une communication filaire. En conséquence, on peut dire que l'aspect sans fil pose un certain nombre de problèmes qui doivent être étudiés et pris en compte dans un environnement mobile :

I.4.3.1 Les déconnexions :

Contrairement aux réseaux fixes où les déconnexions sont considérées comme des pannes réseaux, dans un environnement mobile, une déconnexion est un évènement normal qui peut se produire à tout moment. Dans les réseaux mobiles, la déconnexion d'une unité peut être volontaire ou involontaire, prévisible ou soudaine, de longue ou de courte durée [Nou01].

Dans le premier cas, déconnexions volontaires, c'est l'utilisateur qui décide de se déconnecter pour une raison ou une autre. Par exemple, s'il détecte une faible connectivité [DSO03], ou s'il sait qu'il va rentrer dans un endroit où les connexions sans fil sont interdites, comme dans un avion[Den03], ou pour préserver sa batterie. Dans ce cas de déconnexion, plusieurs techniques et systèmes ont été proposés pour permettre à un utilisateur de travailler tout en étant déconnecté et ceci en faisant du pré chargement de certaines entités sur le terminal mobile, et puis d'effectuer une réconciliation dès sa reconnexion. Exemple : le système coda [KS92] [SKK90].

Le second type de déconnexion ou déconnexions involontaires, sont généralement le résultat de coupures imprévisibles, comme par exemple, lors du passage de l'utilisateur dans une zone non couverte ou devant un obstacle qui coupe le signal. Ces déconnexions fréquentes et imprévisibles restent un des grands défis de l'environnement mobile et posent plusieurs problèmes au niveau de: la récupération des données, la récupération de l'état de la session, l'exclusion mutuelle, …etc.

I.4.3.2 La faible bande passante :

L'une des limites des liens sans fil est la largeur de la bande passante des technologies utilisées, qui malgré toutes les avancées technologiques dans le domaine reste faible. Cette bande qui est partagée entre les utilisateurs appartenant à la même cellule, fait que la largeur de bande exploitable par un utilisateur soit encore plus faible. Pour augmenter la capacité de

service d'un réseau et pour pallier à ce problème, deux techniques sont utilisées : la technique de recouvrement des cellules sur différentes longueurs d'onde et celle qui réduit la portée du signal pour avoir plus de cellules [Bad98] [GJ94]. Malgré cela la bande passante reste relativement faible et pose toujours des problèmes aux niveaux de la transmission de données.

Certaines techniques logicielles peuvent également intervenir pour faire face à ce problème, parmi lesquelles : la compression de données avant leurs envois, la technique de pré chargement pour réduire l'accès réseaux et le transfert de données sur la bande [CD91].

I.4.3.3 La grande variabilité de la largeur de bande :

La variabilité dans la largeur de bande est liée à la diversité des réseaux sans fil, qui d'une technologie à une autre proposent des capacités différentes.

Cette variabilité est un des problèmes qui doit être pris en compte pendant la conception d'un système mobile [GJ94].

I.4.3.4 La sécurité :

La sécurité des communications sans fil peut être compromise beaucoup plus facilement que les communications filaires, particulièrement si les transmissions s'établissent sur de grandes surfaces. [GJ94][DRN95]

En effet, l'interface air des réseaux sans fil ne bénéficie pas de protection physique et le médium de transmission reste accessible à tous.

Parmi les problèmes rencontrés [Pre01] :

- Les virus qui représentent un danger potentiel aux unités, aux stations de base et aux réseaux sans fil.
- L'écoute et la surveillance des déplacements.
- L'usurpation d'identité et le refus de service.

I.5 Conclusion :

Dans ce premier chapitre, nous avons présenté les caractéristiques des réseaux mobiles ainsi que les différents problèmes engendrés par les trois contraintes principales de ce type de réseau, à savoir : la mobilité, la portabilité et les liens sans fil.

Ces contraintes ne sont pas liées au niveau courant de la technologie, mais sont intrinsèques à cet environnement. Ensemble, elles compliquent la conception et l'exécution des applications, d'où la nécessité d'un processus qui ajuste de façon dynamique le fonctionnement des applications selon les variations induites par ces contraintes.

Chapitre II :
L'Adaptation et le Context-aware Computing

II.1 Introduction :

Contrairement aux applications traditionnelles qui s'exécutaient sur des environnements fixes caractérisés par contexte statique, les applications informatiques de nos jours ont comme support des réseaux hétérogènes comportant des unités fixes ou mobiles, connectées via une liaison filaire ou sans fil. Ceci fait que l'application pendant son exécution aura à subir des changements très fréquents dans son contexte d'exécution (ressources disponibles, la location, serveur d'attachement, etc.).

Cette nature très dynamique de changement dans les conditions d'exécution engendrées par les différentes contraintes du réseau mobile a impliqué de nouveaux besoins : l'adaptation dynamique et le context-aware computing

II.2 Concepts de base :

II.2.1 Définition de l'adaptation :

La définition du dictionnaire est la suivante :

Adapter v. : rendre (un dispositif, des mesures, etc.) apte à assurer ses fonctions dans des conditions particulières ou nouvelles [S1]

Qu'entend-on par adaptation dans les systèmes logiciels ? En reprenant la définition sémantique du terme, une adaptation correspond au processus de modification du système, nécessaire pour garantir un bon fonctionnement dans un contexte donné.

Dans un environnement mobile, l'adaptabilité est un mécanisme essentiel aux applications, qui tente d'ajuster quelques aspects de l'application aux changements détectés dans l'environnement d'exécution. Elle nécessite l'existence de plusieurs façons d'exécuter une application, ou des configurations alternatives qui reflètent différents profils d'utilisation des éléments de calcul.

II.2.2 Définition du contexte

Le mot « contexte » est un mot très vaste qui peut avoir plusieurs sens et peut être utilisé de plusieurs façons dans différents domaines. Cependant dans notre travail, on s'intéresse au contexte utilisé par les applications dans un environnement mobile.

Dans [ST94], où le thème context-aware a été introduit pour la première fois, le mot contexte faisait référence à la localisation, les identifications des personnes et objets proches, et les changements dans ces objets. Dans une définition similaire [BBC97], le contexte est la localisation, l'identité des personnes proches de l'utilisateur, l'heure de la journée, la saison, la température, etc. Dans ces articles ainsi que dans beaucoup d'autres [RPM97][Dey98], la définition du contexte a été donnée par des exemples , ce qui pose problème au moment de décider si un type d'informations qui n'a pas été cité dans la définition est du contexte ou non.

Ultérieurement, dans [Bro96][HNB97][FF98], le contexte a été tout simplement défini par des synonymes. En le définissant par exemple comme étant l'environnement utilisateur ou l'environnement de l'application. Ce qui est difficilement exploitable dans la pratique.

En tentant de régler ces problèmes, les chercheurs ont essayé de donner d'autres définitions plus opérationnelles, parmi lesquelles :

La définition de Dey et al. [DAW99], qui réfère par le mot contexte à l'état informationnel, émotionnel, social, ou physique de l'utilisateur. Schilit et al. [SAW94], considèrent que les aspects les plus importants du contexte sont : où êtes vous, avec qui êtes vous, et quelles sont les ressources proches, et divisent le contexte en trois catégories :

- Contexte de calcul : tel que la connexion réseau, les coûts de communication, la largeur de bande, et les ressources proches (imprimantes, écrans, et postes de travail).
- Contexte utilisateur : tel que le profil utilisateur, la localisation, les personnes proches de lui, et même la situation sociale courante.
- Contexte physique : tel que l'éclairage, les niveaux de bruit, l'état du trafic, et la température.

Auxquelles, Chen et Kotz ajoutent une quatrième catégorie qui est :

- Le contexte temps : tel que l'heure de la journée, la semaine, le mois, et la saison de l'année.

N'étant pas satisfait par ces définitions, en les trouvant trop spécifiques, Dey [Dey01] propose une nouvelle définition plus formelle et dans laquelle le contexte est :

> « *Toute information qui peut être utilisée pour caractériser la situation d'une entité. Une entité est une personne, place, ou objet qu'on considère approprié à l'interaction entre un utilisateur et une application, y compris l'utilisateur et les applications elles-mêmes* »

Chen et Kotz [CK01] quant à eux exposent une différence qui jusqu'ici n'a été illustrée dans aucune des définitions précédentes. C'est la spécification de deux aspects de contexte. L'un, dit actif, comporte les caractéristiques de l'environnement d'alentour et détermine le comportement de l'application. Et l'autre, appelé Passif, est relatif à l'application mais non nécessaire pour l'adaptation. Décider de la classe d'un contexte précis (Active ou Passive) dépend de la manière dont il est utilisé. Par exemple, la localisation de l'utilisateur peut être utilisée d'une façon active dans un système [WHF92], et d'une façon passive dans un autre [Wei93].

Pour formuler cette différence, la définition suivante a été donnée :

> « *Le contexte est l'ensemble d'états et d'arrangements environnementaux qui soit détermine le comportement d'une application soit dans lequel un évènement d'application se produit et est intéressant à l'utilisateur* »

Finalement, on peut dire que la plupart des définitions formelles qui ont été proposées restent ouvertes. Cependant, dans la pratique un ensemble non exhaustif d'informations de contexte est étudié tel que : la bande passante, la localisation, l'énergie, les ressources proches, …etc.

II.2.3 Le context-aware computing

Le calcul context-aware a été défini pour la première fois par Schilit et Theimer [ST94] : « un logiciel qui s'adapte selon l'endroit de son utilisation, l'ensemble d'objets et personnes proches, et les changements de ces objets à travers le temps ». Depuis cette première interprétation, plusieurs définitions et catégorisations des applications context-aware ont été données. Parmi lesquelles :

i) La définition de Schilit et al. [SAW94] qui décrit les applications context-aware comme étant des applications qui *adaptent dynamiquement leurs comportements selon le contexte utilisateur et le contexte de l'application*. Ils catégorisent les applications context-aware selon que la tâche soit une collecte d'informations ou l'exécution d'une commande, et selon qu'elle soit exécutée manuellement ou automatiquement :

- Les applications « *proximate selection* » : Ce sont les applications qui cherchent les informations manuellement pour les utilisateurs, c'est une technique d'interaction où une liste des objets et places proches est présentée.
- Les applications de reconfiguration contextuelle automatique « *Automatic contextual reconfiguration* » : c'est un processus d'ajout de nouveaux composants, de suppression de certains existants, ou de modification des liens entre les composants selon le contexte.
- Les applications de commandes contextuelles « *contextual command applications* » : sont des applications qui exécutent manuellement des commandes pour l'utilisateur, en se basant sur le contexte disponible. Ce sont des services exécutables qui soit sont rendus disponibles selon le contexte, soit leur exécution est modifiée selon le contexte.
- Les applications qui exécutent des commandes pour l'utilisateur de façon automatique en se basant sur des actions qui se déclenchent selon le contexte « *Context-triggered actions* », ce sont des services qui sont exécutés automatiquement quand la bonne combinaison du contexte existe, ils sont basés sur de simples règles de si-alors.

ii) Plus récemment Pascoe et al. [Pas99][PRM98][RPM97] ont défini le « context-aware computing » comme étant « *la capacité des équipements de calcul à détecter et sentir, interpréter et répondre aux aspects de l'environnement local d'un utilisateur et des équipements même ».* Pascoe dans son travail [Pas99], propose une taxonomie des caractéristiques de base du context-awareness. La première caractéristique étant *la collecte contextuelle* qui est la capacité de détecter les informations contextuelles et les présenter au système de détection de l'utilisateur. La deuxième c'est l'adaptation contextuelle, qui consiste en l'habilité d'exécuter ou de modifier automatiquement un service en se basant sur le contexte courant. La troisième caractéristique est la découverte de ressources, elle permet aux applications context-aware de localiser et d'exploiter les ressources et services relatifs au contexte d'utilisation. Quatrième et dernière caractéristique est l'augmentation contextuelle. Cette caractéristique est la capacité d'associer des données digitales avec le contexte utilisateur.

- Dey [Dey01] a essayé de donner une définition plus générale du context-aware computing : « *Un système est context-aware s'il emploie le contexte pour fournir l'information et/ou des services appropriés à l'utilisateur, et dont la pertinence dépend de la tâche de l'utilisateur* ». Dey a combiné les deux taxonomies précédentes et les a résumées en trois catégories de caractéristiques que les applications context-aware peuvent supporter : présentation des informations et services à l'utilisateur, une exécution automatique de services, et l'étiquetage du contexte pour la récupération postérieure.

iii) Dans le travail de Chen et Kotz [CK01], deux définitions du context-awareness ont été données selon que le contexte soit actif ou passif :

- Un context-awareness actif : c'est le cas où une applications s'adapte automatiquement au contexte détecté, en modifiant son comportement

- Un context-awareness passif : dans ce cas, l'application soit présente le nouveau contexte (ou sa mise à jour) à l'utilisateur intéressé soit le rend persistant pour une utilisation ultérieure.

De l'étude de ces différentes définitions et taxonomies, nous pouvons remarquer que les deux premières définitions se croisent sur certains points qui sont : la détection contextuelle de la taxonomie de Pascoe qui correspond à la sélection approximative de Schilit. De même pour l'adaptation contextuelle qui correspond aux actions context-triggered et la découverte de service qui concorde avec la reconfiguration contextuelle automatique. Donc, on peut dire que les deux catégorisations exposent la capacité d'exploiter les ressources relatives au contexte utilisateur, d'exécuter une commande automatiquement en se basant sur le contexte et d'afficher les informations relatives à l'utilisateur.

Nous pouvons également remarquer que la définition de Dey [Dey01] est une définition plus générale et plus ouverte que les deux premières : elle n'exige pas que le comportement de l'application soit modifié pour qu'elle soit context-aware. Par exemple une application qui affiche le contexte de l'environnement utilisateur sans changer son comportement est considérée context-aware. De plus et contrairement à la définition de Pascoe et al. [Pas99][PRM98][RPM97] qui exige que les systèmes context-aware puissent détecter, interpréter et répondre au contexte, la définition de Dey exige seulement que le système réponde aux changements dans le contexte, permettant ainsi que la détection et l'interprétation soient faites par d'autres entités de calcul.

II.3 Taxonomie des différents travaux de recherche

L'adaptation et le context-aware computing sont deux termes très liés que, depuis l'émergence des réseaux mobiles, ont été utilisés de différentes façons et à chaque fois pour répondre à un besoin spécifique :

II.3.1 L'adaptation un besoin urgent :

Au début, l'adaptation a été vue comme un besoin urgent pour permettre aux applications déjà existantes de continuer à travailler dans le nouvel environnement. L'objectif de cette adaptation est de garantir le bon fonctionnement de l'application et donc un certain niveau de QdS (Qualité de Service). Pour répondre à ce besoin, plusieurs solutions qui placent la responsabilité entière de l'adaptation sur le système et qui garantissent un masquage complet de la mobilité et des différents problèmes engendrés aux applications ont été proposées. Dans ces solutions c'est au système d'être sensible (aware) des variations de contexte. Notons que le contexte dans ce cas représente un ensemble restreint de variables liées essentiellement aux trois contraintes du réseau mobile (les variations dans la bande passante, l'hétérogénéité des terminaux, etc.)

Une solution générale dans cette approche consiste en l'utilisation d'un Proxy local sur l'hôte mobile pour cacher les problèmes posés par la mobilité :

II.3.1.1 Un Proxy pour les systèmes de fichiers

Ce genre de Proxy a pour but de garantir le transfert de fichiers entre le serveur de fichiers et l'hôte mobile.

Le système de fichier CODA, le système le plus utilisé dans cette approche, utilise un Proxy de système de fichiers qui permet aux applications de travailler sans aucune modification. Il fournit aux clients, une grande disponibilité d'accès aux informations même dans le cas déconnecté.

Fonctionnement :

Les applications s'exécutant sur CODA utilisent l'interface standard du système UNIX. A chaque client est associé un gestionnaire de cache local appelé venus, qui enregistre et cherche dynamiquement les données des différents volumes [Kis92].

Deux mécanismes sont utilisés pour assurer la disponibilité des données. Le premier est la réplication, qui permet à un volume d'avoir des copies sur plus d'un serveur, définissant ainsi un VSG (Volume Storage Group) pour chaque volume donné. Le sous-ensemble du VSG nommé AVSG est défini pour chaque client comme l'ensemble des VSG accessibles. Les modifications du client sont propagées à tous les sites AVSG et éventuellement aux sites VSG manquants. Le deuxième mécanisme est la gestion des opérations en mode déconnecté.

La gestion des déconnexions peut être schématisée par la figure suivante (Figure2.1.):

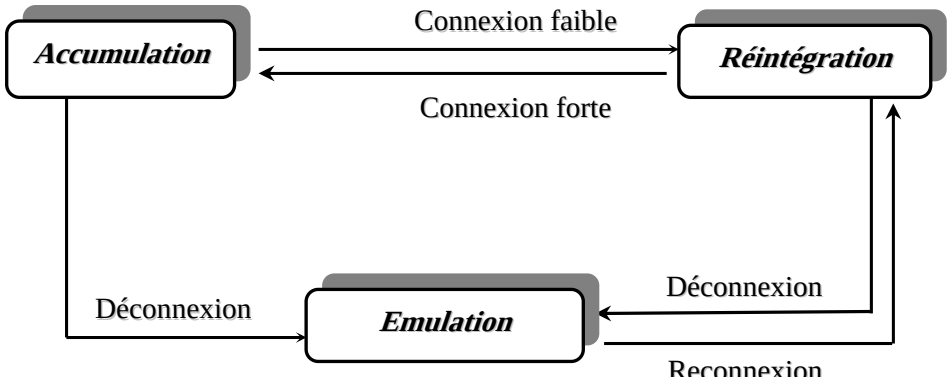

Figure2.1. *Mode de fonctionnement d'un client CODA*

Pendant l'étape d'accumulation, le client CODA travaille dans un mode connecté assimilé à une connexion LAN. Dans ce cas *venus (le gestionnaire du cache local)* est dit pleinement connecté. Si une application demande un fichier, *venus* vérifie si le fichier est dans le cache ou bien le cherche à partir du serveur [BNS95]. Une grande responsabilité est affectée à *venus* qui est chargé d'assurer que les objets critiques susceptibles d'être utilisés à l'avenir seront dans le cache au moment de la déconnexion donc une collecte de fichiers est faite dans cette étape. *Venus* doit gérer son cache de façon à effectuer un équilibre entre les besoins des applications dans les deux modes connecté et déconnecté, exemple : un utilisateur peut désigner un ensemble de fichiers qui va être probablement utilisé dans l'avenir en travaillant sur un autre ensemble de fichiers. *Venus* utilise une réplication optimiste, il refait automatiquement une opération d'accumulation chaque 10 minutes, mais cette opération peut être faite à la suite d'une demande de l'utilisateur dans le cas d'une déconnexion volontaire.

Au moment de la déconnexion le client CODA se trouve dans l'étape de l'émulation et y reste pendant toute la période de déconnexion. Quand il y a une demande d'un fichier, *venus* vérifie s'il est dans le cache sinon il retourne un état d'erreur. Les opérations s'effectuent sur les fichiers qui ont été mis dans le cache au moment de l'accumulation. Les mises à jour sont stockées par *venus* par des méthodes d'optimisation pour réduire la consommation des ressources. Exemple, si un fichier est créé, nommé et plus tard supprimé *venus* ne garde pas trace de ces opérations au moment de la réintégration.

L'étape de réintégration est un état intermédiaire qui se produit au moment de la reconnexion. *Venus* réintègre les modifications faites dans la période de l'émulation, et met à jour son cache pour refléter un état récent du serveur.

Le Proxy du système de fichiers de Coda fournit également un mécanisme pour la gestion de conflits entre les transactions concurrentes.

II.3.1.2 Web Proxy

C'est un autre type de Proxy qui permet aux applications de navigation sur le web de fonctionner dans les réseaux sans fil sans imposer aucun changement pour les navigateurs ou les serveurs. Le web Proxy peut être utilisé pour télécharger et stocker, compresser et transférer les pages web sur les clients mobiles, et supporter ainsi les opérations de navigations asynchrones et en mode déconnecté.

WebExpress [Moh02] est une des solutions proposées dans cette approche, il utilise un modèle pour intercepter et contrôler les communications sur les liens sans fil dans le but de réduire le volume du trafic et d'optimiser les protocoles de communication. Ce modèle se base sur deux composants : le processus d'interception côté client (CSI) qui s'exécute sur le client mobile et le processus d'interception côté serveur (SSI) qui s'exécute sur le réseau fixe (Figure2.2.).

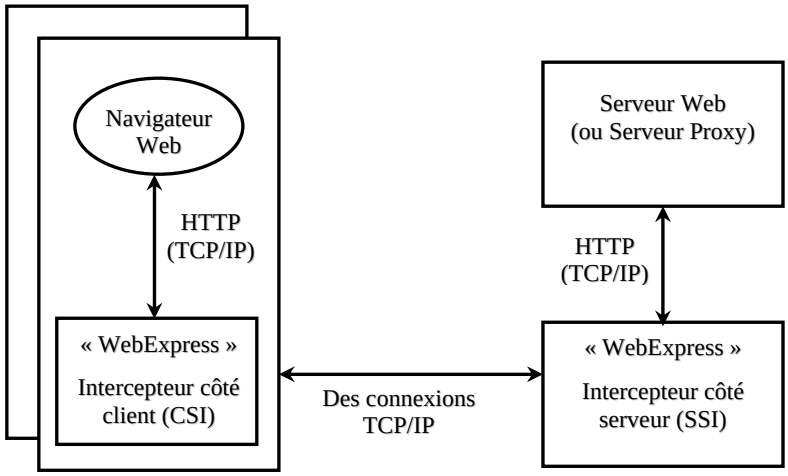

Figure2.2. *Le modèle d'interception dans le WebExpress*

Le CSI intercepte les requêtes HTTP venant du Web Browser, leur applique des optimisations pour réduire la consommation de la bande passante et les délais de transmission sur le lien sans fil, et les envoie au SSI à travers une connexion TCP en utilisant une version réduite du protocole http. SSI de son côté en recevant une requête, reconstitue les données et les envoie à son serveur Web. Mettant en œuvre ainsi un modèle d'interception indépendant du navigateur et du serveur.

WebExpress offre quatre méthodes d'optimisation :

1. L'utilisation de cache : Le SSI et le CSI stockent des objets graphiques et HTML à leurs niveaux. Lorsque le CSI reçoit une URL, il vérifie si l'objet demandé se trouve dans son cache. Si c'est le cas, il envoie la réponse immédiatement, sinon l'URL est envoyée au SSI. Ce dernier vérifie au niveau de son cache avant d'envoyer la requête au serveur.

2. Le concept de 'Differencing' : Ce concept consiste en le stockage d'un objet de base au niveau du CSI et SSI. Ensuite chaque fois que le CSI reçoit une requête, il calcule la différence entre l'objet de base et la requête reçue et puis envoie cette différence au SSI. Ce dernier fusionne cette différence avec son format de base pour retrouver la requête de base. (Le même principe est utilisé lors d'une réponse)
3. Réduction du protocole : pour réduire le coût élevé des communications, causé par les connexions répétitives, WebEpress propose de multiplexer entre les requêtes et les réponses. [Wan01]
4. Réduction des en-têtes : Pour réduire le transfert des en-têtes, le CSI lorsqu'il établit une connexion avec le SSI, envoie les capacité du navigateur sur la première requête. Le SSI de son côté maintient ses informations pour toute la durée de la connexion.

Cette approche répond bien au besoin initial qui est : étendre les solutions déjà existantes. Cependant gérer toute la complexité du système indépendamment des applications mène à des middlewares très complexes qui ne peuvent tourner de façon efficace sur des terminaux mobiles. D'autre part, quelques applications peuvent bénéficier considérablement de savoir ce qui se produit dans leurs environnements pour améliorer la qualité de service. Par exemple, une application de vidéoconférence peut obtenir des améliorations considérables de sa qualité de service en choisissant un protocole de transport de réseau qui convient à l'infrastructure (LAN, WLAN, … etc.) et à la largeur de bande disponible.

II.3.2 L'adaptation Application-aware

Avec le temps, la nature très dynamique de changement conjuguée aux différents problèmes de la première solution, a fait ressentir le besoin d'impliquer l'application dans le processus d'adaptation. Ce qui a donné naissance à une nouvelle approche (Application-aware adaptation) et donc un nouveau type d'applications. Des applications qui doivent tenir compte des limitations engendrées par la contrainte de portabilité en étant plus flexibles et en modifiant leurs dépendances aux ressources selon leurs disponibilités, et qui ajustent leurs comportements de façon dynamique selon leurs localisations et les ressources ou objets disponibles dans leurs voisinages. Par exemple, une application de vidéoconférence qui détecte la présence d'un afficheur plus grand dans le voisinage peut basculer l'affichage de la vidéo sur ce dernier, améliorant ainsi la qualité du service. Et des applications qui détectent les changements dans la bande passante et adapter leurs modes de fonctionnement en conséquence. Par exemple pour couvrir la grande variabilité de la bande passante l'application peut adapter son comportement selon la largeur de bande disponible en diminuant ou en augmentant le transfert de données sur les liens sans fil. Nous parlons ainsi des applications context-aware. Plusieurs solutions ont été proposées dans ce sens, dans lesquelles l'application et le système collaborent pour faire face aux différents problèmes et garantir un certain niveau de QdS.

II.3.2.1 Un middleware réflectif pour des applications adaptables

Ce middleware [CEM01], a été proposé dans le but de fournir aux applications un accès dynamique à leurs contextes d'exécution. Pour la conception de ce middleware deux principes ont été utilisés : la réflexion et les méta données :

1/ Le principe de réflexion[2] est un des principes largement utilisé dans le développement des middlewares context-aware. Un système réflectif peut apporter des modifications à son comportement au moyen de deux opérations qui sont l'inspection et l'adaptation. A travers l'inspection, le comportement interne du système est exposé et via l'adaptation, ce comportement interne peut être modifié dynamiquement en modifiant les caractéristiques existantes ou en ajoutant d'autres. Ceci signifie que le noyau du middleware avec un ensemble minimal de fonctionnalités peut être installé sur le terminal mobile et c'est à l'application par la suite de gérer et d'adapter le comportement du middleware selon ses besoins.

Ce qui implique que le noyau du Middleware seulement avec un minimum de fonctionnalités est installé et c'est à l'application de gérer et d'adapter le comportement du middleware selon ses besoins.

L'information de contexte peut être gardée par le Middleware dans ses structures de données internes et, par des mécanismes de réflexion. Les applications peuvent acquérir des informations sur leur contexte d'exécution et accorder le comportement du middleware en conséquence. Aucun paradigme spécifique de communication n'est lié au principe de la réflexion, et est laissé sans précision et dépend du système Middleware établi.

2/ une structure de méta données a été utilisée pour faire la différence entre ce que le middleware fait et comment il le fait.

II.3.2.1.1 Les principes de ce middleware réflectif :

Cette solution définit deux niveaux de conscience, device-awareness et environment-awareness. Alors que le premier concerne les ressources internes, et fait référence à tout ce qui réside dans le dispositif d'exécution de l'application (mémoire, batterie, etc.), le dernier se réfère aux ressources externes tel que la bande passante, la localisation, les services, etc.

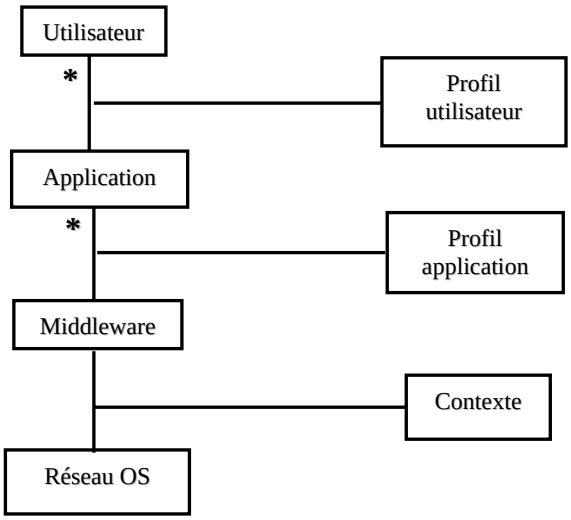

Figure2.3. *Les profils applications et utilisateurs*

Le système prend en considération qu'on peut avoir plusieurs applications fonctionnant sur le même middleware, et plusieurs utilisateurs utilisant la même application

[2] Le principe de réflexion a été initialement introduit par Smith en 1982 [Smi82] comme le principe qui permet à un programme d'accéder, traiter et altérer sa propre interprétation.

(Figure2.3.). Chaque utilisateur peut adapter l'application de différentes façons selon ses besoins; les utilisateurs peuvent, par exemple, adapter la barre de tâche de l'interface d'application en utilisant quelques icônes au lieu d'autres, ou demander à l'application de se déconnecter automatiquement quand la batterie s'affaiblie, etc.

Pour cela, l'utilisateur doit installer un profil utilisateur qui informe l'application sur la façon dont elle doit se comporter dans différentes circonstances. Du côté application, deux types de données sont définies, les données simples 'data' relatives aux conditions fonctionnelles de l'application et les métas données qui représentent le profil utilisateur.

L'application filtre les tâches qu'elle peut faire de façon indépendante (comme par exemple la disposition de la barre des tâches), et translate le reste en un profil application qui est ensuite transféré au middleware.

Au niveau du middleware, les informations de contexte telles que la valeur de la bande passante, l'état de la batterie, etc. représentent les données internes, alors que le profil application représente les 'méta données'. A partir de cela, c'est le middleware qui a la charge de maintenir une représentation valide du contexte; et à chaque fois qu'un changement dans le contexte d'exécution est détecté, il consulte ses méta données pour voir comment l'application veut s'adapter dans une telle configuration.

Il est à noter que l'application peut accéder dynamiquement à son profil et le modifier en utilisant le principe de réflexion.

Toutes ces étapes sont résumées dans la figure suivante (Figure2.4.):

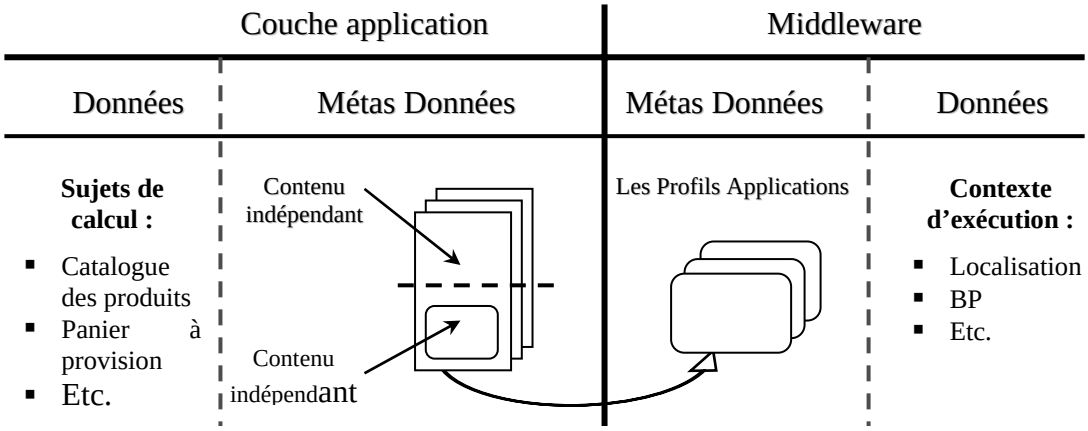

Figure2.4. *La notion de données et de métas données au niveau de l'application et du middleware*

II.3.2.1.2 Le modèle conceptuel

Pour que le middleware puisse manipuler les métas données, écrites par le concepteur de l'application, il faut qu'il y ait un accord entre les deux parties sur la représentation du profil.

Comme solution à ce problème, le middleware présenté définit une grammaire en un schéma XML qui doit être suivie lors de l'écriture du profil. Par la suite et pendant la génération du profil, le concepteur crée une instance de cette grammaire, et toute modification ultérieure doit se faire en respectant les règles de la grammaire.

Pour éclaircir le choix de l'information à coder, deux manières différentes que l'application peut utiliser pour influencer sur le comportement du logiciel personnalisé sont définies :

1. **Un changement dans le contexte d'exécution :** l'application peut demander au middleware de détecter les changements de contexte et de réagir en conséquence indépendamment de la tâche courante de l'application, par exemple se déconnecter en cas de faiblesse de la bande passante. Pour cela, une association est faite entre les différentes configurations de contexte (ces configurations dépendent des valeurs des ressources surveillées), et les politiques qui doivent être appliquées. (la partie droite de la figure Figure2.5.)
2. **Demande de service :** un service demandé peut généralement être fourni de différentes manière, par exemple un service d'accès aux données peut être fourni de deux façons : soit télécharger une copie de la donnée localement, soit créer un lien à la copie principale. Le choix de la manière dépend des ressources disponibles, c'est pourquoi le profil application doit spécifier pour tous les services que l'application aura à utiliser les politiques possibles ainsi que les conditions nécessaires pour l'exécution de chacune d'elle (la partie gauche de la figure Figure2.5.).

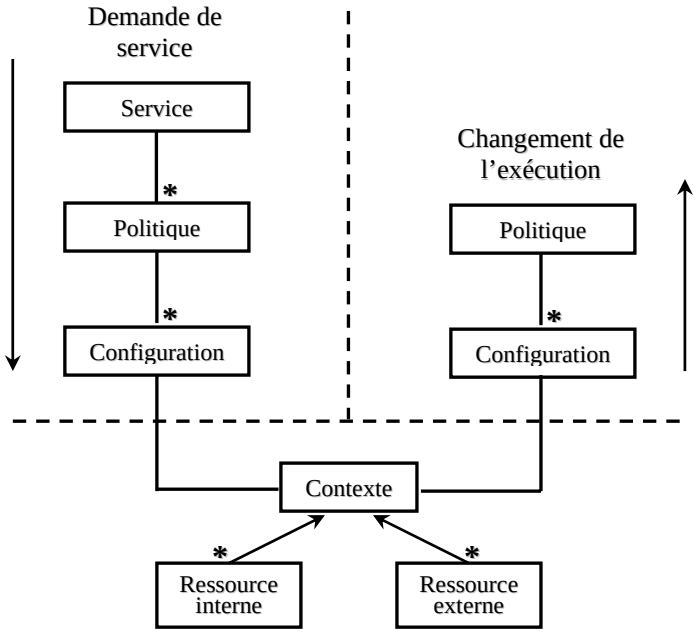

Figure2.5. *Le profil application*

II.3.2.1.3 L'architecture du système

Comme le montre la figure (Figure2.6.), le middleware conserve ses données et méta données dans des structures de données internes. Les données simples font référence à une présentation unique du contexte d'exécution; on trouve à ce niveau les valeurs réelles de toutes les ressources internes et externes que le middleware peut surveiller (par exemple, mémoire, puissance de batterie, largeur de bande, raccordement de réseau, centres serveurs proches, … etc.). Les métas données incluent tous les profils des applications qui s'exécutent sur le middleware.

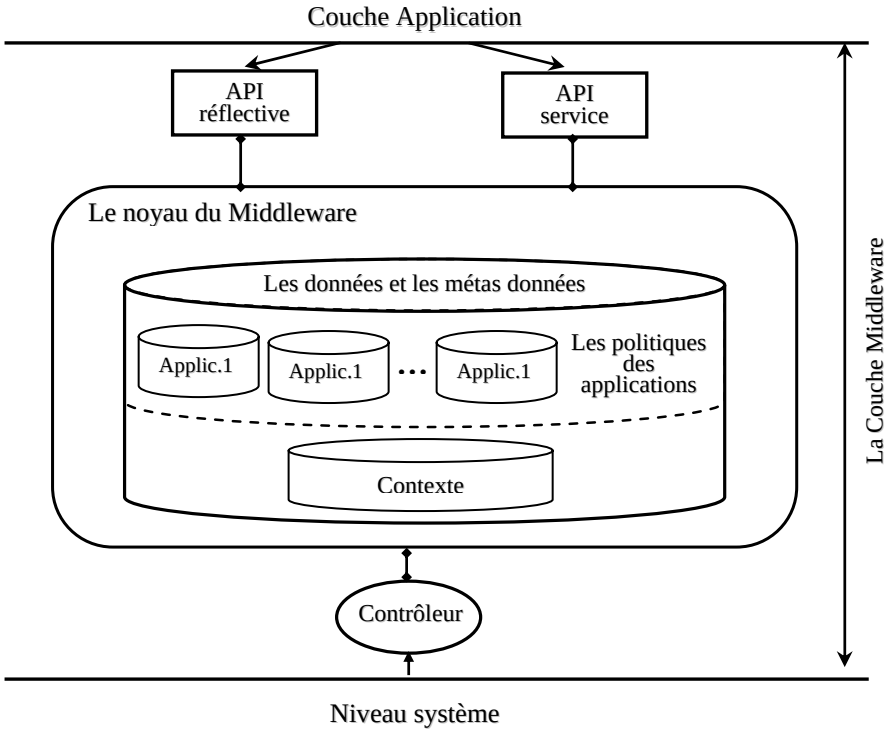

Figure2.6. *Architecture du middleware*

Les données peuvent être lues et modifiées par le composant de contrôle 'Contrôleur'. Sa tâche est d'interagir avec le système afin de capter les mises à jour de la configuration du contexte. A chaque fois qu'un changement de contexte est détecté, le NOYAU du middleware cherche dans ses métas données si une application parmi celles courantes a enregistré un intérêt pour les ressources changées, et déclenche les actions correspondantes.

Quand aux métas données, elles peuvent être créées, lues, et modifiées par la couche application en utilisant une API Réflective. Les profils applications sont conservés dans des documents XML (sous forme d'arbre) ; l'information peut alors être manipulée par une API DOM (Document Object Model) [ABC98], qui fournit des primitives pour consulter, ajouter et supprimer des noeuds d'un arbre XML, par la technologie de XPath [CD99], qui permet l'adressage de points spécifiques d'un document XML. Tous les profils sont stockés de manière permanente par le middleware, mais l'activation ne se fait que pour les profils des applications en cours. Quand un profil est transféré pour la première fois au middleware, et toutes les fois qu'une mise à jour est apportée, le NOYAU est responsable de vérifier l'uniformité d'information qu'elle contient, avant de la stocker de manière permanente. À cette fin, le NOYAU exécute un analyseur qui traite le document et vérifie si c'est un exemple valide de la grammaire XML fournie par le middleware. En cas de contradiction, la mise à jour échoue et un message d'erreur est envoyé à l'application.

OpenCorba [Led99], dynamicTAO [KRL00] et OpenORB [BCA01] sont d'autres exemples de middleware réflectifs qui peuvent être utilisés pour garantir une adaptabilité dynamique.

II.3.2.2 Odyssey

Odyssey est un middleware de partage de données avec des politiques de synchronisation dépendantes de l'application. Cette plate-forme est un autre système context-aware qui gère la disponibilité des ressources et intègre l'adaptation au niveau applicatif, permettant de prendre en compte des informations liées à l'environnement. Odyssey surveille les ressources, détecte les changements, envoie des notifications aux applications intéressées et permet ainsi d'opérer une adaptation de leur comportement d'exécution.

II.3.2.2.1 La fidélité

Face à une rareté de ressources, le client mobile peut agir de deux façons : soit réduire sa demande pour une ressource critique ou bien dégrader la qualité des données accédées en minimisant la consommation de la ressource [NS99]. Pour quantifier cette notion de qualité une nouvelle propriété a été définie : 'la fidélité'.

La fidélité est définie par le degré auquel une copie de données présentée s'accorde avec la copie de référence (donnée originale). La *fidélité* a plusieurs dimensions qui sont utilisées comme des axes naturels pour l'adaptation. La consistance est une des dimensions les plus connues. D'autres dimensions dépendent du type de données à manipuler, exemple : les données vidéo ont deux dimensions additionnelles qui sont le nombre d'images par seconde et la qualité de l'image [NSP95]. Donc l'adaptation agit sur les données considérées (dégradation, distillation des images, réduction du nombre d'images par seconde etc.).

II.3.2.2.2 Environnement d'une application

Odyssey définit l'environnement d'une application par l'ensemble des ressources disponibles pour cette application. Ces ressources peuvent être soit génériques tels que la largeur de bande du réseau, la puissance de la batterie et l'espace disque disponible. Soit spécifiques d'un type particulier de données, comme par exemple dans le cas d'un service de vente sur le web qui permet à un client de faire un certain nombre de demandes par jour, ce nombre de demander est une ressource spécifique [NSP95]. ODYSSEY gère la disponibilité de ces ressources et leurs variations : la disponibilité d'une ressource est mesurée par une valeur unique par exemple, la largeur de la bande est mesurée en bit/seconde, l'espace disque en Kilo-octet etc.

II.3.2.2.3 Architecture et fonctionnement d'ODYSSEY

ODYSSEY peut être vu comme un ensemble d'extensions d'un système d'exploitation supportant des applications adaptatives. Il a été implanté comme un composant noyau dans l'espace utilisateur, mais peut être aussi implanté directement dans le noyau du système ou bien comme une couche de middleware [Nob00].

L'architecture d'un client ODYSSEY est schématisée par la figure suivante :

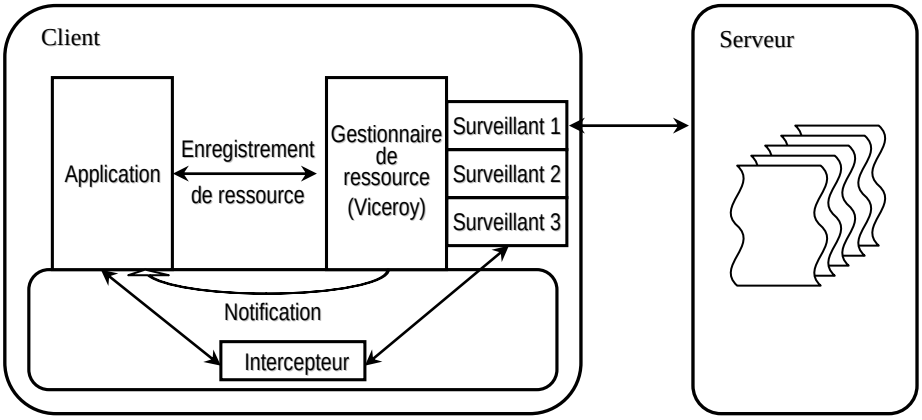

Figure2.7. *L'architecture d'un client ODYSSEY*

ODYSSEY fournit un accès aux objets qu'il gère à travers le système de fichier du client. Le module intercepteur dans le noyau redirige les opérations sur ces objets vers la plate-forme ODYSSEY, qui contient deux composants : le gestionnaire de ressources « viceroy » et un ensemble de surveillants « wardens ».

Le viceroy est responsable de toutes les tâches indépendantes du type au niveau du client. La plus importante est la gestion de l'utilisation des ressources et la notification des applications sur les changements significatifs concernant la disponibilité des ressources. Il agit comme un point de contrôle de ressources, donnant ainsi un support aux applications concurrentes.

Les applications doivent être capables de spécifier quels sont les changements dans la disponibilité des ressources qui sont importants ainsi que les seuils à partir desquels elles doivent être notifiées, par exemple dans le cas de la bande passante, l'application définit une borne inférieure et une autre supérieure selon lesquelles elle dégrade ou améliore la fidélité. Ensemble ces deux limites forment une fenêtre de tolérance de la bande passante. L'application envoie au système une requête contenant les fenêtres de tolérance de chaque ressource ainsi que les fonctions à appeler en cas de viol de ces fenêtres.

Plus tard quand le viceroy met à jour les estimations de la disponibilité de la ressource. Il notifie l'application, via un upcall, de chaque estimation liée à ses fenêtres de tolérance déclarées, et c'est à l'application de réagir par la suite suivant cette nouvelle valeur.

Les wardens sont responsables de l'implémentation des méthodes d'accès aux objets, pour changer la fidélité des données cachées et présentées à l'utilisateur. Ils gèrent toutes les communications entre le client et les serveurs, et offrent un menu de fidélité pour les applications.

II.3.2.3 MoLèNE (MObiLE Networking Environment)

Le projet MoLèNE [AS99][SA00] a pour but de construire un système générique et personnalisable qui peut dynamiquement s'adapter aux changements des conditions d'exécution pour faciliter l'adaptation des applications existantes et la construction de nouvelles applications. Pour cela, MoLèNe fournit des services qui permettent plusieurs types d'adaptation selon les besoins des applications qui les utilisent.

II.3.2.3.1 Les caractéristiques de MoLèNE

Le système MoLèNE est caractérisé par :

- **Une nature générique :** la conception de MoLèNE permet aux programmeurs de le personnaliser à leurs applications particulières.
- **Adaptabilité aux conditions d'exécutions de l'environnement :** les composants constituants le système MoLèNE sont capables de modifier dynamiquement leur comportement en tenant compte du contexte d'exécution et des exigences des applications.
- **Flexibilité et extensibilité :** MoLèNE n'a pas été conçu comme *un système fermé* mais comme un système flexible, il permet aux applications d'injecter de nouveaux composants dans le système fournissant ainsi des moyens d'extension du système.

II.3.2.3.1.1 La nature générique de MoLèNE

MoLèNE est un prototype orienté objets utilisé pour la construction de middlewares. Il offre un ensemble de services sous forme d'une généralisation des concepts souvent utilisés par les approches existantes dans la gestion des données dans les environnements mobiles.

Les programmeurs ont besoin juste d'implémenter les parties du framework qui dépendent de l'application au lieu de concevoir toute l'application.

II.3.2.3.1.2 L'adaptation dans MoLèNE

Dans le but de garantir une adaptation en fonction des changements de l'environnement, des méthodes qui assurent la gestion de l'environnement et l'adaptation sont fournies dans MoLène comme des outils. MoLèNE est un framework orienté objet qui fournit des moyens de changement dynamique de l'implémentation d'un composant (plus petite unité logicielle dans MoLèNE). Chaque service est représenté par un objet service dont le rôle est divisé en des micro services encapsulés dans des objets micro services. Plus tard un micro service est obtenu par un objet implémentation. L'adaptation dans MoLèNE est assurée par le changement de la structure d'un service ou bien le changement de l'implémentation et/ou la localisation d'un micro service. Les décisions d'adaptation sont prises selon les besoins de toutes les applications et non pas d'une application particulière.

Dans le but de faciliter l'adaptation, trois types d'objets ont été identifiés pour implémenter les services adaptatifs (Figure2.8.) :

L'entité *interaction* reçoit les éléments (par exemple, un ensemble de composants des applications qui ont besoin d'être déplacés) et les transmet à l'objet *implémentation* qui représente le code assurant la fonctionnalité d'un composant. Le *contrôleur* pourra donc décider d'appliquer ou non une autre stratégie. La décision est prise en tenant compte des conditions d'exécution et de l'algorithme d'adaptation. Dans le cas où une décision de changement est prise, l'implémentation courante est remplacée par un autre code.

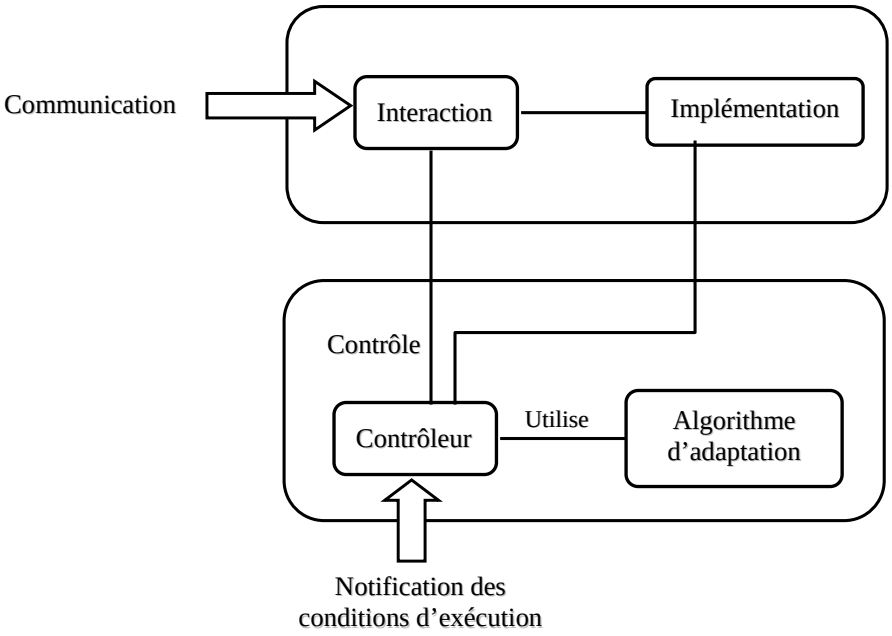

Figure2.8. *Infrastructure d'un composant MoLèNe*

II.3.2.3.2 L'architecture de MoLèNE

Dans le but de maximiser l'utilisation des composants et de faciliter l'adaptation MoLèNE est divisé en deux niveaux présentés dans la figure suivante (Figure2.9.):

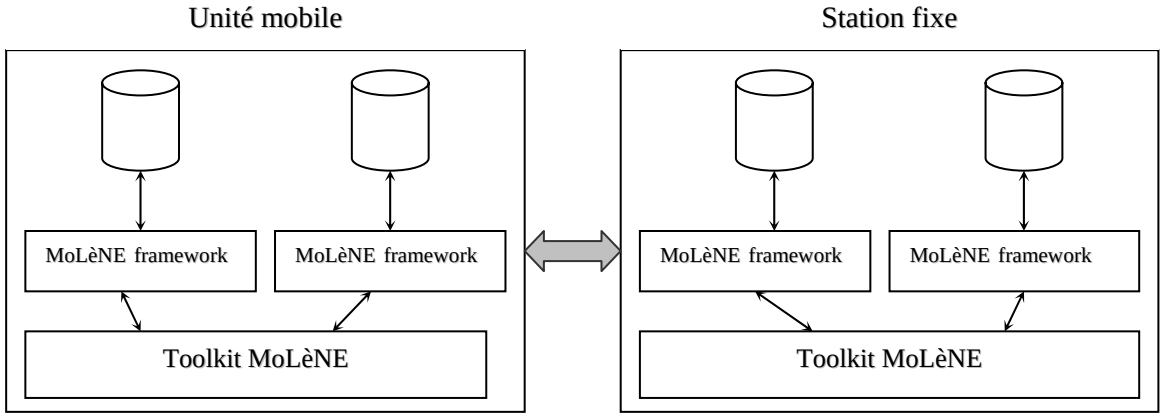

Figure2.9. *Architecture générale de MoLèNE*

- **Le *toolkit :*** il offre des fonctionnalités, indépendantes des applications, nommés *outils.* Ces derniers n'ont pas besoin d'être personnalisés pour une application particulière, pour l'architecture des supports (système d'exploitation, unités extérieures) ou bien pour l'environnement (localisation de l'utilisateur). Ils sont exécutés au niveau de la machine et sont partagés par toutes les applications.
- **Le *framework :*** il offre des fonctionnalités qui dépendent des applications nommées, *services.* Ils sont personnalisables pour une application particulière et son environnement.

Ces services sont partagés par des applications utilisant le middleware MoLèNE et vont être utilisés pour le développement des nouvelles instances du framework

Les outils et les services de MoLèNE sont schématisés par la figure (Figure2.10):

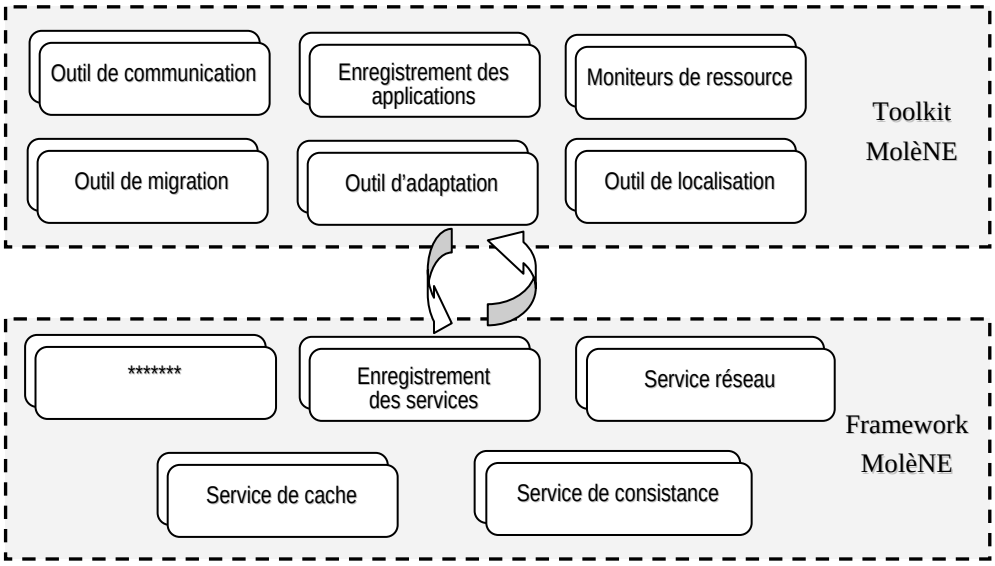

Figure2.10. *Les services et outils de MoLèNE*

II.3.2.4 L'adaptation par la distribution adaptative des applications (AeDEn)

La plupart des approches dans le domaine de la mobilité utilisent des ressources extérieures au mobile soit pour assurer du stockage d'informations, soit pour effectuer des traitements ou encore pour effectuer de la recherche d'informations. Cette approche consiste à généraliser cette utilisation des ressources extérieures au moyen de la distribution. Elle se base donc sur le fait que l'unité mobile (i) connaisse les différentes ressources et services de son environnement, (ii) distribue ensuite ses applications selon leurs besoins et de manière à profiter de cet environnement, (iii) et enfin réagisse dynamiquement en adaptant la distribution et les politiques de distribution selon les changements de l'environnement.

II.3.2.4.1 La distribution : un moyen d'améliorer la qualité de service rendu à l'utilisateur :

La distribution permet de surmonter les inconvénients liés à la mobilité en améliorant de manière globale la qualité de service en terme de [MA01]:

- *Qualité d'interactivité utilisateur :* la qualité de l'interactivité utilisateur se mesure au moyen du temps de réponse du système ou des applications. Ce temps de réponse dépend des contraintes physiques inhérentes au mobile : la mémoire, le processeur, la taille disque, et la bande passante réseau. La distribution permet de mettre sur le mobile les composants de l'application en interaction avec l'utilisateur et les autres composants sont exécutés dans l'environnement. On obtient donc (i) une diminution de l'utilisation des ressources et des services du mobile, (ii) une meilleure utilisation des ressources et des services de l'environnement permettant d'optimiser les traitements, (iii) un moyen de surmonter les connexions/déconnexions puisque les traitements peuvent être effectués hors-ligne, les résultats étant envoyés lors de la reconnexion.

- *Qualité de données :* plusieurs éléments constituent les données : le texte, les images, les sons, la vidéo, etc. Ces données sont souvent amenées à subir des transformations pour des raisons d'économie de bande passante ou d'adaptation à l'affichage du mobile. En utilisant les ressources et services de l'environnement, les transformations peuvent être d'une meilleure qualité tout en satisfaisant le même temps de réponse, ou alors elles peuvent être effectuées plus rapidement tout en satisfaisant la même qualité.

- *Accessibilité :* la distribution permet d'utiliser les différentes ressources et services de l'environnement de manière transparente pour l'application par un placement approprié de ses composants. Ce placement change dynamiquement, par exemple, lorsque des ressources et services apparaissent et disparaissent dans l'environnement.

Dans le but d'atteindre ces buts, le mécanisme de distribution est conçu pour être personnalisable aux différentes applications ayant différents besoins et contraintes. Ceci est obtenu en exploitant les aspects génériques de MoLèNE (un framework orienté objet) ; le mécanisme de distribution doit être dynamique et ceci à deux niveaux :

- Une notification des changements de l'environnement, une stratégie donnée peut réagir en changeant la distribution courante.

- Le mécanisme de distribution lui même doit être capable de changer radicalement sa stratégie.

II.3.2.4.2 L'environnement de distribution AeDEn (Adaptive Distribution Environment):

Construire un algorithme prenant en compte tous les changements qui pourraient intervenir dans un environnement mobile paraît irréaliste et peu extensible. C'est pourquoi, pas un algorithme de distribution mais un cadre général de distribution, appelé AeDEn [Mou03], a été proposé dans lequel les concepteurs d'application peuvent utiliser les politiques de distribution existantes ou insérer leurs propres politiques selon les besoins de leurs applications. En outre, le changement dynamique de politique est possible de manière à tenir compte des changements intervenant dans l'environnement.

Traditionnellement un algorithme de distribution se décompose en trois fonctionnalités :(i) une politique d'information qui définit la nature des informations collectées et la manière dont ces informations sont collectées et mises à jour, (ii) une politique d'éligibilité qui utilise ces informations pour élire les composants qui doivent être placés, migrés ou suspendus et (iii) une politique de placement qui détermine la station sur laquelle les composants élus doivent être transférés.

II.3.2.4.3 Services d'AeDEn:

AeDEn met en place ces trois fonctionnalités au sein de différents services [MA00]:

- Service de distribution.

- Service de gestion de l'environnement.

- Service de détection et de notification.

Ce découpage en services est effectué pour des raisons d'extensibilité et de réutilisation. En effet, un service doit pouvoir être étendu avec une nouvelle politique sans que cela n'influe sur les autres services. De plus, chacun des services peut être utilisé indépendamment de la distribution. L'application peut, par exemple, consulter la ressource

bande passante et déclencher une détection des variations et cela indépendamment de la distribution.

L'architecture d'AeDEn est schématisée par la figure suivante (Figure2.11.):

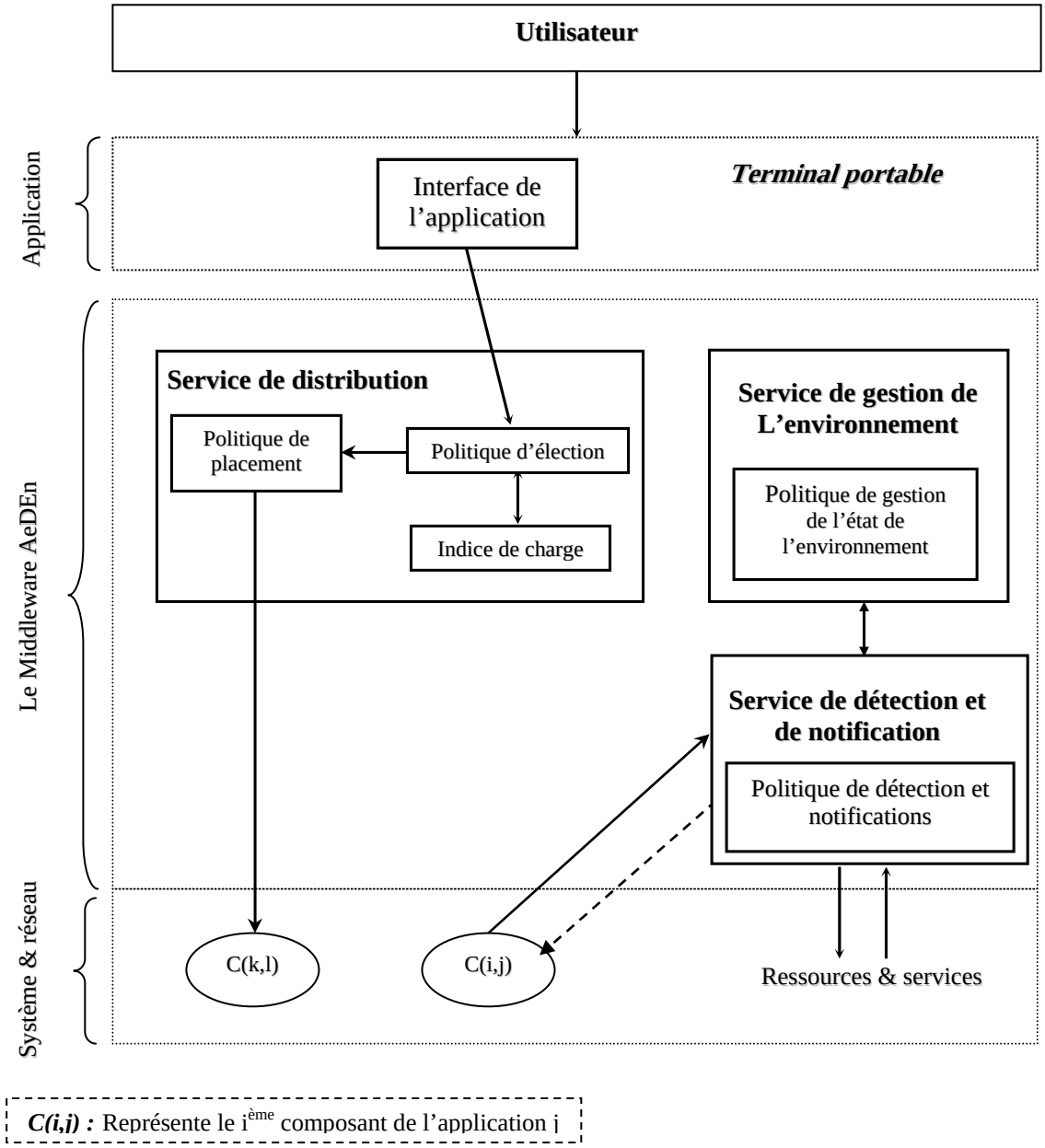

Figure2.11. *Architecture d'AeDEn*

II.3.2.4.3.1 Le service de détection et de notification :

Son rôle principal est de fournir au service de gestion de l'environnement les changements détectés dans l'environnement. Le service de gestion de l'environnement enregistre les ressources et les services au niveau du service de détection et de notification selon les besoins des applications. Les services et les ressources peuvent être exprimés sous

forme de conditions simples décrivant un état d'une ressource (ou un service) ou complexes décrivant plusieurs ressources et/ou services.

II.3.2.4.3.2 Le service de gestion de l'environnement :

Le service de gestion de l'environnement maintient la description des ressources et des services utilisés. Ce service est implémenté sous forme d'une base de données constituée d'éléments (ressources et services). A chaque élément est associé un nom, une valeur, le terminal auquel il est attaché et un ensemble d'applications qui l'utilisent. Quand une application fait une référence à un composant, le service indique sa localisation et les ressources et services utilisés par ce composant. Il reçoit aussi les informations du service de détection et de notification quand un changement est détecté et met à jour, par conséquent, l'état de l'environnement.

II.3.2.4.3.3 Le service de distribution :

Le service de distribution est responsable de la distribution des composants des applications ; il reçoit les demandes des composants des applications et interroge le service de gestion de l'environnement afin de pouvoir déterminer les composants élus pour la distribution. Le service de distribution effectue alors une opération de placement selon les informations fournies par le service de gestion de l'environnement.

II.3.2.4.4 Distribution adaptée à l'application:

Au lancement d'une application, les interactions entre cette application et AeDEn constituent la première phase d'adaptation. Celle-ci consiste à prendre en compte les besoins des composants de l'application ainsi que les ressources et services disponibles pour effectuer la distribution la plus adéquate des composants.

L'application fournit au service de distribution, et plus particulièrement à la politique d'élection, l'ensemble des composants pouvant être distribués. La politique d'élection interagit avec le service de gestion de l'environnement pour connaître l'état des ressources et services. Sa base d'informations est mise à jour après une notification de la part du service de détection et notification. L'indice de charge peut être calculé. Les composants élus sont alors soumis à la politique de placement qui détermine la station appropriée et effectue le transfert sur celle-ci. Le service de distribution enregistre enfin les composants dans le service de gestion de l'environnement et active une détection de variation sur ces composants. Ces derniers sont placés et le système peut tenir compte des changements de l'environnement pour reconsidérer ses choix [MA01].

II.3.2.4.5 Distribution dynamique en fonction de l'environnement

Les changements dans l'environnement sont pris en compte par les services d'AeDEn à deux niveaux différents :

- Au niveau de l'application, en fonction des changements de l'environnement et des décisions des différentes politiques, une politique de placement dynamique est appliquée qui prend en charge le placement de nouveaux composants (suspension, migration, exécution) .

- Au niveau du système de distribution en lui même, les changements de l'environnement influent sur les politiques de distribution elles-mêmes et conduisent à un changement dynamique des politiques utilisées. Une politique dynamique peut changer dynamiquement de variante, par exemple lorsque des changements surviennent dans l'environnement, chaque

politique dynamique possède une seule variante à un instant *t* mais peut dynamiquement en changer à un instant *t+1*.

II.3.2.5 ISAM

Le but du projet ISAM [YAB02] (Infrastructure de Support d'Applications Mobiles - Support Infrastructure for Mobile Applications) est la conception d'une infrastructure pour le développement et l'exécution des applications mobiles réparties.

Dans ISAM, l'adaptation des applications mobiles peut avoir lieu en quatre dimensions : Temporelle, Spatiale, Personnelle et Sociale [AYB01]. L'adaptation temporelle fait référence au moment de l'adaptation : à la demande (Semi statique) ou à l'exécution (dynamique). L'adaptation spatiale est liée à qui subit l'adaptation : qu'elles sont les entités essentielles à l'application qui influencent son comportement ? Cela peut inclure les entités, les activités et la localisation. La dimension personnelle est relative au modèle de comportement et des préférences de l'utilisateur mobile. Quand à la dimension sociale, elle apparaît quand l'environnement composé par d'autres applications influence le comportement de l'application s'exécutant sur une unité mobile.

Ce système est conçu tel qu'il offre une adaptation multi niveaux. Pour cela l'architecture suivante a été proposée :

II.3.2.5.1 Architecture d'ISAM :

Dans son architecture ISAM se base sur le scénario de mobilité le plus général qui permet une mobilité physique et logique des unités :

II.3.2.5.1.1 L'architecture Software

L'architecture d'ISAM (Figure2.12.) est pour des raisons d'organisation, structurée en modules. Celui de l'adaptation rentre dans la conception de beaucoup d'autres.

L'un des points importants dans cette architecture est lié à la décision d'adaptation qui est généralement prise en se basant sur les profils comportementaux de trois entités, qui sont : l'utilisateur mobile, l'application, et le système.

- La première supérieure, est constituée d'applications distribuées mobiles programmées avec les abstractions du Holoparadigm[3] permettant ainsi d'exprimer la mobilité de façon normale. L'adaptabilité est exprimée au niveau application (ISAMadapt) en ajoutant de nouvelles fonctionnalités à l'Holoparadigm.
- Une couche intermédiaire qui représente le noyau fonctionnel de l'architecture. Cette couche comporte trois niveaux :
    - le premier constitué de deux modules application : le scheduleur est l'élément clé de l'adaptation et l'environnement virtuel de l'application. Celui-ci est composé des différents éléments qui intègrent l'interface utilisateur Il est conçu par les informations dynamiquement obtenues, sur le travail de l'utilisateur, les préférences, … etc.

[3] L'approche de Multiparadigm intègre des paradigmes de langage de programmation. L'Holoparadigm (Holo) a été proposé comme modèle de multiparadigm orienté au développement des systèmes répartis. [BG01], [LBY02]

- C'est un autre module important d'écoute, qui obtient les informations en observant et en enregistrant le comportement de l'application et de l'utilisateur. Ces informations permettent d'obtenir l'évolution historique et quantitative des entités surveillées.
- Le dernier niveau de cette couche intermédiaire comporte les services de base de l'environnement d'exécution d'ISAM qui fournissent les fonctionnalités nécessaires pour le deuxième niveau

Une troisième et dernière couche, comprend les systèmes distribués existants, tels que la machine virtuelle Java, ainsi que des supports pour accéder aux deux types de réseaux (fixe et mobile)

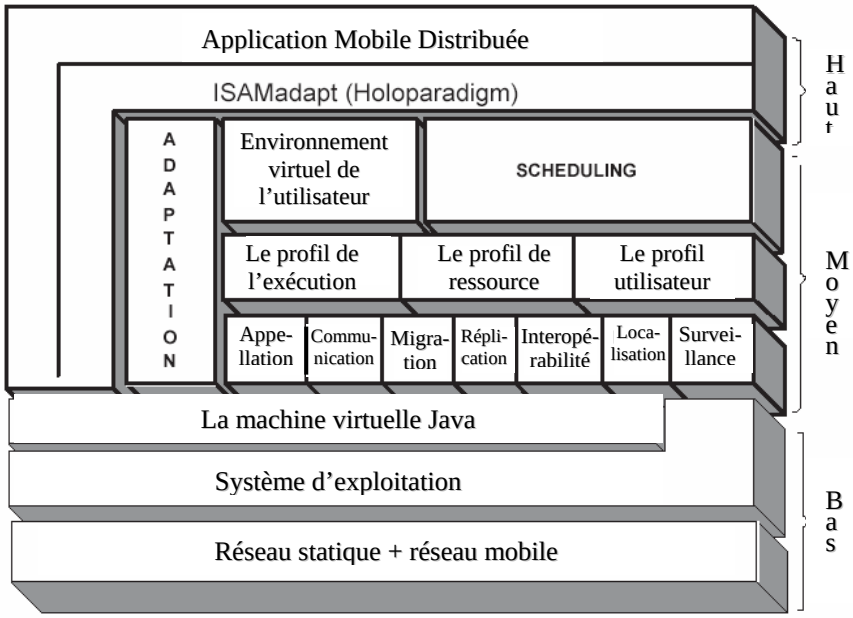

Figure2.12. *Architecture d'ISAM*

II.3.2.5.1.2 Une adaptation Multi niveaux

L'un des objectifs principaux de l'architecture d'ISAM est de rendre la totalité du système aussi adaptable que possible pour répondre aux besoins des applications mobiles. Pour cela, l'architecture logicielle fournit des modèles de programmation et d'interaction qui expriment la mobilité et l'adaptabilité, ainsi qu'un environnement d'exécution qui dirige le processus d'adaptation.

Dans le modèle d'adaptation combiné multi niveau d'ISAM (Figure2.13.) :

- Le système est responsable de certains cas d'adaptation qui sont en rapport avec les performances ou la gestion de ressources;
- L'application est responsable de prendre les décisions d'adaptation relatives à l'usage ou spécifique à son domaine;
- Et dans certains cas, les deux parties (application et système) sont responsables de prendre les décisions d'adaptation, suivant une négociation.

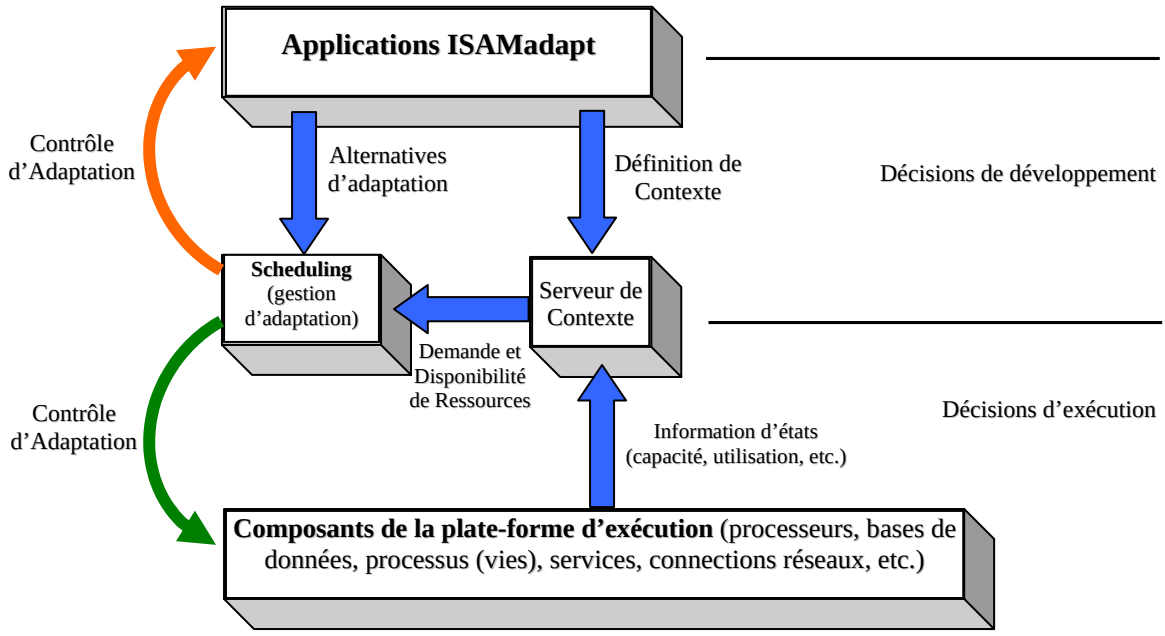

Figure2.13. *Une adaptation combinée multi niveaux*

II.3.2.6 Gaia : une infrastructure de développement pour les espaces actifs

Dans le projet Gaia, Cerqueira et al. [CHR01] définissent un environnement informatique générique qui convertit les espaces physiques et leurs dispositifs de calcul omniprésents en un système de calcul programmable, appelé espace actif (Active space), au niveau desquels les services, les utilisateurs, les données et les emplacements sont représentés et manipulés dynamiquement.

L'infrastructure de Gaia est basée sur trois éléments de base: un système d'exploitation middleware à base de composants qui fournit un environnement informatique générique pour le calcul omniprésent, un modèle d'application qui définit un mécanisme standard pour établir des applications omniprésentes, et un langage de script utilisé pour assembler les applications à base de composants et pour coordonner les activités.

II.3.2.6.1 GaiaOS :

C'est un système d'exploitation à base de composants qui opère sur les systèmes déjà existants, tel que Windows2000, …etc. Il utilise CORBA comme infrastructure de communication.

GaiaOS se compose de deux parties :

1. **L'UOB (Unified Object Bus):** c'est la base sur laquelle les autres composants de GaiaOS se fondent. Il fournit des outils communs pour manipuler le cycle de vie des composants fonctionnant dans un espace actif, tel que la création, l'inspection, la suppression, et l'appellation d'un composant, et définit quatre abstractions de base qui sont : l'*Unified Component*, *Component Manager*, *Component Container*, et l'*UOBHost*.
2. **Les services du noyau :** le noyau de GaiaOS comporte les services essentiels pour l'implémentation des fonctionnalités de base du système.

Le service fondamental est le gestionnaire d'évènement '*Event Manager*' qui est utilisé pour supporter les applications context-aware ; les applications peuvent s'enregistrer à un évènement spécifique pour qu'elle soit notifiée en cas de changement dans l'environnement. Le service 'découverte' utilise ce gestionnaire d'évènement pour repérer les composants logiciels, les personnes, et les entités physiques présentes dans un espace.

II.3.2.6.2 Le modèle d'application Gaia :

Le modèle d'application de Gaia [RC01] fournit un cadre standard pour la construction des applications omniprésentes, appelé 'Model-Presentation-Adapter-Controller-Coordinator' (MPACC). Ce cadre définit quatre composants de base (Figure2.14.) :

1. **Le modèle :** Le modèle consiste en l'implémentation de la structure centrale de l'application, qui se compose normalement des données et d'une interface programmable pour manipuler ces données. Un modèle a une ou plusieurs vues jointes, qui sont responsables de présenter les données d'une manière particulière.
2. **La présentation :** c'est la présentation physique du modèle qui permet à des utilisateurs de le percevoir.
3. **Le contrôleur :** Il exporte des mécanismes pour modifier l'état du modèle et coordonne entre n'importe quelle source de contexte qui peut affecter l'application.
4. **L'adaptateur :** c'est le composant responsable d'adapter le format des données du modèle aux caractéristiques d'un appareil de sortie arbitraire.
5. **Le coordonnateur :** Il contrôle la composition des applications et applique des politiques d'adaptation en se basant sur les aspects fonctionnels et non fonctionnels de l'application.

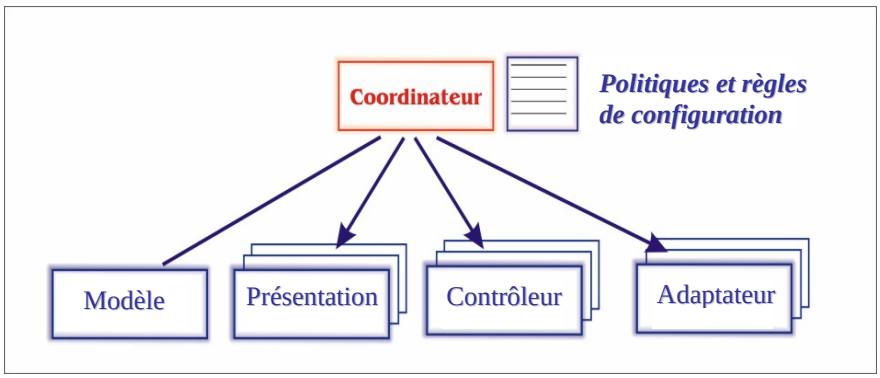

Figure2.14. *Le cadre d'application de Gaia*

II.3.2.6.3 La coordination dans Gaia :

Pour faciliter la configuration et la coordination entre les applications et le système 'GaiaOS', l'équipe de Gaia a présenté un mécanisme basé sur un langage de script qui est 'LuaOrb' [CRR99].

Cet outil permet d'automatiser les tâches de gestion et de configuration, de décrire et créer des scénarios de calcul omniprésent (Ubiquitous computing), tester les composants, créer des prototypes pour de nouvelles applications, … etc. Il est également utilisé pour implémenter deux types importants de composants dans Gaia :

- Le premier c'est le contrôleur de contexte 'context controller'. Ce type de composant intercepte les évènements relatifs aux informations de contexte en provenance du gestionnaire d'événement et déclenche les actions spécifiques. Les composants de ce type peuvent être utilisés pour implémenter l'interpréteur du contexte qui détecte des évènements spécifiques, les traduit, les intègre, ou les distille pour produire de nouveaux évènements.
- Le deuxième type c'est le coordinateur du cadre d'application, qui attache d'autres composants, coordonne leurs activités et définit des politiques de configuration.

LuaORB collabore avec le cadre d'application de Gaia et l'UOB pour offrir un support de composition et d'adaptation dynamique des applications ; LuaORB en offrant un mécanisme de reconfiguration dynamique des raccordements entre composants, et MPACC et UOB en offrant les moyens qui permettent cette reconfiguration.

II.3.3 Les applications et systèmes context-aware à service :

Mener toujours par l'émergence et le développement croissant des nouvelles technologies, les développeurs et chercheurs se sont orientés vers un nouveau type d'applications context-aware adaptables qui n'utilisent plus les valeurs de contexte pour améliorer leurs qualités de service mais pour offrir de nouveaux services liés essentiellement au type de contexte auquel on s'intéresse ; Dans ce nouveau domaine d'application, les valeurs de contexte ne sont plus limitées aux variations liées aux trois contraintes de mobilité mais prennent des valeurs beaucoup plus larges (température, orientation, le temps, la date, etc.)

Plusieurs travaux dans ce sens ont été faits, allant de la gestion de contexte, le développement des applications context-aware à service, aux middlewares soutenant de telles applications.

II.3.3.1 La détection de contexte :

Pour pouvoir utiliser les valeurs de contexte, des mécanismes qui captent et délivrent les différentes variations doivent être définis.

La technologie des capteurs (sensors) a été utilisée au début dans les domaines de la robotique, de la vision par ordinateur, etc. Cependant, les avancées technologiques dans le développement des capteurs (la taille réduite, la consommation d'énergie, le coût, etc.) ont encouragé leur intégration dans les calculateurs ultralégers.

L'une des valeurs de contexte largement utilisée par les applications context-aware à service est la localisation. C'est pourquoi plusieurs solutions de détection de localisation ont été proposées, allant de la localisation en plein air, à la localisation à l'intérieur des immeubles. Le système GPS (Global Positioning system) représente la solution la plus utilisée pour la localisation en plein air, mais comme certains appareils ne sont pas forcément dotés du GPS, d'autres solutions qui se basent sur la connectivité ont été proposées [BHE00]. Pour la localisation dans les immeubles où le système GPS ne peut être utilisé, plusieurs solutions selon les différents besoins ont été développées, certaines basées sur l'approche cellulaire [HHS+99] [PCB00], et d'autres non [BP00] [CM00].

En plus de la localisation, plusieurs mécanismes et capteurs ont été développés pour les différents types de contexte que cela soit un contexte de bas niveau (le taux de lumière, la température, les objets proches, la pression, l'orientation, etc.), ou de haut niveau (l'activité actuelle de l'utilisateur).

Quand à la modélisation des différentes valeurs de contexte, plusieurs modèles de présentation de contexte ont été développés dans les laboratoires de recherche ; la plupart des systèmes utilisent leurs propres modèles selon le type d'informations de contexte qu'ils exploitent et les différentes propriétés.

II.3.3.2 Quelques applications context-aware à service :

L'information de localisation en particulier a été largement utilisée et plusieurs exemples d'applications exploitant ces informations ont été définis parmi lesquels :

- ***Le Shopping Assistant*** [ACK94] : avec lequel l'appareil peut guider un acheteur à travers un magasin, lui fournissant des détails sur les articles à vendre, l'aidant à localiser des articles spécifiques, et faisant une comparaison entre les prix.

- ***Le CyberGuide*** [LKA96] : qui a comme but de fournir aux utilisateurs des services relatifs à leurs emplacements et orientations. Cette application a été développée en deux versions. La première pour un usage interne, utilise un système de positionnement à base d'infrarouge. Et la deuxième pour un usage externe qui se base sur le système GPS pour collecter les informations de localisation.

- ***Le Teleporting*** [BRH94] : appelée aussi '***followme***', elle met en place dynamiquement une carte sur laquelle on trouve les outils et services proches.

- ***Le Conference Assistant*** [DFS99] : c'est un assistant qui utilise une variété d'informations de contexte. Il examine le programme de la conférence, les sujets des présentations, la localisation de l'utilisateur, et les intérêts de ce dernier pour lui faire des suggestions. Et puis chaque fois que l'utilisateur rentre dans une salle de présentation, l'assistant affiche le nom du présentateur, le titre de la présentation, et d'autres informations relatives. Il peut également être utilisé pour envoyer un message particulier à toutes les personnes se trouvant dans un espace précis.

- ***Le People and Object Pager*** [Bro98], pour envoyer un avertissement dépendant de la localisation etc.

- ***L'office Assistant*** [YS00]: c'est un agent installé au seuil d'un bureau de haute circulation dans un environnement bruyant où le propriétaire du bureau préfère avoir une porte fermée. La tâche de l'agent est d'interagir avec les visiteurs, leur donnant des renseignements au sujet de leurs rendez-vous ou leur donnant de nouveaux rendez-vous, et met à jour la liste des visiteurs pour refléter des interactions récentes.

La plupart des applications qu'on trouve dans les articles sont des prototypes développés dans des laboratoires de recherche. Et rares sont les solutions commercialisées. Ceci est dû à la complexité de collecte et de représentation des données contextuelles et à la complexité de développement de telles applications qui doivent inclure un certain niveau d'intelligence pour traiter l'information et déduire la signification.

II.3.3.3 Quelques systèmes supportant les applications context-aware à service

Pour développer les différents types d'applications context-aware à service, plusieurs plateformes ou middleware ont été développées :

II.3.3.3.1 Nexus :

Nexus [FKV00], est un middleware qui a été développé dans le but d'offrir une plate-forme générique qui supporte des applications location-aware. Il fournit une infrastructure qui

supporte un environnement de communication hétérogène et gère des modèles spatiaux dynamiques qui représentent le monde réel ainsi que des objets virtuels. Ces derniers agissent comme des bus entre la plate-forme Nexus et les services et sources d'informations externes, de sorte que les informations stockées peuvent être spatialement structurées.

II.3.3.3.1.1 Architecture de Nexus :

Comme le montre la figure suivante (Figure2.15.), l'infrastructure de Nexus est constituée de quatre composants qui travaillent en collaboration :

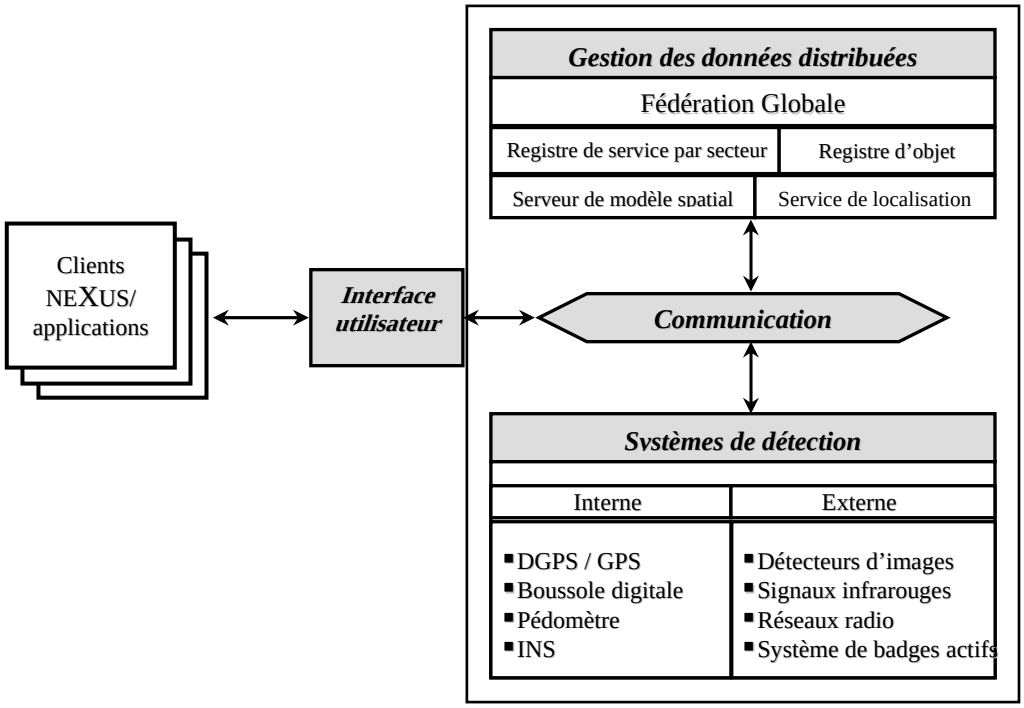

Figure2.15. *Architecture de la plate-forme NEXUS*

A/ L'interface utilisateur :

Le support de l'Interface Utilisateur s'exécute sur l'appareil mobile et contient les fonctionnalités de base exigées par les clients pour interagir avec la plate-forme NEXUS ainsi que pour afficher et naviguer à travers le modèle. Il fournit également un support d'adaptation aux différents appareils mobiles (Par exemple, un PDA exige une interface complètement différente de celle d'un PC portable). L'interface de l'utilisateur doit autoriser une communication facile entre le client et la plate-forme.

B/ L'environnement de communication :

Pour accéder à une information, l'utilisateur, via son client NEXUS, doit être capable de se connecter aux sources d'information en utilisant des liens sans fil. Ces liens peuvent varier d'un réseau cellulaire tel que GSM ou GPRS, à un réseau WLAN. Pour dissimuler la différence entre les différents réseaux hétérogènes existants, la couche communication de NEXUS agit comme un courtier.

C/ Les systèmes de détection :

L'étape de détection de la localisation est une phase très importante pour les applications location-aware. Dans le système NEXUS, les applications s'exécutent généralement dans les deux zones (en plein air et à l'intérieur d'un building). Cependant, utiliser la même technique de détection dans les deux environnements est une chose presque impossible. En se basant sur ce fait ainsi que sur les différents travaux de recherche qui ont montré l'importance de combiner différents outils basées sur différents principes, NEXUS propose un outil de positionnement qui comporte différents systèmes parmi les quels :

- le GPS (Global Positioning System) un système de positionnement dans les grandes surfaces, cette technique permet de détecter la position avec une précision de quelques mètres.
- Les détecteurs d'images (Imaging sensors) : Ce sont des techniques de positionnement dans les building, ils utilisent des caméras digitales pour la capture d'images et des techniques d'interprétation [AS97]
- Les systèmes de badges actifs : c'est un autre système de localisation interne. L'idée de ce système est très simple, il suffit juste d'implanter un réseau de localisation de badge actif et chaque utilisateur doit porter un badge unique [WHF92].

D/ La gestion de données :

L'élément de base de la plate-forme NEXUS est responsable de la gestion de données. Pour cela, des modèles du monde réel sont stockés, des requêtes sont exécutées, et des mises à jour sont apportées. Les données spatiales doivent être fournies sous plusieurs représentations selon la demande des différentes applications context-aware. C'est pourquoi des algorithmes appropriés pour déduire les niveaux de détail nécessaires sont implantés dans la plate-forme. De plus et pour garantir l'interopérabilité, NEXUS définit des dépendances entre les différents modèles.

Le stockage et traitement des données dans NEXUS sont divisés en deux parties différentes. Quant aux objets mobiles, les spécifications de l'utilisateur tel que son nom et son profil sont manipulées dans le registre d'objets " *Object Register* ", alors que les positions de l'utilisateur sont administrées par le service de localisation. D'autre part, le registre de service par secteur " Area Service *Register* " est responsable de l'assignement de modèles de région plus augmentés aux serveurs de modèles (*Spatial Model Servers*), ce dernier se charge du stockage et de la gestion de ces modèles.

Pour la gestion de plusieurs modèles de région augmentés, des serveurs qui supportent le stockage et le traitement de données spatiales peuvent être implantés, tel que Oracle, DB2.

Toujours dans le même sens et pour améliorer le développement des services et applications basés sur la localisation et réduire leurs cycles de développement, d'autres systèmes middleware qui intègrent des technologies de positionnement ont été développés, parmi lesquels : OracleiASWE [S2], Alternis [S3], SignalSoft [S4], …etc.

II.3.3.3.2 Context Toolkit :

Context toolkit [DSA99], a été développé dans le but de supporter les applications context-aware. Son architecture supporte l'acquisition et la livraison du contexte en utilisant trois types de composants (Figure2.16.):

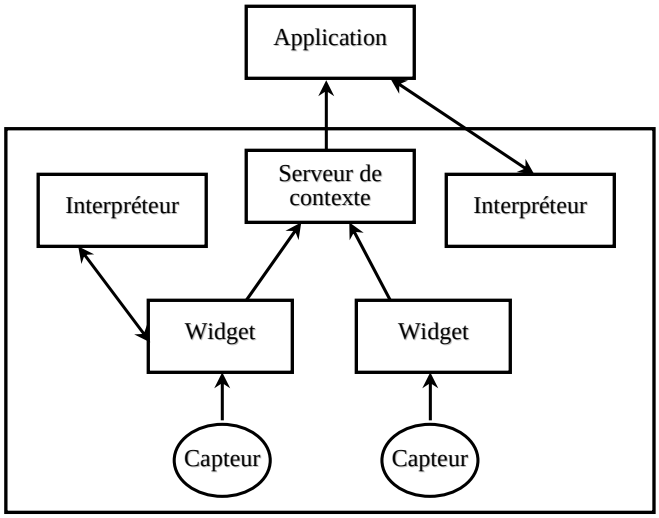

Figure2.16. *Architecture du Context Toolkit*

1. **Les composants 'Context widgets' :** sont des composants logiciels qui fournissent aux applications l'accès à leurs informations de contexte tout en leur cachant la complexité du processus d'acquisition.
   Le widget est défini par des attributs qui sont des segments de contexte rendus disponibles aux autres composants, et par des appels de service qui représentent les types d'évènements qu'il peut utiliser pour notifier les applications enregistrées. Il encapsule des informations sur une pièce de contexte, tel que l'endroit ou l'activité, et maintient l'état contextuel permettant à d'autres composants de rechercher l'historique du contexte.
2. **Les serveurs de contexte 'context servers' :** ces éléments sont utilisés pour rassembler tout le contexte d'une entité particulière. Ils sont responsables de la souscription à chaque widget et agissent comme des Proxy entre l'application et le 'widget'.

   Le serveur de contexte peut être vu comme un widget composé. En tant que tel, il a des attributs et des appels de service, et son historique peut être recherché
3. **Les interpréteurs de contexte :** ces objets sont responsables d'implémenter l'interprétation des informations de contexte. Ils peuvent translater ces informations entre différents formats de représentation, ou combiner différentes informations de contexte pour fournir de nouvelles représentations.

Chacun de ces objets s'exécute de façon autonome. Ils peuvent êtres instanciés sur le même dispositif ou sur des dispositifs multiples. Le HTTP et le XML sont utilisés pour garantir la communication entre les différents composants.

II.3.3.3.3 TEA (The Technology for Enabling Awareness):

TEA [SAT99] est une autre approche qui supporte les applications context-aware et convertit les valeurs de sortie des capteurs en des profils de contexte.

Ceci est réalisé via une architecture de quatre niveaux (Figure2.17.) et principalement par l'utilisation des '*sensors*', '*cues*', et des '*context profils*'.

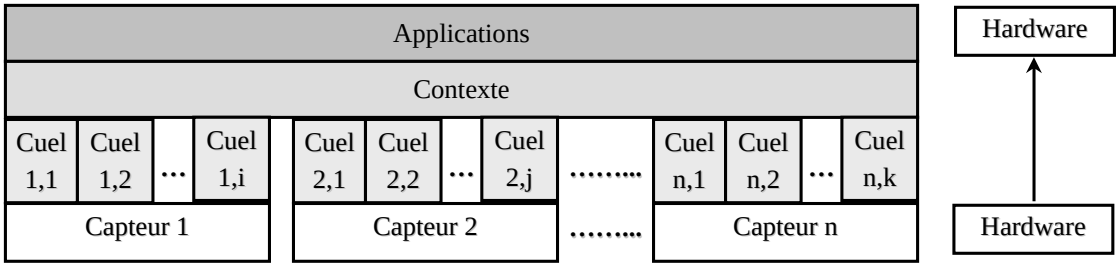

Figure2.17. *Architecture du TEA*

1. **La couche des capteurs (sensors) :** dans cette architecture deux types de capteurs sont définis, les capteurs physiques sont des composants hard responsables de mesurer les paramètres physiques de l'environnement. Toutes les informations recueillies de l'hôte (ex. l'heure, la cellule GSM, etc.) sont considérées comme des capteurs logiques. Chaque capteur C_i est considéré comme une fonction dépendante du temps qui renvoie une valeur scalaire, vectorielle, ou symbolique, et pour laquelle un ensemble (fini ou infini) des valeurs possibles est défini.
2. **La couche des cues :** Les valeurs de sortie des capteurs sont considérées comme du contexte de bas niveau qui ne peut être exploité par les applications et nécessite donc des transformations. Une 'Cue' peut être considérée comme une fonction qui a en entrée les valeurs d'un capteur *i*. Le filtrage est un exemple de 'cue' qui peut être utilisé pour donner une meilleure interprétation aux informations interceptées dans l'étape précédente. Notons que chaque 'Cue' est dépendante d'un capteur unique et pour chaque capteur un ensemble de 'Cues' peuvent être calculées.
3. **La couche de contexte :** Les données résultantes de la couche des 'cues', sont transformées au niveau de la troisième couche en des profils de contexte suivant les spécifications de l'utilisateur. C'est l'étape principale de cette approche.
4. **La couche application :** au niveau de cette couche, trois sémantiques différentes pour l'utilisation d'une information de contexte sont définies. Des actions de base peuvent être effectuées au début d'un contexte (situation), durant un contexte, et à la fin d'un contexte.

II.4 Discussion :

Le problème d'adaptation est un des problèmes largement étudié dans les systèmes distribués classiques. Cependant, avec l'arrivée des réseaux mobiles, le problème s'est accentué par l'aspect dynamique de changement ; le contexte d'exécution qui peut influer sur le comportement d'une application (tels que la disponibilité des ressources : batterie et bande passante, la location, l'ajout de nouveaux terminaux ou le changement de service) est très dynamique et les changements sont fréquents. Contrairement à celui des réseaux distribués fixes qui est plus au moins statique.

Il ressort de l'étude précédente que les domaines d'adaptation et de context-aware computing sont deux domaines très liés et présentent plusieurs facettes, abordés à chaque fois suivant un axe spécifique et en se basant sur différentes contraintes que nous allons utiliser pour établir notre étude comparative (Tableau2.1) :

Objectif de l'adaptation :

- Déjà si nous commençons par voir l'évolution historique des différents travaux, nous remarquons que l'adaptation et le context-aware computing ont suivi l'évolution des différents besoins. Au début, l'adaptation a été vue comme une urgence pour permettre aux applications déjà existantes de continuer à travailler, et plusieurs middlewares qui garantissent cette adaptation ont été développés (exp. Coda). Par la suite, le besoin d'impliquer l'application dans l'adaptation pour améliorer la qualité de service a donné naissance à une nouvelle classe de solutions, dans laquelle l'application et le système collaborent pour exécuter le processus d'adaptation (exp. Odyssey). Actuellement, les avancées technologiques dans les réseaux mobiles (calculateurs ultralégers, les capteurs, etc.) ont ouvert le chemin vers de nouvelles possibilités et une nouvelle classe d'applications et de systèmes qui n'utilisent plus l'adaptation que pour améliorer la qualité de service, mais également et principalement pour garantir un service spécifique lié à des valeurs de contexte spécifiques. En plus de tous ces travaux, le context-aware computing et l'adaptation commencent a être utilisés pour aborder et traiter des problématiques très ambitieuses liées aux réseaux ad Hoc, tels que les mécanismes de routage, la détection des ressources, et l'établissement d'un réseau Ad Hoc.

Le niveau d'implantation des stratégies d'adaptation et de context-awareness :

- Si nous prenons comme critère de classification le niveau d'implantation des stratégies d'adaptation et de context-awareness, nous remarquons que les différentes solutions peuvent être divisées en deux groupes. Un premier groupe dans lequel l'adaptation est sous la responsabilité entière du système ; on parle alors d'une adaptation transparente à l'application, et c'est au système d'être sensible (aware) aux différents changements et d'appliquer les différentes stratégies d'adaptation. Et un deuxième, où le processus d'adaptation est garanti à travers une collaboration entre le système et l'application, ce qui est connu sous le nom de Application-aware adaptation. Dans ce cas, les deux entités système et applications sont sensibles (aware) aux différentes variations.

Le type de contexte pris en compte :

- Durant notre étude, nous avons constaté que l'ensemble des valeurs de contexte pris en compte a évolué en même temps que les besoins et les objectifs. Au début lorsque le besoin était de garantir le bon fonctionnement des applications et donc un certain niveau de qualité de service, les valeurs de contexte prises en compte par les systèmes d'adaptation (variation de la bande, taille d'affichage, niveau d'énergie, etc.) étaient principalement liées aux trois contraintes de base d'un réseau mobile (portabilité, mobilité, et liens sans fil). Avec l'apparition des nouvelles applications context-aware à service et des différents plateformes les soutenant, cet ensemble s'est enrichi avec une infinité de valeurs ou de types (la température, l'orientation, etc.).
- Sous un autre œil, nous pouvons voir que chacun des systèmes étudiés et relativement au domaine d'application considéré n'a pris en compte qu'un niveau restreint de contexte. Par exemple, le système Odyssey qui s'intéresse au domaine du multimédia a pris en compte un des problèmes relatifs à la contrainte de portabilité qui est la source d'énergie limitée ainsi que les problèmes de déconnexion et de la faible bande passante. NEXUS de son côté est destiné aux applications location-aware et pour cela il se concentre sur les problèmes relatifs à la mobilité tels que la détection de ressources, de services, et de localisation. Chacun utilisant des principes différents.

Le niveau architectural pris en compte :

- Prenant en compte le niveau architectural et les problèmes étudiés, nous remarquons que chacun des systèmes s'est intéressé à des problèmes spécifiques. Certains comme Odyssey et le Middleware réflectif, se sont intéressés à l'interaction entre application et système context-aware (comment l'application est notifiée des changements ? comment peut elle influencer le comportement du système ? etc.) sans se soucier de la détection ou de l'interprétation des valeurs de contexte et en supposant que ces traitements sont faits automatiquement. D'autres tels que NEXUS et TEA, traitent les problèmes de bas niveau ; la détection, l'interprétation, etc.

Pour résumer tout ce qui a été dit et pour plus de détail, nous proposons le tableau suivant :

	Objectif	**Niveau d'implantation des stratégies**	**La problématique**	**(Paramètres de contexte considérés)**	**Principes et outils Utilisés**	**Domaine d'application (exe d'application)**
Odyssey	Améliorer la Qualité de service	Application et système context-aware	Politiques d'adaptation Et interaction entre application et système	• Energie • Déconnexion • Faible bande passante	• Cache • Fidélité aux données • Les APIs	• Vidéo-player • Web Browser • Speech recognizer
Coda	//	Application-Transparente, système sensible aux changements	Comment garantir l'adaptation (Politiques d'adaptation)	• Déconnexion	• Cache	• Accès aux BD
WebExpress	//	//	Comment garantir l'adaptation (Politiques d'adaptation)	▪ Les problèmes liés aux liens sans fil	• Cache • Autres technique tq la compression	• Navigation sur le Web
Middleware refléctif	Améliorer la Qualité de service	Application et système context-aware	L'interaction entre application et système	• Pbs de portabilité • Pbs des liens sans fil	• Réflexion • Métadonnées	• e-Shopping
AeDEN	Améliorer la Qualité de service	L'architecture context-aware	Comment garantir l'adaptation	• Pbs de portabilité • Déconnexion	• Décomposition et distribution des applications	•
NEXUS	Pour les applications à service	Application et système context-aware	Détection et modélisation du contexte	• La localisation • Découverte de service	• Une BD spatiale • Technique d'objets virtuels	• Les applications Location-aware
TEA	//	Application et système contxte-aware	Détection et modélisation du contexte	• Différentes valeurs de contexte	• Profil de contexte • Techniques de traitement	• Les applications context-aware à service
Context toolkit	//	Application et système contxte-aware	Détection et modélisation du contexte	//	• Programmation par composant	• Les applications context-aware à service
			Du niveau de	• Découverte	• Architecture	

Gaïa	Améliorer la Qualité de service	//	détection au niveau application	de service et de ressources, • Et autre	modulaire et la programmation par composant	• Adaptation de contenu
ISAM	Améliorer la Qualité de service	- Transparente - Coopérative	Comment garantir l'adaptation	• Ressources • Localisation • Le contexte utilisateur	• Comportement collaboratif • Langage exprimant la mobilité	

Tableau2.1. *Etude comparative des différents systèmes d'adaptation*

Deuxième partie : domaine d'application

Le problème d'adaptation et du context-aware computing est un problème très vaste qui a été traité de différentes manières dans les environnements mobiles hétérogènes, principalement suivant deux axes d'objectif : soit pour améliorer la qualité de service, soit pour offrir des services spécifiques. Dans notre travail, nous nous sommes intéressés au premier axe, qui est d'utiliser l'adaptation et le context-awareness pour améliorer la qualité de service. Notre intérêt porte sur le multimédia comme domaine d'application.

Cette partie sera structurée en deux chapitres. Le premier chapitre présente les caractéristiques du flux multimédia, la notion de qualité de service et le besoin d'adaptation, puis donne un aperçu sur les travaux liés au domaine. Le deuxième chapitre, présente notre contribution qui se résume en une architecture d'adaptation d'une application de vidéo à la demande et plus précisément d'une application de vidéo formation. L'objectif de cette architecture est de garantir le bon fonctionnement de l'application, et donc un certain niveau de qualité de service perçu par l'utilisateur, en utilisant un processus d'adaptation basé sur des données du contexte.

CHAPITRE III :
Les Applications Multimédias Mobiles

III.1 Introduction :

Durant cette dernière décennie, le monde des communications et des systèmes d'informations a subi une profonde mutation sous l'impulsion conjuguée de plusieurs facteurs technologiques et socio-économiques parmi lesquels : l'explosion des réseaux mobiles de seconde et de troisième génération, la forte progression des capacités de transmission avec les technologies photoniques offrant de gros débits, la forte progression des capacités de stockage et de traitement de l'ensemble des calculateurs, la variété croissante du contenu publié par ces sources de données, avec une prédominance des données multimédia (documents XML, séquences audio MP3, vidéo MPEG4 ...), et la prolifération des calculateurs ultralégers connectables à Internet (agendas électroniques, téléphones cellulaires, cartes à puce …).

Toutes ces évolutions ont permis l'émergence d'une variété d'applications multimédias mobiles. Ces applications tirent partie de la possibilité d'avoir accès aux mêmes informations à distance que localement, via différents types de terminaux et de liaison (omniprésence). Il s'agit par exemple des applications de télé éducation, de télémédecine, de télé présence, de visioconférence, ou de travail coopératif.

III.2 Les données multimédia :

La caractéristique commune la plus évidente des applications multimédias est que les communications internes s'effectuent principalement au moyen de flux de données. Pour une application multimédia, un flux est constitué par une suite de messages émis régulièrement. Ces flux sont le plus souvent de type audio ou vidéo, auxquels sont ajoutées parfois des informations de contrôle ou de complément : sous-titres, état des joueurs dans un jeu, etc.

III.2.1 Contraintes temporelles :

Les contraintes temporelles en général peuvent être classées en deux classes : les contraintes temporelles dures ou fortes et les contraintes temporelles faibles. Contrairement aux contraints dures qui s'appliquent aux applications critiques et qui doivent être respectées tout le temps (le non respect de la contrainte ne serait ce qu'une fois peut mener à des conséquences graves). Les contraintes temporelles faibles, peuvent tolérer leur non respect de temps en temps sans remettre en cause le service fourni.

Les contraintes temporelles fortes sont décrites par des expressions du type : « le temps pour effectuer le service σ doit être strictement compris dans un intervalle I » et les contraintes faibles par « le temps du service σ doit être dans un intervalle I avec une probabilité p » [Dem02].

Les contraintes temporelles qui pèsent sur les applications multimédias se répartissent en trois catégories : les contraintes de bout en bout, les contraintes de synchronisation et la gigue. Les ***contraintes de bout en bout*** portent sur le délai qui s'écoule entre la création d'une information et son utilisation. Les ***contraintes de synchronisation*** mesurent la simultanéité de l'arrivée d'informations diverses qui composent un même message (exp. décalage entre son et image). Et enfin, ***la gigue*** caractérise la régularité d'un flux ; il s'agit de la variation des délais par rapport à la moyenne.

Pour chacun des médias employés, un seuil est fixé pour le taux d'erreur de synchronisation. Ce seuil dépend de ce qui peut être supporté par les utilisateurs [Dem02].

III.2.2 Algorithmes de compression :

Avant la transmission du flux et quelque soit son type, les données doivent être codées et compressées. Plusieurs protocoles pour l'encodage des données multimédia existent. D'une manière générale, on distingue deux classes de techniques : les techniques réversibles (procédés de compression dits sans perte), le flux une fois décompressé est exactement identique à celui d'origine. Et les techniques irréversibles (procédés de compression avec perte), qui impliquent que le flux décompressé contient moins de données que le flux initial.

Dans cette partie nous nous intéressons principalement au flux vidéo qui consomme beaucoup plus de ressources que l'audio et qui doit donc être compressé pour réduire le besoin en bande passante. En général pour la vidéo deux types de redondances peuvent êtres supprimées :

- Les redondances spatiales, on utilisera pour ce faire des techniques de transformées en fréquence de type DCT (transformées en cosinus discrète);
- Les redondances temporelles (au moyen de techniques d'estimation et compensation de mouvements).

Deux classes de normes de codage de la vidéo existent, les H26x développées par l'ITU-T utilisées surtout pour la visioconférence, et les MPEG de ISO utilisées principalement pour la télévision numérique.

Nous insistons sur les protocoles MPEG destinés à la compression des flux vidéo, en raison de la possibilité de disposer de formats liés hiérarchiquement. Ceci nous semble un atout dans notre travail d'adaptation en fonction de la disponibilité des ressources. L'originalité de ce protocole est qu'il permet de définir plusieurs qualités pour un même flux. Le flux se décompose en trois couches complémentaires (décomposition hiérarchique) : la première est de qualité minimale, l'ajout de la deuxième permet d'améliorer la qualité, et enfin la troisième couche permet d'obtenir une qualité optimale.

Le groupe MPEG a été établi en 1988 dans le but de développer des standards internationaux de compression, décompression, traitement et codage d'image animées et de données audio.

MPEG permet l'encodage d'une vidéo grâce à plusieurs techniques :

- Intra coded frames (Frames I, correspondant à un codage interne): les images sont codées séparément sans faire référence aux images précédentes. De telles images sont nécessaires dans une vidéo MPEG car ce sont elles qui assurent la cohésion de l'image (puisque les autres sont décrites par rapport aux images qui les entourent), elles sont utiles notamment pour les flux vidéo qui peuvent être pris en cours de route (télévision), et sont indispensables en cas d'erreur dans la réception. Il y en a donc une ou deux par seconde dans une vidéo MPEG.
- Predictive coded frames (frames P ou codage prédictif): les images sont décrites par différence avec les images précédentes. L'encodeur recherche les différences de l'image par rapport à la précédente et définit des blocs, appelés macroblocs (16x16 pixels) qui se superposeront à l'image précédente. Par rapport aux frames-I (compressant directement), les frames-P demandent d'avoir toujours en mémoire l'image précédente.

- Bidirectionally predictive coded frames (Frames B): De la même façon que les frames P, les frames B sont travaillées par différences par rapport à une image de référence, sauf que dans le cas des frames B cette différence s'effectue par apport à l'image précédente et l'image suivante.

Dans la pratique et afin d'optimiser le codage MPEG, les séquences d'images sont codées suivant une suite d'images I, B, et P dont l'ordre a été déterminé expérimentalement. La séquence type appelée GOP (Group Of Pictures ou en français groupes d'images) est la suivante: IBBPBBPBBPBBI

Pendant l'évolution de MPEG, plusieurs standards ont vu le jour parmi lesquels nous citons :

- Le MPEG-1, développé en 1988, est un standard pour la compression des données vidéo et des canaux audio associés (jusqu'à 2 canaux pour une écoute stéréo). Il permet le stockage de vidéos à un débit de 1.5Mbps dans une qualité proche des cassettes VHS sur un support CD appelé VCD (Vidéo CD).
- Le MPEG-2, un standard dédié originalement à la télévision numérique (HDTV) offrant une qualité élevé à un débit pouvant aller jusqu'à 40 Mbps, et 5 canaux audio. Le MPEG-2 permet de plus une identification et une protection contre le piratage. Il s'agit du format utilisé par les DVD vidéo.
- Le MPEG-4, un standard destiné à permettre le codage de données multimédia sous formes d'objets numériques, afin d'obtenir une plus grande interactivité, ce qui rend son usage particulièrement adapté au Web et aux périphériques mobiles. Elle s'adapte également à tous les types de réseaux (y compris le sans fil, les réseaux locaux et l'Internet).

Ces dernières années une nouvelle norme a vu le jour, la H.264 / MPEG4 AVC, cette norme H.264/AVC est le projet de normalisation vidéo du groupe VCEG (Video Coding Experts Group) de l'UIT-T et du groupe MPEG de l'ISO/CEI. Ce projet vise à mettre au point un système de codage vidéo simple et direct, doté de performances accrues en matière de compression, et à permettre une présentation vidéo « conviviale» compatible avec les modes conversationnel (vidéo-téléphonie) ou non (stockage, radiodiffusion et diffusion en continu). Ce nouveau codeur/décodeur vidéo est notamment destiné aux applications de transmission sur les réseaux mobiles et sur Internet.

Les travaux de l'UIT-T et de l'ISO peuvent être résumés dans le schéma suivant :

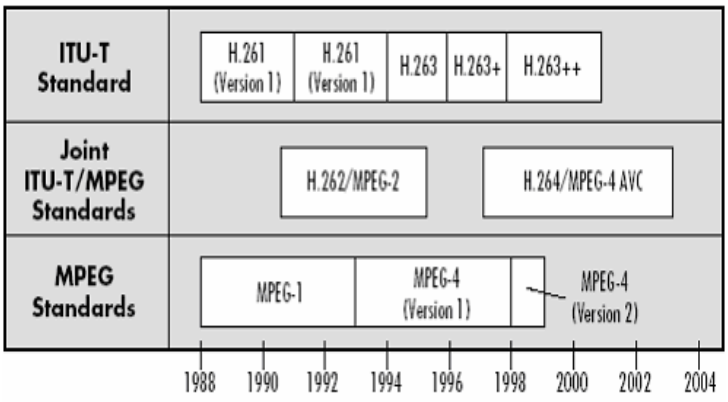

Figure3.1. *Evolution des standards de codage*

III.3 Les différents types d'applications multimédia :

Les applications multimédias en général peuvent être classées en deux grandes catégories qui sont les applications client/serveur et les applications peer-to-peer.

Dans une application peer-to-peer, chaque partie a les mêmes possibilités et l'une ou l'autre partie peut lancer une session de communication. Un utilisateur peut communiquer avec un ou plusieurs utilisateurs de la même application en échangeant des données multimédia (speech, image, texte, ou vidéo). Dans certaines applications de ce type, on peut avoir des nœuds qui jouent le rôle de client et de serveur en même temps

Dans une application client/serveur, l'utilisateur interagit avec des portions de programmes. Généralement, les applications de ce type sont géographiquement distribuées. La partie cliente de l'application est exécutée sur le terminal utilisateur, elle permet la représentation du contenu à l'utilisateur et l'interaction application/utilisateur via une interface utilisateur. De l'autre côté, la partie serveur se charge de fournir le contenu à l'utilisateur. Notons qu'un serveur peut fournir du contenu à plusieurs clients en même temps.

Sous un autre œil, les applications multimédia peuvent être classées en quatre classes : la vidéo sur demande, les applications interactives, la visioconférence, et la télésurveillance.

III.3.1 La vidéo à la demande :

Un système de vidéo sur demande est un système multimédia interactif qui travaille comme une télé câblée, à la seule différence que l'utilisateur peut sélectionner la séquence vidéo qu'il souhaite visualiser à partir d'une base de données. Dans ce type d'application, les données stockées sur le serveur sont transmises en un format codé compressé et mises en marche sur le terminal utilisateur suite à une requête de ce dernier.

Plusieurs exemples d'application vidéo sur demande existent parmi lesquelles :

- La télévision sur demande qui permet de voir des films sur demande.
- Les applications de vidéo formation grâce auxquelles on peut prendre un cours donné par les meilleurs professeurs dans n'importe quel domaine.
- Les applications de jeux, etc.

III.3.2 La visioconférence :

La visioconférence permet d'organiser des meetings ou conférences quel que soit l'emplacement des participants, et ceci en utilisant les techniques de visiocommunication pour pallier leur éloignement

La visiocommunication est l'ensemble de techniques numériques permettant la communication visuelle et audio via un réseau.

Parmi les applications de ce type on trouve :

- Un ordinateur personnel qui communique avec un autre.
- La diffusion d'un cours à partir d'un studio, etc.

III.3.3 La télésurveillance :

Les systèmes de télésurveillance sont des systèmes qui consistent à capturer les images des caméras de surveillance, les transporter sur le réseau et les acheminer vers les terminaux. On peut les installer n'importe où : dans les écoles, les bases militaires, les hôpitaux, les centres de recherche, les supermarchés, etc.

Ces différents types d'applications multimédias affrontaient déjà beaucoup de défis dans les réseaux traditionnels, aussi bien au niveau des capacités de calcul qu'au niveau de la transmission. Toutefois, avec tous les développements récents dans les domaines de télécommunication et de la fabrication des calculateurs ultralégers, qui permettent actuellement d'exécuter ces applications n'importe où, n'importe quand et à partir d'une variété de terminaux (omniprésence), d'autres défis et problèmes viennent s'ajouter à ceux déjà posés. Ceci est dû à l'hétérogénéité d'un tel environnement d'exécution, que ce soit au niveau des terminaux à partir desquels on exécute l'application ou au niveau des supports de transmission.

III.4 Les protocoles du multimédia

Dans cette partie, nous allons présenter les protocoles les plus utilisés pour la transmission vidéo ou multimédia sur les liens sans fil [FSR].

III.4.1 Le protocole RTP

Le streaming consiste à diffuser une vidéo d'un serveur vers un client à travers un réseau de type Internet (protocole IP). Le serveur segmente la vidéo en paquets susceptibles d'être diffusés sur le réseau. Ces paquets sont ensuite assemblés par le client afin de reconstituer la vidéo. A la différence d'un simple transfert de fichier, la vidéo est jouée au fur et à mesure que les paquets arrivent. Ces paquets sont ensuite détruits.

Le RTP (Real Time Protocol) est un protocole qui permet de faire à proprement parler du streaming, c'est à dire de la diffusion de contenu en temps réel. Le principe du protocole RTP consiste à envoyer les paquets en temps réel sur le réseau. Les paquets sont marqués temporellement de manière à être réordonnancés par le client afin d'afficher la vidéo de manière cohérente. Grâce à ce protocole, on peut aussi bien diffuser des vidéos préenregistrées que des images en direct.

RTP est défini depuis janvier 1996 par la RFC 1889 [SC+96]. Cette RFC (Request For Comment) a elle même été complétée par de très nombreuses autres RFCs. RTP a été créé afin de définir des fonctions de diffusion temps réel comme la diffusion audio et/ou vidéo. Il permet des services de type unicast ou multicast sans garanties de qualité de service (QdS). Les données sont augmentées de l'ajout du protocole de contrôle RTCP (RTP Control Protocol). Par opposition à HTTP et FTP qui fonctionnent au-dessus de TCP (mode connecté), RTP fonctionne au-dessus d'UDP (mode non connecté).

III.4.2 RTCP

Le protocole RTP (RTP control protocol) permet d'encapsuler les données vidéo ou audio dans un format valide pour la transmission réseau, mais n'offre pas de garanties quant à la délivrance des paquets, ni l'ordre d'arrivée des paquets, ni la qualité du réseau sous-jacent. C'est pourquoi RTCP, défini également dans la REF 1889, a été proposé pour contrôler la bonne marche de la transmission par RTP.

Le protocole RTCP est basé sur des transmissions périodiques de paquets de contrôle par tous les participants dans la session.

Il existe 5 types de paquets RTCP pour transporter des informations de contrôle :

- SR : Sender Report, transmission de statistiques des participants actifs en émission.
- RR : Receiver Report, transmission de statistiques des participants passifs.
- SDES : Source Description items (CNAME, NAME, EMAIL, PHONE,...).

- BYE : Fin de participation
- APP : fonctions spécifiques à l'application

Chaque paquet RTCP commence par une partie fixe, suivie par des éléments structurés qui peuvent être de longueur variable selon le type de paquet, mais toujours terminé sur une frontière de 32 bits.

III.4.3 Real Time Streaming Protocol (RTSP)

RTSP est un protocole de niveau applicatif qui a été défini depuis avril 1998 par la RFC 2326 [SRL98]. Il sert à contrôler les propriétés temps réel du contenu délivré. Il est adapté aussi bien à la diffusion de données préenregistrées que de données diffusées en direct.

Le protocole RTSP peut être considéré comme une télécommande lors de la diffusion des médias [FSR]. Il est utilisé pour implémenter les caractéristiques interactives d'un magnétoscope, tel que la pause, l'avance rapide, le retour arrière, etc.

Le protocole RTSP échange des messages RTSP codés en ASCII à travers un protocole de transport tel que TCP ou UDP. Il peut être utilisé en unicast ou en multicast [NTR04] :

Le streaming unicast

Le client contacte le serveur de streaming grâce au protocole RTSP. En réponse à cette requête, le serveur retourne via RTSP une description de la session de streaming qu'il va ouvrir. Une session de streaming est composée d'un ou plusieurs flux (stream), par exemple audio ou vidéo. Le serveur informe le client du nombre de flux. Il donne aussi des informations décrivant les flux comme le type du média et le code de compression. Les flux sont quant à eux diffusés séparément via le protocole RTP.

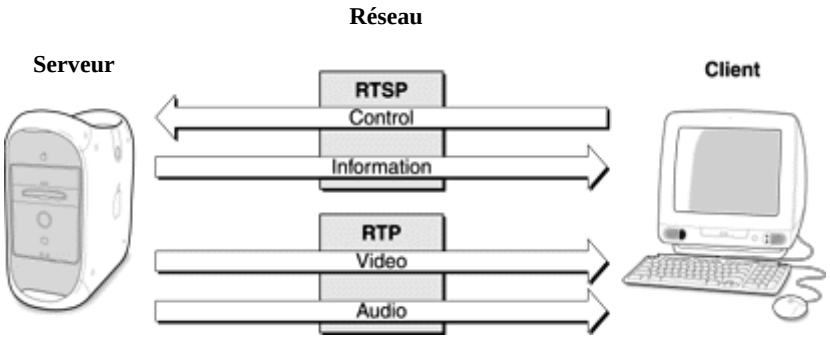

Figure3.2. *Le streaming unicast avec RTSP*

Le streaming multicast

Dans le cas du streaming multicast, une seule copie de chaque flux est envoyée sur chaque branche du réseau, ce qui permet de réduire considérablement le trafic lors d'une diffusion pour de nombreux clients.

Une diffusion multicast est annoncée par un ficher SDP (Session Description Protocol) qui est téléchargé à partir d'un serveur web classique (Apache, IIS,...). Ce fichier contient les informations nécessaires pour recevoir le flux multicast, adresse IP du serveur, numéro du

port et les informations de description des flux (mêmes informations que celle envoyées par RTSP dans le cas d'une diffusion unicast).

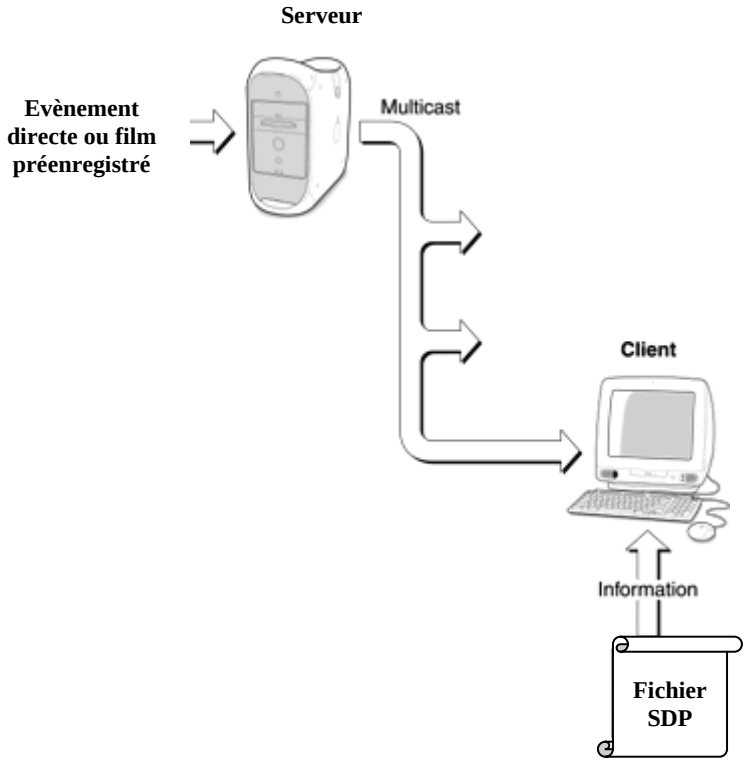

Figure3.3. *Le streaming Multicast avec RTSP*

Actuellement, tous les routeurs ne supportent pas le multicast [NTR04]. Afin de permettre aux clients situés derrière ces routeurs d'accéder aux données multicast, il est possible d'installer un serveur de streaming qui va agir comme une passerelle entre multicast et unicast. Ce serveur est connecté aux flux multicast et sert aux clients qui se connectent à lui en utilisant RTP et RTSP. Cette opération s'effectue en temps réel, ce qui permet de retransmettre aussi bien des vidéos préenregistrées que des images en direct (figure3.4).

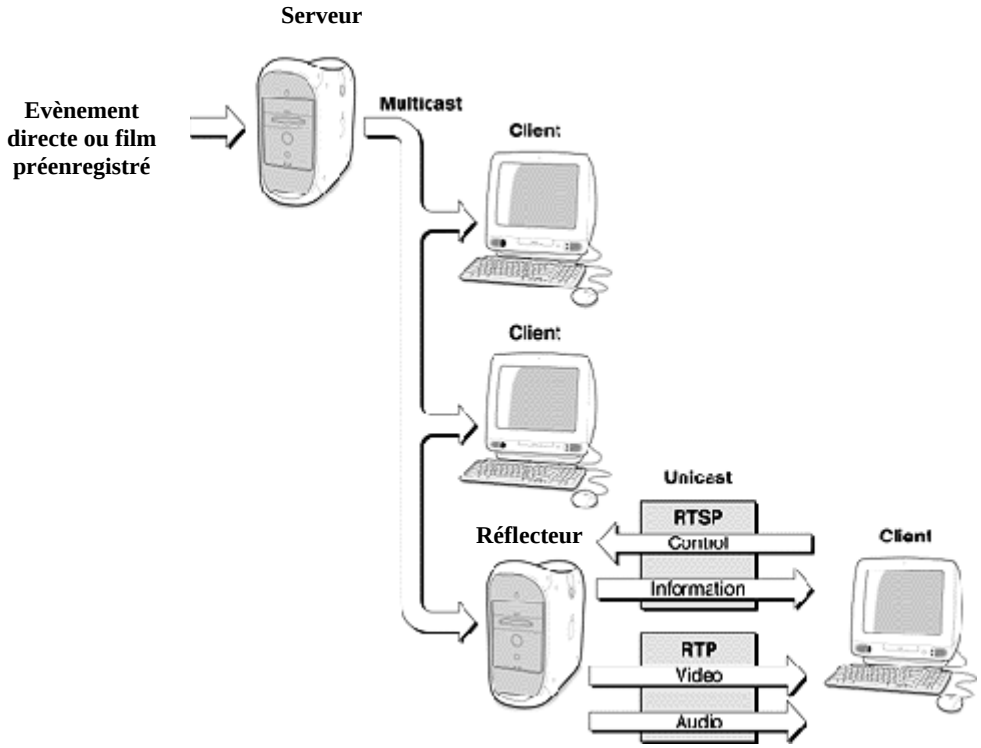

Figure3.4. *Interaction entre multicast et unicast avec RTSP*

III.4.4 Session Initiation Protocol (SIP)

Le SIP, dont l'architecture de base est de type client/serveur, est un protocole de couche application, spécifié par l'IETF dans la RFC3261 en février 1999 [Ros+99], dans le but d'établir en temps réel des appels et des conférences à travers le protocole du réseau Internet. Chaque session de communication peut inclure différents types de données tels que audio, vidéo, fichier, etc. Comme son nom l'indique, SIP est un protocole de signalisation qui permet d'initier, créer, modifier, ou terminer des sessions multimédias avec un ou plusieurs participants, il ne prend en considération qu'une partie du signal de la communication IP, tandis que le flot d'information est pris en charge par différentes ressources (protocole et application). Comme tout protocole de signalisation, SIP permet de localiser un terminal, le contacter pour déterminer sa volonté d'établir une session, échanger des informations sur les médias pour permettre l'établissement d'une session, modifier des sessions media existantes, clore une session média existante, etc. Il est basé sur des messages textuels possédant une en-tête et un corps : alors que l'en-tête définit les paramètres nécessaires au routage du message et à l'établissement de la session, le corps définit les caractéristiques de la session à l'aide d'un protocole de description de session tel que le SDP.

SIP peut être utilisé pour le contrôle de conférences multimédias, d'appels téléphoniques sur IP et bien d'autres types de communications. Les communications peuvent être en unicast ou en multicast. Les participants sont des utilisateurs finaux, des serveurs de media (audio, vidéo…), des serveurs de pure signalisation SIP, ou bien des passerelles vers d'autres réseaux.

L'architecture SIP comprend deux types de composants [CN04] :

Le « User Agent » (Agent utilisateur) peut être un terminal de téléphonie ou de visioconférence sur IP, un serveur audio ou vidéo ou encore une passerelle vers un autre protocole. L'agent utilisateur se décompose en une partie cliente et une partie serveur. La partie cliente, appelée User Agent Client (UAC), envoie les requêtes SIP, et la partie serveur, appelée User Agent Server (UAS), les reçoit. Le principal objectif de SIP est de permettre l'établissement de sessions entre User Agents. Chaque agent a une adresse SIP, représentée par une URL similaire à une adresse e-mail.

Les Serveurs, dans l'architecture de SIP, sont de plusieurs types : les serveurs d'enregistrement, les serveurs de localisation, les serveurs Proxy, et les serveurs de redirection. Chacun des serveurs assure des fonctionnalités spécifiques.

III.4.5 SAP: Session Announcement Protocol (SAP)

C'est un protocole qui a été défini par la RFC 2974 [HPW00], pour annoncer des sessions multicast. Il est particulièrement utilisé avec le protocole SIP pour décrire et définir les caractéristiques de la session. En bref, SAP découvre les sessions multicast et cherche les informations nécessaires pour l'établissement des sessions. Dans le cas d'une session en unicast, ces informations peuvent être connues par les participants. Une fois qu'il détermine ou collecte toutes les informations nécessaires pour l'établissement d'une session, le protocole SIP est employé pour la lancer.

III.4.6 SDP : Session Description Protocol

Le protocole SDP a été proposé à l'origine pour décrire des sessions audio, vidéo, ou multimédias en multicast. Par la suite, il a été également utilisé pour des sessions unicast avec SIP ou RTSP. Le SIP, SAP, et le RTSP utilisent tous le SDP (Session Description Protocol) pour la description des média.

Le message de SDP contient un codage textuel qui décrit la session et plus précisément :

1.) Le protocole de transport des données.

2.) Un champ de type pour distinguer les médias (vidéo, audio, etc..).

3.) Le format de médias (MPEG4, GSM, etc..).

En outre le message de SDP peut contenir la durée de la session, des informations de sécurité (clefs de chiffrage), le nom de session, etc.

III.4.7 SMIL

SMIL est un langage créé par un groupe de travail du W3C (World Wide Web Consortium) nommé SYMM (SYnchronized MultiMedia), dans le but de synchroniser des documents électroniques contenant des objets multimédias (vidéo, son, …). Depuis sa création, plusieurs versions du langage ont vu le jour allant de SMIL 1.0 à SMIL 2.0

La version 2.0 du langage SMIL (Synchronized Multimedia Integration Language) [Lem04] est un modèle de document basé sur le langage XML qui permet la représentation des présentations multimédia interactives. Le langage permet principalement de décrire l'organisation temporelle des objets média, les relations entre les objets utilisés, la disposition spatiale des objets ainsi que les liens qui peuvent être utilisés dans une présentation. D'autre part, la spécification du langage vise à offrir la possibilité de réutiliser la syntaxe et la sémantique de SMIL dans d'autres langages basés sur XML. Les composants temporels et de

synchronisation de SMIL sont ainsi utilisés pour intégrer un aspect temporel tel que c'est le cas dans XHTML [XHT02] et dans SVG [SVG03].

Un document SMIL comporte une structure d'éléments composites appelés opérateurs. Un opérateur porte une certaine sémantique temporelle qui permet de définir le placement temporel des ressources qu'il contient. L'élément 'seq' permet de présenter les ressources médias en séquence. L'élément 'par' permet de présenter les ressources en parallèle. L'élément 'excl' permet de présenter les ressources d'une manière exclusive, c'est-à-dire qu'un seul objet est joué parmi l'ensemble des objets contenus dans l'opérateur.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<smil xmlns="http://www.w3.org/2000/SMIL20/CR/Language">
<head>
       <layout>
              <region id="main" left="10" .../>
       </layout>
</head>
<body>
       <par id="T" dur ="120s">
              <img id="photo" src="Ray_of_Light_CD.jpg" region="main"/>
              <audio id="song" src="Frozen.mp3" begin="id(song)+2s"/>
       </par>
</body>
</smil>
```

Figure3.5. *Exemple d'un document SMIL 2.0*

III.5 La qualité de service et les contraintes de mobilité

La qualité de service a été introduite lors des travaux sur les réseaux [ISO95], il s'agissait de définir pour un service donné les ressources réseaux lui permettant d'être effectivement fourni. Depuis, plusieurs définitions ont été données par plusieurs parties (l'ISO et ITU-T, l'IETF, ou le QdS Forum). En général, la qualité de service peut être caractérisée par différents critères de performance de base qui incluent la disponibilité, le taux d'erreurs, le temps de réponse, le temps d'établissement de connexion, le débit des données, la perte de connexion ou des données à cause de congestions du réseau et la rapidité de détection et de correction de fautes, etc.

Dans le cadre des applications multimédias, où il est nécessaire de gérer des ressources autres que celles liées au réseau, des modèles avec différents niveaux de qualité de service sont définis, allant du niveau utilisateur, formé par un ensemble réduit de choix de qualité (un flux de haute, moyenne, ou basse qualité), au niveau applicatif défini par les paramètres des médias manipulées (exp. Les paramètres d'un flux vidéo : le nombre d'images par seconde, la résolution, la chrominance, etc.), jusqu'au niveau système qui détermine les ressources physiques nécessaires (exp. occupation du processeur) (Figure3.6.).

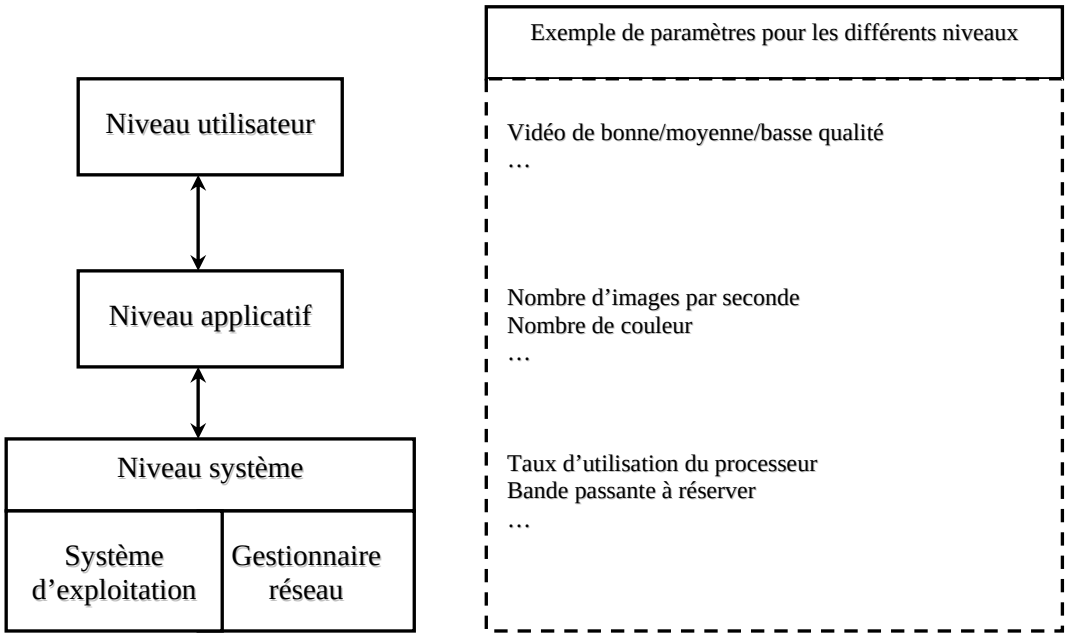

Figure3.6. *Les différents niveaux de qualité de service*

Le respect des contraintes de qualité de service par le système s'effectue selon trois niveaux d'exigence [Dem02] :

Au mieux (*best-effort*) : les ressources ne sont pas réservées et le service n'est fourni que si ces ressources sont disponibles au moment où l'application en a besoin. Cette technique est généralement utilisée dans les applications distribuées sur Internet.

Contraintes déterministes : contrairement au cas précédent, les ressources dans cette technique sont réservées de façon définitive au moment de l'admission de l'application. Une ressource réservée ne peut être utilisée par une autre application, tant qu'elle n'a pas été libérée.

Contraintes probabilistes : dans ce cas les ressources nécessaires doivent être présentes au minimum pour une certaine fraction du temps où le service est demandé. Cependant, il est possible que la qualité de service demandée ne puisse pas être fournie constamment ; le problème est de déterminer si la fraction du temps durant laquelle les contraintes de qualité de service peuvent être respectées est compatible avec les exigences de l'utilisateur.

Un des nouveaux challenges des applications multimédias est de s'ouvrir sur l'utilisation des nouvelles possibilités de mobilité (la mobilité des utilisateurs, la mobilité du matériel, et la mobilité des interfaces) et de connexions sans fil. En effet, les différentes contraintes de qualité de service font face dans ce nouvel environnement à plusieurs facteurs (voir chapitre I). Quelques unes sont des conséquences directes de la mobilité, d'autres liées à l'infrastructure de communication, et même à la portabilité des terminaux mobiles.

Lors de l'exécution d'une application multimédia dans un environnement mobile, on est souvent amené à utiliser des réseaux d'accès sans fil, qui présentent non seulement des capacités réduites en bande passante (WLAN, GPRS, etc.), et des taux d'erreurs plus élevés, mais également des problèmes de connectivité intermittente. Ce qui a un effet direct sur les contraintes de qualité de service de bas niveau qui sont, le taux de perte, la gigue, et les retards. De même, la mobilité provoque des migrations fréquentes et donc des handovers qui provoquent des pertes de paquets et des retards.

La contrainte de portabilité quant à elle, a des effets conséquents sur les contraintes de qualité de service du niveau système (exp. occupation du CPU) et celles du niveau utilisateur (la taille de l'image, la résolution), ceci est dû aux différentes limitations engendrées par la taille réduite des terminaux.

III.6 L'adaptation et les applications multimédias mobiles

Les défis majeurs lors de l'exécution d'une application multimédia dans un environnement mobile résident dans la capacité de calcul limitée pour effectuer l'encodage des données, dans la transmission des données multimédia sur un lien sans fil, et dans les variations rapides et significatives de leur environnement opérationnel auxquels les applications multimédias sont très sensibles.

Au début les chercheurs et développeurs ont essayé d'utiliser les techniques de réservation de ressources pour garantir un certain niveau de qualité de service. Cependant, faire de la réservation de ressource revient à utiliser un protocole de gestion de ressource global, installé sur chaque nœud intermédiaire. Ce qui est peu commode pour un environnement sans fil à connexion intermittente et à fréquentes déconnexions. C'est pourquoi, chercheurs et développeurs, se sont orientés vers une solution qui semble plus adéquate et qui consiste à adapter l'exécution de l'application aux ressources disponibles.

III.6.1 Adaptation au terminal :

1. **Hétérogénéité des terminaux :** une des caractéristique d'un environnement mobile hétérogène est le pouvoir d'exécuter l'application sur une variété de terminaux : des stations fixes conçues pour offrir à l'utilisateur les meilleures possibilités d'utilisation, des ordinateurs portables offrant des possibilités d'utilisation proches de celles des stations fixes mais conditionnées par une autonomie limitée, des ordinateurs de poche, un peu plus puissants que les assistants personnels, leur autonomie est plus au moins bonne par rapport à celles des ordinateurs portables, et les assistants personnels numériques de type Pocket PC et Palm-size PC [Casio, Compaq, HP], dont les fonctionnalités permettent d'implanter les systèmes d'exploitations (Windows CE, Linux). Les informations relatives au type du terminal, ses capacités et limites, sont des données importantes qui doivent être utilisées pour adapter l'application aux conditions spécifiques de chaque terminal.
2. **Ressources limitées :** La portabilité des terminaux impose des propriétés spécifiques, telle que la petite taille du dispositif et le poids léger, ce qui implique des limitations au niveau des ressources et de l'énergie. Pour faire face à cela, les applications devraient tenir compte de ces limitations en étant plus flexibles. De plus et avec tous les développements technologiques, il est à présent possible d'ajouter et d'enlever des périphériques au terminal (un changement de configuration dynamique) tout en exécutant l'application. Ce qui nécessite, que les applications soient conçues de telle manière qu'elles puissent s'adapter à de tels changements, en modifiant leurs dépendances aux ressources selon leur disponibilité ainsi que leurs comportements. Par exemple, adapter l'interface d'affichage à la taille de l'écran utilisé.
3. **Autonomie limitée :** la durée de vie de la batterie est un paramètre limitant incontournable. Actuellement, les mécanismes d'économie de la batterie sont principalement matériels et n'interviennent que lorsque la batterie est très basse ou lorsque le terminal n'est pas utilisé. Avertir les applications du niveau de la batterie pourrait leur permettre d'adopter des modes dégradés adéquats et d'économiser encore un peu plus la

batterie. Une application pourrait, par exemple, passer d'un affichage graphique à un affichage textuel.

III.6.2 Adaptation au support de transmission :

1. **Hétérogénéité des supports :** une deuxième caractéristique de l'environnement mobile est la possibilité d'exécuter l'application n'importe où. Cette contrainte d'espace implique une hétérogénéité au niveau des supports de transmission. Allant des réseaux filaires permettant des communications à haut débit notamment dans des réseaux locaux, aux réseaux sans fil souffrant d'importantes variations dans la bande passante, et qui peuvent également varier, de réseau infrarouge, réseau radio, réseau cellulaire, au satellitaire. L'application doit pouvoir détecter son support de transmission (le type du support, les contraintes et limites, etc.) pour s'adapter. Par exemple, en réduisant ou en augmentant le débit de transmission selon la capacité du lien.
2. **Variations de la bande passante :** lors d'une variation de la bande passante, si on est entrain de télécharger un fichier, le téléchargement peut être interrompu pour quelques secondes sans qu'on s'en rende compte. De même si on fait une navigation sur le Web, la page pourrait s'afficher lentement sans que cela n'ait aucun effet sur l'application. Cependant, pour une application multimédia la dégradation de la bande passante peut avoir des conséquences graves sur l'exécution de l'application. C'est pourquoi avertir les applications de telles variations pourrait leur permettre d'adapter la qualité de service en conséquence. Une application de visioconférence pourrait, par exemple lors d'une variation de la bande passante, décider de réduire le nombre de couleurs ou la taille de l'image.
3. **Déconnexion :** les déconnexions peuvent totalement paralyser une application multimédia ayant besoin d'un accès distant. L'application peut prévoir une éventuelle déconnexion et adapter son exécution (pré chargement, basculer d'un serveur à un autre, etc.)

III.6.3 Adaptation à l'environnement :

Pouvoir adapter une application multimédia aux deux principales contraintes de l'environnement, qui sont la localisation et les ressources disponibles, peut avoir une grande influence sur l'exécution de l'application et donc sa qualité de service.

1. **L'information de localisation :** c'est un renseignement qui peut être utilisé par l'application pour adapter son exécution et améliorer sa qualité de service. Par exemple, dans le cas d'une application de visioconférence lancée en même temps sur plusieurs serveurs, l'application peut basculer son exécution d'un serveur à un autre plus proche pendant le déplacement de l'utilisateur. Cette information peut également être utilisée par d'autres applications pour améliorer leurs services. Exemple, une application de télésurveillance où nous avons plusieurs agents, et qui utilise cette donnée pour ne fournir à l'agent que les informations de surveillance, relatives à sa localisation.
2. **Les ressources de l'environnement :** De même, les terminaux doivent pouvoir profiter des ressources offertes par leur environnement local. Si par exemple une application de visioconférence détecte la présence d'un afficheur plus grand, elle pourra adapter son exécution et basculer l'affichage de la vidéo sur ce dernier, améliorant ainsi la qualité de service.

III.7 Aperçu sur les travaux liés au domaine :

Durant ces dernières années, l'adaptation des applications multimédias en fonction des ressources disponibles, pour augmenter le niveau de qualité de service, a fait l'objet de nombreux travaux menés par plusieurs équipes de recherche. Dans cette section nous allons donner un bref aperçu sur ces différents travaux, allant des techniques d'adaptation au niveau de la couche application, aux systèmes implantant ces techniques.

III.7.1 Quelques techniques d'adaptation au niveau de la couche application

Les techniques d'adaptation du flux multimédia peuvent être introduites sur les différentes couches de la pile protocolaire : au niveau de la couche physique en introduisant des techniques de contrôle adaptatives pour atténuer les variations du lien sans fil, au niveau de la couche liaison en utilisant des techniques de réservation et de contrôle d'erreur pour faire face aux variations du taux d'erreur, au niveau de la couche réseau en implémentant des mécanismes de routage dynamiques qui permettent d'éviter les congestions et atténuer les variations de l'environnement mobile, au niveau de la couche transport via une négociation et renégociation dynamique des paramètres de connexions, et finalement et principalement au niveau de la couche application où plusieurs techniques peuvent être utilisées pour adapter l'application multimédia aux conditions d'exécution :

1. **Création de versions multiples :** cette technique consiste tout simplement à créer des versions multiples du même document, chacune caractérisée par un certain débit et une certaine qualité.
2. **Le codage en couche :** dans un codage en couche, l'information vidéo est codée en plusieurs couches, dont la première couche est la plus importante (couche de base) et doit être présente pour que les autres couches puissent être décodées. Les couches supérieures, appelées couches d'amélioration, sont utilisées pour améliorer la qualité de la vidéo de façon progressive, chacune d'elles correspond à un incrément de qualité de service et exige l'existence des couches inférieures ; nous pouvons ainsi dire que l'encodeur assigne aux différentes couches, un ordre de priorité décroissant, tel que la couche de bas niveau ait la priorité la plus élevée.

   Le premier schéma utilisant le codage en couche pour une transmission contrôlée a été proposé en 1996 [MJV96], depuis plusieurs travaux ont été faits dans ce sens [LP+97] [LPA99].
3. **Le taux du trafic (Rate shaping) :** Ce sont des techniques réactives qui ajustent le taux de trafic généré par l'encodeur selon les conditions réseaux, en utilisant les paramètres de compression (ex. le taux de frame) [BT94] [BG96]. Ces techniques utilisent des mécanismes de rétroaction pour détecter les changements dans le réseau et contrôler le taux de l'encodeur.
4. **Le contrôle d'erreur :** le taux d'erreur et le taux de perte sont variables dans un environnement mobile à lien sans fil. Pour atténuer les taux de perte et d'erreurs et adapter une application multimédia à ces variations, deux approches ont été proposées [VFJ+99]. L'ARQ (Automatic Repeat Request) basée sur un mécanisme réactif dans lequel c'est au client de demander la retransmission des paquets perdus, ce qui est peu commode aux réseaux mobiles à lien sans fil. Et la FEC (Forward Error Correction), une approche passive dans laquelle la source envoie des informations redondantes pour recouvrir les paquets perdus. Bolot et Turletti [BT98], ont proposé un schéma adaptatif à base de l'approche passive (FEC).

III.7.2 Classification des systèmes d'adaptation

Les différents systèmes ou solutions proposés pour permettre aux applications multimédias de s'adapter aux ressources disponibles, peuvent être classées en trois catégories ou classes selon la partie (Serveur, client, ou Proxy) responsable de l'adaptation :

1. **Adaptation côté serveur :** dans la littérature nous trouvons plusieurs travaux (ex. [TH96] [IC+97]), qui attribuent au serveur la responsabilité sur la qualité de service et donc la charge d'adapter l'exécution de l'application en conséquence. Cependant, le serveur peut répondre aux dégradations de la qualité de service tout simplement en adaptant le débit de transmission. Pour cela, il se base sur des mécanismes de rétroaction et des techniques qui permettent de mesurer les paramètres réseaux (l'occupation des tampons, le taux d'erreurs, les variations dans les délais, etc.) [SS98] [SWW94] [KMR93].
2. **Adaptation côté client :** pour alléger le serveur des différents traitements de contrôle qui augmentent considérablement sa charge, des solutions dont le processus d'adaptation est sous la responsabilité des client ont été proposées [NS+97] [NS99]. Ces solutions ou approches, se basent essentiellement sur des mécanismes d'adaptation liés aux techniques de codage et de transmission employées (codage en couche, envoi de versions multiples). Le client dans ce cas n'aura qu'à choisir entre les différentes versions et les différentes qualités, celles qui s'adaptent à son état et à ses ressources.
3. **Adaptation à base de Proxy :** pour pallier aux problèmes d'hétérogénéité des clients et des serveurs, une troisième classe de solutions, qui se base sur l'ajout d'une troisième entité entre le serveur et le client appelée Proxy a vu le jour. Les solutions de cette classe placent la responsabilité d'adaptation principalement sur le Proxy, qui reçoit le contenu à la place du client, lui applique un certain traitement avant de le transmettre au client cible. Cette façon de procéder permet de garder les caractéristiques déjà existantes de l'environnement et pus précisément des entités essentielles de l'application (le client et le serveur).

   La majorité des solutions proposées dans cette classe visent à diminuer le débit de transmission sur les liens sans fil. C'est pourquoi elles proposent comme mécanisme d'adaptation la technique de transcodage qui consiste à transformer les données multimédias d'un format d'encodage en entrée à un autre format d'encodage en sortie. Parmi ces solutions, citons celle proposée dans [AMZ95] basée sur une technique de transcodage qui change automatiquement le format d'encodage d'une vidéo de MPEG à H.261, ce qui permet de réduire le débit de transmission. D'autres solutions, et toujours dans l'optique de réduire le débit de transmission, se basent sur un format d'encodage spécifique et proposent des techniques liées directement à ce format [LG02].

   Hagimont et Layaida [LH02] quand à eux proposent une solution qui se base essentiellement sur des techniques de transcodage d'un ensemble de format en entrée {H.261, H.263, Mpeg1, etc.} à un ensemble de format en sortie {H.261, H.263, Mpeg1, etc.}. Après décodage, le flux multimédia intermédiaire représenté dans un format brut {RGB24, YUY2, etc.} pourra subir trois types de transformation : redimentionnement de la taille, réduction du nombre de couleurs, et intégration de nouveaux services (ex. un logos ou un spot publicitaire).

   NAC [Lem04] est une architecture à base de Proxy proposée dans le but d'adapter le contenu multimédia sur le Web. Cette solution consiste en un schéma de présentation de profil, où toute entité du système aura un profil bien défini, un protocole de négociation de

contenu, et une architecture d'adaptation de contenu, basée essentiellement sur les techniques de transformation de contenu.

III.8 Conclusion :

L'évolution rapide dans les réseaux de communication (WPAN, WALN, et WWAN) et dans la fabrication des calculateurs ultralégers offre aujourd'hui de nouvelles possibilités d'utilisation qui étaient jusqu'ici inimaginables. Ceci a conduit à la demande croissante pour les applications multimédias et au désir d'utiliser ces applications dans un environnement mobile hétérogène présentant de nouveaux défis technologiques pour une bonne exécution de ces applications.

Les recherches dans le domaine des applications multimédias mobiles et des problèmes engendrés par ce nouvel environnement sont nombreuses, aussi bien chez les opérateurs télécoms que chez les industriels de l'informatique, et même aujourd'hui dans les entreprises traitant du multimédia. Chacun de ces acteurs possède une vision et une démarche propre de la recherche pour offrir une bonne gestion de qualité de service. Pendant l'étude des différentes solutions, nous avons remarqué que chaque solution s'est intéressée à un certain nombre de problèmes en dépit du reste.

Dans notre travail, nous nous focalisons sur un type particulier d'applications multimédias, qui est la vidéo à la demande, et plus précisément une application de vidéo formation. Nous nous intéressons principalement à l'adaptation de l'exécution de l'application aux variations dynamiques de son contexte d'exécution (les capacité du terminal, le type du support, la variation de la bande passante, etc.). Nous allons utiliser l'adaptation à différents niveaux afin de régler des problèmes spécifiques. Dans notre architecture, chaque partie du système doit jouer un rôle pour garantir le bon fonctionnement de l'application.

CHAPITRE 4 :
Conception d'une Architecture d'Adaptation d'une Application de Vidéo à la Demande dans un environnement hétérogène

IV.1 INTRODUCTION ET PROBLEMATIQUE

Les immenses progrès dans les domaines de télécommunication et de fabrications des calculateurs ultralégers (allant jusqu'à 3Go de capacité de stockage) nous permettent actuellement de suivre des cours n'importe où, n'importe quand, et à partir d'une variété de terminaux. Ceci via les applications de vidéo formations mobiles qui nous donnent la possibilité de réaliser ce qui a été, jusqu'à un passé proche, classé comme rêve ou projet futuriste. En effet, l'utilisateur pourra suivre ses cours lorsqu'il se sentira capable d'assimiler au mieux, selon son heure d'étude préférée (matins, soirs, week-ends). Il n'aura plus à arranger son emploi du temps selon les heures prévues pour les cours mais plutôt le contraire, arranger les heures de cours selon le temps libre qu'il a, même s'il est hospitalisé ou en voyage. Nous pouvons ainsi dire que l'utilisateur peut suivre à distance des cours donnés par les meilleurs professeurs dans n'importe quel domaine sans se soucier des contraintes d'espace ou de temps.

Cependant, l'ouverture sur le monde des applications multimédias utilisant les nouvelles possibilités de mobilité (la mobilité des utilisateurs, la mobilité du matériel, et la mobilité des interfaces), et de connexions sans fil, nous oblige à faire face à plusieurs contraintes et défis. Quelques uns sont des conséquences directes de la mobilité, d'autres sont liés à l'infrastructure de communication, et même à la portabilité des terminaux mobiles :

- Hétérogénéité des terminaux.
- Les ressources limitées des calculateurs mobiles ainsi que la capacité réduite de présentation des médias.
- Hétérogénéité des supports de transmission et donc des interfaces réseaux.
- La grande variabilité de la bande passante dans les réseaux sans fil.
- Les déconnexions fréquentes et leurs effets sur la vidéo Streaming.

Toutes ces informations (type de terminal, les ressources disponibles, le type de support, etc.) constituent ce qu'on appelle « le contexte d'exécution d'une application ». Le problème majeur des applications multimédias mobiles est lié aux variations très dynamiques de ce contexte, des variations qui peuvent avoir des effets conséquents sur l'exécution de l'application. Afin de prendre en considération tous ces changements et permettre à l'application de continuer à travailler dans un tel environnement, l'application doit pouvoir détecter les changements et s'adapter de façon dynamique aux conditions d'exécution.

D'où vient l'idée de notre contribution qui consiste en la conception d'une architecture qui adapte l'exécution d'une application de vidéo formation et la qualité du flux aux variations dynamiques dans le contexte utilisateur. L'application est de type client/serveur, dans laquelle la partie cliente est installée sur les terminaux utilisateurs, et les cours sont stockés et répliqués sur plusieurs serveurs. Nous supposons que les étudiants et les professeurs sont déjà inscrits au service. L'architecture vise à garantir une bonne exécution de

l'application (assurer la continuité du service tout au long d'une session) et donc un bon fonctionnement du service en dépit des différents problèmes qui peuvent se poser dans un environnement mobile hétérogène. Pour cela nous définissons les principaux composants nécessaires pour garantir les fonctionnalités de base d'une telle architecture et d'un tel service.

Dans cette partie, nous présentons l'application considérée et l'architecture proposée, tout en justifiant le choix d'une telle architecture. Après une description générale, nous donnerons deux vues d'ensemble de l'architecture, une vue selon les principales fonctionnalités et une vue en termes de composants tout en mettant l'accent sur nos principaux apports :

IV.2 ARCHITECTURE GENERALE

Notre architecture [BNB05] a été proposée dans le but d'offrir une adaptation dynamique de l'exécution d'une application de vidéo formation en fonction des différentes variations dans son environnement d'exécution.

L'application qu'on considère est basée sur le modèle client/serveur, pour cela nous proposons une architecture à base de Proxy. Le Proxy est placé dans un emplacement approprié sur la partie fixe du réseau. Les clients se connectent aux serveurs à travers le Proxy, qui reçoit les données de la formation, effectue une adaptation selon les différentes valeurs de contextes détectées avant de les acheminer vers les clients. Le choix d'une architecture à base de Proxy est motivé par le fait que nous travaillons dans un environnement hétérogène, dans lequel l'utilisateur peut se connecter en utilisant différents types de terminaux, pas forcément aussi puissants les uns que les autres, et à travers des réseaux de faible ou de bonne bande passante. Car l'exécution de tous les traitements sur le client peut revenir cher à un terminal de capacité réduite, et consommer beaucoup d'énergie et de bande passante. L'utilisation d'un Proxy nous permet d'améliorer les performances de notre système :

1. Réduire la consommation des ressources du terminal tel que le processeur, car une part des traitements s'effectue sur le Proxy, ce qui permet d'économiser de l'énergie.
2. Réduire le trafic sur la partie sans fil du réseau et donc la consommation de la bande passante.
3. Améliorer les performances du serveur et éviter sa surcharge, en lui masquant les problèmes introduits par la mobilité, et en le laissant dédié à sa tâche principale et classique qui est l'accès au contenu de la formation.

Comme le montre la figure ci-dessus (Figure 4.1.), l'architecture est constituée des trois parties suivantes :

1. **Le terminal utilisateur :** peut être de différents types et muni de plusieurs moyens de communication. Ces interfaces de communication lui permettent de se connecter au Proxy. Il comporte la partie cliente de l'application qui permet à l'utilisateur de s'enregistrer au service dans une phase initiale et puis d'accéder aux cours après une procédure d'authentification et une ouverture de session.
2. **Le serveur :** comporte la partie serveur de l'application, il se trouve toujours dans la partie fixe du réseau, et c'est à son niveau que les fichiers multimédias sont stockés. Seul l'administrateur a le droit de modifier son contenu en ajoutant ou en supprimant des cours.

3. **Le Proxy** qui représente le noyau de l'architecture : c'est une partie intermédiaire entre le client et le serveur, installé dans la partie fixe du réseau ; sa connexion avec le serveur est établie via un lien filaire stable, et celle avec le client à travers différents types de liens beaucoup moins sûrs. Le Proxy comporte les différents modules essentiels pour garantir le bon fonctionnement de notre architecture et les modules nécessaires pour la communication réseau avec les deux autres parties (le client et le serveur). C'est à son niveau que la plupart des traitements d'adaptation sont effectués.

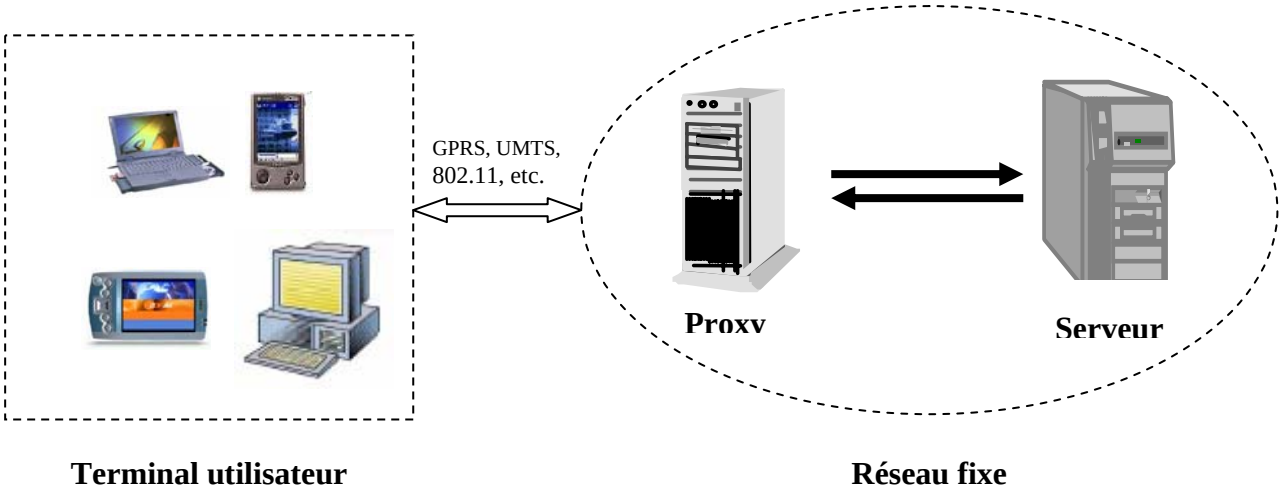

Figure 4.1. *Architecture générale du système*

IV.2.1 Les différentes étapes de l'application

L'application de vidéo formation que nous considérons, est une application client serveur, nous supposons que des étudiants et des professeurs sont inscrits au service. Les étudiants peuvent ainsi consulter des cours suivant la technique du Streaming et passer des tests d'évaluation en téléchargeant les fiches correspondantes. Les cours sont stockés sur le serveur, et sont transmis dans un format codé compressé et mis en marche sur le terminal utilisateur suite à une requête de ce dernier.

IV.2.1.1 La phase d'inscription

Nous avons défini différentes classes utilisateurs, chacune engendrant un certain nombre de critères et de contraintes, qui vont être utilisés par la suite par le processus d'adaptation :

1. La classe 1 (C1) : c'est la classe des étudiants aveugles.
2. La classe 2 (C2) : c'est la classe des étudiants sourds.
3. La classe 3 (C3) : c'est la classe des utilisateurs « normaux ».

Pour décider de la classe de chaque utilisateur, nous utilisons une classification paramétrée ; ie. selon les paramètres saisis par l'utilisateur et liés principalement à ses préférences, nous décidons de sa classe.

Nous définissons également et pour chaque classe, deux autres sous classes, PR et NP, qui dépendent de l'état de l'utilisateur au moment de sa connexion. PR montre que l'utilisateur est pressé (ex. la veille d'un examen) et donc peut permettre toutes les dégradations possibles du service en dépit de sa continuité. Et NP indique que l'utilisateur n'est pas vraiment pressé et donc il peut tolérer une qualité plus ou moins moyenne.

IV.2.1.2 Ouverture de session

Au début de chaque session l'utilisateur doit passer par une phase d'authentification après laquelle et si l'accès est permis, l'utilisateur doit choisir entre ses deux sous classes (PR, NP). C'est également pendant cette étape que les différentes caractéristiques du terminal client sont détectées et transférées.

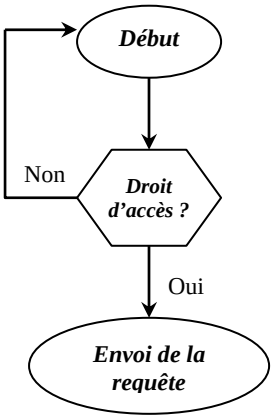

Figure 4.2. *Ouverture d'une session*

IV.2.1.3 Interaction au cours d'une session

Pendant la visualisation du cours, les ressources au niveau du terminal et celles du réseau peuvent changer. Dans ce cas, les composants responsables de la détection des variations du contexte, notifient le composant responsable de l'adaptation du flux multimédia en conséquence (Figure 4.3.).

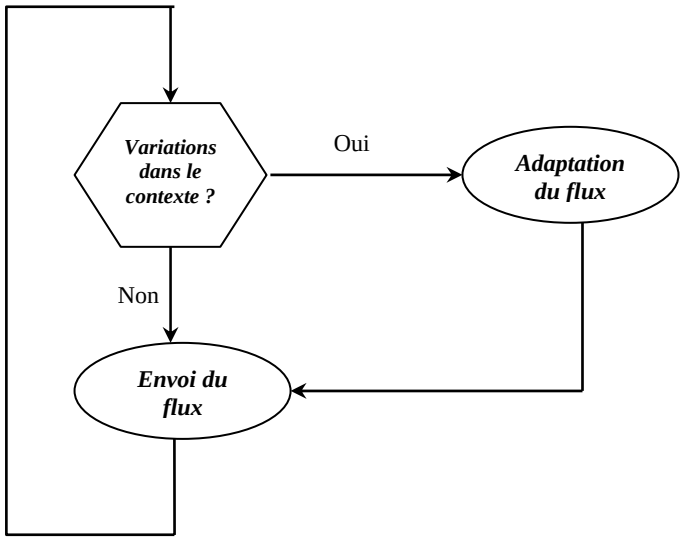

Figure 4.3. *Adaptation du flux multimédia*

IV.3 LES PRINCIPALES FONCTIONNALITES DE L'ARCHITECTURE

Pour garantir l'objectif de notre architecture qui est le bon fonctionnement du service dans les différentes contraintes, nous définissons quatre fonctionnalités de base qui vont être assurées par les différents composants de l'architecture : détection et gestion du contexte, adaptation de l'exécution de l'application au contexte détecté, gestion de la session pendant le déplacement de l'utilisateur, et sécurité du service.

IV.3.1 Détection et gestion du contexte :

Pour un processus d'adaptation, la fonctionnalité de détection et de gestion du contexte est très importante. Dans cette partie, nous allons définir les types de contexte que nous prenons en considération, la façon dont on l'extrait, et sa représentation.

Le mot « contexte » est un mot très vaste qui peut avoir plusieurs sens et peut être utilisé de plusieurs façons dans différents domaines [ST94] [DAW99] [CK01] [Dey01][]. Dans notre cas nous nous intéressons à un ensemble restreint qui englobe les paramètres suivants :

- **Le profil utilisateur** (User awareness) qui représente l'identité de l'utilisateur et ses préférences: Pendant la phase d'inscription, l'utilisateur doit saisir ses informations personnelles (nom, prénom, mot de passe, etc.) et ses préférences (ex. la langue). Toutes ces informations en plus de la classe à laquelle il sera affecté, seront regroupées dans un Profil utilisateur. Chaque classe englobe un certain nombre de caractéristiques et de contraintes. Par exemple pour la classe C1 les utilisateurs sont aveugles et donc le flux de type audio prend la priorité pendant le processus d'adaptation. Les profils utilisateurs de base, sont transférés à la fin de la phase d'inscription pour être stockés sur une base de données profil au niveau du Proxy.
- **Le profil du terminal** (Device awareness) qui représente les caractéristiques générales du client. Il inclut les capacités physiques du terminal (la capacité d'affichage, les capacités de calcul, l'énergie maximale), et les différents formats supportés (ex. pas de sortie pour l'audio). Ce profil est envoyé par le terminal durant la session, afin de laisser l'environnement ouvert à une large diversité d'appareils et pour supporter le changement dynamique des profils (le niveau d'énergie, la capacité courante du CPU).

- **Le profil du réseau** (network awareness) : représente le type de réseau (WLAN, Ethernet, …etc.), la bande passante maximale supportée, la bande passante disponible, ….
- **La localisation** (location awareness), **l'orientation** et **le temps**.

Selon le type de contexte, le moment de son exploitation et la façon d'interagir avec lui, nous avons défini trois catégories suivant lesquelles les différentes valeurs de contexte seront classées :

1. **Le contexte statique** qui inclut le profil utilisateur (il y'a une probabilité faible pour qu'il ait à changer dans le temps) : les valeurs de contexte de cette catégorie sont saisies une fois pour toute et stockées au niveau d'un dépôt de profil. Et puis lors de l'ouverture d'une session et après une phase d'authentification de l'utilisateur, le profil correspondant est téléchargé au niveau du Proxy pour être utilisé.
2. **Le contexte semi dynamique :** les valeurs de contexte de cette classe sont extraites au début de chaque session et resteront stables durant une même session. Il contient quelques valeurs du profil du terminal tel que la résolution, les capacités d'affichage, …etc.
3. **Le contexte dynamique :** les valeurs de cette catégorie (vitesse du réseau, niveau de l'énergie, etc.) sont évaluées en temps réels et dès qu'un changement est détecté le processus d'adaptation est notifié.

Sachant que les environnements hétérogènes présentent beaucoup de limitations, que ce soit sur le plan réseau de communication ou terminaux utilisés, la gestion (le stockage, le traitement, la transmission..) des descriptions doit être optimisée afin d'éviter l'augmentation de délais de réponse. Ce besoin d'optimisation représente un vrai problème lorsqu'on veut joindre un bon niveau de détail des descriptions à un transfert rapide et un traitement minimal de ces descriptions.

Notons que dans notre travail, nous n'avons pas essayé de définir un nouveau modèle de description de contexte, mais juste utiliser le modèle qui satisfait au plus les besoins de notre architecture. C'est pourquoi nous avons opté pour le CC\PP (Composite Capabilities \ Preference Profiles) [S5], grâce auquel nos différents profils resteront extensibles et supporteront l'ajout de nouvelles dimensions de contexte.

IV.3.2 Adaptation du flux multimédia :

L'adaptation de l'application aux variations de contexte représente la fonctionnalité cœur de notre architecture, son objectif principal est d'exploiter au maximum et d'une façon intelligente les différentes ressources disponibles.

Le processus d'adaptation sera utilisé dans notre architecture à plusieurs niveaux et à chaque fois pour résoudre un problème spécifique :

1. Au début de la session pour adapter la requête reçue du client suivant les préférences utilisateur et surtout sa langue.
2. Au cours de la session pour gérer les déconnexions (voir la section suivante).
3. Et pendant l'envoi des données pour adapter le flux multimédia reçu du serveur selon les différentes valeurs de contexte (bande passante, occupation du processeur, etc.) et ceci pour satisfaire au mieux les besoins de l'utilisateur.

Gestion adaptative de la qualité de service :

IV.3.2.1 Aperçu sur la solution

Satisfaire une requête de l'utilisateur revient à satisfaire toutes les contraintes qu'elle peut comporter, c'est pourquoi nous avons opté pour une adaptation qui se fait suivant trois principaux axes. Premièrement, adaptation au profil utilisateur et donc à ces préférences, deuxièmement, adaptation aux caractéristiques du terminal qui consiste à adapter les flux selon les capacités de traitement et d'affichage des terminaux, le format d'encodage supporté, ou le niveau d'énergie. Et finalement, adaptation au support de transmission, qui prend en compte l'hétérogénéité des supports de transmission, les variations de la bande passante, et les déconnexions possibles et pour principalement réduire le volume de données émis sur le réseau lorsque le débit du réseau baisse.

Selon la catégorie de contexte, nous avons défini deux étapes d'adaptation : une première, exécutée au moment de la connexion au service, elle prend en considération les valeurs de contexte statiques et semi dynamiques (le profil utilisateur, la capacité du terminal, le type de réseau, etc.) (Figure 4.5.). Et une deuxième, en cours d'exécution, activée à la suite d'une variation dans le contexte dynamique (le niveau d'énergie, les paramètres de connexion, les variations dans la bande passante, le changement de l'interface de connexion, etc.). Lors de l'ouverture d'une session, les capacités du terminal et l'identité de l'utilisateur sont envoyés au Proxy qui télécharge le profil utilisateur correspondant et ouvre avec le serveur une session adaptée aux préférences de ce dernier (ex. la langue, la sous classe) (Figure 4.5.).

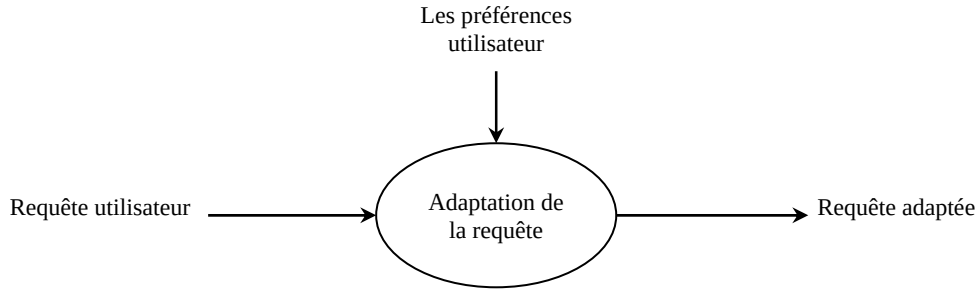

Figure 4.5. *Adaptation de la requête*

Après cela, le Proxy reçoit en provenance du serveur le flux vidéo dans la langue spécifiée, lui applique les politiques d'adaptation correspondantes aux différentes valeurs de contexte, et l'achemine vers le client (Figure 4.6.).

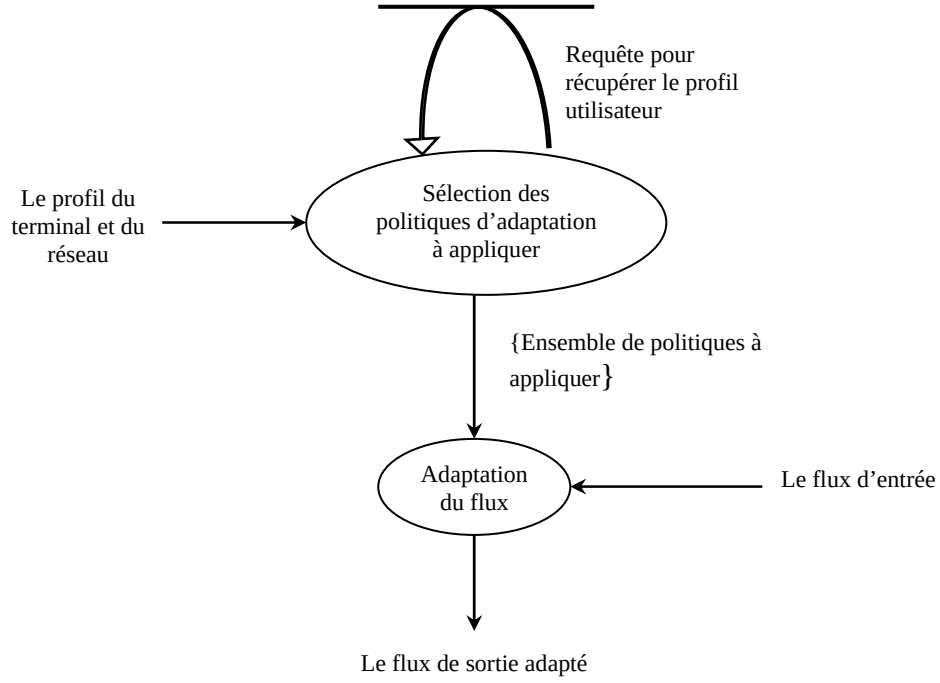

Figure 4.6. *Adaptation initiale du flux*

Par la suite (au cours de l'exécution) dès qu'une variation dans le contexte est détectée, l'exécution est adaptée en conséquence et selon la variation notifiée (Figure 4.7.).

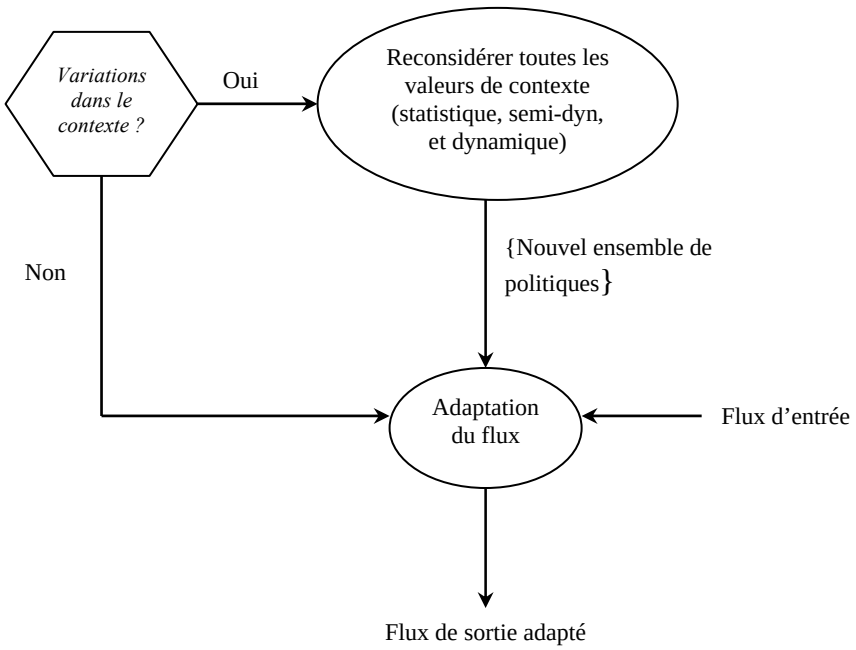

Figure 4.7. *Adaptation en cours d'exécution*

Comme l'objectif de notre processus d'adaptation est de gérer la qualité de service perçue par l'utilisateur, et ceci selon les ressources disponibles, nous avons commencé par

faire une classification des différentes contraintes ou paramètres de QdS qui vont être utilisés par la suite par notre processus d'adaptation :

- Les paramètres de niveau utilisateur, représentés par la qualité de service telle que vue par l'utilisateur (Vidéo de bonne \ moyenne \ ou mauvaise qualité, Son de bonne qualité, etc.).
- Les paramètres de niveau applicatif, tels que le nombre d'images par seconde, la chrominance, le nombre de couleurs, etc.
- Et finalement, les paramètres de niveau système (le nombre de paquets perdus, l'occupation du CPU, etc.).

Notons que pour arriver à un bon niveau de qualité de service au niveau utilisateur, les différents paramètres des autres niveaux doivent être pris en compte et contrôlés. C'est pourquoi une mise en correspondance entre les paramètres des différents niveaux doit être faite. Par exemple une bonne qualité d'image correspond à un ensemble de paramètres de niveau applicatif tel que le nombre d'images par seconde, le nombre de couleur, la chrominance, etc.

Nous avons par la suite, établi une correspondance entre les paramètres de qualité de service et les attributs de contexte pris en compte. Cette mise en correspondance a donné naissance à deux classes d'attributs de contexte :

1. **Les attributs liés à l'utilisateur :** représentent les attributs du profil utilisateurs (la classe, l'identité, les préférences, etc.) et sont utilisés principalement pour définir un ordre de priorité entre les paramètre de QdS de niveau utilisateur (Par exemple, pour un utilisateur de la classe C1, les paramètres liés au son et à la qualité du son ont une priorité maximale). Cet ordre de priorité est, à un certain degré, fixe pour un même utilisateur excepté certaines modifications qui peuvent être appliquées au début de chaque session, selon la sous-classe choisie.
2. **Les attributs liés aux ressources appelés également paramètres d'adaptation :** ces attributs englobent ceux du profil du terminal et ceux du réseau (taille de l'écran, CPU). Nous les avons également classé en deux catégories selon la façon dont ils varient : les attributs statiques durant le processus d'adaptation (les paramètres de contexte semi dynamiques tels que la capacité d'affichage) pris comme des restrictions sur la qualité de service, et les attributs dynamiques durant le processus (les paramètres de contexte dynamiques), et qui sont pour la plupart liés aux contraintes de qualité de service niveau système.

Nous pouvons ainsi résumer notre processus d'adaptation par :

Début

{

Déclaration :

Définir (Ensp{})

{

1. Un contrôle dynamique des contraintes de QdS de niveau bas (ex. taux de perte) et des paramètres de contexte liés aux ressources et qui ont un effet direct sur ses contraintes (la capacité du CPU, l'énergie, etc.)
2. si détection d'une variation alors notification en temps réel pour définir un nouvel ensemble de politiques ensp{} qui ajustent les contraintes de niveau applicatif. Et ceci pour un objectif bien défini qui est satisfaire au mieux et dans la mesure du possible les contraintes de QdS niveau utilisateur.

}

Adapter (flux, ensp{})

{

Appliquer les politiques de l'ensemble ensp{} sur le flux d'entrée.

}

Le corps

{

Début

1. Etape initiale (au début du Processus) :
   - Définir l'ordre de priorité entre les paramètres de QdS niveau utilisateur.
   - Prise en compte de tous les attributs liés aux ressources.
   - Définir toutes les modifications qui doivent se faire sur le flux multimédia au niveau applicatif. Ensp{} := l'ensemble initial des politiques à appliquer.

2. forbegin

{

Tant que non fin de la présentation

{

Adapter (flux, ensp{})
Envoi du flux adapté

}

Tant que non fin de la session

{

Définir (ensp{})

}

Forend

}

Fin

}

}

IV.3.2.2 Les politiques d'adaptation

Nous définissons pour chaque paramètre d'adaptation, les paramètres d'exécution adaptables ou modifiables qui peuvent avoir un effet sur la ressource déclenchant l'adaptation, ou qui exploitent aux mieux les ressources disponibles. Par exemple, si nous prenons la bande passante disponible comme paramètre d'adaptation, nous devons sélectionner tous les paramètres d'exécution qui ont un effet sur la consommation de la bande (le nombre d'images par seconde, le taux d'envoi, etc.), et donc les adapter pour réduire ou augmenter la consommation de la bande passante.

Nous représentons chacun de ces paramètres d'exécution par un axe d'adaptation. Ainsi pour chaque paramètre d'adaptation un ensemble d'axes et de politiques pouvant être appliqués sera défini.

Nous définissons également, et pour chaque type de données, des degrés de fidélité selon les axes d'adaptation utilisés.

Exemple (pour la vidéo) :

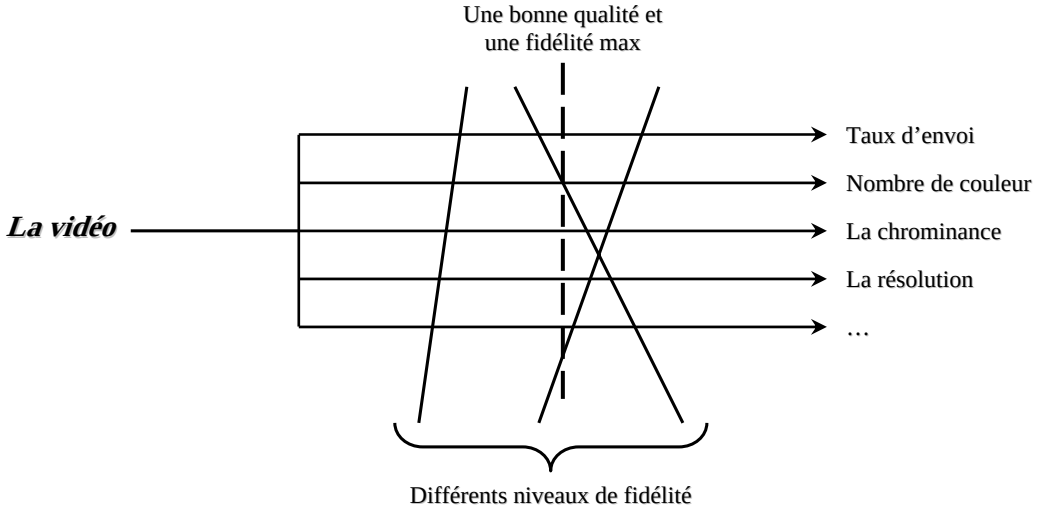

Figure 4.8. *La notion de fidélité des données*

Adapter un flux multimédia revient à considérer donc tous ces axes et toutes les politiques proposées pour un ensemble de paramètres ou de contraintes (Tableau4.1) :

1. Les contraintes du client : peuvent se résumer principalement en :
   a. Des contraintes spatiales : elles sont liées à la taille de l'écran et l'espace d'affichage. Pour ce genre de contraintes, des politiques qui jouent sur les dimensions spatiales doivent être appliquées. Par exemple : réduire la taille de l'image, ou partager les régions d'affichage entre plusieurs objets en introduisant une dimension temporelle.
   b. Des contraintes de capacité : lorsque le client reçoit une vidéo, il doit la traiter pour pouvoir l'afficher. Si ce traitement ne s'effectue pas en temps réel à cause d'une surcharge du CPU, les images sont ignorées. Ce qui signifie que les ressources réseaux et plus précisément la bande passante a été mal exploitée puisque les paquets sont ignorés après leur arrivée au poste client. C'est

pourquoi des politiques qui réduisent le nombre de paquets doivent être appliquées dans ce cas (réduire la qualité de l'image, jouer sur le type de flux, etc.).

c. Et l'énergie : pour décider des politiques à exécuter dans le cas d'une faiblesse de la batterie. Nous devons penser à réduire la charge de tout composant qui a un effet sur la consommation de l'énergie (ex. le CPU).

2. Les contraintes de l'utilisateur : Pour chaque classe d'utilisateur, nous définissons l'ensemble de méthodes qui peuvent être appliquées et celles qui ne peuvent pas l'être. Par exemple : pour un utilisateur de classe C1, le flux audio doit être de bonne qualité et nous ne devons pas lui envoyer du texte à la place.
3. Les contraintes réseau : la principale contrainte réseau consiste en la limitation de la bande passante. Dans ce cas plusieurs politiques qui réduisent le volume des données transmises peuvent être proposées. Exemple : réduire la qualité de l'image (nombre d'images par seconde, le nombre de couleurs, etc.), ou se limiter à l'envoi du texte ou du son.

<table>
<tr><th rowspan="2">Paramètres d'adaptation :</th><th colspan="3">Axes d'adaptation :</th></tr>
<tr><th>Classe1</th><th>Classe2</th><th>Classe3</th></tr>
<tr><td>Tailles de l'écran</td><td>X</td><td colspan="2">▪ Résolution
▪ Taille de l'image
▪ Partager la région d'affichage
▪ Etc.</td></tr>
<tr><td>CPU</td><td rowspan="2">▪ Algorithme de compression du flux audio
▪ La technique de préchargement
▪ Etc.</td><td>▪ Nbr d'images/seconde
▪ Algorithme de compression du flux vidéo
▪ Nbr de couleur
▪ La technique de préchargement
▪ Etc.</td><td>▪ Nbr d'images/seconde
▪ Algorithme de compression
▪ Nbr de couleur
▪ La technique de préchargement
▪ Le type de flux envoyé
▪ Etc.</td></tr>
<tr><td>Energie</td></tr>
<tr><td>Bande passante</td><td>▪ Algorithme de compression
▪ La technique de préchargement
▪ Etc.</td><td>▪ Algorithme de compression
▪ La technique de préchargement
▪ Nbr d'images/seconde
▪ Etc.</td><td>▪ Algorithme de compression
▪ La technique de préchargement
▪ Nbr d'images/seconde
▪ Le type de flux envoyé
▪ Etc.</td></tr>
<tr><td>Interface de connexion</td><td colspan="3">▪ Le lieu d'exécution
+ (les techniques liées à la bande passante)</td></tr>
</table>

Tableau4.1. *Echantillon sur les principaux paramètres et axes d'adaptation*

Pour la compression des flux vidéo, nous nous basons principalement sur les protocoles MPEG4/AVC10 en raison de la possibilité de disposer de formats liés hiérarchiquement. Ceci nous semble un atout dans notre travail d'adaptation en fonction de la disponibilité des ressources. L'originalité de ce protocole est qu'il permet de définir plusieurs qualités pour un même flux. Le flux se décompose en trois couches complémentaires (décomposition hiérarchique) : la première est de qualité minimale, l'ajout de la deuxième

permet d'améliorer la qualité, et enfin la troisième couche permet d'obtenir une qualité optimale. Un autre aspect dans les méthodes de compression MPEG, qui peut être utilisé lors de l'adaptation, consiste en le taux de compression qu'on peut faire varier selon les ressources disponibles

Les différentes politiques d'adaptation sont initialement stockées sur un dépôt de politique comme des processus prêts à être utilisés. Le choix de la politique d'adaptation à appliquer dépend des paramètres (valeurs de contexte) détectés ; c'est selon l'ensemble de paramètres déclenchant l'adaptation que nous pouvons décider des politiques d'adaptation qui satisfont au mieux l'ensemble de contraintes, et qui conservent au mieux ces ressources. On peut donc dire que certaines conditions (valeurs de contexte) déclenchent certaines actions (politiques d'adaptation).

IV.3.3 Gestion adaptative de la session

Le problème de gestion de la session d'une application multimédia lors des déconnexions est un des problèmes cruciaux dans le monde de l'informatique mobile.

Dans notre travail, nous avons comme objectif de minimiser le nombre de paquets perdus lors des déconnexions et de pouvoir reprendre l'état de la session après reconnexion. Nous supposons que le client dépend d'un mécanisme tel que MobileIP, Mobile SCTP, ou autre pour assurer que la même connexion reste établie et nous essayons de proposer des solutions niveau applicatif pour atteindre nos objectifs.

Pour cela nous considérons les trois cas suivants :

IV.3.3.1 L'ouverture de session

Pour l'ouverture de session, nous proposons un mécanisme au niveau client qui nous permet de choisir parmi les différentes interfaces disponibles au moment de la connexion, celle via laquelle la connexion ou la session sera établie. Ce mécanisme se base sur plusieurs critères qui sont : la qualité du lien, la bande passante maximale, celle disponible au moment de la connexion, un ordre préalablement établi, et le coût.

Au niveau du Proxy, la requête reçue est adaptée selon la langue choisie avant d'ouvrir une session avec le serveur.

IV.3.3.2 Les déconnexions sans changement de Proxy

Dans le cas d'une déconnexion de courte durée sans changement de Proxy, nous proposons une solution qui consiste en un *PRECHARGEMENT DYNAMIQUE*. Comme nous l'avons déjà cité, notre application se base essentiellement sur la vidéo Streaming, cependant dans le cas d'une déconnexion le pré chargement nous semble la meilleure solution. Le pré chargement consiste à télécharger les données sur l'unité utilisateur pour être utilisées lors des déconnexions.

Pour cela nous définissons pour un client trois états : connecté, partiellement connecté, et déconnecté ; le mode de connectivité est défini selon la disponibilité de la bande passante.

Au cours d'une session, dès que l'état du client passe de l'état 'connecté' à 'partiellement connecté', la procédure de préchargement est lancée. Cette solution peut se résumer en les trois étapes suivantes (Figure 4.9.) :

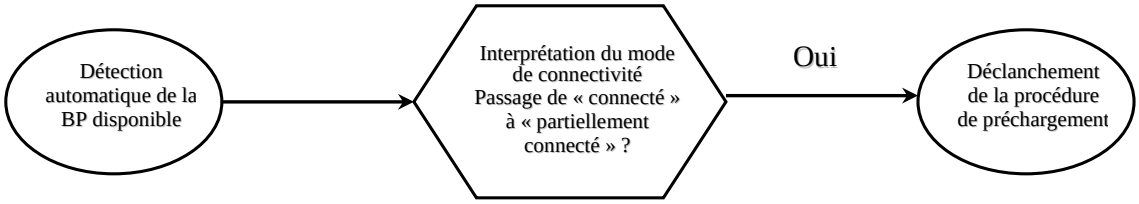

Figure 4.9. *Pré chargement dynamique*

IV.3.3.3 Les déconnexions avec changement de Proxy ou avec migration

Pour ce cas, nous proposons une solution qui se base essentiellement sur :

1. Un stockage temporaire et transfert de l'état de la session.
2. Et une prédiction et détection du mouvement.

Dans cette solution, nous définissons au niveau de chaque Proxy une table de correspondance, dans laquelle nous précisons l'ensemble de tous les Proxy voisins et leurs localisations. Nous nous basons sur ces tables et sur des mécanismes de détection de localisation et d'orientation pour positionner le client.

Le Proxy pourra ainsi décider à l'aide de ces mécanismes, et dès que le client passe à un mode partiellement déconnecté, s'il va y avoir une migration ou pas. Dans le cas où une migration est prévue, le Proxy sélectionne le Proxy qui va reprendre la session et établit une session avec lui durant laquelle il lui envoie toutes les données liées à la session :

- le fichier de contexte statique et semi dynamique,
- et l'état de la session.

Par la suite et une fois que le client établit une connexion avec le nouveau Proxy, ce dernier pourra reprendre la session là ou elle a été interrompue et terminer le processus d'adaptation selon le fichier de contexte transféré et les différentes autres valeurs dynamiques détectées.

Cette procédure peut se résumer en les étapes suivantes (Figure 4.10) :

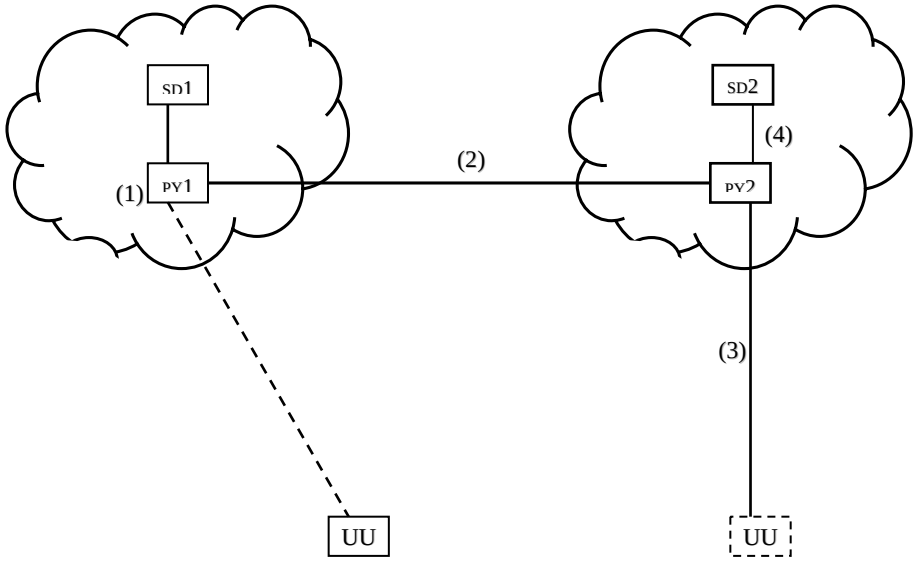

Figure 4.10. *Procédure de migration d'une unité*

SD1 : le serveur de données d'origine

PY1 : le Proxy d'origine

SD2 : le serveur qui va continuer la session

PY2 : le Proxy qui va reprendre la session

UU : l'unité de l'utilisateur

(1) Le Proxy détecte que le client est partiellement connecté et qu'il va y avoir une migration, il déclenche alors un stockage temporaire à son niveau (utilisation d'un cache), et sélectionne le Proxy qui va reprendre la session.

(2) Ouverture d'une session entre les deux Proxy et envoi de l'état de la session, des informations de contexte et du contenu du cache.

(3) Ouverture d'une session avec l'unité utilisateur, et envoi des médias stockées sur le cache avant de reprendre la session avec le serveur, étape (4).

Ce transfert de contexte a pour objectif de réduire les effets d'un « handover » sur notre application. Le principe est de mettre en place le plus rapidement possible le même niveau de service dont bénéficiait l'utilisateur dans son Proxy précédent.

IV.3.4 Sécurité

Dans un système informatique, les éléments à prendre en compte en ce qui concerne la sécurité sont :

- Les utilisateurs humains ou les processus qui agissent pour eux, appelés sujets.
- Les objets utilisés dans le système.

L'objectif de la sécurité informatique est la mise en œuvre des mécanismes de protection qui permettent d'assurer l'intégrité, la confidentialité, et la disponibilité.

Dans notre travail, nous considérons deux cas de figure, le cas d'une visualisation d'un cours avec la technique du Streaming, et le cas de passage des tests d'évaluation avec la technique de pré chargement.

IV.3.4.1 L'authentification et le droit d'accès

L'authentification a été utilisée dans notre système pour assurer la confidentialité des données et gérer les droits d'accès. C'est le Proxy qui est responsable de gérer les différentes informations relatives aux utilisateurs inscrits et de leurs droits d'accès.

IV.3.4.2 Le watermarking

Le watermarking est un procédé qui permet l'insertion d'informations numériques (watermarks ou marques) dans le médium. Cette insertion de données modifie légèrement le contenu du fichier sans en altérer la perception au niveau humain [NMF00].

Dans notre architecture, nous utilisons le watermarking pour signer les fiches d'évaluation. La marque contiendra l'identité de l'utilisateur et une estampille d'envoi. L'utilisateur aura ainsi un temps limite pour faire son évaluation.

IV.4 LES DIFFERENTS COMPOSANTS DE L'ARCHITECTURE :

Pour garantir les différentes fonctionnalités citées dans la partie précédente, nous proposons une architecture modulaire dans laquelle les différents modules collaborent pour garantir le bon fonctionnement du système. Dans cette partie, nous allons présenter deux vues sur la collaboration entre composants et le lieu d'exécution des fonctionnalités : une vue horizontale qui s'intéresse aux entités de l'architecture et une autre verticale dans laquelle les modules de chaque entité sont décrits de façon plus précise :

IV.4.1 Une vue horizontale de l'architecture

Si nous posons une vue horizontale sur l'architecture, le déroulement d'une session peut être résumé en les étapes suivantes (Figure 4.11.) :

(1) Inscription au service (saisie du contexte statique et envoi au Proxy).

(2) Stockage du contexte statique (identifiant, mot de passe, la classe, la langue, etc.) dans le dépôt de profil.

(3) Choix de l'interface de connexion, authentification, détection de contexte. (exp. Profil du terminal), et puis envoi de la requête et d'un fichier de contexte.

(4) Récupération du profil utilisateur et adaptation de la requête en conséquence (exp. Les paramètres de la classe).

(5) Ouverture d'une session avec le serveur.

(6) Début du transfert.

(7) Détection des variations, adaptation du flux et gestion de la session.

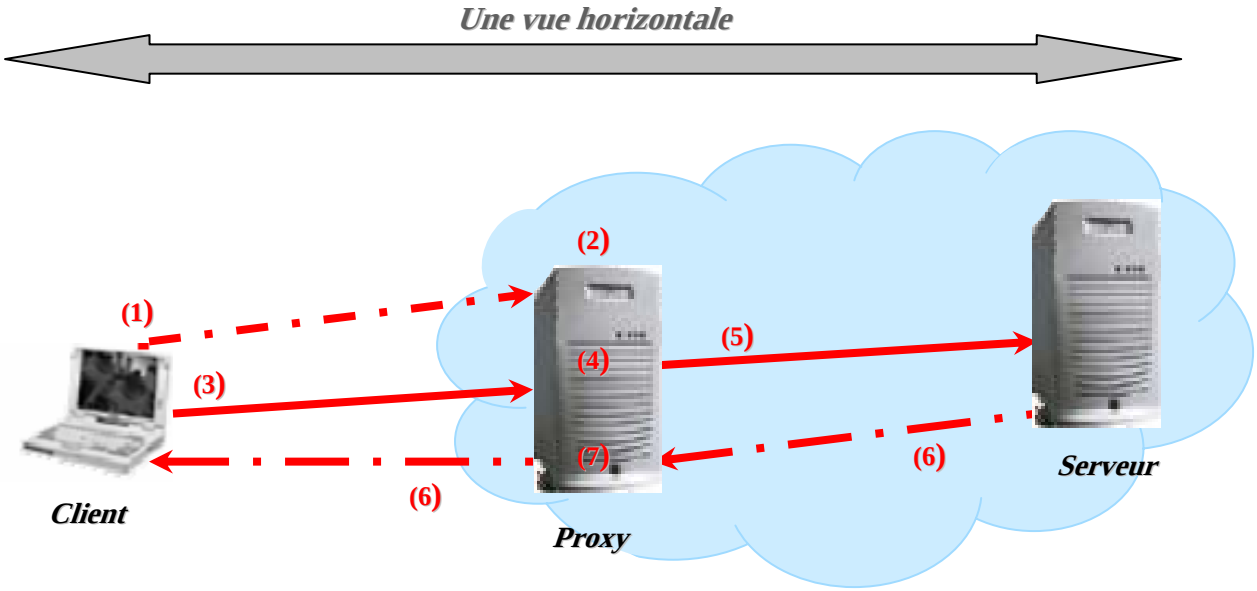

Figure 4.11. *Une vue horizontale de l'Architecture*

IV.4.1.1 Une vue verticale de l'architecture

Donner une vue verticale de l'architecture, revient à donner deux vues une sur le client et une autre sur le Proxy :

IV.4.1.1.1 Le client

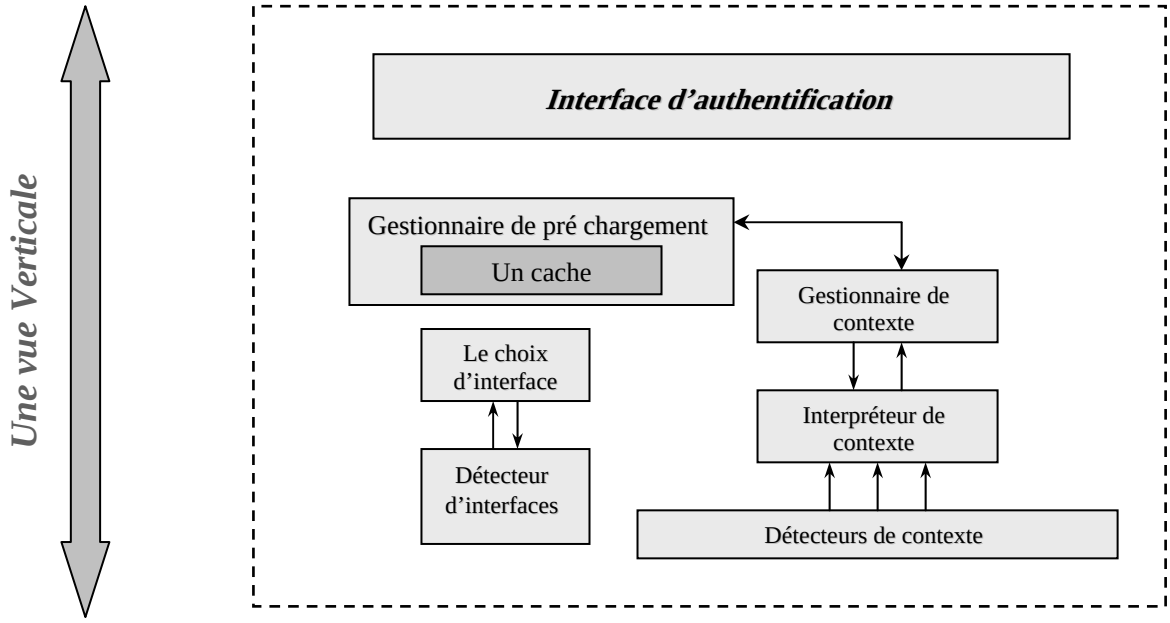

Figure 4.12. *Les composants sur l'unité utilisateur*

1. Interaction avec l'utilisateur

Pour l'interaction avec le client, nous définissons une interface utilisateur via laquelle il pourra :

a. S'inscrire,

b. S'authentifier,

c. Sélectionner, et visualiser les cours,

d. Et passer des tests d'évaluation.

Cette interface doit s'adapter aux capacités d'affichage des différents terminaux. Par exemple, pour un PDA de capacité réduite, certaines barres peuvent ne pas être affichées.

Les différentes fenêtres de l'interface peuvent prendre deux états, qui sont actives ou pas, selon la fonctionnalité et la période (ex. la fenêtre d'inscription et la fenêtre des tests).

2. La détection et gestion de contexte

La fonctionnalité de détection et de gestion de contexte côté client est structurée en couche selon le type de contexte pris en compte.

Nous définissons dans l'architecture plusieurs détecteurs ou capteurs, chacun responsable de surveiller une ressource particulière. Les valeurs de contexte après interprétation sont transmises au gestionnaire de contexte.

Pour l'interaction avec le gestionnaire de contexte, nous définissons deux cas de figure :

1. Inscrire un intérêt à une variation spécifique dans un type de contexte dynamique précis. Dans ce cas, des politiques d'intérêt sont définies pour chaque composant qui s'inscrit. Par exemple, pour chaque mode de connectivité nous définissons une plage de valeurs, et puis dès qu'un changement de mode est détecté, le composant responsable du pré chargement est alerté. A chaque fois qu'un changement est détecté, le gestionnaire de contexte vérifie à son niveau dans l'ensemble des politiques définies, si un composant a enregistré un intérêt à ce changement (Figure 4.13.). Nous pouvons ainsi dire que les valeurs de contexte reçues de l'interpréteur représentent les données du gestionnaire, et les politiques représentent les métas données.

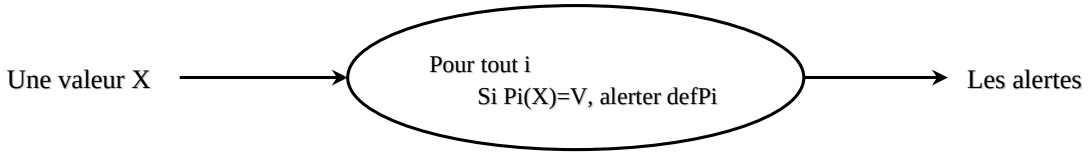

Figure 4.13. *Gestionnaire de contexte*

defPi : c'est le composant qui a défini la politique Pi.
Pi(X) : la politique Pi a un intérêt au changement X.

2. Utiliser le gestionnaire comme une base de contexte, qui peut être interrogée par une requête pour récupérer des valeurs de contextes précises (le contexte semi dynamique)

3. Le choix de l'interface de connexion

Pour garantir cette fonctionnalité, deux composants sont définis. L'un d'un niveau bas qui détecte les différents signaux et donc les différentes interfaces disponibles, et l'autre d'un niveau supérieur qui décide via laquelle des différentes interfaces il va se connecter. La décision est basée sur plusieurs critères : le type d'interface (la bande maximum et celle disponible), le coût, et un ordre de priorité défini principalement suivant la qualité du lien et le débit maximum.

4. La gestion des déconnexions

Pour gérer les déconnexions, nous proposons côté client un gestionnaire de pré chargement et un cache. Le gestionnaire de pré chargement inscrit son intérêt d'être notifié dans le cas d'un changement dans le mode de connectivité. Et si le mode passe de 'connecté' à 'partiellement connecté' le pré chargement est lancé et les données sont stockées au niveau du cache. Le choix de la technique de pré chargement est utilisé comme un axe d'adaptation qui dépend du type de réseau, des protocoles utilisés et de la qualité du lien.

IV.4.1.1.2 Le Proxy

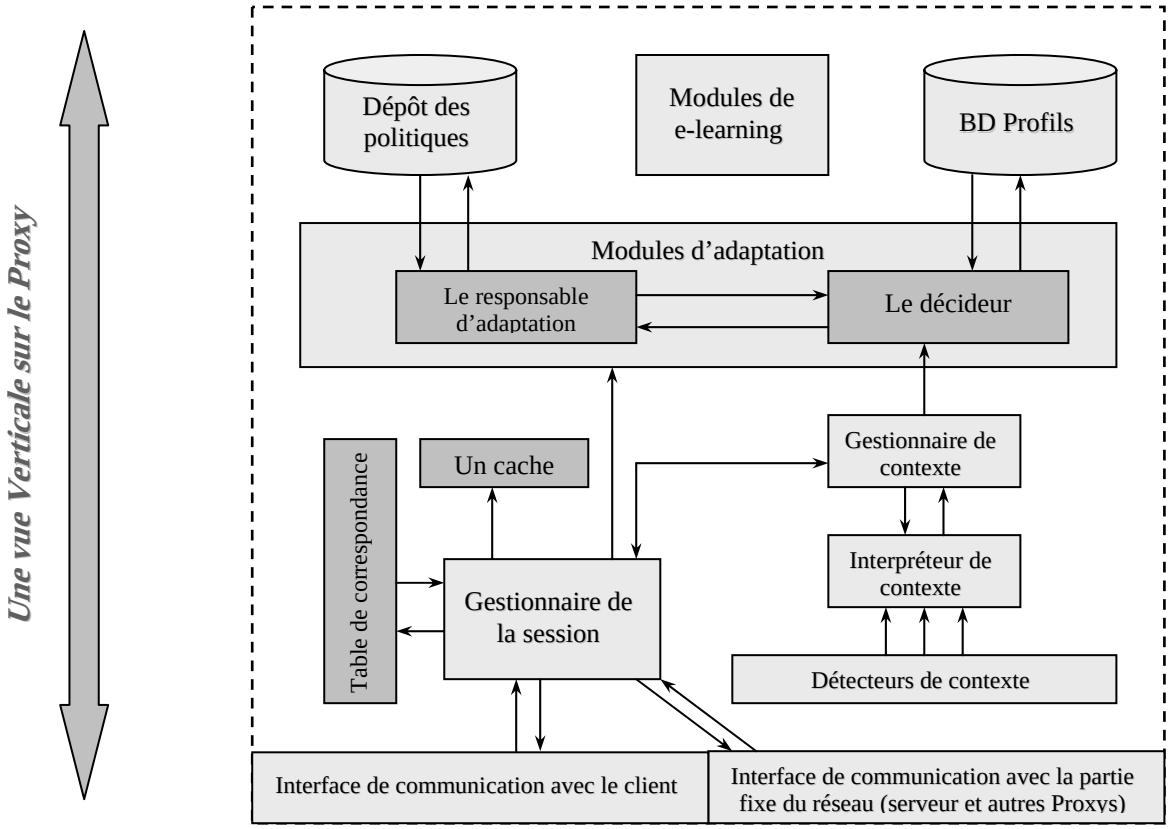

Figure 4.14. *Architecture détaillée du Proxy*

1. La détection et gestion de contexte

De même que pour le client, nous définissons plusieurs *détecteurs* ou *capteurs* selon le type de contexte pris en compte (ex. localisation, bande passante, etc.), et un *gestionnaire de contexte* regroupant les différentes politiques qui décrivent les intérêts de chaque composant inscrit à des changements spécifiques dans les ressources. Dès qu'il détecte un changement dans ses données, le gestionnaire vérifie au niveau de ses métas données (l'ensemble des politiques) s'il existe des composants qui ont enregistré un intérêt à ce changement, auquel cas, il leur envoie des messages spécifiques selon l'intérêt défini.

De même que le gestionnaire de contexte au niveau du client, ce composant peut être interrogé par l'envoi d'une requête.

Pour le stockage du contexte statique (ex. les mots de passe, la classe), nous proposons une base de données ou dépôt de profils. Cette base sera interrogée au début de chaque session pour :

a. vérifier les droits d'accès,

b. adapter la requête selon la langue préférée de l'utilisateur,

c. et télécharger un fichier de contexte (la classe, les préférences, etc.) comme paramètre dans le processus d'adaptation du flux multimédia reçu du serveur.

2. Les modules de e-learning

Ces modules sont utilisés pour vérifier certains critères ou contraintes de la vidéo formation ou l'enseignement à distance. Par exemple, un test d'évaluation ne peut être passé qu'après un certain nombre de cours.

3. La communication dans le Proxy

Pour interagir avec les deux autres entités du système (Client et serveur), différents types d'interfaces de communication sont définis. Cette interaction peut se faire via différents protocoles destinés essentiellement aux transferts des données multimédia tels que RTP et SDP.

Les différents modules du Proxy quant à eux communiquent par échange de messages.

4. Gestion de la session

Pour gérer la session au niveau du Proxy, trois composants sont définis : le gestionnaire de session, la table de correspondance et un cache.

Le rôle du gestionnaire de session est de gérer une session de sa création à sa clôture. Pour l'interaction avec le gestionnaire de contexte, il s'inscrit et définit une politique qui décrit son intérêt d'être notifié dans le cas d'un changement dans le mode de connectivité. Dans cette politique, on précise que si on passe d'un mode à un autre, le gestionnaire de session doit être seulement notifié, sauf pour le cas d'un passage du mode 'connecté' à 'partiellement connecté' où même la localisation doit être envoyée avec la notification au gestionnaire de session. Ce dernier, après consultation de la table de voisinage ou la table de correspondance, pourra prévoir s'il va y avoir une migration ou non :

- Dans le cas où il ne prévoit pas de migration, le Proxy continue l'envoi de données normalement, et dès qu'il détecte qu'il n'y a plus de connexion, il stocke le flux reçu du serveur au niveau de son cache local.
- Dans le cas contraire (il prévoit une migration), le gestionnaire de session sélectionne le Proxy qui va reprendre la session et stocke au niveau de son cache local les données reçues du serveur. Dès que le signal atteint une certaine borne inférieure, préalablement définie, le gestionnaire de session envoie au Proxy sélectionné l'état de la session, le fichier de contexte lié à cette session, et les données stockées dans le cache.

La taille du cache est déterminée en fonction du temps nécessaire pour le nouveau Proxy pour ouvrir une session avec le serveur. Il pourra ainsi commencer le transfert des données vers l'unité de l'utilisateur le temps d'ouvrir la session avec le serveur.

5. Adaptation du flux multimédia :

Pour garantir la fonctionnalité de base de notre architecture, qui est l'adaptation du flux multimédia aux différentes valeurs de contexte statique, semi dynamique, et dynamique, nous définissons trois composants de base qui sont :

1. Le dépôt des politiques : où les différentes politiques d'adaptation sont stockées comme des processus prêts à être utilisés.

Notons que notre dépôt de politique restera ouvert à d'éventuelles modifications, nous pourrons ainsi ajouter ou enlever des politiques existantes.

2. Le module de décision : qui a en entrée les différentes valeurs de contexte :

- Le profil utilisateur est récupéré de la base de profils au début de la session suite à une requête.
- Le fichier de contexte reçu du client quand à lui est automatiquement envoyé au module de décision au début de la session.

Pour le contexte dynamique, le module de décision est inscrit au niveau du gestionnaire de contexte pour être notifié dans le cas d'une variation spécifique dans l'un des paramètres. Dans la politique qui décrit l'intérêt et pour chaque type de contexte, des plages de valeurs sont définies, et chaque fois que la valeur de contexte passe d'une plage à une autre, le module de décision doit être notifié.

Au niveau de ce module, plusieurs dimensions qui correspondent aux différentes plages de valeurs sont associées à chaque paramètre de contexte. Exemple : Pour chaque plage de valeurs de la bande passante, une qualité de flux est associée.

Le module, après acquisition des différentes valeurs de contexte, et d'après la combinaison des différents paramètres, décide de l'ensemble des politiques à appliquer.

Notons que le temps est une valeur de contexte très importante qui influence la décision prise par ce module. Exemple : dans le cas où on a un niveau d'énergie réduit, c'est selon le temps qui reste pour la présentation qu'on peut décider des actions à prendre. S'il reste peu de temps avant la fin de la présentation et du cours, on peut se permettre d'appliquer certaines dégradations de la qualité des images ou autres. Mais si on voit qu'il reste beaucoup de temps, on doit soit suspendre la présentation, soit faire un filtrage et n'envoyer que le texte si l'utilisateur est pressé (*la classe C3/PR*).

3. Le module d'adaptation

Ce composant a pour rôle d'appliquer les différentes politiques d'adaptation, il aura en entrée, en plus de l'ensemble des politiques sélectionnées par le module de décision, les objets multimédias ou le flux multimédia reçu du serveur. Ce flux va ainsi passer dans une file de politiques, téléchargées à partir du dépôt de politiques. A la sortie nous aurons un flux multimédia adapté (Figure 4.15.).

Au cours d'une session, dès que le module de décision est notifié par le gestionnaire de contexte, l'ensemble des politiques à appliquer est modifié, et un message de type {Nvl ens, {ref1, ref2, …}} où Nvl ens : nouvel ensemble de références, et {ref1, ref2, …} : les références aux politiques à appliquer, est envoyé au module de décision qui crée un nouveau tunnel.

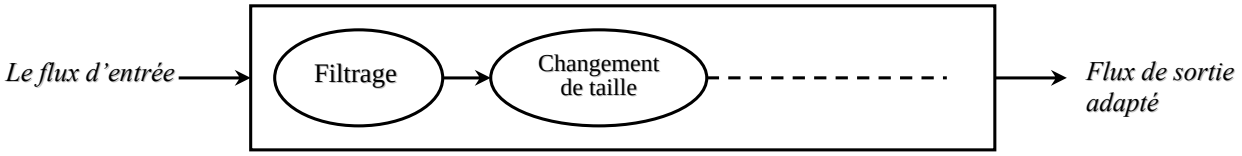

Figure 4.15. *Le module d'Adaptation*

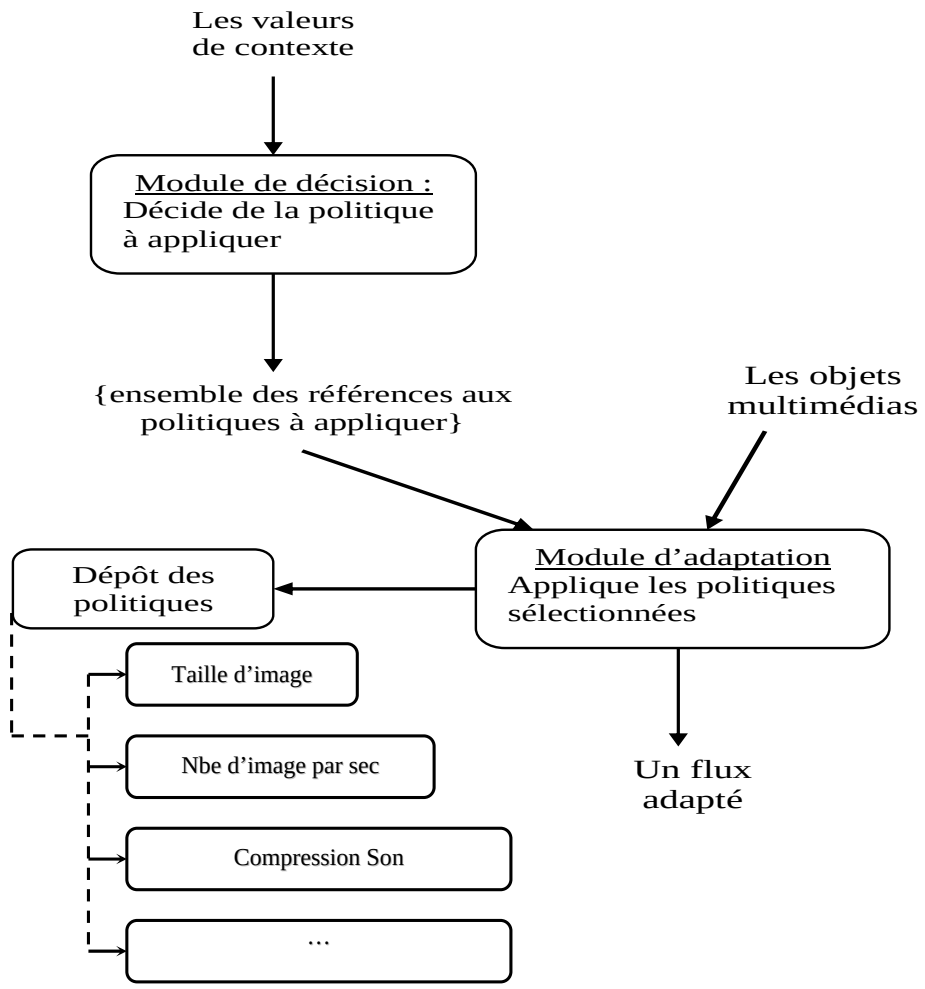

Figure 4.16. *Interaction entre les composants d'adaptation*

IV.5 Conclusion

Dans le but d'assurer l'exécution context-aware d'une application multimédia mobile en général, et d'une application de vidéo formation en particulier, nous avons proposé une architecture d'adaptation qui permet à l'application en collaboration avec le système d'adapter son exécution selon les variations dynamiques de contexte. L'architecture proposée est modulaire à base de Proxy, elle permet grâce aux différents composants situés sur l'unité utilisateur et sur le Proxy, de garantir le bon fonctionnement de l'application.

Nous ne nous somme pas contentés de proposer des politiques d'adaptation de contenu qui vont jouer sur la qualité de service selon les différentes valeurs de contexte pris en compte. Mais nous avons également utilisé l'adaptation et les informations de contexte pour apporter des solutions à d'autres problèmes, tel que les déconnexions sans et avec migration en proposant une solution basée sur les préchargements et une technique de prédiction.

La sécurité a également été abordée comme aspect indispensable qui doit être pris en compte dans tout travail visant à offrir un service.

Conclusion générale

Le problème d'adaptation est un des problèmes largement étudié dans les systèmes distribués classiques. Cependant, avec l'arrivée des réseaux mobiles, le problème s'est accentué par l'aspect dynamique du changement ; le contexte d'exécution qui peut influer sur le comportement d'une application (tels que la disponibilité des ressources : batterie et bande passante, la location, l'ajout de nouveaux terminaux ou le changement de service) est très dynamique et les changements sont fréquents. Contrairement à celui des réseaux distribués fixes qui est plus au moins statique.

Dans la première partie de notre travail, nous nous sommes focalisé sur l'étude du problème d'adaptation et des différents systèmes existants. Nous avons constaté à la fin de cette étude que le problème d'adaptation et du context-aware computing est un problème à plusieurs facettes qui peut être abordé de différentes manières, pour répondre à différents besoins. Cette étude parallèle des différents systèmes nous a permis de les classer suivant différents paramètres et essentiellement suivant l'objectif de l'adaptation et le niveau d'implémentation des politiques. En effet, si nous regardons l'objectif de l'adaptation, nous remarquons que certains ont utilisé l'adaptation et le context-aware computing pour garantir le bon fonctionnement des applications dans l'environnement mobile et donc un certain niveau de QdS, alors que d'autres les ont utilisé pour offrir de nouveaux services liés à des valeurs spécifiques de contexte. De même, si nous considérons le niveau d'implémentation des politiques, les solutions étudiées peuvent être classées en deux groupes. Le premier groupe englobe les solutions qui placent la responsabilité entière de l'adaptation sur le système et garantissent donc une adaptation Application-transparente. Et le deuxième, regroupe les solutions qui se basent sur une collaboration entre le système et l'application pour garantir le bon fonctionnement des applications.

Apres ce constat, nous nous sommes orientés vers le domaine du multimédia. Nous avons commencé par étudier le domaine (caractéristiques, problèmes, etc.) et principalement le problème de la qualité de service et l'effet des contraintes de l'environnement mobile sur cette dernière. Nous avons montré le besoin d'une adaptation dynamique en fonction des variations dans les valeurs de contexte. Nous nous sommes essentiellement intéressés au concept de qualité de service et tous les paramètres et contraintes qui peuvent l'influencer.

Dans ce cadre, nous avons proposé une architecture modulaire qui permet l'exécution d'une application multimédia de façon adaptable dans un environnement mobile hétérogène, et plus précisément une application de vidéo formation. Pour garantir l'objectif principal de l'architecture qui est le bon fonctionnement de l'application, nous définissons au sein de l'architecture quatre fonctionnalités de base qui sont : la détection et gestion de contexte, l'adaptation dynamique de la qualité de service, la gestion de la session, et la sécurité.

Les principales fonctionnalités offertes par cette architecture peuvent se résumer en :

- La gestion dynamique du contexte en utilisant la notion de données et de méta données.

- La gestion dynamique de la qualité de service en utilisant l'adaptation dynamique de l'exécution et la notion de fidélité de données.
- La gestion prédictive des migrations basée sur un stockage temporaire et le transfert de l'état de la session, ainsi qu'une prédiction et détection de mouvement.
- La gestion dynamique des déconnexions de courtes durées en proposant un pré chargement dynamique.
- Une fonction objective pour le choix de l'interface de connexion.
- L'aspect sécurité n'a pas été oublié.

Ce travail nous a permis de définir une architecture globale garantissant les fonctionnalités de base nécessaires à l'adaptation context-aware de notre application. Cependant nous pensons qu'il nous a surtout ouvert plusieurs autres voix de recherche :

Dans notre architecture, nous avons pris en compte le cas d'un Proxy par serveur mais comme perspective, nous pouvons reconsidérer notre solution dans le cas où un Proxy peut gérer plusieurs serveurs et vice versa. Dans ce cas le choix du serveur ou du Proxy peut dépendre d'une fonction objective calculée en fonction de plusieurs paramètres dont la surcharge, la distance, etc.

L'aspect sécurité a été abordé dans notre architecture comme une fonctionnalité qui doit être assurée, surtout dans le cas d'une application à service. Néanmoins beaucoup de travail reste à faire dans ce sens pour offrir une vraie sécurité, ce qui peut faire l'objet d'une suite en perspective.

La gestion de la session durant la mobilité mérite elle aussi une recherche plus approfondie pouvant faire l'objet d'un autre travail.

Références bibliographiques

[ABC98] V. Apparao, S. Byrne, M. Champion, S. Isaacs, I. Jacobs, A. Le Hors, G. Nicol, J. Robie, R. Sutor, C. Wilson, and L. Wood.
Document Object Model (DOM)
Level 1 Specification. W3C Recommendation
http://www.w3.org/TR/1998/RECDOM- Level-1-19981001, World Wide Web Consortium, October 1998.

[ACK94] Abhaya Asthana, Mark Cravatts, and Paul Krzyzanowski.
An indoor wireless system for personalized shopping assistance.
In *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications*, pages 69-74, Santa Cruz, California, December 1994. IEEE Computer Society Press.

[AMZ95] E. Amir, S. McCanne, H. Zhang,
An application level video gateway, Proc.
Of ACM Multimedia '95, San Francisco, Novembre 1995.

[AS97] AHN, S.J. and SCHULTES, M
A new circular coded target for the automation of photogrammetric 3D-surface measurements.
In: Optical 3-D Measurement, Wichmann Verlag, München, Germany. .(1997).

[AS99] F. André, M.T Segara-
On building a file system for mobile environments using generic services.
Proc. of the12th International Conference on Parallel and Distributed Computing Systems, August 1999, Fort Lauderdale, Florida, USA

[AYB01] I. Augustin, A. Yamin, j. V. Barbosa, C. R. Geyer,
"Requirements for Design of Mobile Distributed Applications",
Proc. VIII Argentinian Congress of Computer Science (CACIC), Argentina, oct, 2001.

[Bad95] N. Badache,
La mobilité dans les systèmes répartis,
Publication interne N°962, IRISA, Octobre 1995

[Bad98] N. Badache,
Ordre causal et tolérance aux défaillances en environnement mobile,
Thèse de doctorat, Institut d'Informatique, USTHB, Octobre 1998

[Bag96] Baggio Aline,
Environnements mobiles: Caractéristiques et problèmes
Séminaire sur les systèmes et applications réparties, CENT-Issy les moulineaux, Février 1996

[BBC97] Brown, P.J., Bovey, J.D. Chen, X.
Context-Aware Applications: From the Laboratory to the Marketplace.

IEEE Personal Communications, 4(5) (1997) 58-64

[BCA01] Gordon S. Blair, Geoff Coulson, Anders Andersen, Lynne Blair, Michael Clarke, Fabio Costa, Hector Duran Limon, Tom Fitzpatrick, Lee Johnston, Rui Moreira, Nikos Parlavantzas, and Katia Saikoski.
The Design and Imple mentation of Open ORB 2.
IEEE Distributed Systems Online, 2(6), 2001

[BG96] J.C Bolot and A.V. Garcia.
Control mechanisms for packet audio in the internet.
In *IEEE INFOCOM*, November 1996.

[BG01] J.L.V. Barbosa, C.F.R. Geyer,
A Multiparadigm Language Oriented to Distributed Software Development,
V Brazilian Symposium of Programming Languages. (SBLP), mai., 2001.

[BHE00] Nirupama Bulusu, John Heidemann, and Deborah Estrin.
GPS-less low-cost outdoor localization for very small devices.
IEEE Personal Communications, 7(5):28-34, October 2000.

[BNB05] F. Bouabache, N. Nouali, N. Badache
Architecture d'Adaptation d'une Application de Video on Demand
SETIT'2005

[BNS95] Brian D. Noble, M. Satyanarayanan, M. Price-
A programming interface for application-aware adaptation in mobile computing.
Proceedings of the Second USENIX Symposium on Mobile & Location-Independent Computing Apr. 1995.

[BP00] Paramvir Bahl and Venkata N. Padmanabhan.
Radar: An in-building RF-based user location and tracking system.
In Proceedings of IEEE INFOCOM 2000, Tel-Aviv, Israel, March 2000. IEEE Computer Society Press.

[BRH94] Frazer Bennett, Tristan Richardson, and Andy Harter.
Teleporting - making applications mobile.
In Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, pages 82-84, Santa Cruz, California, December 1994. IEEE Computer Society Press.

[Bro96] Brown, P.J.
The Stick-e Document: a Framework for Creating Context-Aware . Applications.
ElectronicPublishing '96 (1996) 259-272

[Bro98] Peter J. Brown. Triggering information by context.
Personal Technologies, 2(1), March 1998.

[BT94] J.C Bolot and T. Turletti.

A rate control mechanism for packet video in the internet.
In *IEEE INFOCOM*, November 1994.

[BT98] J-C. Bolot and T. Turletti.
Experience with rate control mechanisms for packet video in the Internet.
Computer Communications Review*, 28(1), 1998.*

[CD91] Carl D. Tait and Dan Duchamp.
Detection and Exploitation of File Working Sets.
*I*n 11th International Conference on Distributed Computing Systems, pages 2--9, May 1991.

[CD99] J. Clark and S. DeRose.
XML Path Language (XPath).
Technical Report World Wide Web Consortium, November 1999.

[CEM01] Licia Capra, Wolfgang Emmerich and Cecilia Mascolo
Reflective Middleware Solutions for Context-Aware Applications
Dept. of Computer Science, University College London

[CHR01] R. Cerqueira, C. K. Hess, M. Romn, and R. H. Campbell.
Gaia: A Development Infrastructure for Active Spaces.
In Workshop on Application Models and Programming Tools for Ubiquitous Computing (held in conjunction with the UBICOMP 2001), Sept. 2001.

[CK01] Guanling Chen and David Kotz,
A Survey of Context-Aware Mobile Computing Research,
Department of Computer Science Dartmouth College. Dartmouth Computer Science Technical Report TR2000-381

[CLW02] Cecilia Mascolo, Licia Capra and Wolfgang Emmerich,
Mobile Computing Middleware, 2002

[CM00] Paul Castro and Richard Muntz.
Managing context data for smart spaces.
IEEE Personal Communications, 7(5):44-46, October 2000.

[CNB04] F. Chehbour, N. Nouali, L. Boukantar
Gestion de la mobilité au niveau de la couche application : état de l'art.
Conférence Internationale SETIT'2005

[CRR99] R. Cerqueira, R. Ierusalimschy, and N. Rodriguez.
The LuaOrb Project.
Project home page: http://www.tecgraf.puc-rio.br/luaorb, 1999.

[DAW99] Dey, A.K., Abowd, G.D., Wood, A.
CyberDesk: A Framework for Providing Self-.Integrating Context-Aware Services.
Knowledge-Based Systems, 11 (1999) 3-13

[Dem02] Erwan Dematry
Aide à la conception des applications multimédias
Thèse pour l'obtention d'un grade de docteur à l'université de Rennes

[Den] Denis Conan
INT, Département Informatique, Équipe Systèmes Répartis, 9 rue Charles Fourier, 91011 Évry cedex

[Dey98] Dey, A.K.
Context-Aware Computing: The CyberDesk Project.
AAAI 1998 Spring . Symposium on Intelligent Environments, Technical Report SS-98-02 (1998) 51-54

[Dey01] ANIND K. DEY
Understanding and Using Context
Future Computing Environments Group, College of Computing & GVU Center Georgia Institute of Technology

[DFS99] Anind K. Dey, Masayasu Futakawa, *Daniel Salber, and Gregory D. Abowd. The Conference Assistant: Combining Context-Awareness with Wearable Computing.*
In Proceedings of the 3rd International Symposium on Wearable Computers (ISWC '99), pages 21-28, San Francisco, CA, October 1999. IEEE Computer Society Press.

[DRN95] *Untracability in mobile network*

[DSA99] Dey A.K., Salber, D., Abowd, G.D. (1999).
The Context Toolkit: Aiding the Development of Context-Enabled Applications, CHI'99..

[DSO03] Denis Conan, Sophie Chabridon, Olivier Villin, and Guy Bernard
Domint: A Platform for Weak Connectivity and Disconnected CORBA Objects
On Hand-Held Devices GET /INT CNRS Samovar 5157

[FF98] Franklin, D., Flaschbart, J.
All Gadget and No Representation Makes Jack a Dull .
Environment.AAAI 1998 Spring Symposium on Intelligent Environments, Technical Report SS-98-02 (1998) 155-160

[Fit99] Tom Fitzpatrick
Open Component-Oriented Multimedia Middleware for Adaptive Distributed Applications
Thèse pour l'obtention d'un grade de docteur à l'université de Lancaster

[FKV00] D. Fritsch, D. Klinec, and S. Volz.
NEXUS Positioning and Data Management Concepts for Location Aware Applications.

In Proceedings of the 2nd International Symposium on Telegeoprocessing, pages 171–184, Nice-Sophia-Antipolis, France, 2000.

[FSR] Frank Fitzek and Patrick Seeling and Martin Reisslein
Video Streaming in Wireless Internet

[HHS+99] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, and Paul Webster.
The anatomy of a context-aware application.
In Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking, pages 59-68, Seattle, WA, August 1999. ACM Press.

[HNB97] Hull, R., Neaves, P., Bedford-Roberts, J.
Towards Situated Computing.
1st International Symposium on Wearable Computers (1997) 146-153

[HPW00] M. Handley, C. Perkins, and E. Whelan,
"RFC 2974: *SAP: Session announcement protocol,*" Oct. 2000.

[IC+97] J. Inouye, S. Cen, C. Pu, J. Walpole.
System Support for Mobile Multimedia Applications.
Proceedings of the 7th NOSSDAV; May 1997.

[Kis92] James J.Kistler and M. Satyanarayanan-
Disconnected operation in CODA file system.
ACM Transactions on Computer Systems, vol.10, No.1, february 1992, pages 3-25.

[KMR93] H. Kanakia, P. Mishra, A. Reibman,
An adaptive congestion control scheme for real-time packet video transport,
ACM/IEEE SIGCOMM Symposium on Communications Architectures
and Protocols, San Francisco, PP. 20-31. Septembre 1993.

[KRL00] F. Kon, M. Rom´an, P. Liu, J. Mao, T. Yamane, L. M. aes, and R. Cambpell.
Monitoring, Security, and Dynamic Configuration with the dynamicTAO Reflective ORB.
In International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware'2000), pages 121–143, New York, Apr. 2000. ACM/IFIP.

[KS92] Kistler, J.J., Satyanarayanan, M.
Disconnected Operation in the Coda File System.
ACM Transactions on Computer Systems 10(1), February, 1992

[LBY02] Jorge Luis, Victória Barbosa, Adenauer Corrêa Yamin, Patrícia Kayser Vargas, Iara Augustin, Cláudio Fernando et Resin Geyer
Holoparadigm: a Multiparadigm Model Oriented to Development of distributed Systems
2002

[Led99] T. Ledoux.
OpenCorba: a Reflective Open Broker.
In Reflection'99, volume 1616 of LNCS, pages 197–214, Saint-Malo, France, 1999. Springer

[Lem04] Tayeb LEMLOUMA
Architecture de Négociation et d'Adaptation de Services Multimédias dans des Environnements Hétérogènes.
Thèse pour l'obtention d'un grade de docteur de l'INPG, Juin 2004.

[LG02] Z.Lei and N.D.Georganas,
H.263 Video Transcoding for Spatial Resolution Downscaling, Proc.
IEEE International Conference on Information Technology: Coding and Computing (ITCC 2002), Las Vegas, Nevada , Avril 2002

[LH02] O. Layaida, D. Hagimont,
Dynamic Adaptation in Distributed Multimedia Applications.
Rapport technique n° 0266, August 2002, INRIA

[LKA96] Sue Long, Rob Kooper, Gregory D. Abowd, and Christopher G. Atkeson. *Rapid prototyping of mobile context-aware applications: the Cyberguide case study.*
In Proceedings of the Second Annual International Conference on Mobile Computing and Networking, pages 97-107, White Plains, NY, November 1996. ACM Press.

[LPA99] X.Li, S. Paul, and M.H. Ammar.
Multi-Session Rate Control for Layered Video Multicast.
In Proc. IS&T/SPIE Multimedia Computing and Networking, *January* 1999.

[LP+97] Li, S. Paul, P Pancha, and M.H. Ammar.
Layered Video Multicast with Retransmission (LVMR): Evaluation of Error Recovery.
In *Proc. NOSSDAV'97*, May 1997.

[MA00] F. Le Mouël, F André-
Distribution over Mobile Environment
Proc. Of the 2000 ACM Symposium on Applied Computing

[MA01] F. Le Mouël, F André-
AeDEn : un cadre générale pour une distribution adaptative des applications en environnements mobiles.
Revue électronique sur les réseaux et l'informatique répartie No 11, mars 2001.

[MJV96] Steven McCanne, Van Jacobson, and Martin Vetterli.
Receiver-driven layered multicast.
In *ACM SIGCOMM*, Stanford, CA, August 1996.

[Moh02] Kedar .B. Mohare
Web Proxy Mechanisms for Wireless Networks
Department Of Computer Science & Engineering University Of Texas At Arlington kbm6471@omega.uta.edu

[Mou03] Le Mouël, 2003.
Environnement adaptatif d'exécution distribution d'applications dans un contexte mobile.
Thèse de doctorat Université de Rennes 1, 2003.

[NMF00] Nicolas Paris, Maxime Rouca, Franck Fauverte, Jérôme segard.
Le watermarking
DESS Informatique Multimédia, 1999/2000.

[Nob00] Brian D. Noble-
System support for mobile, adaptive applications.
IEEE personnal Communications february 2000

[Nou01] N. Nouali,
Impact de la mobilité sur les modèles de transactions,
Publication Interne, Cerist, Janvier 2001

[NS99] B.Noble, M. Satyanarayanan.
Experience with adaptive mobile applications in Odyssey.
Mobile Networks and Applications Vol. 4, 1999 .

[NS+97] B.D. Noble, M. Satyanarayanan, D. Narayanan, J.E. Tilton, J. Flinn, K.R. Walker.
Agile application-aware adaptation for mobility.
ACM SIGOPS Operating Systems. Rev. 31, 5, 276–287. 1997.

[NSP95] Brian D. Noble, M. Satyanarayanan, M. Price-
A programming interface for application-aware adaptation in mobile computing.
Proceedings of the Second USENIX Symposium on Mobile & Location-Independent Computing Apr. 1995.

[NTR04] Nolwenn, Thérèse & Guillaume Rincé
Le principe du streaming : Catégorie QuickTime 4 et le streaming.
Publié le 20 février 2001 par Guillaume. Mise à jour le vendredi 23 juillet 2004. http://guillaume.rince.free.fr/spip

[Pas99] Jason Pascoe.
Adding generic contextual capabilities to wearable computers.
In Proceedings of the Second International Symposium on Wearable Computers, Pittsburgh, Pennsylvania, October 1998. IEEE Computer Society Press.

[PCB00] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan.

The Cricket location-support system.
In Proceedings of the Sixth Annual ACM International Conference on Mobile Computing and Networking, Boston, MA, August 2000. ACM Press.

[Pre01] *how wireless works,*
International standard book Number: 0-7897-2487-1

[PRM98] Jason Pascoe, Nick Ryan, and David Morse.
Issues in developing context-aware computing.
In Proceedings of First International Symposium on Handheld and Ubiquitous Computing, HUC'99, pages 208-221, Karlsruhe, Germany, September 1999. Springer Verlag.

[RC01] M. Rom´an and R. H. Campbell.
A model for ubiquitous applications.
Technical Report UIUCDCSR-2001-2223 UILU-ENG-2001-1730, University of Illinois at Urbana-Champaign, May 2001. Avaliable at http://devius.cs.uiuc.edu/gaia/papers/appmodel-tech01.pdf.

[Ros+99] J. Rosenberg and et. al.,
"RFC 3261: *SIP: Session initiation protocol,*" Feb. 1999.

[RPM97] Ryan, N., Pascoe, J., Morse, D.
Enhanced Reality Fieldwork: the Context-Aware .
Archaeological Assistant. Gaffney,V., van Leusen, M., Exxon, S. (eds.) Computer Applications in Archaeology (1997)

[SA00] M.T Segara, F. André-
A generic approch to satisfy adaptability needs in mobile environments.
proc. Of the 33rd Annal Hawaii International Conference on System Sciences, january 2000, Maui, Hawaii, USA.

[Sat95] M. Satyanarayanan,
Fundamental Challenges in Mobile Computing
School of Computer Science Carnegie Mellon University

[SAT99] Albrecht Schmidt, Kofi Asante Aidoo, Antti Takaluoma, Urpo Tuomela, ristof *Van Laerhoven, and Walter Van de Velde.*
Advanced interaction in context.
In Proceedings of First International Symposium on Handheld and Ubiquitous Computing, HUC'99, pages 89-101, Karlsruhe, Germany, September 1999. Springer Verlag.

[SAW94] Bill Schilit, Norman Adams, and Roy Want.
Context-aware computing applications.
In Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, pages 85-90, Santa Cruz, California, December 1994. IEEE Computer Society Press.

[SKK90] Satyanarayanan, M., Kistler, J.J., Kumar, P., Okasaki, M.E., Siegel, E.H., Steere, D.C.
Coda: A Highly Available File System for a Distributed Workstation Environment.
IEEE Transactions on Computers 39(4), April, 1990.

[SC+96] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson,
Audio-Video Transport Working Group
"RFC 1889: *RTP: A transport protocol for real-time applications*," Jan. 1996.

[SRL98] H. Schulzrinne, A. Rao, and R. Lanphier,
"RFC 2326: *Real time streaming protocol (RTSP)*," Apr. 1998.

[SS98] D. Sisalem, H, Schulzrinne,
The loss-delay adjustment algorithm: a TCPfriendly adaptation scheme, Proc. International Workshop on Network and Operating System Support for Digital Audio and Video NOSSDAV, (Cambridge), Juillet 1998.

[ST94] Schilit, B., Theimer, M.
Disseminating Active Map Information to Mobile Hosts.
IEEE Network, 8(5) (1994) 22-32

[SVG03] SVG Working Group,
Scalable Vector Graphics (SVG) 1.0 Specification.
W3C Recommendation, http ://www.w3.org/TR/SVG, 14 Janvier, 2003.

[SWW94] J. Sandvoss, J. Winckler, H. Witting.
Network Layer Scaling: Congestion control in Multimedia Communication with Heterogenous Networks,
Rapport n° 43.6401, IBM European Networking Center, Heidelberg, 1994.

[TH96] T. Turletti et C. Huitema.
Videoconferencing On the Internet.
IEEE, CM Transactions on Networking, vol. 4, p.340-351, Juin 1996.

[VFJ+99] Bobby Vandalore, Wu-chi Feng, Raj Jain, Sonia Fahmy
A survey of Application Layer techniques for Adaptive Streaming of Multimédia
OSU technical Report, OSU-CISRC-5/99-TR14

[Wan01] Sheng Wanli
Vom mobilen zum ubiquitären Gebrauch von Rechnern"
Abteilung Verteilte Systeme, Fakultät Informatik, Universtät Stuttgart

[Wei93] Mark Weiser.
Some computer science issues in ubiquitous computing.
Communications of the ACM, 36(7):75-84, July 1993.

[WHF92] Roy Want, Andy Hopper, Veronica Falcão, and Jonathan Gibbons.
The Active Badge location system.
ACM Transactions on Information Systems, 10(1):91-102, January 1992.

[XHT02] XHTML 1.0 : *The Extensible HyperText Markup Language XHTML 1.0*.
W3C Recommendation, http ://www.w3.org/TR/xhtml1/, 26 Janvier 2000, revised 1 Août 2002

[YAB02] Adenauer Yamin, Ira Augustin, Jorge Barbosa et Claudio Geyer
ISAM, A pervasive view in distributed mobile computing

[YS00] Hao Yan and Ted Selker.
Context-aware office assistant.
In Proceedings of the 2000 International Conference on Intelligent User Interfaces, pages 276-279, New Orleans, LA, January 2000. ACM Press.

[S1] *Hachette. Le dictionnaire universel francophone en ligne*.
http ://www.francophonie.hachette-livre.fr.

[S2] *Oracle Technology Network. Oracle9i Application Server Wireless*.
http://technet.oracle.com/products/iaswe/content.html.

[S3] Alternis S.A. *Solutions for Location Data Mediation*.
http://www.alternis.fr/.

[S4] SignalSoft. *Wireless Location services*.
http://www.signalsoftcorp.com/,

[S5] Composite Capabilities / Preferences Profile,
http://www.w3.org/Mobile/CCPP/