Potacco [Texte imprimé] : noeud polymorphique transparent pour l'adaptation de contenu adapté au contexte

Bertrand Mathieu

Laboratoire d'Informatique de Paris 6

Thèse de doctorat de

l'Université Pierre et Marie Curie, Paris Universitas

Spécialité

INFORMATIQUE, TELECOMMUNICATIONS ET ELECTRONIQUE

Présentée par

M. Bertrand MATHIEU

Pour obtenir le titre de

Docteur de l'Université Pierre et Marie Curie, Paris Universitas

Potacco: nœud POlymorphique Transparent pour l'Adaptation de Contenu adapté au COntexte

Soutenue le 06 Février 2008 devant le jury composé de

Prof. Francine KRIEF	Rapporteur
Prof. Ahmed KARMOUCH	Rapporteur
Prof. Guy PUJOLLE	Directeur de thèse
Prof. Dominique GAITI	Présidente du jury
Prof. Ken CHEN	Examinateur
Dr. Yvon GOURHANT	Examinateur
Dr. Olivier MARCE	Examinateur

Tables des matières

Liste des figures

Remerciements

Après l'obtention de mon diplôme d'ingénieur et de mon DEA, suite à un stage effectué à France Telecom R&D (anciennement CNET) deux possibilités s'offraient à moi : réaliser une thèse ou accepter l'offre d'embauche de France Telecom en tant qu'ingénieur de recherche et développement. La conjecture économique à l'époque n'étant guère favorable et les offres d'emploi peu fréquentes, j'ai décidé d'accepter cette offre. Mais mon choix était toutefois lié au fait que cet emploi se déroulerait dans un centre de recherche de renommée internationale.

Les années passèrent ainsi, de projet en projet, à France Telecom jusqu'à ce que la direction de la recherche de France Telecom propose à ses employés de réaliser une thèse tout en restant sur leur poste actuel et sur leurs activités actuelles. J'ai donc profité de cette opportunité pour continuer les travaux de recherche dans ce cadre et ainsi les valider.

Bien évidemment, ce n'est pas toujours facile de concilier études de recherche et projets internes et notamment ayant un intérêt pour France Telecom, mais c'est toutefois dans cette optique que mes travaux se sont déroulés pendant ces cinq dernières années.

Je remercie donc d'abord France Telecom et principalement la direction de recherche, représenté par Bruno Choquet de m'avoir offert cette possibilité. Je remercie aussi Prosper Chemouil, responsable de pôles de recherche à France Telecom durant ces années.

Je remercie chaleureusement Yvon Gourhant, mon responsable à France Telecom depuis maintenant huit ans, pour son soutien permanent, ses conseils avisés, et pour avoir toujours défendu mes idées.

Je remercie Guy Pujolle, une source de connaissance impressionnante, dans laquelle il fait bon aller puiser et au contact duquel nous apprenons toujours quelque chose, pour avoir accepté de diriger ma thèse, pour avoir soutenu mes travaux en discussions directes ou lors de conférences.

Pour leurs recommandations et lecture pertinente, je tiens à remercier Francine et Ahmed, les 2 rapporteurs de cette thèse.

Je remercie aussi les personnes avec lesquelles j'ai collaboré pour ces études dans le cadre de projets internes à France Telecom, dans le cadre de projets collaboratifs nationaux ou européens, mais aussi les personnes rencontrées lors de conférences internationales, pour leurs retours sur mes travaux qui m'ont permis de progresser.

Enfin, un grand merci à mes collègues de travail pour les discussions toujours intéressantes, les avis partagés, la bonne humeur, et l'utilisation de la cafetière commune: Yannick, Djamal-Eddine, François, Riadh, Pierre, Christophe et tous les autres…

Résumé

Avec l'évolution des réseaux fixes et mobiles, des terminaux de plus en plus nombreux et diversifiés, il est maintenant possible d'accéder à n'importe quel type de service, depuis n'importe quel type de terminal, en étant connecté sur n'importe quel type de réseau. En ajoutant le souhait des utilisateurs de recevoir un contenu personnalisé, l'adaptation de contenu est devenue une problématique majeure.
Cette thèse définit une solution de nœud intermédiaire, flexible permettant l'adaptation dynamique de tout type de contenu en fonction du contexte de l'utilisateur. Ces travaux ont abouti à la définition d'une architecture de nœud, dénommé Potacco pour *nœud **PO**lymorphique **T**ransparent pour l'**A**daptation de **C**ontenu adapté au **CO**ntexte*, à sa mise en œuvre et sa validation. Ce nœud :

- collecte et met à disposition le contexte courant pour permettre aux modules applicatifs de réaliser des adaptations en fonction de ces valeurs
- gère/coordonne les modules applicatifs et les collecteurs de contexte
- permet le déploiement sécurisé de code dans le nœud avec authentification du fournisseur du code, mais aussi du nœud cible
- peut être transparent en réalisant des traitements sans que les points terminaux puissent s'en apercevoir.

Deux démonstrateurs constituent une preuve de concept de ce nœud générique, intégré dans un réseau physique: une passerelle filaire/sans-fil réalisant l'adaptation de contenu média et un nœud dans un réseau ADSL insérant dynamiquement le contexte des utilisateurs.
Ensuite, l'apport de ce nœud dans le cadre des réseaux "overlays" a fait l'objet d'une nouvelle preuve de concept. Deux cas ont été étudiés: la première pour la fourniture de service adapté au contexte de l'utilisateur dans un réseau overlay de service, où un cas d'usage de service d'IPTV personnalisé est présenté; la deuxième relative à l'adaptation de contenu de flux multimédia diffusé sur un réseau P2P où le nœud Potacco est lui-même membre du réseau P2P.
Des évaluations, par simulation et expérimentation réelle, ont permis d'évaluer ces solutions.

Abstract

With the evolution of fixed and mobile networks, the increasing number of diversified devices, it is now possible to access any type of services, from any type of devices, being connected to any type of networks. By adding the wish of users to receive personalized contents, adapting content has become a major problem.
To help to reach this goal, this thesis defines an intermediate flexible node, allowing dynamic adaptation of any type of content depending on the context of the users. This study resulted in the definition of a node architecture, called Potacco (for *nœud **PO**lymorphique **T**ransparent pour l'**A**daptation de **C**ontenu adapté au **CO**ntexte*, in French or Transparent Polymorphic node for content adaptation to the context in English) and its implementation and validation. This node:

- collects and provides current context information to enable application modules to make adaptation based on such values
- manages/coordinates the application modules and the context collectors (e.g. context sensors)
- enables the secured deployment of code in the node (the supplier code, but also the target node are authenticated)
- may be transparent to the applications and may process data without no mean for the endpoints to detect it.

Two demonstrators have been implemented as a proof of concept of this generic node being located in a physical network: as a wired/wireless gateway performing media content adaptation and as a node in an ADSL network inserting transparently the users' context.
Then, the use of that node in "overlay" networks has been the subject of a new proof of concept. Two cases were studied: the first one for the provisioning of services adapted to the user's context in a service specific overlay network service, where a case of a personalized IPTV service is presented, and the second one related to the content adaptation of multimedia stream broadcasted on a P2P network where the Potacco node itself is a member of the P2P network.
Validation by simulation and real experiments, permitted to evaluate these solutions

Chapitre 1
Introduction

1.1 Contexte

Le monde des télécommunications et de l'informatique a subi une véritable révolution lors des quinze dernières années. Bien dissociées auparavant, ces deux communautés se sont rejointes: le monde des télécommunications est maintenant un monde informatique fournissant des services de données et vice-versa, puisque de nombreuses applications informatiques ont pour objectif la fourniture de service de télécommunications (téléphonie sur IP, vidéoconférence…). Cette révolution a entrainé une évolution des réseaux physiques (réseaux câblés, xDSL, FTTH..) mais aussi introduit la possibilité de communiquer et d'accéder à l'internet depuis des réseaux mobiles, maintenant nombreux et différents (GSM, UMTS, Wifi, ad hoc, Wimax, DVB...) en utilisant des terminaux mobiles de plus en plus nombreux et diversifiés (téléphone GSM, UMTS, PDA, tablettes PC...). Avec les dernières générations, certains terminaux peuvent même se connecter sur des réseaux différents, voire basculer d'un réseau à l'autre dynamiquement, comme l'UMA (Unlicensed Mobile Access). L'explosion de l'internet a eu aussi pour conséquence l'apparition d'une multitude de services offerts (Web, messagerie, téléphonie, blogs personnels, partage de photos/vidéos, espace de stockage ou d'échange, jeux en ligne, sites d'achat...). Il est ainsi clair que nous sommes rentrés dans l'époque où il sera possible d'accéder à n'importe quel type de service, depuis n'importe quel type de terminal, en étant connecté sur n'importe quel type de réseau [Kim'03] [Nie'04].
De plus, les utilisateurs eux-mêmes ont changé de comportement et d'exigence et veulent maintenant un service personnalisé, qui offre une qualité satisfaisante et qui est adapté à l'utilisation courante. Le fait de prendre en compte le type de terminal, le réseau d'accès ainsi que les préférences de l'utilisateur forme un ensemble connu sur l'appellation de "context-awareness", ou connaissance du contexte [Pas'99] [Dey'00] [Dey'01] [Anag'02] [Xyn'04]...
Ces états de fait illustrent clairement le besoin d'adaptation dynamique de contenu au contexte des utilisateurs que doivent inclure les services à venir.

1.2 Outils d'adaptation de contenu

La variété de services disponibles offerts sur Internet implique qu'il n'existe pas une solution et une technologie miracle qui permettrait d'adapter le contenu de manière universelle. Au contraire, plusieurs technologies existent ou ont été étudiées pour fournir les informations de contexte ou pour réaliser les adaptations elles-mêmes,

pour les services de type Web ou les applications multimédias par exemple. Cette section vise à introduire rapidement ces solutions.

1.2.1 Format de déclaration du contexte

Pour la déclaration des informations de contexte, plusieurs formats sont définis tels que CC/PP, SDPng, CSCP…

Parmi les solutions les plus avancées, CC/PP (Composite Capabilities / Preferences Profiles) défini au W3C (World Wide Web Consortium) est une des plus connues et préconisées. CC/PP [CCPP1'04] [CCPP2'07], défini dans le groupe initialement nommé CC/PP puis DIWG (Device Independence Working Group) puis maintenant UWAWG (Ubiquitous Web Applications Working Group), permet de décrire les caractéristiques du terminal ainsi que les préférences de l'utilisateur en définissant une structure utilisant la description RDF (Resource Description Framework). CC/PP définit aussi comment cette structure peut être transmise par le terminal utilisateur au serveur [CCPP-Ex'99]. L'utilisation la plus courante est l'insertion du profil dans une requête HTTP ou WSP (Wireless Session Protocol) pour les terminaux mobiles. En effet, CC/PP est préconisé par le W3C pour les applications Web classiques, mais aussi par l'OMA (Open Mobile Alliance), organisme en charge de définir les normes pour les terminaux mobiles, qui a défini un vocabulaire dénommé UAProf [UAProf], basé sur CC/PP, pour décrire les caractéristiques des terminaux mobiles.
De nombreuses études sur l'adaptation de contenu utilisent CC/PP pour fournir les informations relatives au profil utilisateur. Parmi celles-ci, on peut, entre autres, citer [Sur'01], [Gil'03], [Pap'02], [Cou'04], [Yasu'01] et [Ind'03]…
Pour remédier à certaines limitations de CC/PP, notamment la structuration de profils complexes [Held'02] et [Buch'04] ont défini CSCP (Comprehensive Structured Context Profiles), un autre langage de représentation du contexte, plus structuré, basé aussi sur RDF. Dans CSCP, les informations de contexte sont rattachées aux profils de sessions (et donc du contexte de session) des utilisateurs. Cela permet de mieux prendre en compte des informations spécifiques à la session comme les informations réseau.
Les travaux présentés dans [Lem'04] [Leml'03] [Leml'04] sont intéressants, car ils permettent d'étendre l'utilisation de CC/PP en définissant un nouveau moyen de description du contexte (UPS : Universal Profiling Schema) qui prend en compte plus d'informations.

SDP (Session Description Protocol) [SDP] ou SDPng (SDP new generation) [SDPng] [SDPngTrans] est un format de déclaration, principalement utilisé pour les applications multimédias. Il est d'ailleurs recommandé et intégré au protocole SIP (Session Initiation Protocol) [SIP], qui est maintenant le protocole de référence pour l'établissement de communications multimédias (audio et vidéo). SDP est un format textuel simple ou chaque attribut est représenté par une lettre suivi du signe "=" et de la valeur. Les papiers relatifs aux communications multimédias basés sur SIP utilisent principalement SDP ou SDPng, comme [Singh'00], [Ho'01], [Glas'01], [Guen'06], [Guen'05] …

1.2.2 Solutions d'adaptation d'applications Web

Historiquement, l'adaptation de contenu Web commence avec l'arrivée des premiers terminaux mobiles avec un accès Internet et implémentant le protocole WAP (Wireless Access Protocol). Suivant le cas, le contenu Web est encodé en format HTML (Hypertext Markup Language) si le terminal utilisé est un PC par exemple et en format WML (Wireless Markup Language) [WML'99] dans le cas d'un téléphone portable utilisant le protocole WAP au lieu de HTTP (Hypertext Transfer Protocol). Ensuite, le WAP forum, maintenant renommé OMA (Open mobile Alliance) a défini WML2 [WML2'02] pour faire évoluer WML. Dans cette nouvelle version, XHTML Basic [XHTMLBasic'07] est défini et reprend des fonctions de bases de XHTML [XHTML'02] qui sont les mieux adaptées pour présenter le contenu sur des terminaux à capacités limitées comme les terminaux mobiles.

D'autres travaux similaires et relatifs au concept de séparation des données elles-mêmes et de la présentation des données sont définis au W3C (World Wide Web Consortium) et ont abouti à la définition de XML (Extensible Markup Language), maintenant à la 4ème version [XML'06]. L'utilisation de XML simplifie l'adaptation de contenus Web puisque les données sont clairement séparées de la présentation.
Pour la présentation, le W3C a défini CSS (Cascading Style Sheets) [CSS2'98] qui permet de définir comment présenter les données, notamment la police, la couleur…
Ensuite le W3C a défini XSLT (eXtensible Stylesheet Language Transformations) [XSLT'07] et XPath [XPath'07] qui permettent de transformer un document XML en un autre format de type HTML ou XHTML ou autre à partir de feuilles de style bien définies…
L'utilisation conjointe de XML et de XSLT permet donc d'adapter un contenu Web vers différents terminaux, simplement en créant plusieurs feuilles de style correspondant à chaque cible souhaitée. De nombreux papiers de recherche utilisent ces solutions basées sur XML/XLST principalement, dont par exemple [Ha'04], [Kan'04], [Mor'04]...

1.2.3 Solutions d'adaptation d'applications multimédia

SMIL (Synchronized Multimedia Integration Language) [SMIL2] [SMIL2.1], défini au W3C, a pour objectif de permettre la présentation de différents contenus multimédias. Les contenus peuvent être du texte, des images, des sons, des vidéos ou encore des animations …. SMIL permet aux concepteurs du service de définir comment arranger les différents objets médias entre eux, de les relier, de les placer aussi bien dans la dimension spatiale que temporelle ….Dans la version 2.1, des nouveautés permettent de gérer des profils pour les mobiles. De nombreux papiers traitant de l'adaptation et de la composition de différents médias utilisent ou se basent sur SMIL: [Lem'03], [Stee'04], [Chan'05], ….

Les codages hiérarchiques, encore appelés "scalable", [Ohm'05] [Vit'05] [Sik'05] représentent une solution d'adaptation de contenus audio ou vidéo où la caractéristique principale est que le contenu n'est encodé qu'une seule fois, avec le

débit le plus élevé possible et peut être fourni suivant différentes qualités en fonction des capacités du terminal client et des capacités du réseau physique. Ceux qui décoderont la totalité auront une qualité optimale, ceux qui ne décoderont qu'une partie auront une qualité inférieure. L'avantage des flux hiérarchiques est que le contenu n'est encodé qu'une seule fois au lieu d'avoir X contenus différents, encodés et stockés X fois pour pouvoir être délivré dans X configurations différentes de qualité. En effet, les systèmes de codage "hiérarchiques" encodent les signaux multimédias (audio et/ou vidéo) en fournissant un flux binaire composé de couches successives. La couche de base, encore appelée "cœur", est formée des éléments binaires absolument nécessaires au décodage du train binaire, et déterminant une qualité minimum de décodage. Les couches suivantes permettent d'améliorer progressivement la qualité du signal issu de l'opération de décodage, chaque nouvelle couche amenant de nouvelles informations, qui, exploitées par le décodeur, fournissent en sortie un signal de qualité croissante.

L'une des particularités des codecs hiérarchiques est la possibilité d'intervenir à n'importe quel niveau de la chaîne de transmission pour supprimer une partie du train binaire sans devoir fournir d'indication particulière au codeur ni au décodeur. Le décodeur utilise les informations binaires qu'il reçoit et produit un signal de qualité correspondante. Ceci peut donc se faire aussi bien au niveau du serveur, du client que d'un élément intermédiaire.

Des types de codage hiérarchiques sont les codecs G.727 (16/24/32/40 kbit/s) [G727] défini à l'ITU-T, CELP-TDAC [Kov'04] [Tad'99] pour l'audio et les codecs MPEG4 CELP-AAC [Grill'97], H264-AVC (Advanced Video Coding) [H264AVC'03] [Wie'03] [AVC'05] pour la vidéo…

MPEG-21 [MPEG-21] est un framework défini au MPEG (Moving Picture Experts Group) forum [MPEG] visant à simplifier la fourniture de services multimédia. L'objectif est de couvrir toute la chaine de diffusion: du producteur au consommateur final (utilisateur) en passant par d'éventuels éléments intermédiaires.
Dans MPEG-21, 2 concepts primordiaux ont été définis: le DI (Digital Item), qui est un objet numérique représentant un contenu multimédia, entité fondamentale dans MPEG-21, et qui est traité par les "users" et le terme "user" (utilisateur) qui fait référence à n'importe quelle entité interagissant avec le framework MPEG-21 ou réalisant des actions (créations, traitements, adaptations…) sur les DIs. Un "user" est donc aussi bien le fournisseur du contenu, que l'utilisateur final, qu'un élément tiers agissant sur les DI de ce contenu.
MPEG-21 recouvre de nombreux domaines et est défini en plusieurs parties. Dans le cas de cette thèse, ce qui nous intéresse est la partie 7 de la spécification MPEG-21, nommé " Digital Item Adaptation (DIA)" [DIA], relative à l'adaptation de contenu. Il n'est pas spécifié dans la norme quelle adaptation réalisée ni comment la réaliser, mais ont été définis deux moteurs ("engine"): le "resource adaptation engine" et le "resource description engine" et des outils qui permettent de réaliser les adaptations: le "Usage Environment Description Tools" (UED), le "Digital Item Resource Adaptation Tools" et le "Digital Item Declaration Adaptation Tools".
De nombreux travaux de recherche actuels relatifs à l'adaptation de contenu multimédia se basent sur MPEG-21. Parmi les plus courants, on retrouve [Kazi'04] où un cas d'usage utilisant MPEG-7 et MPEG-21 est illustré, un autre [Wolf'04] où les auteurs proposent d'utiliser conjointement MPEG-21 et SDPng [Vetro'05] qui

présente quelques études sur le sujet ou [Hutt'05] qui présente un cas d'adaptation en fonction du contexte.
En parallèle à ces mécanismes d'adaptations applicatives sont apparues les études relatives au fonctionnement inter-couches (cross-layer approach) [VdS'05] [Kaw'05]. Pour l'adaptation dynamique, plusieurs études visent à prouver l'intérêt de l'approche inter-couches [Gros'04] [Shan'02]. Pour MPEG-21 plus précisément, plusieurs travaux ont été abordés pour fournir des informations de niveaux réseaux aux modules applicatifs par une approche intercouches. Parmi ces travaux, il est possible de citer [Xu'06], [Ahm'03], [Ahm'06]…

Les solutions présentées précédemment sont des solutions ayant rapport avec des organismes de normalisation et donc à vocation à être plus utilisées ou déployées. Cependant, de nombreux papiers de recherche sont relatifs à l'adaptation de contenus et proposent des solutions spécifiques, notamment par l'utilisation de transcodeurs intermédiaires entre le serveur et le terminal utilisateur. Certains auteurs proposent leurs propres transcodeurs [Leo'04], [Xie'02], mais beaucoup basent leurs démonstrateurs sur des produits existants tels que JMF (Java Media Framework) [JMF], FFMPEG [FFMPEG] ou encore Live555 [Live555] comme par exemple [Curr'05], [Bell'03], [Hash'03], [Scho'06], [Bosz'07], [Agh'03]…

1.2.4 Bilan sur les solutions d'adaptation

Pour conclure cette section, on remarque qu'il existe plusieurs solutions et formats de déclarations de contexte utilisateur et aussi plusieurs solutions et mécanismes permettant de réaliser l'adaptation de contenus, chacune étant relative à son cadre d'application. Ceci induit qu'il faudra une coexistence de solutions et d'architectures différentes pour permettre l'adaptation de contenu pour tous les types de service, ce qui n'est guère intéressant. Essayer de définir une nouvelle solution générique qui pourrait prendre en compte tous les cas, tous les environnements et qui soit applicable pour toutes les applications, actuelles et futures, semble peu probable. L'idée qui ressort est donc qu'il faudrait pouvoir définir un environnement d'exécution où ces différentes solutions pourraient être mises en œuvre indépendamment d'une configuration donnée, en définissant une architecture ouverte qui puisse offrir la possibilité de déployer et activer différentes solutions sur ce nœud en fonction des besoins.

1.3 Où localiser ces modules d'adaptation ?

La question qui se pose concernant la mise en œuvre des modules d'adaptation est de savoir où ils doivent être localisés. En effet, l'adaptation peut être réalisée à divers points de la chaine de livraison du service: sur le serveur lui-même, sur le client lui-même ou sur un nœud intermédiaire. Une analyse rapide de ces différents points possibles est ici faite.
Avoir un mécanisme d'adaptation sur le serveur lui-même peut simplifier l'architecture réseau, mais cela introduit des mélanges de rôles entre les entités

impliquées dans la fourniture du service, des problèmes de passage à l'échelle de la solution et de performance du service global si de nombreux utilisateurs se connectent au serveur. Bien évidemment, une solution pour y remédier est de rajouter des capacités physiques (voire des serveurs eux-mêmes) mais cela a un coût non négligeable et le problème de passage de l'échelle pourra toujours survenir un jour ou l'autre. De plus, on peut imaginer que l'adaptation de contenu puisse intégrer des contenus de sources différentes dans la même présentation. Par exemple, afficher le message SMS reçu ou une annonce publicitaire dans un bandeau en bas de l'écran d'une vidéo que l'utilisateur est en train de visualiser. De ce fait, avoir le mécanisme d'adaptation sur les serveurs peut limiter cette adaptation puisque le contenu peut provenir de plusieurs sources différentes.
Enfin, avec l'émergence des réseaux P2P (Peer-to-Peer) pour tout type de service, et donc la mise en relation directe entre utilisateurs ou en passant par d'autres utilisateurs, il n'y a plus de serveur précis dans l'architecture et donc il est impossible de réaliser une quelconque adaptation dans un tel environnement si la solution retenue consiste à la réaliser sur un serveur dédié.

Réaliser l'adaptation sur le client peut s'avérer une solution intéressante à mettre en œuvre. Cela peut être le cas pour certains terminaux, actifs sur certains réseaux, ayant des capacités suffisantes. Elle doit donc être prise en compte dans certaines configurations, mais cette solution pas suffisante par elle-même puisqu'il a été dit, que les terminaux utilisateurs peuvent avoir des caractéristiques très différentes et donc certains peuvent être très limités et incapables de réaliser les adaptations nécessaires. De la même manière, l'utilisateur se connectant depuis n'importe quel réseau d'accès, en cas de réseau avec une bande passante disponible très limitée, des pertes de paquets ou des corruptions de données peuvent intervenir durant le transfert jusqu'au terminal et l'adaptation sur le terminal peut être ainsi erronée. De plus, cette solution ne permet pas du tout de diminuer l'utilisation du réseau pour accéder à cet utilisateur, puisque toutes les données y sont envoyées, alors que le réseau peut être déjà bien utilisé.

La localisation de ces modules sur des nœuds intermédiaires, localisés entre le client et le serveur peut se présenter comme une solution satisfaisante. En effet, il est possible de déployer ces mécanismes sur différents nœuds des réseaux physiques, à différentes localisations géographiques, par exemple proche des réseaux d'accès des utilisateurs, comme dans les Points de Présence des réseaux ADSL, les points d'accès pour des hotspots Wifi, ou encore des nœuds proches des utilisateurs, avant les contraintes physiques. Du fait de leur distribution dans le réseau, la charge de traitement est mieux répartie que si toutes les adaptations étaient à réaliser sur un serveur et la performance rendue peut être meilleure.
De plus, si ces nœuds de traitements sont localisés à certains points stratégiques du réseau, comme proche des réseaux d'accès de l'utilisateur afin d'optimiser l'utilisation de leurs ressources (en particulier dans le cas des mobiles, car la ressource radio est coûteuse), un module applicatif activé sur le nœud intermédiaire peut permettre de réduire le débit d'un flux ou modifier le contenu pour utiliser moins de bande passante permettant ainsi d'optimiser l'utilisation du réseau d'accès tout en assurant la fourniture d'un service acceptable pour l'utilisateur. L'opérateur ayant une connaissance de ses réseaux, de la topologie physique, des flux et des éventuels

points d'engorgement a un rôle à jouer dans le déploiement de ces nœuds intermédiaires en choisissant l'emplacement pertinent. Dans cette même optique, avoir le nœud intermédiaire localisé à un endroit stratégique du réseau, comme par exemple sur le « chemin » IP entre les deux points terminaux permet d'optimiser l'échange de données et ainsi améliorer les temps de réponse.
Ces nœuds intermédiaires peuvent aussi réaliser des traitements applicatifs sur les flux afin de personnaliser le contenu (adaptation au contexte bien sûr mais aussi ajout de publicité ciblée, ajout d'information telle que la réception de SMS, issu d'un autre serveur…). Cela peut réduire l'utilisation du cœur du réseau et des traitements des serveurs, puisque ceux-ci peuvent envoyer par exemple toujours le même contenu, la personnalisation étant faite sur le nœud intermédiaire ; ceci est particulièrement intéressant pour les flux diffusés, comme un flux vidéo Live par exemple, qui pourraient être émis en multicast vers les différents nœuds intermédiaires de traitement. Il est aussi possible d'imaginer des fonctions de cache associées à ces nœuds intermédiaires qui permettent ainsi d'optimiser le temps de réponse et donc la qualité fournie à l'utilisateur.
Finalement, pour rejoindre le cas précédent, ces nœuds intermédiaires pourraient aussi être des nœuds utilisateurs, ayant des capacités suffisantes et supportant un environnement d'exécution hétérogène permettant la mise en œuvre de modules d'adaptation qui réaliseraient l'adaptation pour d'autres nœuds utilisateurs, ayant des capacités limitées.
Pour ces différentes raisons, c'est cette solution de nœud intermédiaire de traitement qui a été retenue dans cette thèse.

1.4 Objectifs de la thèse

L'objectif principal de cette thèse est de permettre l'adaptation dynamique de contenu en fonction du contexte utilisateur par un nœud intermédiaire; le contexte des utilisateurs regroupant le contexte courant des réseaux physiques, les caractéristiques des terminaux, les préférences des utilisateurs ainsi que les politiques d'adaptation des fournisseurs de contenus. Le nœud intermédiaire doit être capable de connaître, détecter, analyser des informations de contexte et de les mettre à disposition des modules d'adaptation (si besoin) pour effectuer l'adaptation recherchée. Pour cela, il est nécessaire d'intégrer des modules de détection de contexte dans le nœud et de définir un composant en charge de gérer les communications intercouches et notamment les relations entre les différents modules de contexte et les modules applicatifs. De plus, selon sa localisation, le nœud peut être connecté à plusieurs réseaux physiques (comme une passerelle par exemple). Il faut donc prévoir des modules de routage et d'interception des paquets pour les "remonter" aux modules d'adaptation avant réémission. Ce module d'interception doit être dynamiquement configurable puisque les modules d'adaptation peuvent être activés seulement selon certains critères (par exemple bande passante saturée). La définition de l'architecture d'un nœud intermédiaire ayant cette vocation fait partie du premier objectif de cette thèse.

Le nœud intermédiaire peut être utilisé en tant que passerelle entre deux réseaux ou en tant que proxy. Cependant, certaines applications n'utilisent pas de tel proxy et

l'utilisation d'un proxy implique que la connexion entre le serveur et le client est "coupé en deux": une connexion entre le client et le proxy et une entre le proxy et le serveur. Or il peut exister des cas où la configuration dynamique de proxy peut ne pas être possible sur certains terminaux (par exemple une Set top box) ou encore que le chaînage de proxies entre le client et le serveur (chaque proxy effectuant une action spécifique) n'est pas toujours réalisable, notamment pour une configuration dynamique où le chaînage peut être différent selon le contexte des utilisateurs. Il est aussi envisageable d'avoir des utilisations ou la connexion de bout en bout est nécessaire et ne doit pas être coupée. Par exemple, cela peut être le cas pour les services de police ou de renseignements qui possèdent l'autorité pour connaitre des informations issues de certaines personnes ou l'échange de contenus illicites. Avoir un module intermédiaire, non détectable, localisé entre 2 nœuds terminaux permettrait ainsi de pouvoir filtrer, analyser ou réaliser le service d'interception légale. Le nœud intermédiaire doit donc pouvoir fonctionner, en mode proxy, mais aussi fonctionner sans être détectable et conserver cette notion de connexion de bout en bout tout en offrant la possibilité de réaliser des traitements intermédiaires. Dans ce mode de fonctionnement, le nœud est dit "transparent" car il réalise des traitements sur les données sans que les points terminaux (le serveur ou les utilisateurs clients) ne puissent s'en rendre compte. Cela décrit le deuxième objectif de cette thèse qui est que le nœud doit pouvoir être transparent aux entités aux extrémités des connexions.

La section 1.2 a montré que différents modules d'adaptation existent, avec différentes contraintes et différents requis, pour des cas d'usage variés et des applications différentes. La conclusion qui en est ressortie est qu'il fallait définir une architecture de nœud dans lequel de tels modules d'adaptation pourraient être dynamiquement déployés, instanciés, mis à jour et supprimés lorsqu'ils ne sont plus utiles ou utilisés. Cela revient à dire que le nœud doit offrir un environnement d'exécution ouvert et programmable.
De plus, l'adaptation peut être relative aussi bien aux contenus Web, qu'aux applications multimédias, voire des services de type messagerie. Ainsi, il est possible que plusieurs entités de service puissent proposer des mécanismes d'adaptations. Il faut prévoir dans la solution un mécanisme pour sécuriser le déploiement du code et surtout authentifier les parties en jeu, qui peuvent être mobiles et changer de réseau physique. En effet, il faut pouvoir authentifier l'entité tierce, fournisseur du code d'adaptation, mais il faut aussi s'assurer que le nœud sur lequel le code sera déployé est bien le bon et non pas un nœud malicieux, désirant récupérer le code d'un fournisseur tiers, ou tout simplement un autre nœud. Ces nœuds pouvant être mobiles, la solution devra s'affranchir de relation avec les réseaux physiques telle que l'adresse IP. Cela définit ainsi le troisième objectif de cette thèse qui consiste en un mécanisme de sécurisation associé au déploiement de code des modules applicatifs.

Finalement, le nœud doit pouvoir être utilisé dans plusieurs modes d'utilisation; par exemple en tant qu'équipement réseau, localisé dans le réseau et impliqué dans le transfert des données comme une passerelle sans-fil/filaire ou un nœud dans un réseau de collecte (en mode proxy ou en mode transparent), ou encore en tant que nœud de réseau overlay, jouant un rôle dans la distribution des contenus (il faut noter que le nœud du réseau overlay peut être aussi bien un nœud du réseau physique

qu'un nœud terminal utilisateur). Ainsi, la solution décrite devra permettre l'utilisation de nœud intermédiaire dans différentes configurations réseau.

1.5 Travail Réalisé

Les travaux présentés dans cette thèse sont relatifs à la fourniture de contenu et leur adaptation dynamique au contexte de l'utilisateur, mais ne portent pas eux-mêmes sur une solution spécifique d'adaptation. Ils visent plutôt à la définition d'une architecture de nœud intermédiaire, transparent ou non, capable de connaître le contexte courant relatif à l'utilisateur et permettant de déployer de manière sécurisée et d'instancier des modules d'adaptations de contenus en fonction du contexte. Ce nœud a été dénommé Potacco pour *nœud **PO**lymorphique **T**ransparent pour l'**A**daptation de **C**ontenu adapté au **CO**ntexte*. L'appellation de "nœud polymorphique" a été définie pour ce nœud, car il peut prendre plusieurs formes, vis-à-vis des réseaux (proxy, transparent, nœud dans un réseau overlay) et vis-à-vis des services qu'ils supportent (différents modules de traitement peuvent être déployés, activés, supprimés à différents instants).
Parmi les travaux réalisés dans cette thèse, on retrouve :

- La notion de collecte et de mise à disposition du contexte courant pour permettre aux modules applicatifs déployés sur le nœud de réaliser des adaptations de données en fonction de ces valeurs. Des modules de détection de contexte, appelés collecteurs et sondes, ont été définis pour évaluer des informations localement ou pour récupérer des informations de contexte externes, telles que des informations sur la topologie réseau ou encore fournies par les fournisseurs de services.
- La notion de communication inter-couche entre les collecteurs et sondes qui supervisent l'état de certaines ressources et les modules applicatifs via un gestionnaire central qui coordonne les actions. Ce composant gère les abonnements des modules applicatifs à certaines sondes et informe les modules lorsque les valeurs courantes sont supérieures à certains seuils, définis par les applications. Ce composant a aussi en charge de gérer les paquets interceptés et remontés aux modules applicatifs pour adaptation. Un composant d'interception ou de redirection directe des paquets est ainsi intégré.
- La notion de nœud transparent qui permet au nœud polymorphique de réaliser des traitements sans que les points terminaux puissent s'en apercevoir. Pour fonctionner de manière transparente, des modules permettant la gestion des connexions IP et le maintien des champs de contrôle sont ajoutés à l'architecture du nœud Potacco.
- La notion de déploiement sécurisé de code dans le nœud polymorphique, où le fournisseur du code mais aussi le nœud cible sont identifiés, indépendamment de leur identifiant IP.

Deux démonstrateurs ont été réalisés pour montrer les différentes possibilités d'intégration de ce nœud dans un réseau physique (point d'accès Wifi évolué et nœud intégré dans une chaine de collecte ADSL) ainsi que deux autres démonstrateurs illustrant la mise en œuvre de ce nœud à d'autres infrastructures réseau, notamment

dans le cadre des réseaux overlays (fourniture de services adaptés au contexte de l'utilisateur dans un réseau overlay de service et adaptation de contenu de flux multimédia diffusé sur un réseau P2P).
Ces démonstrateurs ont été réalisés en tant que preuve de faisabilité de la solution et des évaluations, par simulation et expérimentation réelle, permettent d'évaluer leur comportement.

Ce rapport de thèse est découpé en 7 chapitres.
Le chapitre 2 présente un état de l'art de différentes solutions de nœuds intermédiaires.
Le chapitre 3 présente l'architecture du nœud intermédiaire permettant d'intercepter les paquets et de permettre l'adaptation de contenu en fonction d'informations de contexte réseau fournies par des collecteurs et sondes. Un cas d'usage possible de ce nœud polymorphique dans une situation où le nœud est utilisé en tant que point d'accès Wifi évolué est présenté.
Le chapitre 4 introduit un composant permettant au nœud intermédiaire de fonctionner en mode transparent et intègre un environnement d'exécution dans le nœud, associé à un mécanisme de déploiement de code pour activer divers modules d'adaptation. Cette solution est évaluée par l'intégration du nœud Potacco en tant qu'élément dans une architecture ADSL pour fournir le contexte utilisateur aux serveurs de contenus.
Le chapitre 5 décrit une évolution de ce nœud polymorphique qui n'est plus utilisé en tant que composant réseau mais en tant que nœud d'un réseau overlay, permettant pour la fourniture de services adaptés au contexte de l'utilisateur. Un démonstrateur de service d'IPTV personnalisé est présenté en tant que preuve de faisabilité.
Le chapitre 6 introduit l'adaptation de ce nœud dans une chaine de distribution de contenu multimédia basée sur une approche P2P (Peer-to-Peer) où l'adaptation est réalisée par les pairs du réseau eux-mêmes, et où la mise en relation de ces nœuds pour former le réseau P2P lui-même est basée sur les informations de contexte, incluant les informations relatives aux réseaux.
Enfin, le chapitre 7 conclut et indique les pistes d'évolution ou de recherche relatives à cette thèse.

Chapitre 2

Etat de l'art des solutions de nœud intermédiaire de traitement

2.1 Etat de l'art

Cette section présente des architectures et des solutions de nœud intermédiaire permettant de réaliser un traitement applicatif sur les paquets circulant sur le réseau entre le serveur et les clients.

- Proxy

La première solution qui vient à l'esprit en parlant de nœud intermédiaire est celle de Proxy (ou mandataire en français). La notion de proxy est apparue avec l'explosion de l'Internet et des services Internet. Les proxies sont souvent utilisés comme élément permettant de cacher de l'information issue du serveur. Cette fonctionnalité de cache permet d'améliorer le temps de réponse vis-à-vis des utilisateurs. Cependant, un tel proxy est limité en fonctionnalité et ne répond pas à nos besoins d'adaptations dynamiques de contenus.
De plus, dans une architecture de type Proxy, la communication entre le serveur et l'utilisateur est coupée en deux: une connexion entre le serveur et le proxy et une entre le proxy et l'utilisateur. Il n'y a donc pas de possibilité de réaliser des traitements de manière transparente et l'utilisateur ainsi que le serveur sont conscients de l'utilisation d'un nœud intermédiaire. Il existe maintenant des proxies dits transparents [Cha'96] [Coh'99] [Rod'01], mais dans ce cas, la transparence signifie que les requêtes utilisateurs destinées au serveur sont interceptées par le proxy transparent. Le proxy ensuite initie une requête similaire vers le serveur et renverra la réponse au client ensuite. Mais dans ce cas, le proxy transparent ne réalise pas de fonctions d'adaptations et se limite à intercepter les requêtes utilisateurs et invoquer le serveur. La transparence ne s'applique donc qu'aux utilisateurs, pas aux serveurs et est limité car il n'est pas vraiment transparent puisqu'en analysant les adresses IP et les en-têtes des messages http, il est possible de détecter que la réponse provient d'un autre élément que le serveur. De plus, le proxy transparent n'inclut pas de mécanismes des gestions des connexions puisqu'il ne modifie pas le contenu.

- ICAP

Les proxies étant limités en fonctionnalité, le forum ICAP a défini ICAP (Internet Content Adaptation Procotol) [ICAP'01] pour étendre les fonctionnalités

intermédiaires et se focaliser sur l'adaptation de contenu Internet; l'adaptation pouvant consister en une traduction de pages Web, en l'insertion de publicité, en un filtrage de pages Web (pour faire du contrôle parental par exemple) …
Voulant se rattacher à l'architecture Internet existante, les concepteurs d'ICAP ont défini dans l'architecture un élément tiers, nommé serveur ICAP, qui est relié au proxy. C'est ce serveur ICAP qui est chargé de l'adaptation de contenu.
ICAP est défini comme un protocole de communication basé sur HTTP (notamment les requêtes GET et POST) entre le serveur d'origine, les proxies et les éléments réseau d'adaptation, les serveurs ICAP.
ICAP a défini deux modes d'utilisation. Le premier consiste à rediriger la requête de l'utilisateur (le proxy redirige la requête vers le serveur ICAP) pour insérer des informations dans la requête HTTP (par exemple insérer le profil de l'utilisateur). Ainsi, le serveur peut éventuellement réaliser l'adaptation lui-même. Le deuxième cas d'utilisation est propre à la réponse. Quand il reçoit la réponse du serveur, le proxy redirige celle-ci vers le serveur ICAP qui modifie le contenu de la réponse et la renvoie au proxy. Le proxy ensuite renvoie la réponse modifiée à l'utilisateur.
La figure suivante (Figure 2. 1) présente l'architecture ICAP où un serveur ICAP est connecté à un proxy (qui est aussi un client ICAP).

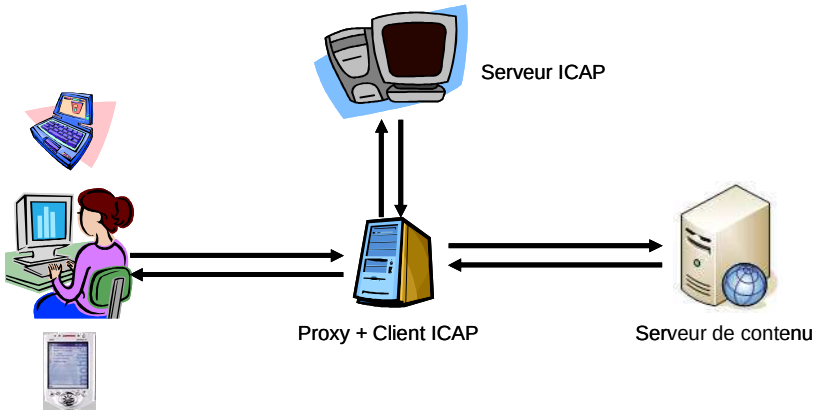

Figure 2. 1: Architecture Proxy/Serveur ICAP

ICAP a été initialement défini par le forum ICAP, composé d'industriels et principalement de vendeurs de solutions relatives aux services Internet. Ensuite a été créé le groupe de travail OPES (Open Pluggable Services) à l'IETF (voir ci-dessous), qui a pour objectif de définir une solution proche de celle définie par le forum ICAP mais plus générique. ICAP a donc été vu comme un précurseur et une instanciation possible d'OPES et ils ont donc décidé de créer un RFC (Request For Comments) pour ICAP [ICAP'03].

ICAP est utilisé dans divers papiers de recherche comme par exemple dans [Lee'03] qui a pour objectif l'adaptation de services sans-fil ou [Pai'03] pour un proxy configurable ou encore [For'06] qui vise à faire de la classification et du filtrage de contenu...

ICAP est donc une solution très liée à HTTP et notamment aux applications Web; elle n'est pas adaptée aux applications multimédias. De plus, les traitements d'adaptations sont statiques, il n'est pas possible de les déployer dynamiquement. Comme dans le

cas des proxies, cette solution ne gère pas les connexions de manière transparente. Finalement, ICAP est une solution de niveau applicative, pouvant prendre en compte un profil utilisateur, mais ne définit rien en ce qui concerne l'acquisition et la mise à disposition du contexte réseau.

- OPES

Le groupe de travail OPES (Open Pluggable Services) [OPES] de l'IETF est le successeur des groupes P1520 et Forces, qui visaient à définir des architectures de réseaux ouverts. OPES est donc dans la continuité et définit une architecture et un protocole permettant d'avoir des éléments intermédiaires dans le réseau réalisant des adaptations.
OPES a pour objectif de définir un framework et un protocole pour invoquer des services applicatifs distribués, localisés entre le serveur d'origine et les clients.
Pour cela, OPES définit des entités, qui sont en charge de réaliser des traitements sur les flux de données dans le réseau, des flux qui circulent entre les entités et des règles qui régissent quand et comment activer les services OPES.
OPES a défini le protocole OCP (OPES Callout Protocol) [OCPReq] [OCPCore] pour communiquer entre nœuds et pour éventuellement distribuer la responsabilité de l'exécution d'un service sur un nœud distant, qui sera chargé de réaliser l'adaptation requise.
OCP est un protocole indépendant des protocoles sous-jacents. Pour l'instant, deux utilisations possibles ont été spécifiées: l'utilisation d'OPES sur HTTP [OCPHTTP] pour l'adaptation de contenu de Web et l'utilisation sur SMTP [OCPSMTP] pour filtrer des messages, rediriger, bloquer...
Plusieurs papiers ont utilisé ou étendu OPES, tels que [Fal'03][Shin'05][Lee'06]...

Comme ICAP, OPES vise à étendre l'utilisation du mode proxy et donc ne peut pas fonctionner de manière transparente, vis-à-vis du serveur ou des utilisateurs. De plus, comme ICAP, rien n'est défini concernant le contexte réseau ou le contexte de manière générale.
Enfin, les problématiques de sécurité sont abordées dans la description et certains requis décrits mais aucune solution n'est vraiment donnée.

- DPI

Les DPI (Deep Packet Inspection) sont des équipements réseau capables d'analyser un flux applicatif (niveau 7 des couches OSI) en temps réel et de prendre des actions associées. Typiquement, l'analyse de flux peut se faire sur les numéros de ports (comme le faisaient déjà les analyseurs réseau de niveau 3) mais aussi à partir de signatures présentes dans les données applicatives. Ces équipements, sont basés sur une architecture hardware performante permettant cette analyse et la décision adéquate en temps réel. Des fournisseurs de DPI sont par exemple Sandvine, Allot, Cisco, Narus...

Malheureusement, pour l'instant, ces équipements sont capables uniquement de transférer les paquets vers une entité tierce ou faire un traitement très succinct (supprimer, favoriser, marquer le paquet…).
De plus, aucune faculté de déploiement de code n'est présente sur ces équipements pour l'instant.
Les DPI sont donc une première étape vers des éléments réseau plus intelligents, capables de réaliser des traitements sur des flux applicatifs et peuvent évoluer à l'avenir pour intégrer des mécanismes tels que ceux décrits dans cette thèse.

- ALAN

ALAN (Application Layer Active Network) [Fry'99] est une plate-forme de réseau programmable [Tenn'97][Camp'99] spécialisée dans les traitements de niveau applicatif, appelé parfois plate-forme de services actifs. En effet, les concepteurs ont plutôt orienté leur développement pour réaliser des traitements applicatifs et non des traitements de niveaux réseaux.
L'architecture d'ALAN (Figure 2. 2) est constituée des principaux éléments suivants :

- Les "**Proxylets**" représentent du code et des données sous forme d'archive java (JAR) qui sont référencés par une URL et qui peuvent être téléchargés dans un serveur d'exécution (DPS: Dynamic proxy Server) depuis des serveurs web où ils sont stockés. Des vérifications sont effectuées avant le téléchargement.
- Les **Serveurs DPS** (Dynamic Proxy Server) sont des nœuds sur lesquels s'exécutent les proxylets pour le compte d'une entité tierce. Dans la dernière version d'ALAN, ces entités sont appelées EEP (Execution Environment for Proxylets). Le serveur maintient un cache des proxylets afin de réduire les téléchargements.
- Les **Interfaces de contrôle** (Control Interfaces) sont utilisées pour initier le téléchargement de proxylets dans un serveur DPS et les configurer ou reconfigurer en cours d'exécution. Il est possible de développer des interfaces de contrôle spécifiques à des proxylets pour interagir avec les utilisateurs.
- Les **Serveurs de protocole dynamique** (Dynamic Protocol Servers) stockent des composants logiciels réalisant une pile protocolaire téléchargeable.

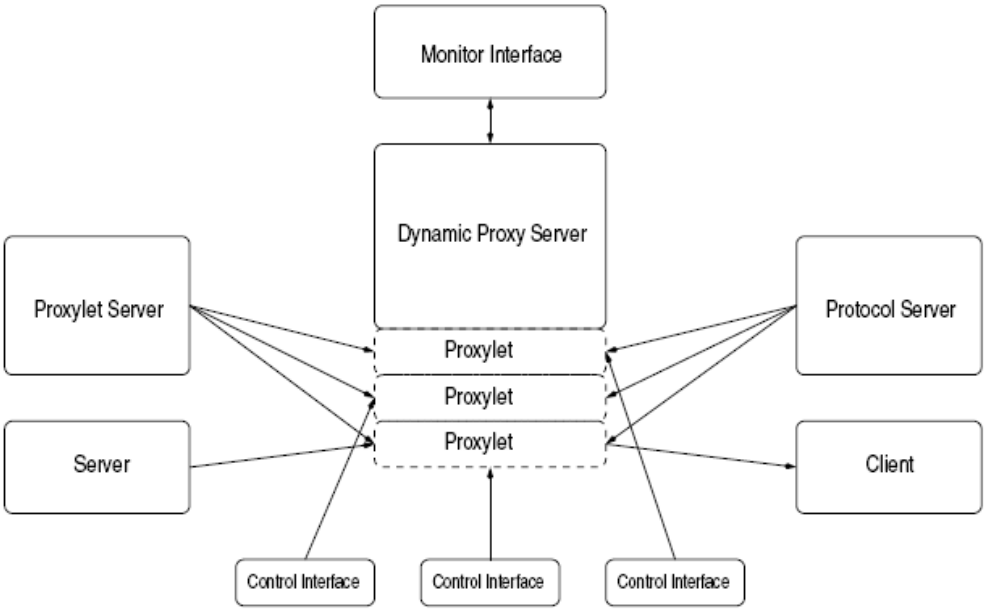

Figure 2. 2: Architecture ALAN

Cette solution présente donc les avantages de déploiement de modules applicatifs dans des nœuds connus, fonctionnalité non présente dans les solutions précédentes. Par contre, il n'y pas de mécanismes de sécurité pour authentifier les serveurs et les nœuds lors du téléchargement. De plus, ALAN ne fonctionne pas en mode transparence mais en mode proxy puisque les connexions sont distinctes. ALAN se limitant à offrir l'environnement applicatif, aucun module de détection ou d'acquisition du contexte n'est présent dans la plate-forme.

- FAIN

FAIN ("Future Active IP Networks") [Gal'00] [Gal'04] est une plate-forme définie dans un projet IST du 5ème PCRD. Le projet a conçu et développé une architecture générique de réseau programmable comprenant des nœuds multi environnement d'exécution, un mécanisme de déploiement dynamique de services dans les nœuds et une plate-forme de gestion à base de politiques.

FAIN se base sur le modèle d'affaire et de rôles défini par le consortium TINA. TINA [TINA] a été définie dans les années 1990 et propose un modèle de rôles entre producteurs, consommateurs et médiateurs via des interfaces définies.

FAIN a défini un mécanisme de déploiement de code [Sol'02] selon une approche dite « out-of-band », ce qui signifie que le code (traitement à réaliser sur les paquets) est déployé par un chemin différent que celui utilisé pour les données. Néanmoins une approche "in-band" ou combinée est possible dans l'utilisation (l'approche « in-band » signifie que le code est véhiculé dans les paquets de données eux-mêmes. Cette approche peut aussi se retrouver sous le nom d'approche « capsule », initialement présent dans la description de la plate-forme ANTS [Wet'98]; première plate-forme de réseaux actifs qui définit des capsules contenant le code à transférer).

L'architecture FAIN (Figure 2. 3) peut se représenter sous la forme d'un cube qui schématise trois plans : transfert, commande et gestion car les concepts développés dans FAIN peuvent s'appliquer aux trois plans (déploiement de code, management...)

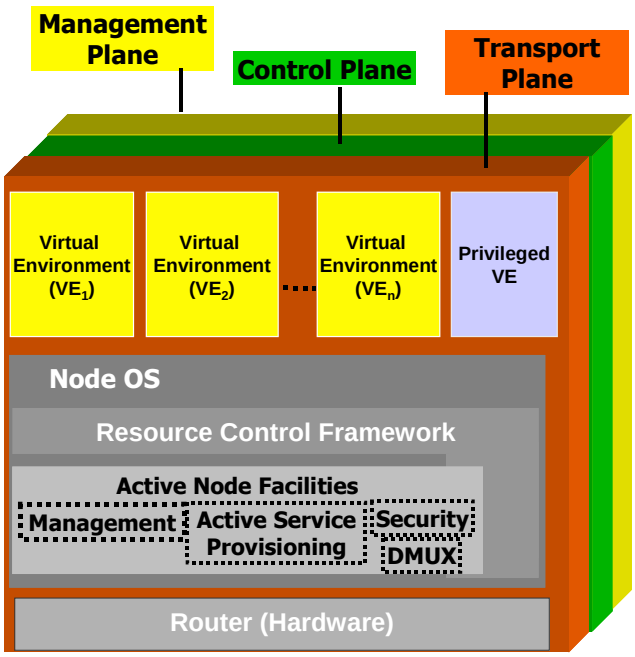

Figure 2. 3: Architecture de référence de FAIN

L'architecture du nœud actif FAIN est basée sur un noyau (NodeOS) autour duquel ont été développés les composants suivants :

- Security : Il contrôle l'intégrité des communications réseau et fournit un service d'authentification pour les paquets actifs et gère les autorisations pour l'accès au noyau. Ces règles sont réalisées par des politiques de sécurité gérées par une autorité principale.
- Resource Control Framework (RCF) : Il reçoit les demandes pour allouer les ressources informatiques et réseaux à différents environnements virtuels (VEs). Il assure également que les VEs sont isolés l'un de l'autre.
- Demultiplexing (Dmux) : Il est responsable de transmettre les paquets actifs aux environnements d'exécution (EEs) correspondants dans les VEs, basé sur l'en-tête du paquet (par exemple, en-tête ANEP).
- Active Service Provisioning (ASP) : Il est responsable du déploiement des services actifs dans les nœuds programmables et de leurs composants de service dans les EEs appropriés.
- Management : Il représente les composants de management (comme les PDP, PEP) responsables de la gestion du nœud actif lui-même et des services déployés
- Un Virtual Environment Manager (VEM) a été développé pour fournir une abstraction de l'environnement d'exécution facilitant le déploiement de service, la configuration, l'intercommunication… En outre, le VEM inclut les facilités qui aident le système de management à imposer ses politiques, par exemple, surveillance des ressources, événements, instanciation de VE, etc.

FAIN présente un environnement assez complet, intégrant un mécanisme de déploiement de code performant. Des mécanismes de sécurité sont aussi définis, mais principalement pour le contrôle des paquets et l'accès aux nœuds et moins relatif au déploiement de code. Malheureusement, il manque aussi les modules relatifs au contexte de l'utilisateur qui permettrait de fournir aux applications les informations de contexte pour qu'elles puissent adapter les services. Ce nœud pourrait être une base intéressante pour cette thèse en l'étendant et l'adaptant pour satisfaire les requis. Cependant après évaluation, cette solution s'est avéré assez complexe, lourde à mettre en œuvre (divers modules et environnements variés sont nécessaires dont des agents mobiles, des interfaces CORBA ...) et donc impossible à instancier sur des nœuds à environnements limités tels que des PDAs ou équipements légers.

- OSGi

Le consortium OSGI (Open Services Gateway Initiative) [OSGI] est un groupe de plus de 80 membres, fondé en 1999, qui vise à définir une plate-forme et un framework (Figure 2. 4) permettant le déploiement des services sur une gamme étendue des réseaux, des WAN à LANs en passant par les réseaux domestiques [OSGI_WP][OSGI_Spec]. Ajouter un framework OSGi à des équipements réseau permet de leur adjoindre la capacité de gestion de services. Les services sont déployés sur des équipements comme une passerelle domestique, embarqués dans des véhicules, des serveurs, des téléphones mobiles...
Devant la multitude de solutions de composants de service, OSGi vise à simplifier la gestion de vie des composants de service et leur dépendance via un framework simple.
Une plate-forme OSGi doit posséder les caractéristiques suivantes :

- Utilisable dans des équipements à mémoire limitée. Les framework OSGi sont la plupart du temps embarquées dans des équipements. L'environnement d'exécution doit être adapté à la faible empreinte mémoire de ces équipements.
- Un environnement sécurisé dans lequel les services pourront fonctionner sans interagir sur les autres services. Les mécanismes de sécurité d'OSGi se basent entre autres sur les mécanismes de Java2, avec l'ajout de quelques mécanismes supplémentaires de sécurité. Par exemple, OSGi gère les dépassements de buffers, les accès au code, les dépendances entre services et les autorisations d'accès entre eux…
- Administration distante : Les services seront contrôlés par un opérateur OSGi et/ou des fournisseurs de services OSGI et non l'utilisateur final.
- Plate-forme Multifournisseurs. Les services contrôlés par différents fournisseurs de service OSGi devront partager les ressources de la même passerelle OSGi.
- Disponibilité du service. Les services déployés sur la plate-forme OSGi auront souvent un cycle de vie long, avec des périodes très courtes d'indisponibilité prévues. L'environnement d'exécution d'OSGi doit donc offrir les mécanismes nécessaires pour permettre aux services d'atteindre ces requis de disponibilité.
- Gestion des versions dynamique. Un fournisseur de service doit pouvoir mettre à jour un service sans générer de période d'indisponibilité de la plate-forme, et sans conséquences pour les autres services déjà déployés.

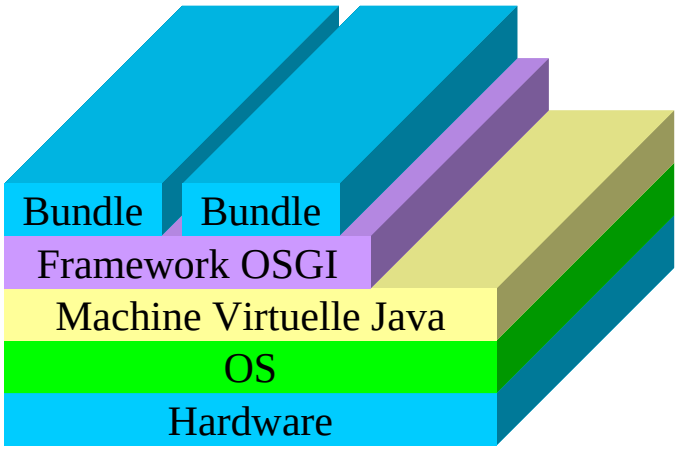

Figure 2. 4: Modèle OSGi

Parmi les propriétés d'OSGI, un des objectifs principaux de l'architecture est d'offrir aux services un certain niveau d'indépendance vis-à-vis du matériel et un mécanisme de téléchargement dynamique de code, afin de faciliter le développement et le déploiement de services. Pour ceci, le groupe OSGi a spécifié des interfaces de programmations (APIs) et des directives, basées sur le langage Java, concernant l'environnement d'exécution dans lequel sont déployés les services. Je m'intéresse ici plus spécifiquement à cet aspect d'OSGI puisque c'est un critère principal de cette thèse. Ce qu'il est possible de retenir de la manière dont sont développés et déployés les services dans un environnement OSGi est :

- Le développement de services est facilement dissociable des spécifications grâce aux notions d'interfaces définies dans Java, que OSGi réutilise. Un service OSGi est implémenté à partir d'un service Java classique en créant une classe « mère » (par exemple ServiceActivator) de la classe principale Java, et qui implémente les méthodes « start » et « stop » (la méthode "start" contiendra le code d'exécution du service). Aucune modification supplémentaire n'est exigée pour transformer le service Java en bundle OSGi.
- La notion de composants est vraiment une partie intéressante de l'architecture, puisque chaque service OSGi peut être vu comme un ensemble de composants de service, appelés « bundles » en langage OSGi. Ces bundles sont « installables » automatiquement, et peuvent être téléchargés et supprimés sur demande. Un bundle est un fichier JAR contenant les classes Java, des bibliothèques (éventuellement natives) nécessaires et un fichier « Manifest » indiquant les dépendances avec d'autres bundles (par exemple, pour importer des interfaces d'un bundle donné ou au contraire pour indiquer à OSGI les interfaces que le bundle exporte vers d'autres). L'environnement d'exécution OSGI fournit un ensemble d'APIs pour contrôler le cycle de vie des composants, et définit un diagramme de transition d'état.

Les services peuvent être déployés, enregistrés, démarrés, stoppés par un administrateur… Cependant, cette dernière partie est loin d'être aussi efficace que le mécanisme de déploiement mis en œuvre dans une plate-forme de réseaux programmables, comme celui de FAIN par exemple. En effet, actuellement, le déploiement est fait manuellement par un administrateur, par l'utilisation d'un portail

Web ou par une interface Telnet. Il doit être amélioré pour proposer une manière automatique de déployer des services.
Les aspects de sécurité du service tournant sur la passerelle sont bien abordés, les fichiers JAR déployés peuvent être signés les autorisations sont décrites avec l'utilisation de certificats, mais peu d'indication sur le déploiement lui-même du code et notamment sur les mécanismes de sécurité de déploiement mis en œuvre.
Finalement, OSGi manque aussi des composants permettre de détecter et d'informer sur le contexte courant.
Bref, pour conclure, OSGi présente un environnement applicatif très prometteur, par rapport aux objectifs de cette étude et répondant à certains requis. Il lui manque cependant quelques aspects qu'il faudrait lui ajouter pour pouvoir remplir le rôle du nœud intermédiaire. On verra au chapitre 4 que la solution d'architecture définie dans cette thèse s'appuiera sur un environnement OSGi en tant qu'environnement d'exécution dans le prototype mis en œuvre.

- **ProAN**

ProAN (Figure 2. 5) est une architecture de passerelle active développée au LSR-IMAG [Nguy'03] [Nguy'04]. L'objectif est de définir une passerelle, localisée en bordure des réseaux d'accès, qui puissent héberger des services tournant dans des environnements d'exécution.

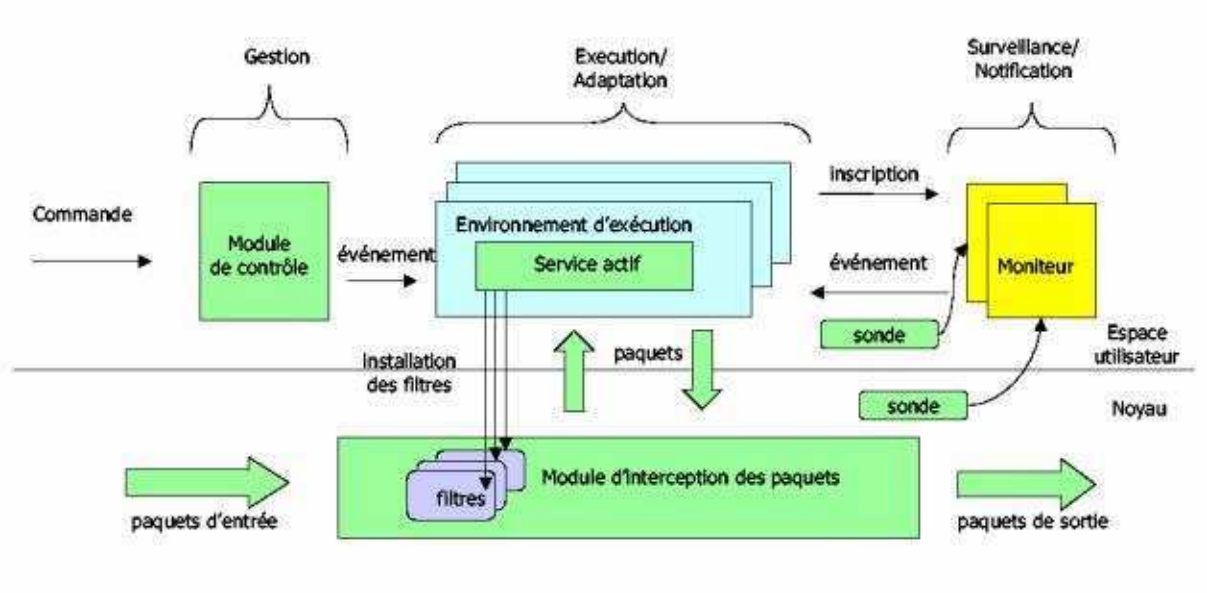

Figure 2. 5: Architecture de ProAN

- Les services actifs de la passerelle sont installés dans un **environnement d'exécution** pour y être exécutés. Trois environnements d'exécution ont été prévus dont l'environnement pour GateScript qui a été spécifié et qui est principalement utilisé dans ProAN. Gatescript [Nguye'03] est un langage de description des PDUs (Protocol Data Unit) avec pour objectif de séparer la fonction d'analyse et de formatage de celle du traitement des données. L'idée est de définir des librairies Gatescript en fonction des protocoles et ainsi les services actifs, écrits en Gatescript, peuvent utiliser ces librairies pour facilement récupérer les valeurs de certains champs des données.

- Le **Module d'interception des paquets** est chargé d'intercepter les paquets transitant par la passerelle et de les remonter aux services actifs ou de les rediriger directement vers l'élément suivant. Les actions sont prises par ce module en fonction de filtres définis par les services actifs. Ce sont les services actifs eux-mêmes qui installent et désinstallent les filtres de paquets.
- Les **services actifs** sont en charge de réaliser des traitements sur les paquets. Ils reçoivent les paquets directement du module d'interception de paquets en fonction des filtres qu'ils ont définis. Les services peuvent configurer (ajouter/supprimer) les filtres dynamiquement. Dans ProAN, le langage GateScript est recommandé pour écrire les services actifs mais laisse la possibilité d'écrire des services en d'autres langages sans vraiment spécifier comment.
- Les **moniteurs** sont des composants capables de superviser des ressources particulières (état de la batterie, espace de stockage...). Les services actifs désirant recevoir des informations de ces moniteurs peuvent s'inscrire auprès des moniteurs.
- Enfin, les accès aux services par les utilisateurs sont gérés par le **module de contrôle**. Une base de données recensant les utilisateurs autorisés à utiliser le service est définie et en fonction de cette base de données, l'accès est validé ou non. Par ce module, les utilisateurs peuvent aussi configurer les services actifs.

ProAN propose donc une architecture de nœud intermédiaire intéressante avec la notion de moniteurs qui permettent d'avoir des informations sur le contexte et les services actifs tournant dans un environnement d'exécution. La notion de module d'interception de paquets est utile et nous la retrouverons aussi dans la solution définie dans cette thèse puisque ce module a été défini en collaboration. Cependant dans ProAN, la passerelle ne fonctionne pas en mode transparent.
Les aspects concernant la sécurité sont limités dans ProAN et le déploiement de code dynamique n'a pas été détaillé. Finalement, bien que l'architecture soit supposée héberger des services en C ou Java, il est clair que la préférence a été mise sur l'environnement GateScript ce qui peut limiter l'étendue des services fonctionnant sur la passerelle, notamment les services multimédias.

- ARFANet

ARFANet [Mok'05] est une infrastructure de nœud actif basée sur la notion de règles actives, chaque règle respectant la forme dite ECA (Event – Condition – Action).

Les services déployés sur le nœud définissent les événements qui les intéressent et peuvent prendre les actions adaptées en fonction des conditions de l'évènement. Cette architecture est principalement valide pour détecter des événements réseau tels qu'une congestion, une perte de connectivité et ainsi prendre les actions qui s'imposent.
Voici un exemple de règle complexe ECA qui définit la règle R1 en fonction des événements E1 et E2, des conditions C1 et C2 pour effectuer les actions A1 et A2.

Define rule R1
On E1 & E2
If C1 | C2
Then A1 ; A2

L'architecture ARFANet peut être schématisée ainsi (Figure 2. 6) :

Figure 2. 6: Architecture du nœud ARFANet

- Le nœud repose sur une **machine virtuelle**. Cette machine virtuelle est installée à la création du nœud. Les services d'applications et les moniteurs actifs sont déployés dans cette machine virtuelle, qui leur fournit ainsi une abstraction des caractéristiques matérielles et logicielles du nœud.
- Les **moniteurs actifs** sont des composants chargés de superviser des événements ou des ressources et d'exécuter les actions définies par les services d'application. Plusieurs moniteurs peuvent être activés sur un nœud.
- Les **services d'applications** sont les composants qui représentent les règles à exécuter par les moniteurs actifs lorsque des évènements sont détectés. Un service d'application est donc lié à un moniteur, mais ce lien n'est pas figé, il est défini lorsque le service d'application est déployé.

ARFANet définit aussi un mécanisme pour l'identification du code utilisé pendant la phase de déploiement. Ceci est géré par un serveur de code et d'identification CISS (Code Identification and Storage Server). Ce serveur étend ainsi la fonctionnalité de base de serveur de stockage de code pour y inclure des mécanismes de sécurité. Le code lui-même est vérifié ainsi que les entités qui le fournissent. En effet, un mécanisme, à base de clés publiques, permet l'authentification des entités concernées par le déploiement. La Figure 2. 7 présente le mécanisme de publication et de déploiement du code dans les nœuds ARFANet.

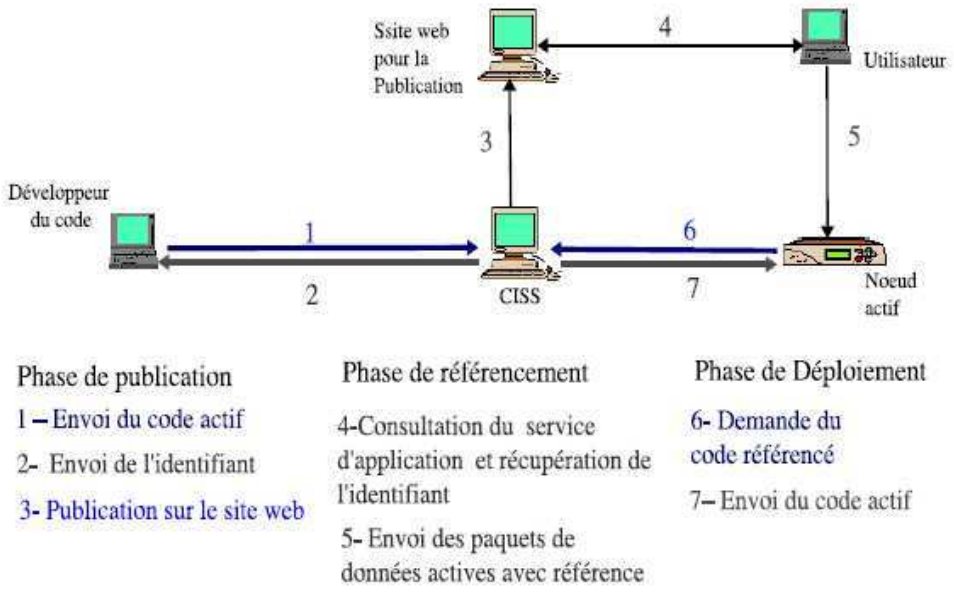

Figure 2. 7: Publication et déploiement du code

L'architecture ARFANet a été définie à peu près en parallèle à ma solution. On y retrouve les problématiques actuelles concernant la détection de contexte (ici les moniteurs) et un mécanisme de déploiement de code sécurisé. Ce dernier point est intéressant dans ARFANet. La solution que je propose repose sur une solution proche mais plus récente utilisant la nouvelle notion HIP (Host Identity Protocol) définie à l'IETF.
Nous voyons que ce principe ECA s'applique surtout aux services de niveau réseau, qui sont déployés sur le nœud, qui peuvent prendre des actions basiques suivant ce qui est défini. Pour l'adaptation et la personnalisation de contenu, cette architecture parait peu adaptée.

- DINA

DINA [Jean'05] est une plate-forme de réseau programmable modulaire, qui permet de déploiement de services et qui peut être instanciée sur différents équipements réseau tels que des routeurs, des passerelles... DINA est basé sur les concepts de la plate-forme ABLE [Raz'99] [Raz'00] et a été réécrit en langage Java en intégrant de nouveaux brokers.

La plate-forme (Figure 2. 8) est constituée de plusieurs composants, sui fonctionnent aussi d'un OS de type Linux :
Le Diverter reçoit les paquets programmables réseau et les envoie au "Session broker".
Le "Session Broker" est le composant principal de la plate-forme en charge d'analyser les paquets et les transmettre aux services concernés. Il a aussi en charge la gestion des services.
D'autres composants, des "brokers" spécifiques peuvent tourner en parallèle et offrir des fonctionnalités dédiées, comme le SIP broker qui gère les connexions SIP ou un

context broker qui peut gérer des informations de contexte (tels que la QoS, le flux réseau…).
Enfin, les applications déployées sur la plate-forme peuvent traiter les paquets reçus des brokers via une interface définie.

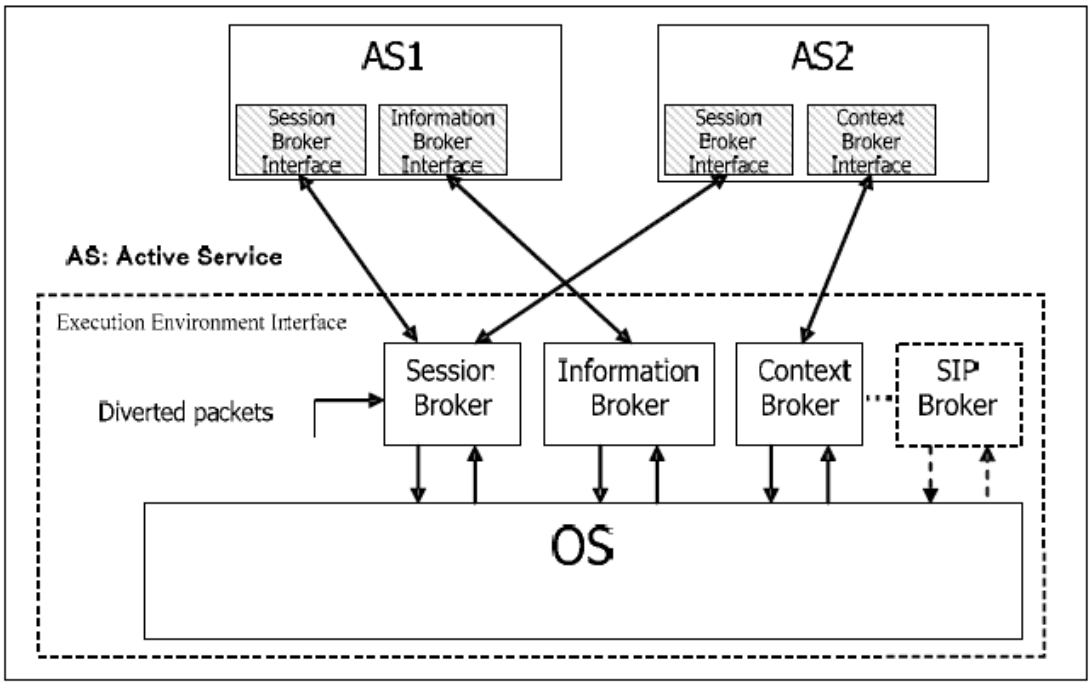

Figure 2. 8: Architecture de la plate-forme DINA

Cette plate-forme DINA peut permettre de détecter des informations de contexte en définissant les brokers qui vont bien. Par exemple, [Ocam'05] présente une solution permettant de déployer des senseurs permettant de connaître des informations de contexte. La technologie des réseaux programmable est utilisée pour le déploiement des senseurs, mais cette solution ne parle pas des mécanismes d'adaptation. DINA pourrait éventuellement fonctionner en mode transparence en intégrant un broker en charge de la gestion transparence des connexions, mais même si cela est faisable, cela reste à faire. Par contre, l'environnement d'exécution n'est pas aussi souple et configurable que je le souhaite car le déploiement de modules ne suit pas la logique requise et la sécurité n'est pas abordée.

- Autres

Ces dernières solutions m'ont paru les plus pertinentes pour les travaux que je souhaitais réaliser, c'est la raison pour laquelle je les ai détaillées. D'autres études de recherche traitent des problématiques d'adaptations par l'utilisation de nœud intermédiaire mais sont souvent spécifiques ou n'incluent pas des fonctionnalités faisant partie des requis pour avoir une solution générique telles que le déploiement de code ou la gestion des connexions. [Ard'01] présente une solution d'adaptation de contenu intéressante, utilisable dans un environnement proxy. L'adaptation est basée sur un contexte fourni par l'utilisateur. Il peut donc manquer des informations de

contexte réseau qui pourraient altérer la transmission du contenu. [Bha'98] présente une architecture de proxy réalisant des adaptations de contenu HTTP pour des utilisateurs mobiles. Mais l'adaptation se limite à envoyer le contenu qu'il faut en fonction par exemple des attributs du champ "Accept" fourni par l'utilisateur dans la requête HTTP. [Mah'02] propose une solution de transcodage et de cache dans des proxies. Les objets sont cachés en fonction de catégories (par exemple les terminaux utilisateurs) en fonction du terminal utilisé par le client, le contenu adéquat est délivré. Dans cette solution, c'est principalement le type de terminal qui guide l'adaptation et les exemples montrés sont relatifs à du transcodage d'images fixes pour s'adapter à l'écran des terminaux. Neon [Schu'05] est une architecture de nœud définie pour permettre le déploiement de services réseau sur des nœuds programmables, basés entre autres sur les concepts issus du groupe Forces à l'IETF. Mais Neon cible plutôt les services réseau, opérant sur les paquets transitant par le nœud pour effectuer des fonctions telles qu'un firewall, un partage de charge, différentiation de services ou contrôle de virus et ne situe pas au niveau applicatif comme l'adaptation de contenu le requiert. [Han'98] propose un framework d'adaptation (principalement de transcodage d'images) sur un proxy permettant de définir s'il faut transcoder ou pas et à quel niveau, en fonction du temps requis pour réaliser le transcodage, pour le transfert. [You'06] présente une solution de déploiement restreint dans le plan de contrôle, permettant d'isoler les codes entre eux et définissant des "routeurs virtuels" (des partitions) dans lesquels les utilisateurs identifiés peuvent déployer leurs services. [Bran'06] propose une passerelle Wifi permettant de faire de l'adaptation de contenu audio et vidéo sur des flux RTP (Real-Time Transport Protocol) en fonction d'un profil utilisateur défini en CC/PP. Cette étude rejoint mes travaux, notamment dans l'idée de l'utilisation du nœud polymorphique en tant que passerelle sans-fil/filaire mais n'est pas transparent et ne prend pas en compte l'aspect de déploiement sécurisé des modules applicatifs.

Il existe aussi des études relatives ou proches des réseaux actifs [Tenn'96] [Tenn'97], networks processors ou des processeurs FPGA (Field-Programmable Gate Array), tels que [Hick'98] [Schm'00] [Neo'03] [Shah'03] [Hass'03] [Kel'02] [Witt'96]... Dans cette thèse, ces solutions ne sont pas considérées car, elles s'appliquent à des niveaux plus bas, présentent des environnements d'exécution limités et n'incluent pas tous les mécanismes permettant d'avoir des informations de contexte utilisateur pertinentes.

Dans cette étude, pour le déploiement de modules, je me focalise sur les principes permettant d'authentifier les fournisseurs et les nœuds et non pas sur la composition de services, qui a fait l'objet de nombreuses publications dans le monde de la recherche. Je la suppose donc comme efficace et éventuellement certaines solutions qui y sont définies peuvent être applicables et transposables dans ma solution. Cependant dans mon cas, je me suis basé sur les mécanismes définis dans OSGi, qui sont, à mon avis, plus valides, ayant plus d'avenir et de possibilité d'utilisation car défendus par un organisme mondial et soutenus par de nombreux partenaires.

2.2 Bilan

Les solutions actuelles de nœud intermédiaire reposent souvent sur l'utilisation d'un proxy ou équivalent. Les extensions réalisées par ICAP ou OPES sont des ouvertures intéressantes pour l'ajout de fonctionnalités de services, telles que les modules d'adaptation. Cependant, ces solutions ne sont pas transparentes vis-à-vis des points terminaux (clients et serveurs) au sens défini et n'incluent pas de possibilité de déploiement dynamique de code. Les solutions de type FAIN, OSGi, ProAN vont plus loin et offrent un environnement d'exécution pour héberger les modules. Les services peuvent ainsi fonctionner de manière isolée, sans interférence entre eux. Cependant, ces solutions n'intègrent pas toutes les fonctions souhaitées, notamment un mécanisme de gestion des communications transparentes entre les utilisateurs et les serveurs et l'intégration dans leur architecture des éléments permettant de connaître le contexte de l'utilisateur. Cet état de l'art fait aussi ressortir que de nombreuses études traitent de la problématique d'adaptation dynamique de contenu par des éléments intermédiaires mais chacune apportant une brique à la recherche ou étant spécifique, sans vraiment définir une architecture globale, intégrant les 4 propriétés définies.
Dans cette thèse, je vais donc étudier une solution visant à la définition d'un nœud polymorphique, transparent ou non, qui soit capable de connaître le contexte courant de l'utilisateur pour permettre l'adaptation dynamique de contenus, par des modules déployés de manière sécurisée sur le nœud.

Chapitre 3

Nœud intermédiaire permettant l'adaptation de contenu en fonction du contexte utilisateur

3.1 Introduction

Les chapitres précédents ont conduit au besoin de définir une architecture de nœud intermédiaire ayant pour objectif de permettre l'adaptation de contenu par des modules applicatifs en fonction du contexte utilisateur.
Adapter le contenu en fonction du contexte utilisateur implique de le connaître. Ainsi, une entité fonctionnelle de contexte utilisateur est définie, dans un plan de connaissance, pour regrouper aussi bien les informations concernant l'utilisateur (type de terminal utilisé, préférences) que les conditions des réseaux physiques, et pour collecter et mettre à disposition le contexte courant.
Pour que les modules applicatifs puissent modifier les données, il est nécessaire de pouvoir rediriger les paquets de données vers ces modules. Un module d'interception des paquets, configurable en temps réel, permettant de "remonter" les paquets de données, est intégré dans le nœud.
Enfin, entre l'entité de contexte utilisateur et les modules d'adaptation, une entité fonctionnelle, le Gestionnaire du nœud intermédiaire, est spécifiée et a pour rôle de gérer les communications et les relations entre les différents composants du nœud.

Dans ce chapitre, l'architecture du nœud, ainsi que les différentes entités le composant, est d'abord présentée avant de décrire un cas d'usage de ce nœud intermédiaire en tant que passerelle physique entre un réseau filaire et un réseau sans-fil (802.11). Une évaluation, basée sur un démonstrateur réalisé et une évaluation réalisée sur simulateur complète ce chapitre [Math'07].

3.2 Architecture du Nœud

La Figure 3. 1 présente l'architecture du nœud, en introduisant les entités fonctionnelles qui le composent, avec les différentes interfaces de gestion ou de données. Ensuite sont décrites les entités avec une attention plus détaillée sur l'entité "Gestionnaire".
Cette architecture a été définie pour être générique et instanciable de différentes manières possibles en utilisant des technologies ou des solutions techniques différentes. Un exemple d'implémentation est présenté dans la section relative au démonstrateur (section 3.3.4.1).

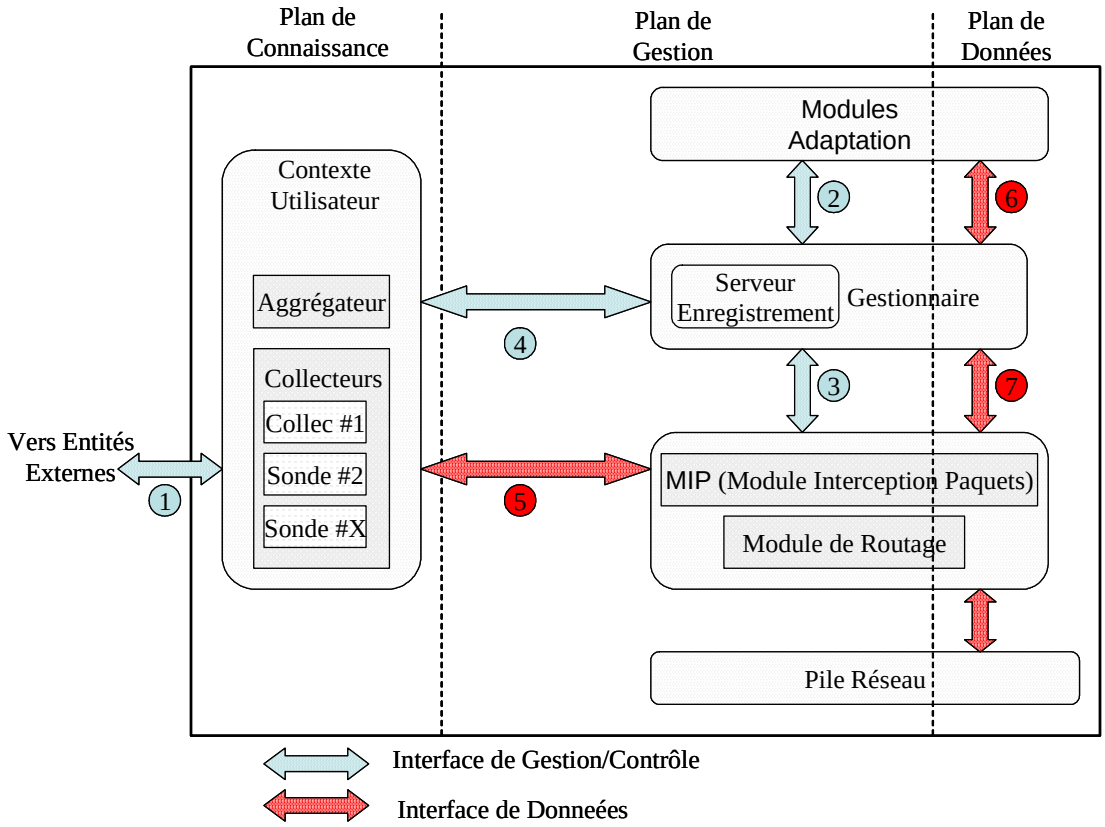

Figure 3. 1: Architecture du nœud intermédiaire

3.2.1 Contexte utilisateur

Pour réaliser l'adaptation, le contexte de l'utilisateur doit être pris en considération. Le contexte regroupe les informations telles que les préférences de l'utilisateur, les caractéristiques de son terminal, les conditions du réseau et les éventuelles conditions ou politiques propres aux services... En effet, les politiques des fournisseurs de service sont à prendre en compte dans le contexte, car l'adaptation ne peut être réalisée qu'uniquement en fonction de ce que le fournisseur de service admet (par exemple, il n'est pas correct de changer les données ou la présentation du service si le fournisseur de service n'est pas d'accord avec ce genre d'adaptation).
Des collecteurs sont chargés de récupérer les informations de contexte, qui peuvent être plutôt statiques ou dynamiques.
Ces informations de contexte se retrouvent dans un plan de connaissance, transverse, qui doit pouvoir être accessible depuis n'importe quelle autre entité: le gestionnaire central du nœud par exemple, mais aussi les modules applicatifs réalisant l'adaptation, les collecteurs/sondes pour enregistrer les valeurs…

- Contexte utilisateur "statique"

Des collecteurs peuvent récupérer, en utilisant l'interface 1, des informations, dites "statiques", en interrogeant des entités externes pour récupérer le profil des utilisateurs (cf démonstrateur chapitre 4) ou bien les caractéristiques matérielles d'un terminal chez le fabricant ou des informations sur la topologie réseau sur une base de l'opérateur…
Ces informations sont dites "statiques" car elles ne varient pas pendant la session de l'utilisateur ou pas souvent (éventuellement en cas de changement de terminal en cours de session par l'utilisateur, ce contexte change).

Parmi ces informations figurent les préférences de l'utilisateur, les caractéristiques du terminal et les politiques du fournisseur de service concernant les adaptations possibles. Voici un exemple de ces informations :

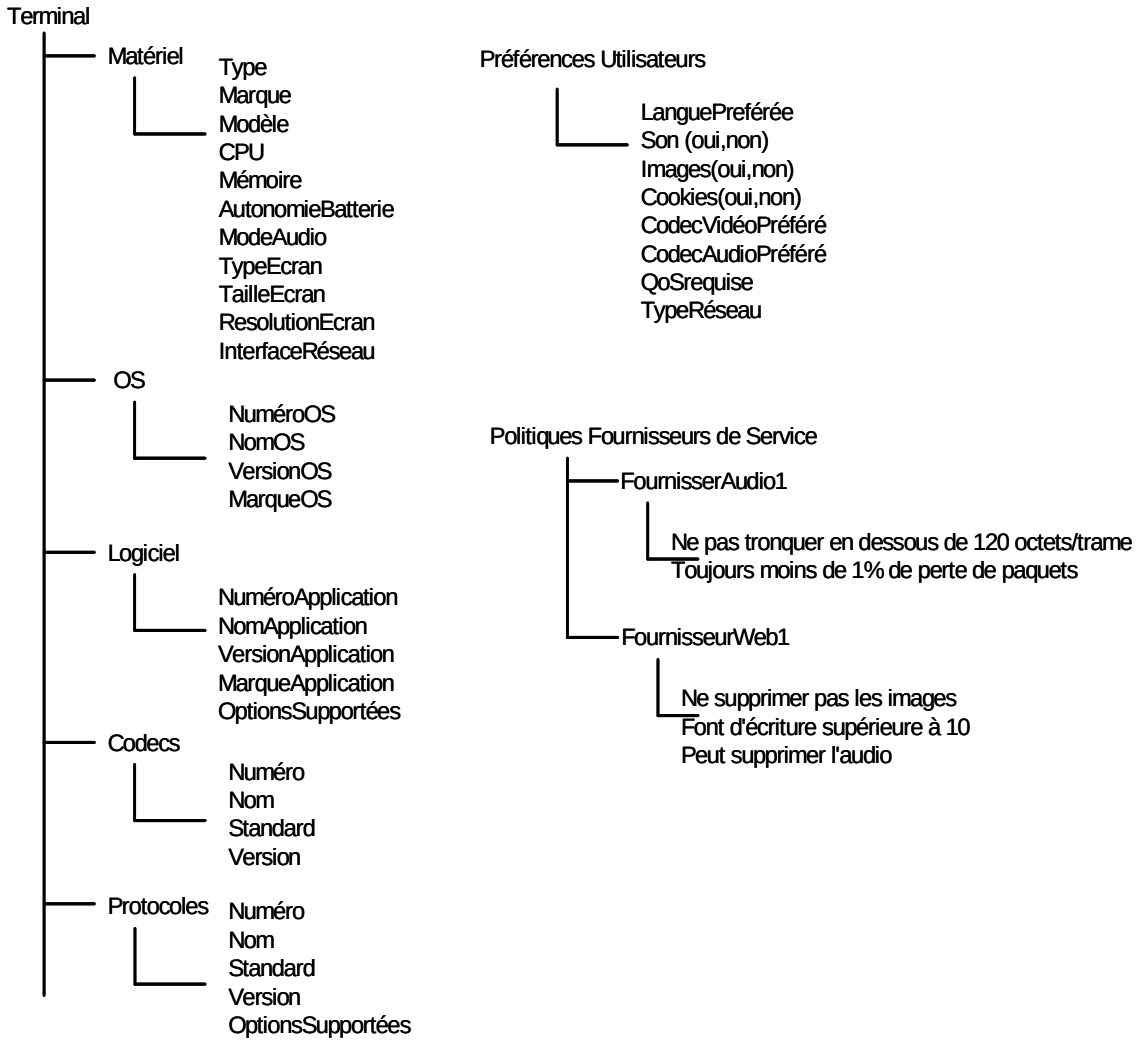

Figure 3. 2: Informations de contexte statiques

- Contexte utilisateur "dynamique"

Les collecteurs peuvent être aussi des sondes qui supervisent l'état de certaines ressources dynamiques. Le terme dynamique signifie ici que les valeurs de paramètres peuvent changer fréquemment et de manière significative. Dans ce cas, la problématique est différente puisque ces collecteurs ne doivent pas uniquement récupérer des informations depuis une base mais sonder, analyser et calculer en temps réel les ressources qui permettent d'obtenir les valeurs des paramètres (par exemple la bande passante utilisée d'un lien, la CPU consommée…). Dans ce cas, la dénomination sonde sera spécifiée pour faire apparaître les traitements nécessaires à mettre en œuvre pour connaître les valeurs des paramètres.
Différentes sondes peuvent être installées sur le nœud en fonction de son utilisation et de ses requis, chacune ayant en charge de superviser des caractéristiques différentes. On peut ainsi citer des :

Sondes informatiques
Le nœud intermédiaire, devant assurer le routage des données entre deux interfaces réseaux, superviser des informations (via les sondes), adapter les données (via les applications déployées), la consommation CPU et de mémoire du nœud peuvent devenir importantes à certains instants. Il est donc nécessaire de pouvoir superviser en temps réel ces données pour détecter une surcharge éventuelle du nœud. Selon ces valeurs, des réactions peuvent être appliquées. Par exemple, si plusieurs modules adaptent des données, menant à une surconsommation des ressources informatiques du nœud, une décision de stopper un ou plusieurs modules d'adaptation peut être prise. Ceci permet de libérer certaines ressources informatiques et ainsi d'éviter un "crash" potentiel du nœud intermédiaire.

Sondes Réseaux
Un des objectifs du nœud intermédiaire est de permettre l'adaptation de flux en cas de conditions du réseau insuffisantes (congestion, bande passante saturée…). En effet, en cas de saturation de la bande passante, de nombreux paquets pourraient être perdus entrainant une qualité de service dégradée. Détecter le taux d'utilisation de la bande passante et anticiper sa surcharge en informant les modules applicatifs d'adapter les données pour consommer moins de débit peut remédier à cette qualité dégradée.
Des sondes qui peuvent surveiller le réseau sont donc nécessaires dans le nœud intermédiaire pour informer sur les conditions des réseaux. De telles sondes sont mises en œuvre en recevant les paquets réseau via l'interface de données 5.

Sondes Applicatives
Diverses sondes applicatives peuvent être développées; des sondes spécifiques aux applications ou aux modules d'adaptation ou des sondes plus génériques, comme des sondes d'évaluation de QoE (Quality of Experience). En effet, il est imaginable qu'en cas de mauvaise qualité perçue du service, qu'un module d'adaptation modifie les données pour améliorer cette QoE. A titre d'exemple, 3 sondes de QoE ont été implémentées dans le chapitre 6 pour connaître la qualité perçue de flux multimédia: une Sonde MDI, chargée d'évaluer le MDI (Media Delivery Index) [Agi'06], une sonde MPQM (Moving Pictures Quality Metric) [Bran'96], basée sur la métrique définie au

MPEG forum, et une sonde SSIM (Structural Similarity) [Wan'04], pour identifier la similarité structurelle entre 2 images, couplée dans ce cas avec la valeur du MDI pour évaluer la QoE.

Les informations de contexte dynamique qui peuvent être définies sont:

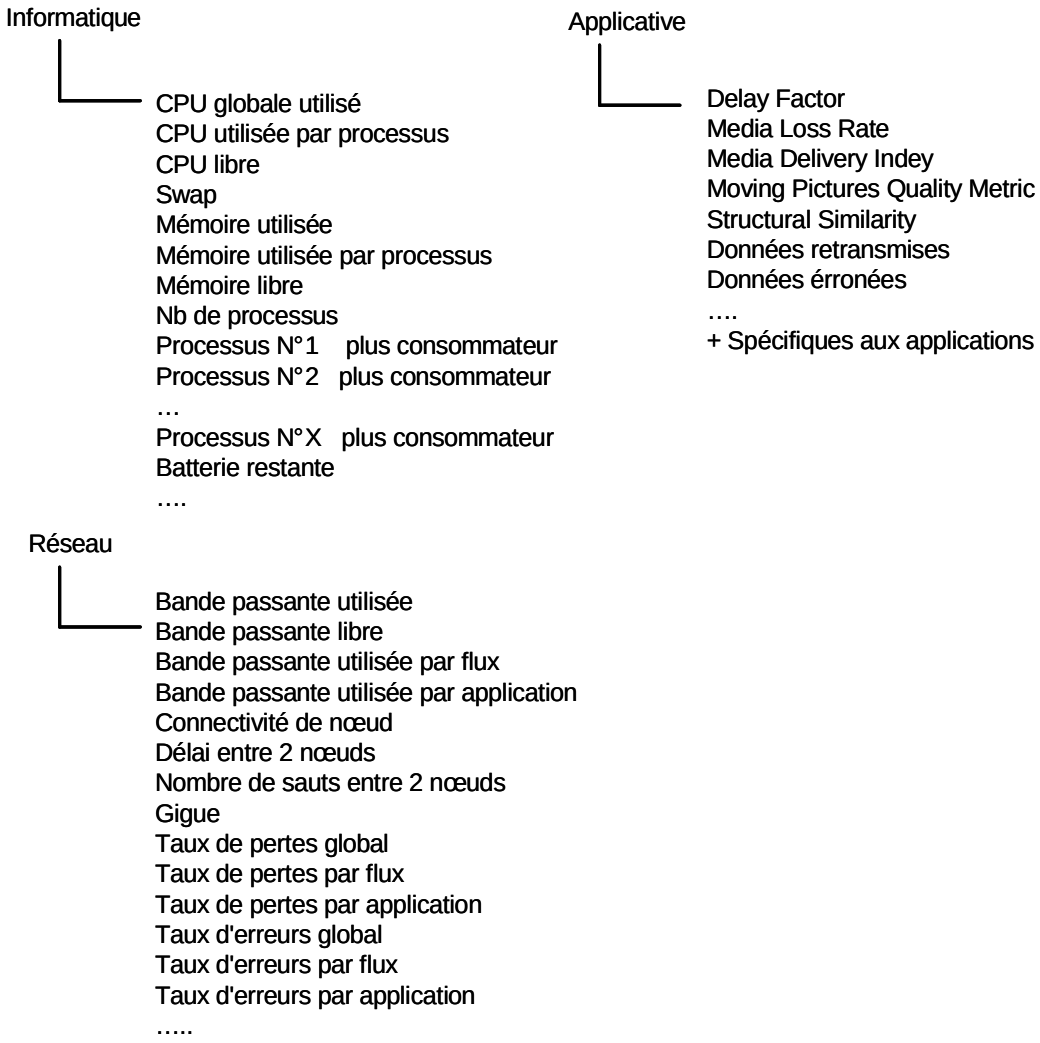

Figure 3. 3: Informations de contexte dynamiques

- Agrégateur

Dans cette entité figure aussi un composant qui a pour rôle d'agréger les informations supervisées par les différents collecteurs ou sondes.
Ce composant doit vérifier la conformité et la cohérence des données et informer le gestionnaire sur les valeurs de contexte selon l'interface 4.
Cette interface est définie par une primitive générique, prenant en arguments, une éventuelle description de flux et le nom du (ou des) paramètre(s) dont la (ou les) valeur(s) est (sont) demandée(s).

La primitive est ainsi de la forme :

Hashtable getInformation (Flow f, String param)

Où est spécifiée la valeur du paramètre "param" pour le Flow "f" concerné.
Cette méthode retourne une "Hashtable" contenant le (ou les) paramètre(s) et sa (ou ses) valeur(s).

Avoir une primitive générique, évite de définir X méthodes pour X paramètres et ainsi de réutiliser toujours la même primitive même si de nouvelles sondes (et donc de nouveaux paramètres supervisés) sont implémentées. Le paramètre *param* peut être "CPU", "memory", "bandwidth"…et dépend des collecteurs (ou des sondes de supervision) déployés sur le nœud et des informations disponibles.
A titre d'exemple, les paramètres peuvent être :

* "***CPU***" : pour le taux d'utilisation CPU
  La valeur retournée est le taux d'utilisation CPU en %.
* "***Memory***" : pour le taux d'utilisation Mémoire
  La valeur retournée est le taux d'utilisation Mémoire en %.
* "***Bandwidth***" : pour la bande passante totale utilisée
  La valeur retournée est la bande passante totale utilisée en Mo/s.
* "*Device/Hardware/model*" : pour le type de terminal de l'utilisateur
* "*UserPreferences/PreferredLanguage*" : pour la langue de l'utilisateur
* ……

L'argument d'entrée est de type "*Flow*", qui est défini de la manière suivante :

```
Flow {
    ipAddrSrc;
    portSrc;
    ipAddrDst;
    portDst;
    protocol;
}
```

Si l'application veut recevoir des informations concernant plusieurs flux ou par exemple tous les flux d'un protocole donné, des champs pourront prendre la valeur "Any".
Pour les sondes de traitements, comme par exemple pour le taux CPU ou la mémoire, la valeur de "Flow" devra être initialisée à "null".

3.2.2 Module de Routage & Module d'Interception de Paquets (MIP)

Le nœud polymorphique pouvant être raccordé à plusieurs réseaux physiques différents, il doit être capable de router les paquets d'un réseau vers l'autre et vice-versa. Le module de routage assure cette fonction.

Comme l'objectif est l'adaptation de contenu, une entité, qui intercepte les paquets et les "remonte" aux modules d'adaptation qui traiteront les données, est nécessaire. Cette entité, appelée MIP (Module d'Interception de Paquets) échange les paquets avec le Gestionnaire via l'interface 7.

Cette entité MIP doit être configurable pour intercepter uniquement les paquets nécessitant une adaptation. En effet, le traitement réalisé par les modules applicatifs engendre un délai dans l'acheminement des données, le temps nécessaire pour transférer les paquets aux modules applicatifs et adapter les paquets. Pour avoir les meilleures performances possible, il ne faut transférer que les trames concernant cette application et à bon escient (c'est-à-dire lorsqu'une adaptation est nécessaire). Ainsi en cas de non-adaptation, le paquet est directement retransmis, assurant une perte de temps de traitement minime, correspondant uniquement à l'analyse de l'en-tête IP pour vérifier si un traitement est nécessaire. Une interface permettant de configurer le MIP est nécessaire. Cette configuration sera réalisée par le gestionnaire, selon les informations de contexte ou selon les conditions de modules de services, via l'interface de gestion 3. Deux primitives sont spécifiées :

boolean addFilter (@IPsource, @IPdestination, portSource, portDestination, Protocole)
Permet de configurer le module MIP pour lui dire d'intercepter et de "remonter" les paquets concernés par les paramètres passés en arguments.

boolean removeFilter (@IPsource, @IPdestination, portSource, portDestination, Protocole)
Permet de configurer le module MIP pour transférer directement (et ne plus "remonter") les paquets concernés par les paramètres.

3.2.3 Gestionnaire du nœud

Le gestionnaire est le composant principal de l'architecture du nœud puisqu'il a pour charge de coordonner, contrôler les interactions entre les différents composants du nœud pour délivrer un service adapté au contexte de l'utilisateur; depuis les modules permettant d'obtenir les informations de contexte tels que les collecteurs jusqu'aux modules applicatifs réalisant les fonctions d'adaptation… Les modules d'adaptation n'ont qu'un ainsi seul interlocuteur, le Gestionnaire.
La Figure 3. 4 suivante présente l'architecture interne du gestionnaire du nœud et les interactions entre les différents blocs fonctionnels le composant.

Le gestionnaire offre une interface (interface de gestion 2) aux applications les permettant de s'enregistrer et de définir les paramètres qui les intéressent et les critères sur lesquels elles souhaitent recevoir des notifications (par exemple dépassement de seuils), de demander les valeurs de certains paramètres gérés par les sondes et pour recevoir les notifications.
Un composant du Gestionnaire, l' "*Analyseur de Requêtes*" est chargé d'analyser la primitive invoquée par le module applicatif et de rediriger la demande vers le module adéquat; à savoir le "*serveur d'enregistrement*", le "*Frontal Lecture*" pour une demande de valeur de paramètre ou le "*Configurateur MIP*" si le module applicatif demande à configurer le MIP pour recevoir les paquets de données (ou au contraire de ne plus les recevoir) via l'interface 3. Le "*Configurateur MIP*" maintient une liste des règles appliquées sur le composant MIP pour vérifier la cohérence entre les règles demandées par différents modules, la redondance…

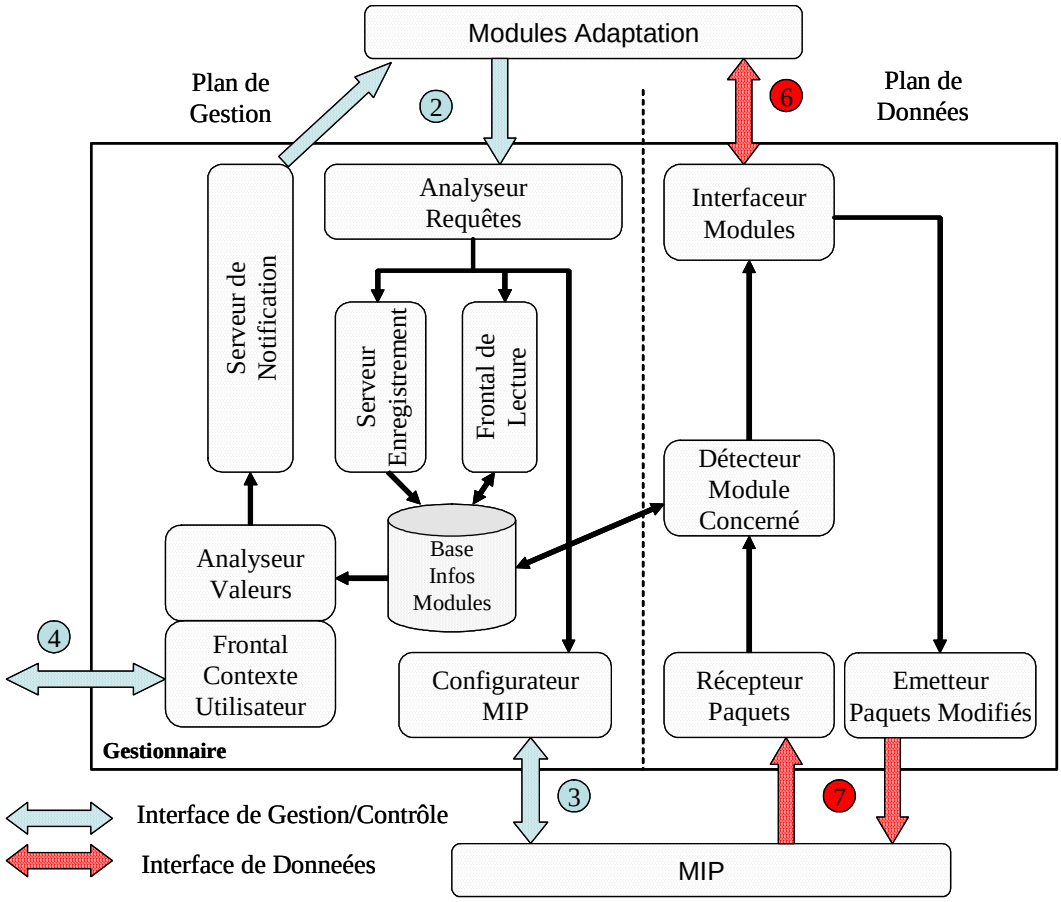

Figure 3. 4: Architecture interne du Gestionnaire du nœud

Lors de l'enregistrement des modules d'adaptation, les informations concernant le module (son nom, son identifiant…) ainsi que les paramètres souhaités et les critères de notifications, sont enregistrés dans une "*Base Info Modules*".

Un "*Frontal Contexte Utilisateur*" a pour rôle de communiquer avec l'entité fonctionnelle "*Contexte utilisateur*", via l'interface de gestion 4, pour connaître les valeurs des paramètres pour lesquelles les modules sont intéressés. Un module "*Analyseur Valeurs*" se charge d'évaluer les valeurs des informations de contexte (notamment les valeurs dynamiques) et en fonction des critères définis par les modules lors de l'enregistrement, d'informer le "*Serveur Notification*" qui lui enverra un message aux modules d'adaptation concernés, via l'interface de gestion 2.

Le Gestionnaire a aussi pour charge de transmettre les paquets des données reçus du module MIP jusqu'aux modules d'adaptation et vice-versa. Pour ceci, les modules "*Réception Paquets*", "*Emission Paquets Modifiés*" sont définis pour communiquer avec le MIP, via l'interface de données 6. Un module "*Détecteur Module Concerné*" a pour rôle d'identifier le module d'adaptation concerné pour les paquets courants. Ceci est réalisé par la communication avec la "*Base Info Modules*". Enfin, le composant "*Interfaceur Modules*" se charge de transmettre les paquets aux modules applicatifs et de recevoir les paquets modifiés, via l'interface de données 6 et de les transmettre au composant "*Emission Paquets Modifiés*" pour être réémis.

Pour permettre la communication entre les modules applicatifs déployés sur le nœud et le gestionnaire, les modules d'adaptation doivent au préalable s'enregistrer dans un serveur d'enregistrement, en précisant les informations de contexte qui les intéressent.
Un format d'échange est nécessaire pour structurer les informations à communiquer. Le format *XML* a été choisi car il est structurellement intéressant, modulaire et couramment utilisé maintenant.
L'exemple suivant présente l'enregistrement du module applicatif, nommé "Truncation_FT" dans le service d'enregistrement et la définition des seuils considérés comme acceptables est passée en argument de ce message. Les valeurs sont normalisées entre 0 et 1 dans cet exemple (0.7 signifie 70 %). Dans cet exemple, on voit qu'il est possible de définir plusieurs seuils (pour la bande passante); cela permet aux applications d'avoir plusieurs niveaux d'adaptation en fonction du contexte.

```
<? xml version="1.0" encoding="UTF-8"?>
<Adaptation_Modules>
        <AM_Name>Truncation_FT</AM_Name>
        <CPU>
                <CPU_Value>0.5</CPU_Value>
        </CPU>
        <Memory>
                <Memory_Value>0.7</Memory_Value>
        </Memory>
        <Bandwidth>
                <Bandwidth _Value_1>0.5</Bandwidth _Value_1>
                <Bandwidth _Value_2>0.7</Bandwidth _Value_2>
<Bandwidth _Value_3>0.8</Bandwidth _Value_3>
        </Bandwidth>
….
</Adaptation_Modules>
```

L'interface 2 offre plusieurs primitives pour communiquer avec le gestionnaire. Comme pour l'interface 4, les primitives sont génériques en prenant les paramètres en argument et le flux (type Flow) concerné.

- *String registerAppli (String thisAppli, String[] ProcessIds)*

Permet d'enregistrer un module d'adaptation, dénommé "*thisAppli*" en précisant le ou les processus qui représentent ce module, transmis sous forme de chaine de numéro de processus. Cette méthode retourne "OK" si l'enregistrement se passe bien, un message d'erreur sinon (par exemple, "nom déjà utilisé", "processus inexistant"…).

- *void unregisterForThresholdReached (String thisAppli, Flow f, String param)*

Pour se désenregistrer et ne plus recevoir d'information sur les dépassements de seuils.

- *Hashtable getInformation (Flow f, String param)*

Permet de recevoir la valeur du paramètre "param" pour le Flow "f" concerné.
Cette méthode retourne une "Hashtable" contenant le paramètre et sa valeur.

- *boolean addFilter (Flow f)*

Permet aux applications de demander au gestionnaire de configurer le module MIP (d'interception des paquets) pour remonter les paquets de ce flow "f" (via le gestionnaire) à l'application qui invoque cette méthode. Mais il faut que cette application soit enregistrée pour ce flow.

- *boolean removeFilter (Flow f)*

Permet de supprimer la règle précédemment configurée sur le module MIP.

3.2.4 Modules Applicatifs

Au plus haut de l'architecture du nœud figurent les modules applicatifs.

Nous avons vu au chapitre 3.2.3 que les applications doivent s'enregistrer dans le service d'enregistrement, via l'interface 2, pour pouvoir ensuite être connues du gestionnaire.

Les modules applicatifs doivent aussi invoquer l'interface 2 offerte par le Gestionnaire pour pouvoir s'enregistrer aux événements relatifs aux sondes (en fonction des seuils définis), pour demander les valeurs supervisées par les sondes ou encore pour configurer le module MIP, via le gestionnaire.

En plus, les modules d'adaptations doivent aussi implémenter 2 primitives pour que le gestionnaire puisse leur envoyer les informations sur dépassement de seuil sur l'interface 2 et les données pour réaliser des adaptations (le cas échéant) sur l'interface 6.

Ces 2 primitives sont :

- *void sendNotif (Flow f, String param, float valeur)*

Sur dépassement d'un seuil (supérieur au seuil ou inférieur si le seuil avait déjà été dépassé), le Gestionnaire invoque cette méthode sur l'application concernée en signalant le flux (si celui-ci est significatif; par exemple il ne l'est pas pour le CPU), le paramètre concerné et sa valeur.

- *byte[] sendPacketToAppli (String ipSrc, String ipdst, int portSrc, int portDst, byte[] message)*

Si une règle est configurée sur le module MIP, alors les paquets relatifs à ce flux sont remontés à l'application concernée via le "Gestionnaire" par cette méthode.
En retour, l'application renvoie le paquet modifié sous forme de tableau d'octets.

3.3 Evaluation

Dans cette évaluation, le nœud intermédiaire est illustré en tant que passerelle entre un réseau filaire et un réseau sans-fil de type 802.11.
Les réseaux locaux (de type sans-fil Wifi 802.11) sont des réseaux à bande passante partagée par tous les utilisateurs. Cette caractéristique implique que la bande passante disponible (ou utilisée) d'un réseau peut être très variable. Elle dépend du nombre de terminaux connectés sur ce réseau mais aussi des applications utilisées. En effet, un téléchargement utilise beaucoup de bande passante, comparé à une consultation de site Web. De la même manière, la visualisation d'un film utilise plus de bande passante que la lecture d'un mail…
Basé sur ce constat, il apparaît que si le réseau est chargé, il se peut que des paquets soient « perdus » et donc même si le client possède un terminal capable d'analyser les données des paquets du service avec une très bonne qualité, la qualité sera en fait fortement dégradée puisque de nombreux paquets seront perdus. Cela se vérifie d'autant plus sur un réseau sans-fil de type Wifi où le nombre de paquets perdus peut être très important.
C'est basé sur ce constat qu'a eu lieu cette évaluation où l'objectif est de superviser la qualité du réseau Wifi et notamment la bande passante et de déclencher les mécanismes d'adaptation de service avant que le réseau sans-fil ne soit trop chargé et donc ne permette plus la fourniture du service dans une qualité acceptable et aboutisse ainsi une indisponibilité du service pour certains utilisateurs (si non tous).
Les services utilisés pour démontrer l'intérêt du nœud intermédiaire permettant l'adaptation de contenu en fonction du contexte sont des services multimédias (audio et vidéo), dits hiérarchiques. Cette approche de codage hiérarchique semble prometteuse pour réduire la bande passante utilisée tout en gardant une qualité satisfaisante. En fonction de la qualité des réseaux (et notamment de la bande passante utilisée) et du type de terminal de l'utilisateur (PC, PDA, téléphones mobiles), un module applicatif tournant sur le nœud intermédiaire est capable "d'adapter" les données applicatives du service. Pour le service audio, la troncature des paquets permet de réduire la taille des paquets avec, certes, une diminution de la qualité selon la quantité de données tronquées mais toutefois acceptable (voir la section 3.3.1). Pour le service vidéo, le choix est de supprimer certains paquets selon l'information qu'ils représentent. Par exemple, des paquets liés aux trames I d'une vidéo MPEG ne seront jamais supprimés tandis que les paquets liés aux trames B et P peuvent l'être selon l'action à effectuer en fonction des conditions du réseau (voir la section 3.3.2).

Les concepts apportés dans cette étude et l'architecture du nœud intermédiaire ont été validés via la réalisation d'un démonstrateur, évalué avec le service audio et via simulation, aussi bien pour le service audio et le service vidéo. Avant de détailler cette évaluation, les deux codecs, qui ont été utilisés, sont décrits pour mieux comprendre le fonctionnement du démonstrateur.

3.3.1 Codec audio hiérarchique

Les codecs hiérarchiques audio ne sont pas vraiment nouveaux dans la communauté audio de recherche. Par exemple, le codec d'ITU-T G.727 est un exemple de codeur hiérarchique qui peut coder de 16 kbit/s à 40 kbit/s avec un débit intermédiaire à 24 et 32 kbit/s. Mais cette solution n'est pas la plus satisfaisante car certains codecs peuvent être très efficaces à de faibles débits mais mauvais pour des débits plus élevés et vice-versa pour d'autres codeurs. Pour de tels codecs, la scalabilité est réalisée mais la qualité est dégradée, comparé à un codage à débit fixe et connu.
Actuellement, il est admis qu'il est préférable d'avoir 2 encodages différents : un pour le cœur du flux hiérarchique, qui fournit une meilleure qualité à bas débit et un autre pour les couches d'améliorations, plus approprié à des débits plus élevés. Ceci permet de couvrir une variété de débits, de qualité satisfaisante aux débits relativement bas et de très bonne qualité à des débits plus élevés.

Les chercheurs de France Télécom R&D, spécialistes en codage ont proposé un nouveau codec hiérarchique, visant à offrir une meilleure qualité, aussi bien à bas débit qu'à haut débit. Dans ce démonstrateur, pour valider le concept d'utilisation du nœud intermédiaire, ce codec sera utilisé. Ce codec est présenté rapidement ici mais des détails peuvent être trouvés dans [Kov'04].

Le codec hiérarchique fonctionne dans la gamme de débit de 12.2 kbit/s à 31.4 kbit/s. Le débit maximum de codec (31.4 kbit/s) a été choisi pour obtenir un débit de 32 kbit/s en incluant l'en-tête RTP qui est généralement utilisé.
La granularité de scalabilité de ce codec est de l'ordre de l'octet, ce qui signifie que le paquet peut être tronqué à l'octet près, aboutissant ainsi à une grande étendue de débits possibles (et non plus uniquement 2 ou 3 comme les codecs hiérarchiques connus). Ceci apporte flexibilité et adaptabilité.
Le bitstream de 12.2 kbit/s produit par le codec est compatible avec le codec AMR standard défini à l'ETSI/3GPP [Grill'97] et également avec le codec utilisé dans les réseaux GSM (GSM-EFR).
Le codec hiérarchique est une combinaison de deux codages cascadés: le premier (codeur de noyau) est le codeur AMR CELP de l'ETSI/3GPP. Ce codeur est défini afin d'assurer la compatibilité avec un codeur AMR simple (non hiérarchique).
Le second est un codeur de transformation MDCT (Modified Discrete Cosine Transform), développé dans les laboratoires de France Telecom sous le nom du « codeur de TDAC » (Time-Domain Aliasing Cancellation) [Bes'02]. La Figure 3. 5 présente la structure de codage de ce codec hiérarchique.
La fréquence d'une trame AMR classique est de 20 ms. Pour ce codec, 3 trames AMR et TDAC sont encodées ensemble aboutissant à une fréquence de 60 ms.
Au niveau des paquets réseau, une trame complète est encodée sur 240 octets.

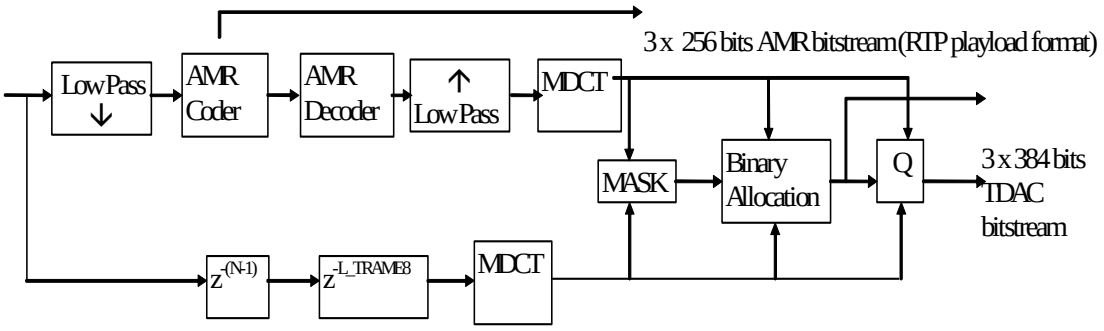

Figure 3. 5: Structure d'encodeur du codec audio

Le processus de décodage des trames est le suivant: les trois trames AMR sont d'abord décodées. Ensuite les trois trames TDAC sont décodées en utilisant les échantillons AMR décodés pour fournir le résultat.
Le codec hiérarchique emploie un algorithme efficace de dissimulation d'effacement de trames [Kov'03] qui peut corriger l'effet de perte de trames même en cas de nombreuses pertes de paquets.

3.3.2 Codage Vidéo Hiérarchique

La diffusion vidéo est prévue par beaucoup d'utilisateurs comme la prochaine "killer application". Celle-ci sera diffusée sur différents réseaux d'accès, tels que les réseaux d'accès fixes à bande large (comme l'ADSL), ou bien encore plus haut débit comme les réseaux FTTH (Fiber To The Home) qui sont en train de se déployer, mais aussi sur les réseaux sans fil mobiles (UMTS ou Wifi…) qui eux offrent une bande passante plus limitée. Les spécialistes du codage vidéo ont donc décidé d'étudier la technologie de codage hiérarchique pour le codage vidéo; l'objectif principal étant de coder le contenu vidéo seulement une fois et de pouvoir le fournir sur différents réseaux d'accès en utilisant les concepts de scalabilité induit. Les normes principales, liées au concept de codage hiérarchique (SVC pour Scalable Video Coding), sont les normes UIT-T H.264 standard et la partie 10 d'ISO/CEI MPEG-4 (ISO/CEI 14496-10), qui sont techniquement identiques depuis le rapprochement de deux organismes de normalisation pour unir leur effort pour faire cette norme.
Dans la norme, trois niveaux de scalabilité ont été définis :

- la scalabilité temporelle définit l'enchainement des trames vidéo et le lien entre eux. Par exemple, supprimer des trames de type B permet de réduire le débit binaire du flux vidéo. Selon la norme, au moins, 2 niveaux de scalabilité temporelle doivent être supportés. L'image suivante (Figure 3. 6) présente un exemple de 3 niveaux de scalabilité temporelle (ligne bleue, pointillé vert et ligne disjointe orange) suivant les trames MPEG qui sont supprimées.

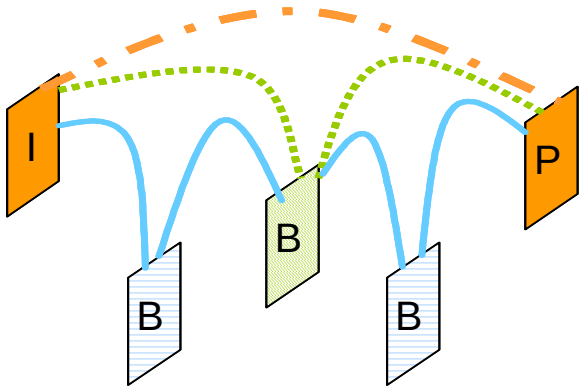

Figure 3. 6: Exemple de scalabilité temporelle dans SVC

- la scalabilité spatiale représente la résolution dans laquelle l'image sera codée. Les formats de type CIF (Common Intermediate Format) ou QCIF (Quarter Common Intermediate Format) sont des exemples de taille d'images qui peuvent être utilisés dans le SVC pour définir le format de l'image.
- La scalabilité de qualité (SNR) définit la qualité dans laquelle l'image sera codée. 3 modes ont été définis: scalabilité SNR grossière (coarse-granular SNR scalability), scalabilité SNR moyenne (medium-granular SNR scalability) et scalabilité SNR fine (fine-granular SNR scalability). Les trois niveaux du scalabilité peuvent être utilisés conjointement. L'encodeur est l'entité qui définit le niveau de scalabilité souhaité le flux vidéo.

Le codage SVC est bien documenté dans la littérature (par exemple [AVC'05] [Wie'03]....).
Dans le démonstrateur mis en œuvre pour le service vidéo SVC, seule la scalabilité temporelle sera appliquée.

3.3.3 Scénario

Dans ce scénario, le nœud intermédiaire représente une évolution d'un point d'accès Wifi, où de nouvelles fonctionnalités sont introduites, tels que les sondes de supervision, le plan de connaissance du contexte de l'utilisateur, les modules applicatifs et le gestionnaire central...

Le scénario cible est le même pour les tests avec le service Audio et le service Vidéo. Il peut être schématisé de la façon suivante, illustré avec le cas du service Audio (Figure 3. 7). Pour le service vidéo, la même architecture est appliquée, uniquement le traitement sur les flux par le module applicatif déployé sur le nœud diffère.
Un fournisseur de service (WebAudio ou WebVideo) envoie un flux de données hiérarchique. Les utilisateurs, localisés dans un réseau sans-fil 802.11 reçoivent et écoutent/visualisent le flux.
Sur le nœud intermédiaire, une sonde réseau supervise sans interruption l'état du réseau sans-fil, notamment la bande passante.
Si la bande passante utilisée excède un seuil spécifié par les applications, le nœud est configuré pour remonter les paquets de données au module applicatif qui se chargera

de tronquer à un niveau donné selon le type de terminal des utilisateurs (ou des paquets du flux vidéo sont supprimés).
Par exemple, la police mise en œuvre dans ce scénario est le suivant: pour des utilisateurs utilisant un ordinateur portable, le flux n'est pas tronqué, pour ceux qui écoutent sur PDAs, le flux est tronqué à un certain niveau et pour des utilisateurs ayant des téléphones portables, seulement le cœur du flux audio hiérarchique est transmis par le nœud intermédiaire.
Dans ce démonstrateur, le contexte de l'utilisateur est utilisé en prenant en compte des informations fixes comme le type de terminal qu'il utilise ou dynamiques comme l'état du réseau sans-fil.

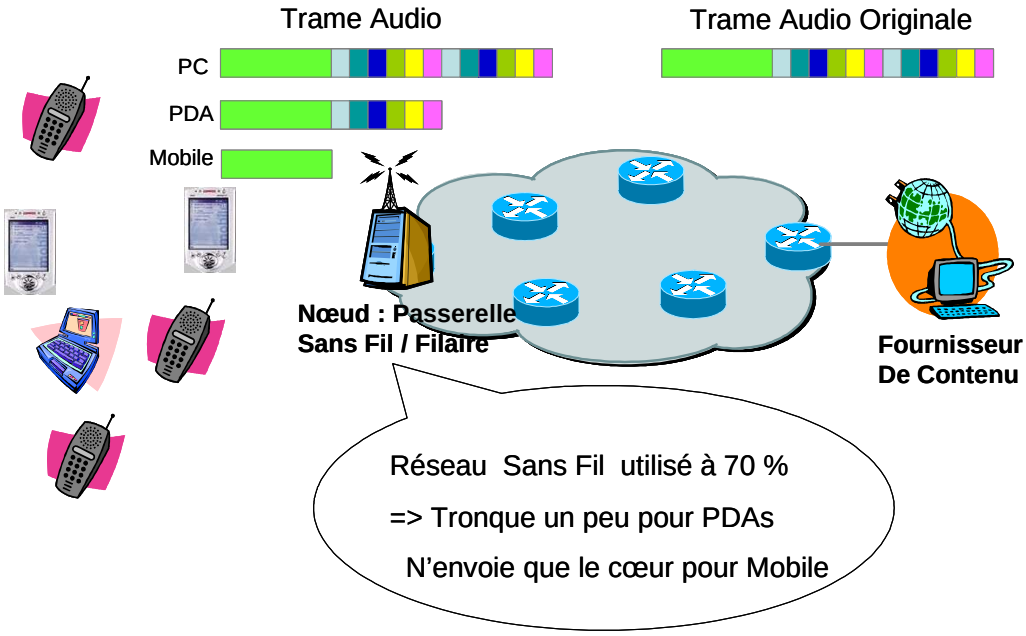

Figure 3. 7: Scénario du démonstrateur

3.3.4 Démonstrateur

Une implémentation du nœud intermédiaire a été mise en œuvre notamment pour illustrer le cas d'utilisation de ce nœud en tant que point d'accès Wifi évolué (passerelle entre un réseau filaire et un réseau sans-fil de type 802.11) [Math'05].

3.3.4.1 Implémentation du démonstrateur

La Figure 3. 8 présente l'architecture du démonstrateur avec les composants mis en œuvre.

Le nœud intermédiaire a été implémenté sur un système *Linux RedHat 9.2 avec un noyau 2.4.20*.
Le driver utilisé pour l'interface Wifi est le driver *Orinoco*.

Le composant de routage repose sur les mécanismes fournis par l'environnement Linux.
Le composant MIP se fonde sur le mécanisme « *Netfilter* » de Linux et entre autres via l'utilisation de filtres, basés sur les adresses IP, les ports IP des entités source et destination et les protocoles…. De plus, il étend une fonctionnalité puisqu'il permet de transférer des paquets à différents modules de l'espace utilisateur et de ne pas conserver de copies des paquets dans le noyau (pour ne pas saturer les files d'attente). Cette évolution de Netfilter, et plus globalement ce composant, n'est pas plus détaillée ici car elle se retrouve aussi dans les travaux [Nguy'04], définis en collaboration.

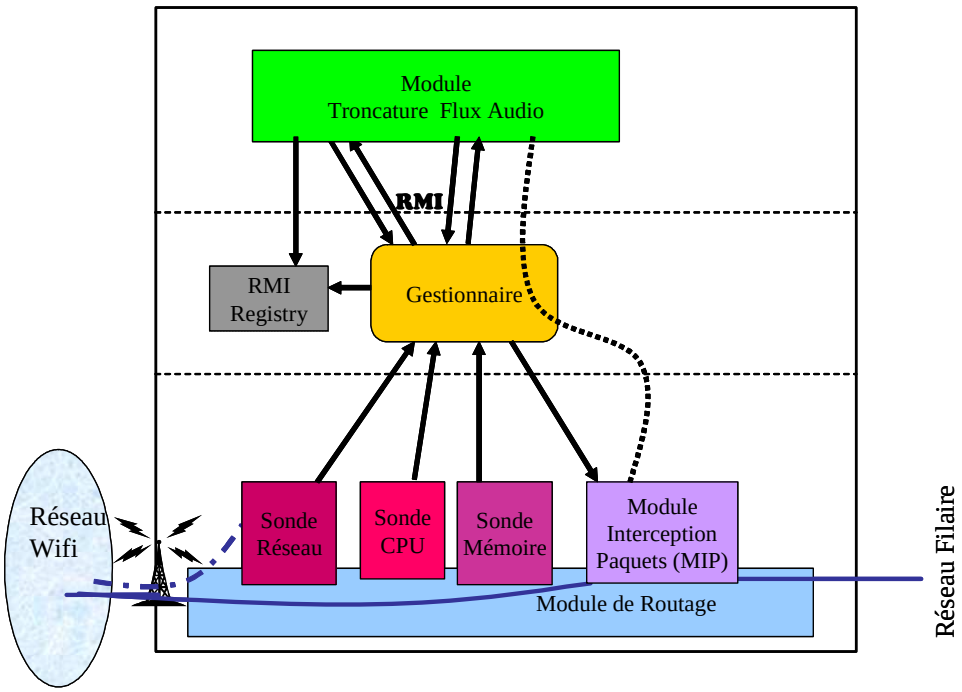

Figure 3. 8: Implémentation du nœud intermédiaire: passerelle filaire/wifi

La sonde de CPU et la sonde de mémoire sont des composants développés en relation avec les informations fournies par la commande *top* du système d'exploitation linux. Pour la sonde CPU, cette valeur regroupe le taux d'utilisation utilisateur, système et nice. Le temps non utilisé correspond donc au temps dans lequel le système est dans l'état "idle". Le taux d'utilisation est défini en pourcentage. La sonde Mémoire concerne le taux d'utilisation de la mémoire vive et retourne aussi le taux d'utilisation en pourcentage.

La sonde réseau, une Sonde Wifi, chargée de superviser la bande passante utilisée dans un réseau sans-fil Wifi, a été implémentée. Etant en mode infrastructure 802.11, tous les paquets traversent le nœud intermédiaire pour le routage des paquets du réseau filaire/sans-fil. La sonde Wifi supervise les paquets qui passent via le nœud, compte le nombre d'octets transmis/reçus dans ces paquets et horodate les captures. Elle calcule ainsi le débit par seconde en fonction du nombre de bits reçus et l'horodatage. La sonde remonte ensuite cette information au gestionnaire. Cette sonde réseau est développée à partir de la librairie « *pcap* ».

Le gestionnaire du nœud est implémenté en langage *Java.*

L'API de communication avec les modules applicatifs (interfaces 2 et 6) est *RMI (Remote Method Invocation)*. En effet, l'utilisation de RMI a été privilégiée par rapport à de simples sockets, car il offre la possibilité d'avoir une architecture distribuée où les modules applicatifs peuvent être sur des nœuds différents du nœud courant.
Le serveur d'enregistrement se base sur le *registry server* de RMI.

La configuration de l'entité MIP (interface 3) se fait par l'utilisation de la librairie *libipq*.
Par exemple, la requête de configuration :
iptables -A FORWARD -j QUEUE -p udp -s 192.168.1.13
configure le composant MIP pour remonter les paquets UPD provenant de l'adresse IP 192.168.1.13

Le flux audio du service mis en œuvre est encodé en utilisant le codage hiérarchique décrit dans la section 3.3.1, avec un cœur fixé à un débit réseau de 12.8 kbit/s (96 octets: cœur du codec pur à 12.2 kbit/s plus l'en-tête RTP) et des couches d'améliorations (octets d'amélioration dans les paquets) allant jusqu'à 32 kbit/s (240 octets).

3.3.4.2 Evaluation du démonstrateur

Le démonstrateur a été évalué de deux manières différentes; une avec de vrais utilisateurs écoutant le flux audio, pour certifier de la qualité perçue du service et l'autre avec un simulateur de clients pour juger du gain de bande passante potentiel en tronquant les flux à certains niveaux et avec un client réel pour écouter le flux reçu en même temps.

Le premier test, subjectif, indique que la qualité audio à 32 kbit/s est très bonne et qu'évidemment, en tronquant le flux audio, la qualité est inférieure. En tronquant le flux pour descendre à un débit de 21 kbit/s, la qualité est encore tout à fait acceptable. A 12,8 kbit/s, le flux audio est encore acceptable puisque nous comprenons clairement, mais évidemment la qualité n'est pas exceptionnelle. Ceci est conforme et en relation avec les tests du codec audio hiérarchique seul.
Cela signifie donc que le nœud intermédiaire fonctionne correctement et n'entraine pas de retard détectable dans la qualité fournie malgré le traitement des données (notamment la troncature), le fonctionnement interne du nœud et en ayant activé les sondes (CPU, mémoire, réseau).

Pour l'évaluation plus technique et plus objective, relative au gain de bande passante (en ayant toujours en tête que l'objectif est de réduire le flux émis puisque le réseau sans-fil est saturé), un banc de test a été installé (Figure 3. 9), comprenant un serveur de diffusion audio (simulant une Web radio par exemple) sur le réseau filaire, le nœud intermédiaire, agissant en tant que passerelle et interconnectant le réseau filaire et le réseau sans fil, un ordinateur hébergeant un client réel pour écouter le flux audio et jugé de la qualité reçue pour les flux tronqués et un ordinateur qui simule des utilisateurs recevant les flux audio. Sur cet ordinateur, X clients sont

simulés en ouvrant des connexions avec des ports IP différents pour simuler des clients différents.

L'objectif de cette évaluation pratique est de réduire la bande passante (lorsque saturée) en tronquant le flux audio pour certains utilisateurs (par exemple des utilisateurs de PDAs) et aussi déterminer le nombre de clients potentiels qui pourraient être ajoutés (et recevoir le flux) lorsqu'un certain nombre de flux sont tronqués.

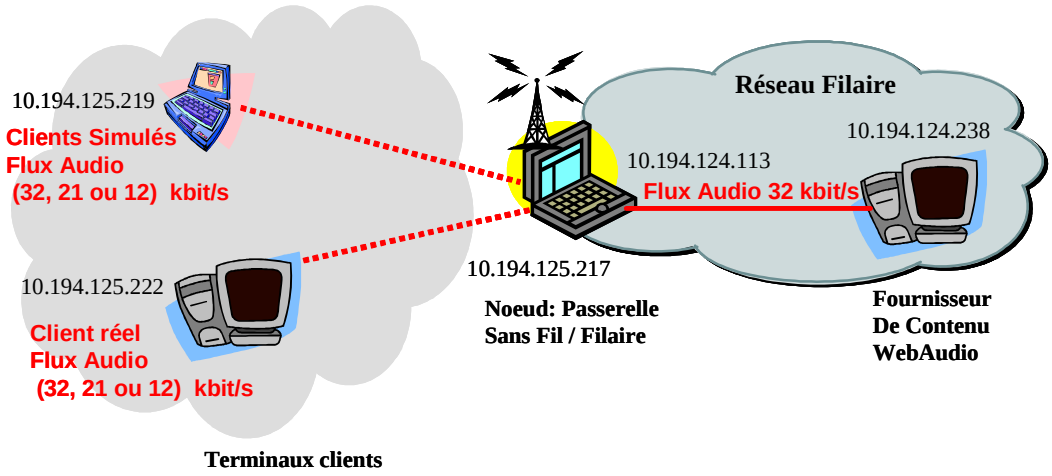

Figure 3. 9: Testbed réseau du démonstrateur

Rôle	Machine	CPU	RAM	OS
Serveur de diffusion audio	Compaq Armada M700	Pentium III 1 Ghz	392 Mo	Windows 2000
Simulateur de client	Compaq nc4000	Pentium M 1,3 Ghz	480 Mo	Windows XP
Client réel	Compaq nc4000	Pentium M 1,3 Ghz	256 Mo	Windows XP
Nœud Intermédiaire	Compaq Evo N610c	Pentium 4 m	520 Mo	Linux Redhat 9.2 Kernel 2.4.20

La procédure suivie pendant le test est la suivante :

- le serveur audio émet sur le réseau filaire un flux encodé à 32 kbit/s
- X clients (plus le client réel) reçoivent le flux audio
- X est augmenté progressivement (de 58 à 65) dans le but de saturer le réseau sans-fil : le taux de perte de paquets pour les clients est mesuré
- Le nœud intermédiaire tronque le flux audio à 21 kbit/s puis 12,8 kbit/s pour Y clients: Y étant 20, 40, 62 ou 65.
- Le gain de clients potentiels d'utilisateurs est évalué en tronquant certains flux pour certains utilisateurs.

Le tableau (Figure 3. 10) donne les résultats de ces tests.

Nb d'utilisateurs tronqués / nb total d'utilisateurs

Débit	0/58	0/60	0/62	20/58	20/60	20/62	40/58	40/60	40/62	40/65	62/62	65/65
32	3,7	67,6	87,6									
21							0,1	15,2	63,7		21,4	
12,8				0	1,1	68,2	0	0,4	0,6	71,6		17,9

Taux de paquets perdus : %

Figure 3. 10: Taux de perte / nb utilisateurs pour flux audio hiérarchique

Ce tableau rassemble des informations sur plusieurs dimensions. Pour l'expliquer, on peut dire que :

- La première ligne signifie : Quand tous les clients reçoivent le flux audio complet (32 kbit/s), aucun paquet n'est perdu jusqu'à 57 clients. Quand le 58ème client écoute le flux, 3,7 % de paquets sont perdus. Quand il y a 60 clients, 67,6 % de paquets sont perdus et 87,2 % sont perdus pour 62 clients. Cela signifie qu'un flux audio de 32 kbit/s peut être diffusé sans perte jusqu'à 57 clients.
- La deuxième ligne signifie : quand le flux audio est tronqué à 21 kbit/s pour 40 utilisateurs, il n'y a aucun paquet perdu jusqu'à 57 clients. 0,1 % de paquets sont perdus pour 58 clients, 63,7 % pour 62 clients. La troncature à 21 kbit/s pour 40 utilisateurs (parmi les X) permet d'atteindre 1 ou 2 utilisateurs supplémentaires (58 et 59).
- La troisième ligne signifie : quand le flux audio est tronqué à 12.8 kbit/s pour 20 utilisateurs, il n'y a aucun paquet perdu jusqu'à 59 clients. 1,1 % de paquets sont perdus pour 60 clients et 68,2 %pour 62 clients. La troncature à 12,8 kbit/s pour 20 utilisateurs permet d'atteindre 3 ou 4 utilisateurs supplémentaires (58, 59, 60 et 61). Quand le flux audio est tronqué à 12,8 kbit/s pour 40 utilisateurs, il n'y a aucun paquet perdu jusqu'à 59 clients. 0,4 % de paquets sont perdus 60 clients, 0,6 % pour 62 clients et 71,6 % pour 65 clients. La troncature à 12,8 kbit/s pour 40 utilisateurs permet d'atteindre 5 ou 6 utilisateurs supplémentaires (58, 59, 60, 61, 62 et 63).

Nous pouvions espérer obtenir plus d'utilisateurs recevant les flux audio, quand les flux sont tronqués alors que l'évaluation nous indique seulement quelques utilisateurs de plus (jusqu'à 5).
En effet, pour 60 utilisateurs et 40 recevant le flux audio à 12,8 kbit/s, la bande passante utilisée est approximativement 1,152 Mbit/s (40 * 12,8 + 20 * 32). La bande passante n'est ainsi pas surchargée par les données elles-mêmes et nous pouvons penser qu'il reste de la bande passante encore disponible.

Mais dans ce cas-ci, la technologie radio sous-jacente est la raison de cette limitation. Les réseaux 802.11 se fondent sur le mécanisme de CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) pour envoyer des paquets. Le CSMA/CA (Figure 3. 11) inclus des mécanismes pour éviter des collisions entre les nœuds (puisque le réseau 802.11 est un réseau radio partagé) et pour donner également une chance à chaque nœud d'envoyer des données (un nœud ayant la permission d'envoyer n'enverra pas toutes ses données sans interruption d'un seul coup). Des temps d'attente (timeout) sont implémentés pour permettre à d'autres nœuds d'envoyer des données. Le mécanisme de CSMA peut être schématisé de la manière suivante:

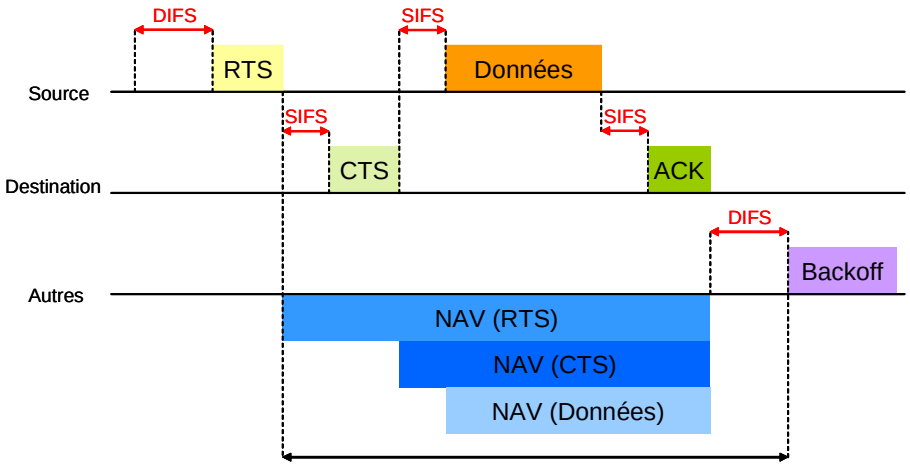

Figure 3. 11: Système de transmission CSMA/CA

L'intervalle SIFS (Short inter Frame Space) est fixé à 10µs, le DIFS (Distributed InterFrame Space) est fixé à 50µs. Mais le temps de contention (backoff) est variable et est un multiple de la fenêtre de collision (50µs). Le multiple est généré aléatoirement. Il n'est alors pas possible de connaître précisément combien de temps sera nécessaire pour envoyer un paquet mais il est évident que ce temps peut être court mais peut être long également. Tout le temps à attendre avant d'envoyer des données empêche évidemment d'envoyer plus de données sur le réseau sans fil. Le nombre de clients additionnels obtenus dans le test dépend donc seulement de la longueur des données envoyées. En effet, plus de données seront à envoyer, plus long ce sera.

Concernant l'évaluation du nœud intermédiaire lui-même et notamment le temps de traitement induit par l'utilisation du nœud, les mesures obtenues ont confirmé le ressenti avec des utilisateurs, c'est-à-dire que cela n'entraine pas un retard détectable et une perturbation de la qualité.
En effet, le temps de traitement, entre l'arrivée d'un paquet dans le nœud et sa réémission après adaptation est de l'ordre de 1 ms, jamais plus de 2 ms pour gérer jusqu'aux 65 utilisateurs, nombre à partir duquel la qualité n'est plus bonne.

Temps Traitement : 1-2 ms

Cette valeur n'a pas mesuré plus précisément puisque l'objectif était de montrer la faisabilité de cette solution dans des conditions réelles et donnant des résultats

satisfaisants. Savoir que le temps de traitement est de 1,2ms ou 1,4ms importe peu, puisque cela dépend aussi de la capacité de la machine.
Ce qu'il faut retenir, c'est que le délai induit est très faible, ne perturbant pas du tout la fluidité et la qualité du flux.

Pour conclure, cette évaluation a prouvé que pour un tel scénario (diffusion de flux audio hiérarchique), la bande passante n'est pas vraiment le facteur principal de limitation mais que c'est plutôt le mécanisme CSMA/CA lui-même et que le nœud intermédiaire gère parfaitement ce cas. Cependant, le concept semble prometteur et pourrait offrir de meilleures performances sur une autre technologie.

3.3.5 Simulations

La simulation a été réalisée avec le simulateur ns2 et avec le simulateur Qualnet. Les résultats pour tous les tests passés sont très semblables; la différence entre les deux simulateurs est seulement de 1 ou 2 utilisateurs. Pour la clarté dans les figures, seulement les résultats fournis par le simulateur ns2 seront tracés et présentés.

La topologie de réseau configurée dans le simulateur est en relation avec le scénario décrit dans la section 3.3.3 c'est-à-dire un serveur de diffusion, un nœud servant de passerelle et plusieurs utilisateurs recevant les flux.
Les utilisateurs sont localisés pour être dans la proximité du nœud intermédiaire, c'est-à-dire à moins 50 mètres et sont fixes.

Tous les tests ont été réalisés pour un réseau sans fil 802.11b. Même si la bande passante théorique d'un tel réseau est 11 Mbit/s, le débit réel est moindre. Dans les tests, le taux de base a été configuré à 7.7 Mbit/s, comme détecté par [Chau'06].

Deux services ont été évalués comme modules applicatifs déployés sur le nœud: un pour les flux hiérarchiques audio, codé comme décrits dans la section 3.3.1 et un pour les flux hiérarchiques audio, comme décrits dans 3.3.2.

La configuration matérielle et logicielle utilisée pour les simulations est la suivante :

Machine	CPU	RAM	OS	Ns2	Qualnet
Compac Evo n610c	Pentium IVm	516 Mo	Linux Debian 2.14.2 Kernel 2.6.16	2.31	3.9

3.3.5.1 Résultats des tests pour le flux audio hiérarchique

Pour le service audio, le scénario est simple. Les données ont été tronquées pour tous les utilisateurs à 3 niveaux différents, menant à 4 qualités différentes, représentant les 4 débits suivants: 32 kits/s, 24 kbit/s, 16 kbit/s et 12.8 kbit/s. Ce test permet de savoir le nombre maximum des utilisateurs qui peuvent recevoir le flux audio pour le débit donné.

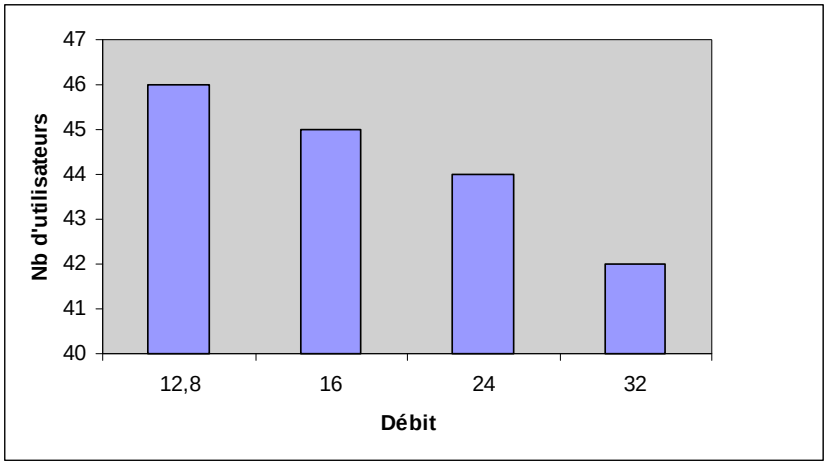

Figure 3. 12: Nb d'utilisateurs du flux audio hiérarchique en fonction du débit

Comme nous pouvons voir sur la Figure 3. 12, le nombre maximum des utilisateurs est limité et inférieur à ce que la bande passante pourrait permettre : entre 42 et 46 utilisateurs, selon le débit.
Nous pouvions penser délivrer le flux à plus d'utilisateurs quand le flux est tronqué (gain seulement de 3 ou 4 utilisateurs) mais la raison est identique à celle expliquée dans l'évaluation du prototype dans la section 3.3.4.2, c'est-à-dire que la bande passante n'est pas le facteur limitant, mais c'est le mécanisme fondamental de CSMA/CA qui l'est (Figure 3. 11). Ce test confirme donc que la troncature de données audio pour de tels réseaux sans fil, se fondant sur des mécanismes de CSMA/CA n'est pas vraiment efficace.

3.3.5.2 Résultats des tests pour le flux vidéo hiérarchique

Pour le service vidéo, 4 scénarios différents ont été réalisés :

1) un flux vidéo codé à 512 kbit/s, délivré aux utilisateurs, en paquets de taille de 1400 octets, puis de 1000 octets et enfin de 600 octets.

Évidemment, plus le paquet est petit, plus la fréquence d'émission est courte (puisque l'objectif est de garder un débit de 512 kbit/s).

La Figure 3. 13 montre que le flux vidéo à 512 kbit/s peut être délivré jusqu'à 6 utilisateurs sans perte de paquet pour des paquets de 600 octets, jusqu'à 8 pour des paquets de 1000 octets et 10 pour des paquets de 1400 octets.
Après ce maximum, un nombre important de paquets sont perdus menant rapidement à une qualité non acceptable.
Les résultats de ce test prouvent que le nombre maximum d'utilisateurs atteint pour un flux composé de petits paquets est moindre que pour un flux composé de paquets de taille supérieure. En effet, cela s'explique par le fait que le nœud doit émettre plus fréquemment et donc doit attendre plus souvent pour accéder au médium du réseau sans fil (voir l'explication de CSMA/CA).

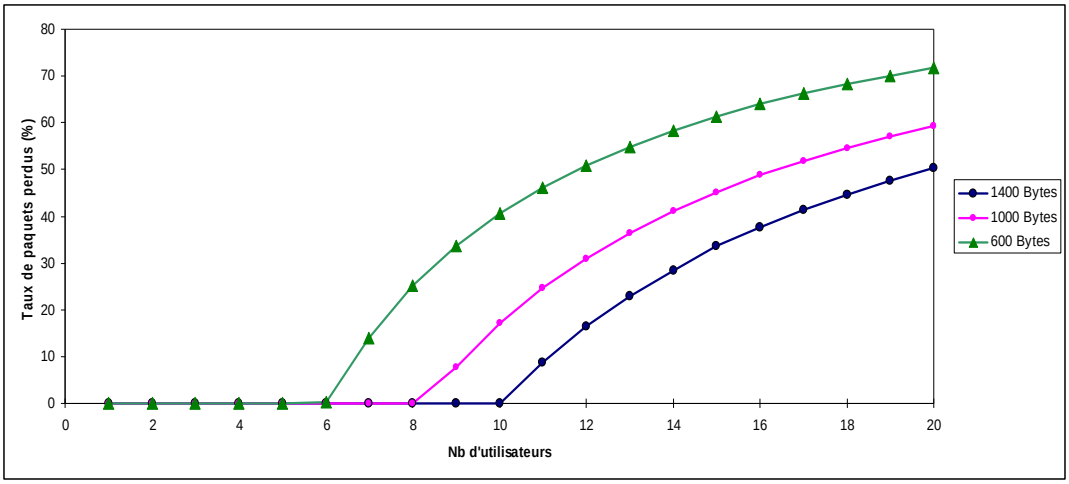

Figure 3. 13: Nb d'utilisateurs pour une vidéo à 512kbit/s avec des tailles de paquets différentes

Pour ce cas, ce test permet de dire qu'il est préférable de coder les paquets vidéo avec une taille de paquet aussi grande que possible (e.g. le MTU, Maximum Transmission Unit).

2) un flux vidéo codé à 512 kbit/s où les paquets sont tronqués pour arriver à un débit de 350 kbit/s et puis de 256 kbit/s pour tous les utilisateurs.

3) un flux vidéo codé à 512 kbit/s où la scalabilité est assurée en utilisant la scalabilité temporelle à 2 niveaux : le premier niveau en supprimant 1 paquet sur 2, le second cas en supprimant 3 paquets sur 4 (cf section 4.3).

La Figure 3. **14** présente les résultats pour ces 2 tests.
Comme vu dans le test avec le flux audio, ces simulations prouvent que la troncature des flux permet d'atteindre un peu plus d'utilisateurs (puisque la taille de paquet et le temps d'émission de paquet sont moindres). Tronquer le flux vidéo à 256 kbit/s permet de fournir le service à 3 utilisateurs supplémentaires.

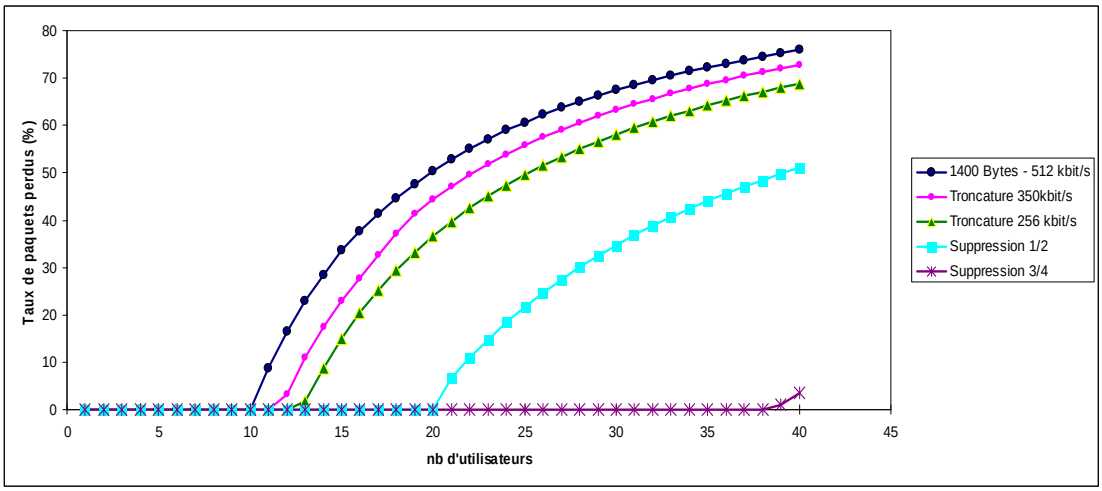

Figure 3. 14: Nb d'utilisateurs avec troncature et scalabilité temporelle du flux vidéo

Mais la Figure 3. **14** prouve également que le gain est plus important quand la scalabilité temporelle du flux vidéo est appliquée. En effet, la suppression de paquets signifie que le nœud n'a pas besoin de demander et d'attendre pour accéder au médium sans fil et ainsi gagne beaucoup de temps.
Les résultats montrent que supprimer 1 paquet sur 2 permet de doubler le nombre d'utilisateurs recevant le flux et l'émission de seulement un paquet sur 4 permet d'atteindre 38 utilisateurs sans perte de paquet. Ce test prouve clairement que même si la troncature des paquets est bonne, la solution d'adaptation la plus efficace pour économiser de la bande passante sur un réseau 802.11b est de supprimer quelques paquets.
Évidemment, il faudrait également examiner la qualité du flux reçu (après adaptation) avec de vrais utilisateurs. Ceci permettrait d'envisager un compromis entre le gain des utilisateurs et la qualité reçue. Cependant, puisque le nœud est configurable et adapte dynamiquement les flux, nous pouvons imaginer commencer le procédé d'adaptation avec une troncature des flux vidéo quand il y a peu d'utilisateurs et basculer sur l'option de suppression de paquets s'il y a beaucoup d'utilisateurs.

4) Le dernier test vise à simuler le scénario expliqué dans la section 4.4, où x utilisateurs visualisent le flux vidéo sur leur PC avec la meilleure qualité (512 kbit/s), y utilisateurs sur leur PDAs avec une qualité inférieure (256 kbit/s) et les z utilisateurs restants visualisent la vidéo sur leur téléphone portable avec la plus basse qualité (128 kbit/s).

Évidemment, le pourcentage du nombre d'utilisateurs en fonction des terminaux devrait être configuré pour représenter un environnement réel. Dans cette simulation, plusieurs cas ont été réalisés avec peu d'utilisateurs ont un PC, un peu plus ont un PDA et encore plus ont des téléphones portables. Ce scénario semble être réaliste pour un réseau sans-fil de type 802.11b, par exemple dans une gare SNCF quand les gens attendent le train.

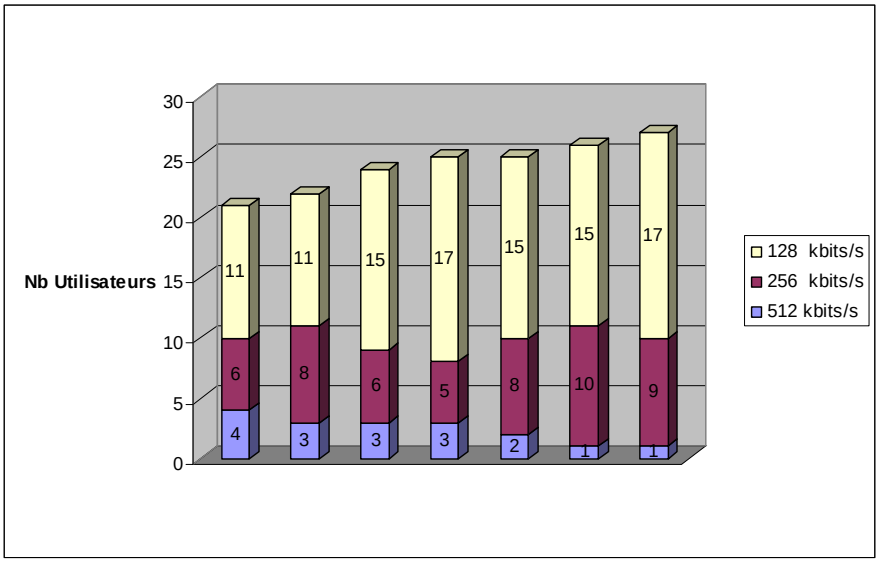

Figure 3. 15: Nb d'utilisateurs possible en fonction des terminaux utilisés

La Figure 3. 15 montre le nombre d'utilisateurs pour chaque catégorie qui peuvent recevoir les flux vidéo sans perte de paquet dans chaque configuration du test. En se rapportant à la Figure 3. 14, cela illustre aussi le fait qu'en adaptant le flux vidéo pour certains utilisateurs, davantage d'utilisateurs peuvent visualiser le flux que si aucune adaptation n'était réalisée.

3.4 Conclusion

Ces tests prouvent que l'insertion d'un nœud intermédiaire, supervisant la qualité d'un réseau sans-fil et offrant la possibilité à des modules applicatifs d'adapter des flux multimédias hiérarchiques, principalement vidéo, entre le serveur et les clients est tout à fait faisable puisqu'il n'engendre qu'un délai de traitement très faible et ne perturbe en rien la fourniture du service.
Si tronquer des flux audio sur de tels réseaux 802.11b s'est avéré peu intéressant, à cause de la technologie sous-jacente (CSMA/CA), les simulations prouvent que la scalabilité temporelle appliquée à des flux vidéo hiérarchiques présente des arguments plus intéressants en termes de gain de bande passante et du nombre d'utilisateurs potentiels supplémentaires.
Pour conclure, la mise en œuvre du nœud intermédiaire en tant que point d'accès Wifi évolué (ou passerelle sans-fil/filaire), illustrée par ce scénario, est une option intéressante. D'autres scénarios d'utilisation du nœud intermédiaire sont réalisables, dans une architecture réseau différente et utilisant d'autres modules applicatifs pour offrir un service adapté aux contextes des utilisateurs.

Chapitre 4

Nœud Potacco transparent permettant le déploiement dynamique de modules applicatifs

4.1 Introduction

Le chapitre précédent a introduit l'architecture du nœud intermédiaire avec des entités essentielles telles que les collecteurs de contexte et le gestionnaire du nœud. Dans ce chapitre, l'architecture du nœud est étendue pour permettre au nœud intermédiaire de fonctionner en mode "transparent", c'est-à-dire de pouvoir intégrer des moyens pour réaliser des traitements applicatifs sans être détectable et conserver intègre la connexion de bout en bout [Mat'04] [Gou'04].
De plus, les modules applicatifs pouvant être spécifiques aux applications ou à certaines configurations, un mécanisme de déploiement dynamique de code est ajouté dans l'architecture du nœud. Ce mécanisme doit authentifier aussi bien le fournisseur de code que le nœud sur lequel il sera déployé et activé.
Ces deux dernières fonctionnalités complètent l'architecture du nœud intermédiaire et aboutissent à la définition du nom Potacco, signifiant *nœud **PO**lymorphique **T**ransparent pour l'**A**daptation de **C**ontenu adapté au **Co**ntexte.* Le terme polymorphique indique la faculté du nœud d'héberger différents modules applicatifs (et variable dans le temps) ainsi que la faculté du nœud de fonctionner suivant différentes configurations réseau: le nœud peut ainsi prendre plusieurs formes.

Ce chapitre présente l'architecture du nœud Potacco intégrant les deux nouveaux modules; le module de Gestion des Connexions Réseau (GCR), intégré dans le Gestionnaire du nœud et l'entité fonctionnelle de déploiement sécurisé de code. Un démonstrateur, consistant à insérer des informations représentatives du contexte utilisateur dans les requêtes utilisateurs HTTP pour permettre à un module de service distant de pouvoir fournir un contenu adapté, est ensuite présenté. Dans cette mise en œuvre; la faculté du nœud Potacco d'héberger des modules applicatifs différents est aussi illustrée. Le démonstrateur est enfin intégré et évalué dans une architecture réseau actuelle de type ADSL.

4.1.1 Architecture du nœud Potacco

La Figure 4. 1 présente la nouvelle architecture du nœud Potacco, en précisant la localisation et les interactions entre les nouveaux modules définis dans ce chapitre et ceux déjà présents.

Le module de Gestion des Connexions Réseau (GCR) est un composant intégré dans le Gestionnaire Potacco. Un environnement d'exécution (EE), dans lequel les modules applicatifs seront activés, est identifié. Enfin, une entité de déploiement de code est associée et offre une interface externe pour la gestion de vie des modules applicatifs.

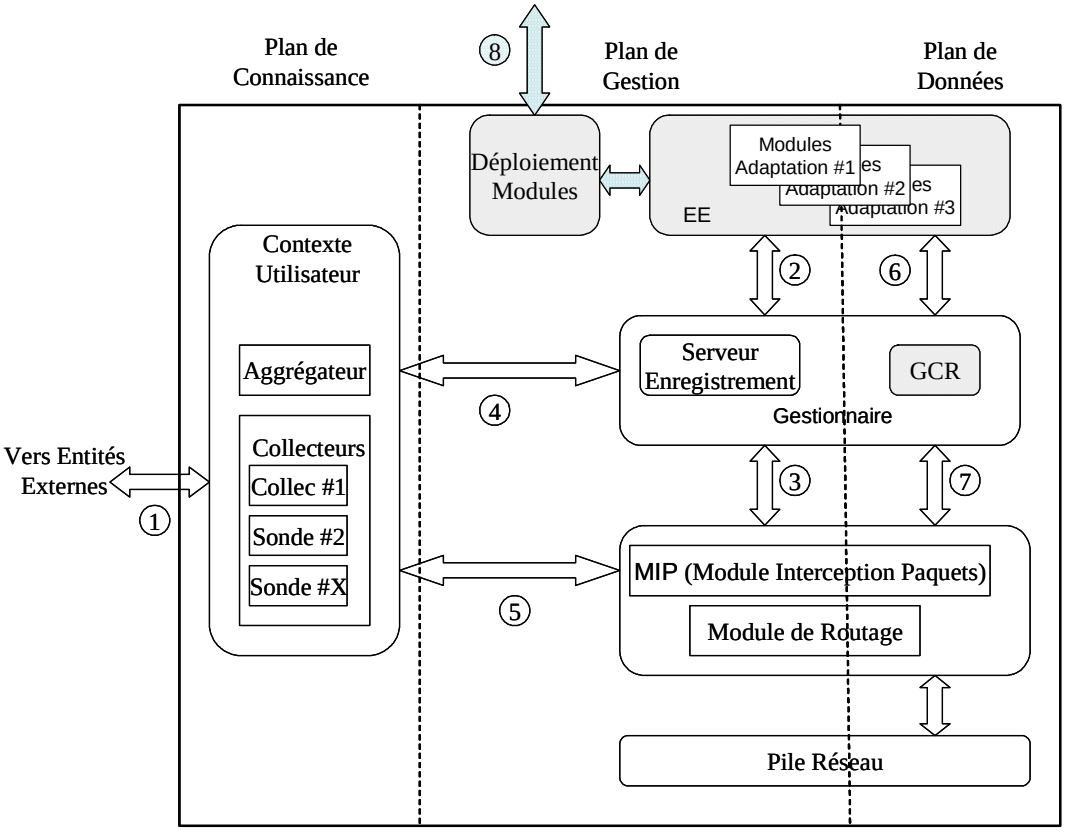

Figure 4. 1: Architecture interne du nœud Potacco

4.1.2 Gestion des Connexions Réseau (GCR)

L'objectif du nœud Potacco est de permettre l'adaptation de données applicatives.
Lorsque les données sont adaptées, cela signifie que la charge utile (payload) des paquets IP est modifiée mais aussi l'en-tête IP, notamment la taille des paquets, la somme de contrôle (checksum) et les éventuels champs de contrôle, doit être adapté en fonction de la modification. Pour éviter à chacun des modules d'adaptation de gérer la cohérence des connexions, un composant du nœud remplit cette fonction: le composant de Gestion des Connexions Réseau (GCR). Ceci permet aux applications déployées sur le nœud de se focaliser sur leur rôle et de ne pas avoir à gérer les connexions réseau et les paramètres associés.
Cette section décrit l'objectif et la spécification du GCR qui doit être opérationnel aussi bien pour les protocoles UDP que TCP.

Si le protocole UDP est utilisé alors, le processus d'adaptation des en-têtes IP est simple puisque seulement la taille du paquet et la somme de contrôle doivent être calculées et modifiées. Ce calcul se fait donc par paquet sans avoir besoin de garder un contexte concernant la connexion.

Par contre lorsque TCP est utilisé, le processus est plus compliqué. En effet, TCP offre une communication de bout en bout, fiable, entre les terminaux d'extrémité. Ceci est assuré par quelques champs dans l'en-tête des paquets TCP tels que le champ "Ack", le champ "Seq" en plus de la taille de paquet et la somme de contrôle. Ces champs sont liés au nombre d'octets transmis et au nombre d'octets reçus (le nombre d'octets reçus par un terminal étant le nombre d'octets envoyés par l'autre). Ces valeurs sont utilisées par les terminaux d'extrémité pour détecter si des paquets ont été perdus ou corrompus pendant la transmission.
Si des données sont modifiées par des nœuds intermédiaires, le nombre d'octets reçus par un terminal sera différent du nombre spécifié par le terminal émetteur dans le champ correspondant. Pour des connexions TCP, il est alors nécessaire de maintenir un contexte des connexions en mémoire et notamment les valeurs correspondantes aux nombres d'octets ajoutés/supprimés (valeur nommée *delta*) pour les paquets précédents pour mettre à jour le paquet courant.
Ainsi même si la charge utile des paquets est modifiée, le terminal ne le détectera pas parce que l'en-tête de protocole sera cohérent. La Figure 4. 2 explique le fonctionnement de ces champs et le rôle du nœud en mode transparent:

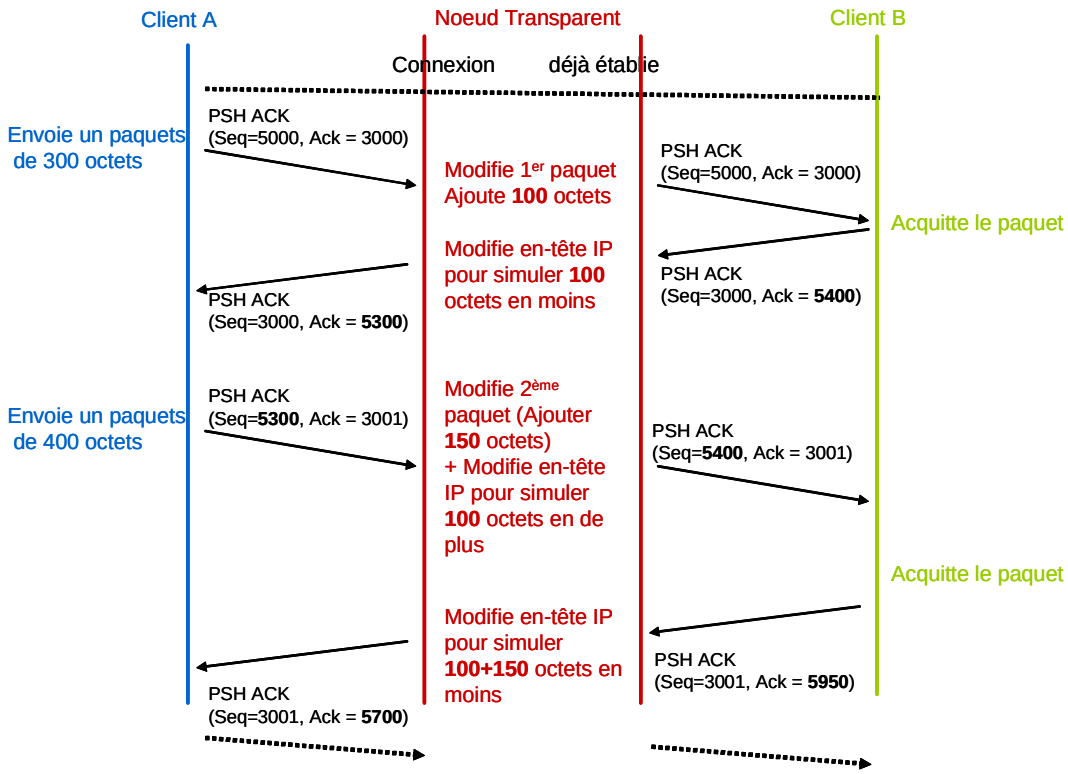

Figure 4. 2: Fonctionnement du Potacco en mode transparent, via le GCR

Pour gérer le contexte des connexions TCP, le GCR gère une liste (1 élément correspond à 1 connexion) contenant pour chaque connexion les informations relatives à la connexion et le nombre d'octets rajoutés ou supprimés dans les paquets (champ *delta*) pour pouvoir adapter les champs Ack et Seq. La structure représentative de la connexion TCP est la suivante :

Connexion = { ipSrc, portSrc, ipDst, port Dst, delta}

Lorsqu'un module applicatif demande la configuration du module MIP pour remonter les paquets le concernant, le gestionnaire Potacco informe le GCR de cette configuration pour que ce dernier puisse définir un nouveau contexte relatif à ces connexions (et initialisé la valeur du champ *delta* à 0).
Ensuite, quand les paquets de cette connexion seront adaptés, la taille des paquets changera et le champ *delta* sera modifié en conséquence. Le GCR doit détecter le nombre d'octets de données dans le paquet initial et calculer celui dans les données renvoyées par les modules applicatifs pour calculer le *delta*.

Ces modifications doivent être réalisées sur les paquets de données qui sont modifiés par les modules applicatifs mais aussi sur les paquets de contrôle (tels que les paquets d'acquittements de type "Ack"…). Les paquets de contrôle sont redirigés au module GCR pour adaptation des en-têtes, mais ces paquets ne doivent pas être transmis aux modules applicatifs puisqu'il n'y a pas de données. Le module GCR a donc aussi le rôle de détecter si le paquet est un paquet contenant des données ou pas et dans ce cas, il doit adapter les en-têtes IP mais n'a pas à recalculer le champ *delta*; celui-ci restant inchangé puisqu'il n'y a pas de données.
La Figure 4. 3 présente l'algorithme interne du GCR.

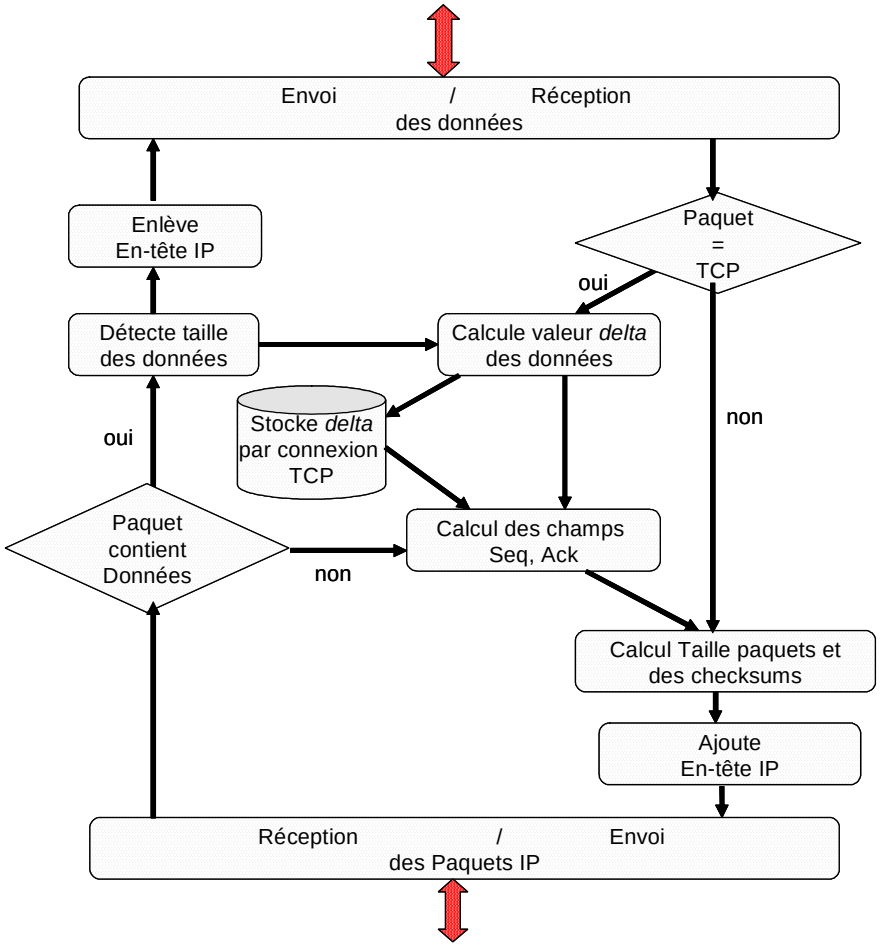

Figure 4. 3: Algorithme de fonctionnement du GCR

Ce composant GCR permet donc au nœud Potacco de réaliser des adaptations de contenus de service, aussi bien sur protocole UDP que sur TCP, sans que les terminaux d'extrémité puissent s'en apercevoir.

4.1.3 Environnement d'exécution

Les modules d'adaptation doivent traiter les données au niveau applicatif. Un environnement d'exécution à l'intérieur du Potacco, dans lequel peuvent fonctionner ces modules de services, est donc défini.
Différents modules doivent pouvoir coexister sur le Potacco et être opérationnels simultanément sans interférer entre eux. Il faut donc que l'environnement d'exécution offre des mécanismes d'isolation entre les modules d'adaptation déployés.
L'objectif étant d'avoir une solution pouvant être implémentée sur divers équipements, de taille et capacité variable, il faut privilégier un environnement à faible empreinte mémoire, pouvant ainsi être instancié sur des équipements à capacité limitée, tels que des passerelles sans-fil/filaire ou des terminaux mobiles. En effet, puisque le monde B3G et même 4G arrivent, il faut prévoir ces terminaux comme une cible de cet environnement (comme c'est le cas au chapitre 5).
L'état de l'art au chapitre 2 a fait ressortir que le framework OSGi présente des caractéristiques intéressantes vis-à-vis de nos requis. Il a donc été décidé d'utiliser OSGi en tant environnement exécution, puisqu'il existe, plutôt que d'en définir un autre.

4.1.4 Entité fonctionnelle de déploiement sécurisé de modules

En fonction de l'utilisation faite de Potacco et de son instanciation sur un nœud fixe ou mobile ou un terminal utilisateur limité, etc., il est possible qu'un module de service ne soit pas présent sur le nœud Potacco à certains moments, alors qu'il est requis pour faire l'adaptation adéquate. De la même manière, des mises à jour de modules applicatifs peuvent être nécessaires. Une entité fonctionnelle de déploiement sécurisé dynamique de modules est ainsi définie dans l'architecture du nœud Potacco.
Ce composant a principalement en charge d'identifier l'entité désirant déployer le code sur le nœud et d'assurer à cette entité que le nœud est bien le bon, et de vérifier l'intégrité du code. L'interface avec l'environnement d'exécution doit aussi être réalisée.
Enfin, ce composant doit permettre une abstraction de l'identification IP en offrant un nommage, adressage unique, toujours valable indépendamment de l'adresse IP du nœud. En effet, en cas de mobilité du nœud (et éventuellement de changement de réseau d'accès), l'adresse IP peut changer.

La Figure 4. 4 présente l'architecture interne de cette entité de déploiement de modules.

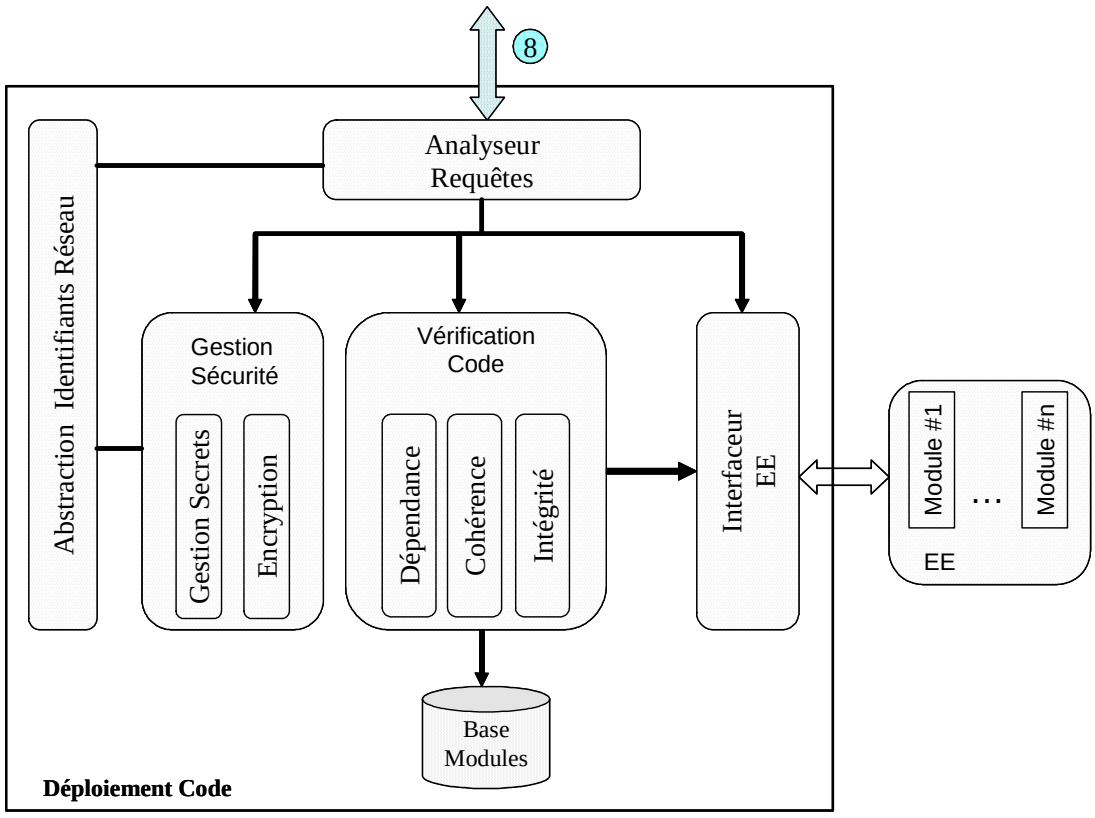

Figure 4. 4: Architecture de l'entité de déploiement sécurisé

L'Analyseur de requêtes a pour rôle de recevoir les requêtes définies selon l'interface 8, qui permettent de gérer la vie des modules applicatifs, et de les rediriger ou d'invoquer les autres composants de cette entité.
Pour assurer la sécurité requise, le fournisseur doit être authentifié. Pour cela, un type *identifier* est défini en tant que paramètre d'entrée des primitives de l'interface 8. Ce type pourrait être une chaine de caractères, une clé cryptographique ou un autre type… Ce type *identifier* est laissé libre au choix des implémenteurs.

L'interface de gestion 8 propose 5 primitives pour gérer les modules applicatifs :

deployCode (identifier providerId, String moduleName, String moduleLocation)
Cette fonction permet de déployer le code du module qui sera identifié avec le nom *moduleName* à partir de l'endroit *moduleLocation* où est stocké le code du module. Le fournisseur du module est identifié par *providerId*.

updateCode (identifier providerId, String moduleName, String moduleLocation)
Cette fonction permet de mettre à jour le code d'un module applicatif déjà présent. Les paramètres sont identiques à ceux du déploiement.

removeCode (identifier providerId, String moduleName)
Cette fonction est appelée par le *providerId*, qui peut être le fournisseur du module lui-même pour enlever un module du nœud Potacco ou l'opérateur du nœud Potacco si ce module n'est plus utilisé.

activateCode (identifier providerId, String moduleName)
Une fois le code déployé, cette fonction permet d'activer le module applicatif *moduleName* qui deviendra opérationnel seulement à partir de ce moment.

desactivateCode (identifier providerId, String moduleName)
Cette fonction désactive le module (stoppe son exécution) mais ne le supprime pas du nœud Potacco.

Le code du module à déployer sur le nœud Potacco doit être vérifié. Le composant *Vérification Code* permet de s'assurer de l'intégrité des données du code, de la dépendance entre les modules (si un module est requis par un autre…), de la composition de modules, et de la cohérence entre modules (si l'on déploie un module ayant un comportement opposé à un autre…), en relation avec une base gérant les modules déployés. De nombreuses études sur le déploiement de code, sur la composition de modules ou en relation ont été publiées [Ben'01] [Dro'03] [Fal'03] [Gal'04] [Car'05]… De ce fait, dans cette thèse, cet aspect ne sera pas abordé mais ces principes pourront être appliqués pour offrir ce rôle.

Par contre, l'accent est mis ici sur la possibilité que ce déploiement soit indépendant de l'adressage des nœuds au niveau IP, assure une authentification entre les deux parties et permette le déploiement sur une connexion sécurisée. Ceci est pris en charge par le composant *Gestion Sécurité* et le composant *Abstraction Identifiant Réseau.*
En effet, les nœuds Potacco fonctionnent dans un environnement IP mais ils peuvent être mobiles, reliés à différents réseaux successivement, il faut donc pouvoir s'abstraire de l'identifiant physique IP qui changera en même temps que le nœud changera de réseau d'accès et avoir un identifiant de nœud unique quelque soit son adresse IP. La communication entre le fournisseur de code et le nœud devra se baser sur un tel identifiant unique. Cette notion d'identifiant de nœud est un concept qui commence à être pris en compte et le groupe de travail HIP (Host Identity Protocol) [Mod'06] à l'IETF a pour objectif la définition d'une telle solution où HIP serait une nouvelle couche au dessus d'IP. Cela entraine qu'un mapping devra être possible entre les identifiants du nœud et son adresse IP courante.

Pendant la phase de déploiement, il faut pouvoir authentifier les parties. En effet, le fournisseur de code doit être authentifié pour être sûr qu'il a les droits de déployer son code dans de tels nœuds. Mais le nœud lui-même doit également être authentifié pour être sûr que le code ne soit pas déployé dans un autre nœud, qui pourrait être un nœud malveillant, visant à perturber la bonne fourniture du service ou ayant pour but de récupérer le code afin de l'analyser et éventuellement le copier…
Le principe de base de l'authentification définie pour le Potacco repose sur des mécanismes semblables aux clefs privées / clés publiques, à savoir le HIT (Host Identity Tag) tel que défini dans les spécifications de HIP.
Les HIT sont des identifiants obtenus à partir des valeurs de hachages des clés publiques. Pour obtenir le HIT, c'est la fonction SHA-1 [SHA1'93] [SHA1'01] qui est utilisé. La fonction de hachage permet d'obtenir des HIT de longueur fixe, facilement

insérable dans les en-têtes des paquets IP. Ensuite, les entités en communication s'échangent leurs HIT au lieu de s'échanger leurs clés publiques.
Cela permet aux entités de toujours communiquer uniquement via leur HIT et non pas leur adresse IP. Ainsi en cas de mobilité, de changement de réseau ou de changement d'adresse IP pour une raison quelconque, les applications n'ont pas à modifier leur comportement et leurs connexions. L'adresse IP est transparente aux applications, c'est la couche du protocole HIP qui est en charge de l'association entre adresse HIT et adresse IP.
Pour initier une communication avec une autre partie, une entité connaît donc son HIT mais pas son adresse IP. Il faut donc une autre entité de confiance, un serveur qui puisse lui fournir cette information. Comme il faut s'assurer de la confiance des entités en relation, il a été choisi d'utiliser un serveur DNSSEC, qui est un serveur DNS pouvant servir d'autorité de confiance pour générer des certificats ou des clés publiques. Un serveur DNSSEC ayant en charge plutôt des informations statiques, il a été décidé d'utiliser un serveur Rendez-Vous (RVS) [Lag'06], qui a lui uniquement en charge de maintenir les associations entre adresse HIT et adresse IP. Ce serveur RVS est lui-même interrogé par le serveur DNSSEC lorsqu'une demande d'association est reçue par le serveur DNSSEC.
En cas de changement d'adresse IP, un message est émis pour mettre à jour le serveur RVS qui permet de faire la correspondance entre les 2 adresses. Dans ce cas d'architecture, seul le serveur RVS supporte les mises à jour d'association des adresses HIT.
Si le changement a lieu pendant que le nœud est en communication avec une autre entité, alors un message est envoyé à l'entité destination pour lui signifier le changement d'adresse et lui permettre de refaire la correspondance entre l'adresse HIT et l'adresse IP, sans avoir à interroger le serveur DNSSEC.

Ce mécanisme d'authentification à base de HIT permet de s'assurer des entités en relation, mais a un coût non négligeable lorsqu'il s'agit de chiffrer et déchiffrer tous les messages des 2 côtés. Il a été ainsi conçu un mécanisme de challenge/réponse pour échanger un « secret » pendant la phase d'authentification. Ce mécanisme repose sur l'algorithme de Diffie-Helmann [DH'76]. Le fournisseur de service et le module de déploiement de code dans le nœud Potacco chiffrent tous les deux la demande et la réponse de secret avec leurs propres HIT. Ce secret est ensuite employé pour les messages suivants entre le fournisseur de code et le module de déploiement du Potacco, facilitant le chiffrement et le déchiffrement des données.

Le logiciel déployé ne doit pas être corrompu ou modifié pendant la phase de déploiement. Pour éviter ceci, le code est transmis en utilisant une connexion sécurisée telle qu'IPSEC. En effet, HIP proposer d'utiliser IPSEC en natif, ce qui permet de s'assurer que les données ne sont pas altérées, interceptées ou modifiées pendant leur transfert.
L'intégrité des données est calculée en faisant une somme de contrôle sur les paquets de données (avec SHA-1) et ensuite la somme de contrôle est signée avec le HIT de l'émetteur. Ainsi, cela permet d'authentifier celui qui envoie le code et de vérifier l'intégrité du code.

4.2 Evaluation

La nouvelle architecture du nœud Potacco a été implémentée et évaluée avec un démonstrateur consistant à utiliser le nœud Potacco pour insérer, de manière transparente, les informations relatives au contexte des utilisateurs dans le flux HTTP de ces derniers lorsqu'ils sont en train de naviguer sur le Web.
L'idée à la base de ce scénario est le cas où les terminaux utilisateurs ne sont pas capables de fournir des informations de contexte, parce que les terminaux n'ont pas le protocole qu'il faut, ou bien parce qu'ils ont des capacités limitées... Ainsi l'insertion du contexte par le nœud intermédiaire permet de se substituer aux navigateurs web des utilisateurs.
Dans ce démonstrateur, le contexte peut être inséré selon le format CC/PP (comme présenté en section 1.2.1) ou selon un format propriétaire, illustrant ainsi le cas où 2 fournisseurs de service ont chacun leur format de contexte (CC/PP pour l'un, propriétaire pour l'autre). Cette possibilité illustre la capacité du nœud Potacco d'héberger des modules applicatifs différents, qui peuvent être déployés dynamiquement, en fonction des besoins dans l'environnement d'exécution.
Le nœud Potacco a été intégré dans une architecture de réseau ADSL réelle en mode de fonctionnement transparent pour étudier son intégration potentielle dans un tel réseau et évaluer son comportement.

4.2.1 Scénario

Cette section décrit le scénario qui inclut les terminaux utilisateurs, un nœud Potacco pour insérer le contexte, un serveur de contenu et un serveur de contexte, incluant une base de données: Figure 4. 5.

Au préalable à toute insertion de contexte, l'utilisateur doit compléter les informations concernant les caractéristiques de son terminal ainsi que ses propres préférences via un serveur Web. Ces informations sont enregistrées dans une base de données. Une fois ces informations renseignées, elles seront toujours valides, l'utilisateur n'est pas obligé de les redéfinir à chaque connexion. Par contre, s'il le souhaite, il peut les modifier quand il le souhaite pendant sa navigation Internet.

Lors de la première requête passant par le nœud Potacco, ce dernier interroge la base de données pour récupérer les informations concernant l'utilisateur et vérifier si une insertion de ces informations est autorisée ou non. En effet, on peut supposer que la fourniture de contenu est un service à valeur ajoutée, payant ou nécessitant un abonnement. Ainsi, il faut vérifier pour l'utilisateur concerné et pour le fournisseur de service concerné, si une adaptation de contenu est autorisée ou pas :

- Si aucune insertion n'est nécessaire, alors le gestionnaire Potacco configure le MIP pour qu'il transfère tous les paquets de ce flux directement sans remonter à l'application (lors de l'intégration du Potacco dans le réseau ADSL, une amélioration est apportée en configurant directement le BAS pour transférer lui-même les paquets du flux concerné).
- Si une insertion est nécessaire, le service applicatif d'insertion de contexte stocke en cache les informations de contexte et les insèrera dans les

requêtes de l'utilisateur au format souhaité. Le Gestionnaire Potacco configure le MIP pour lui indiquer de remonter les requêtes HTTP en provenance de cet utilisateur et à destination de ce fournisseur de service. De plus, le module GCR du Potacco est configuré pour gérer ce flux TCP (champs «Seq» et «Ack» et les codes vérificateurs). La requête HTTP modifiée, enrichie du contexte au format CC/PP ou propriétaire est ainsi routé vers le fournisseur de service qui adaptera le contenu Web avant de la renvoyer à l'utilisateur.

Pendant la session, le module d'insertion sur le Potacco est en attente de notification de la base de données et peut ainsi être informé si le contexte est modifié en cours de session. Ceci permet d'insérer les informations à jour concernant l'utilisateur.

4.2.2 Implémentation du nœud Potacco et des modules de contexte

La Figure 4. 5 présente l'architecture du démonstrateur, avec un détail sur les composants du Potacco.

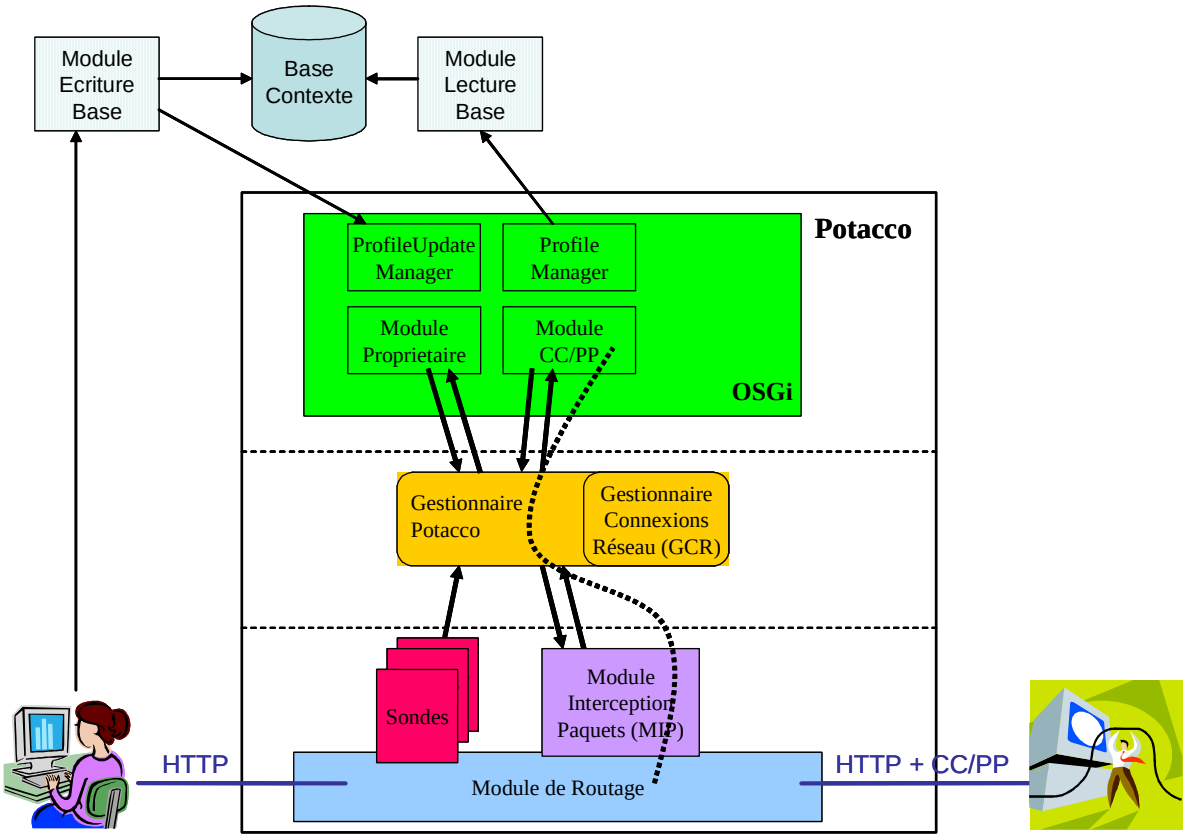

Figure 4. 5: Architecture du démonstrateur - insertion de contexte

Pour gérer le contexte des utilisateurs, 3 composants, externes au Potacco, sont mis en œuvre :

- Une **base de données de contexte** qui stocke les informations sur les utilisateurs et les capacités du terminal. Dans ce démonstrateur, la base de données est basée sur "mySql".
- Un **module d'écriture dans la base** de données qui permet à l'utilisateur de spécifier les informations via un serveur Web écrit en PHP et avec des scripts Java. Le serveur Web analyse les données renseignées et envoie les requêtes SQL adéquates vers la base de données pour remplir les tables pour cet utilisateur.
- Un **module de lecture** qui reçoit les requêtes du nœud Potacco de demande d'informations de contexte utilisateur et les récupère depuis la base de données en faisant une requête SQL.

Le nœud Potacco implémenté se base sur celui défini dans le chapitre précédent.

Cependant, une amélioration a été faite pour améliorer les performances. En effet, dans la première version, le traitement était séquentiel, toutes les fonctions étant lancées dans le même processus. Dans cette nouvelle version, une architecture multithreadée, basée sur un pool de thread alloué dynamiquement en fonction de l'arrivée des paquets, permet de paralléliser les traitements et d'améliorer les performances. Les tests unitaires ont montré que le temps de traitement induit est de l'ordre de 5 ms avec la version parallélisée au lieu d'environ 30 ms.

Concernant les nouveaux modules, le GCR est développé en *langage C* et lié au Gestionnaire Potacco en tant que bibliothèque externe, par l'intermédiaire du mécanisme de *JNI (Java Native Interface*).
L'environnement d'exécution se base sur un framework OSGi. Dans une première version, l'implémentation "*Jeffree*", réalisée par des personnes issues de France Télécom, et mise en code OpenSource disponible dans le consortium ObjectWeb, avait été utilisée. Ensuite, Jeffree n'ayant pas évolué et étant resté à une implémentation de la version 2 d'OSGi, le choix s'est centré sur "*Oscar*", aussi un logiciel Open-Source disponible sur ObjectWeb, qui lui est largement utilisé et régulièrement mis à jour en fonction des avancés du consortium OSGi.
L'implémentation du déploiement sécurisé de code n'a pas été complètement réalisée et surtout pas évaluée en termes de performance. Cependant, [Khu'07] introduit une implémentation de HIP sur tablette Nokia et présente des résultats de performance de HIP sur PC et tablette. Sur tablette, les résultats sont moins bons que PC, mais avec la loi de Moore et les progrès technologiques, on peut penser que les performances vont s'améliorer.

Concernant les modules applicatifs, 4 modules (bundles OSGi) ont été développés :

- **ProfileManager** : pour assurer la communication avec la base de données pour récupérer le contexte utilisateur et pour «cacher» ces informations.

- **ProfileUpdateManager** : pour recevoir les mises à jour de contexte depuis le module d'écriture de la base de données.

- ModuleCCPP : Pour insérer le contexte de l'utilisateur au format CC/PP dans les requêtes HTTP de l'utilisateur

Voici un exemple de requête HTTP, complété par le contexte utilisateur en format CC/PP.

GET /manual/mod/mpm_common.html HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Referer: http://10.192.56.33:8080/welcome.html
Accept-Language: fr
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
Host: 10.192.56.33:8080
Opt: http://www.w3.org/1999/06/24-CCPPexchange;ns=99
99-Profile: "1-hpkMJoxC65VbZ5I9OiZApQ=="
99-Profile-Diff-1:<?xml version="1.0" encoding= 'ISO-8859-1'?>
<rdf:RDF xmlns:rdf=http://www.w3.org/TR/1999/PR-rdf-syntax-19990105#
xmlns:prf="http://www.w3.org/TR/CCPP-ra/#"
xmlns:loc="http://xmlns.rd.francetelecom.fr/2003/moveit/loc">
<rdf:Description rdf:ID="Bertrand">
<prf:component>
<rdf:Description rdf:ID="HardwarePlatform">
<rdf:type rdf:resource="http://www.wapforum.org/profiles/UAPROF/ccppschema-20001430#HardwarePlatform"/>
<prf:Device>PC</prf:Device>
<prf:Brand>Compaq</prf:Brand>
<prf:ScreenSize>1280x1024</prf:ScreenSize>
<prf:ScreenResolution>256</prf:ScreenResolution>
</prf:component>
…..
<prf:component>
<rdf:Description rdf:ID="UserPreferences">
<prf:Language>French</prf:Language>
<prf:Audio>No</prf:Audio>
<prf:Cookies>Yes</prf:Cookies>
</prf:component>
</rdf:Description>
</rdf:RDF>
Connection: Keep-Alive

- ModulePropriétaire : Pour insérer le contexte de l'utilisateur selon un format propriétaire dans les requêtes HTTP de l'utilisateur

Voici un exemple de format propriétaire pour insérer les caractéristiques du terminal utilisateur dans les requêtes HTTP. Dans cet exemple, on suppose que le fournisseur n'a besoin que de ces trois informations (type de terminal, taille de l'écran et si l'utilisateur veut du son ou pas) pour fournir un contenu adapté.

GET /manual/mod/mpm_common.html HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Referer: http://10.192.56.33:8080 /welcome.html
Accept-Language: fr
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
Host: 10.192.56.33:8080
1-Device: PC

1-ScreenSize: 1280x1024
1-Sound: nosound
Connection: Keep-Alive

Il existe des dépendances entre ces modules. En effet, le moduleCCPP et le module propriétaire ont besoin du module "ProfileManager" pour récupérer les informations de contexte de l'utilisateur. Pour spécifier ces dépendances, le fichier "manifest" de OSGI est utilisé pour chaque module. Par exemple, voici la partie du fichier qui définit que le module "ProfileManager" exporte ses interfaces que les autres importeront :

Bundle-Name: ProfileManager
…
…
Export-Package: contentadapt.osgi.ProfileMgt

4.2.3 Introduction du nœud Potacco dans une architecture ADSL

Dans cette évaluation, l'objectif est de voir comment le nœud Potacco pourrait être intégré dans une architecture réseau de type ADSL et de connaitre ses performances. L'horizon pourrait être d'avoir des équipements ADSL de type DSLAM ou BAS évolués qui pourraient intégrer les fonctionnalités présentes dans le Potacco.

Dans ces tests, le Potacco a été intégré dans le réseau ADSL de test mis en place dans les labos de France Télécom R&D à Lannion.
Dans le cas d'une architecture ADSL déployée, les nœuds Potacco devraient être localisés dans les POP ("Point Of Presence") du réseau de l'opérateur, juste derrière les BAS ("Broadband Access Server"). Cela permet de déployer des fonctions de niveau applicatif dans le réseau sans trop surcharger les BAS existants. De plus, cela permet d'instancier rapidement des fonctions avancées sur le nœud Potacco en attendant que la standardisation évolue et que les BAS supportent ces nouvelles fonctions. L'architecture définie peut être schématisée ainsi (Figure 4. 6) :

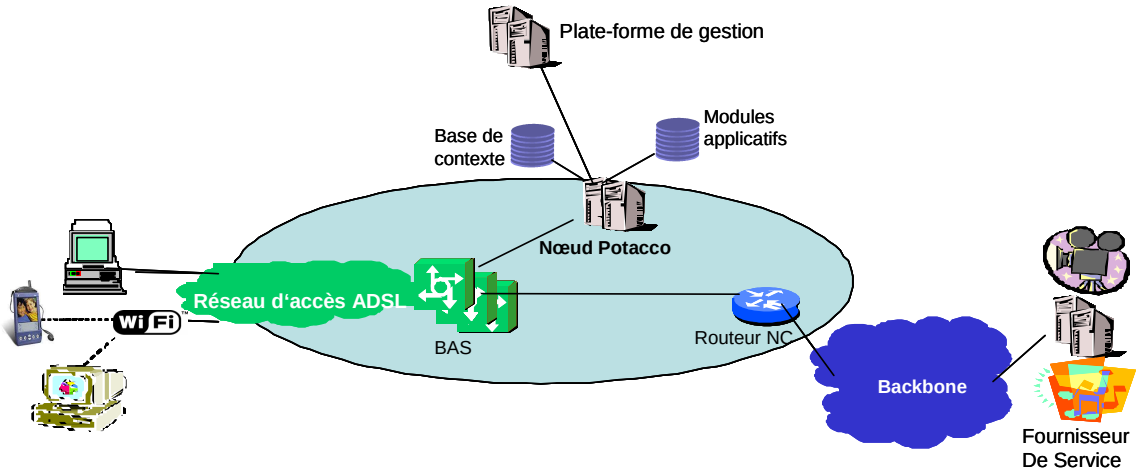

Figure 4. 6: Intégration du Potacco dans un réseau ADSL

Le BAS est positionné en coupure et assure la liaison ATM/IP. La figure montre les deux interfaces distinctes du BAS: le lien ATM entre le terminal et le BAS (qui réalise la terminaison PPP et qui offre une abstraction de la couche ATM entre le modem et le BAS) et le lien IP entre le BAS et le serveur ou le routeur d'accès du fournisseur d'accès Internet. L'intégration du nœud Potacco sur l'interface IP du BAS permet de bénéficier du rôle en coupure du BAS.
Dans cette architecture, 2 nœuds Potacco sont positionnés, l'un étant en secours de l'autre ou les deux en partage de charge. En effet, puisque tous les flux utilisateurs passent par le POP, il faut fournir un premier niveau de tolérance aux pannes avec une supervision de l'état des nœuds Potacco, et une reconfiguration dynamique du BAS en cas de panne ou de surcharge d'un nœud Potacco.

L'architecture définie s'appuie sur le rôle des BAS en coupure et sur leurs capacités avancées pour appliquer des politiques de routage flux par flux. Le déploiement d'un service se traduit ainsi par la commande d'un BAS pour appliquer une politique de filtrage ou de re-routage vers le nœud Potacco, et par l'activation du service sur le nœud Potacco.
La configuration des BAS se fait via l'utilisation des *access-lists* sur le BAS. Cela permet de spécifier à des flux arrivants sur une interface d'être transmis sur une interface sortante précise du BAS.
Dans cette évaluation, il y a donc trois interfaces, une allant vers le routeur NC (chemin normal), une allant vers le nœud Potacco et celle arrivants des clients ADSL. Le choix de routage des paquets se fait en fonction du numéro de port, soit 8080 ou 3128; les 2 ports HTTP utilisés dans ce démonstrateur (ce port est le port destination en émission et le port source en retour).
Les access-lists définies sont de la forme suivante, où 10.194.117.194 est l'adresse du nœud Potacco et 10.194.117.146 l'adresse IP de l'interface du routeur NC.

```
ip access-list interface-adsl
        redirect interface_potacco  10.194.117.194 tcp any any eq 8080
        redirect interface_ potacco 10.194.117.194 tcp any any eq 3128
        permit   ip any any
ip access-list interface_potacco
        redirect Interface_Internet 10.194.117.146 tcp any any eq 8080
        redirect Interface_Internet 10.194.117.146 tcp any any eq 3128
        permit   ip any any
 ip access-list retour-Interface_Internet
        redirect interface_potacco 10.194.117.194 tcp any eq 8080 any
        redirect interface_potacco 10.194.117.194 tcp any eq 3128 any
        permit   ip any any
```

La commande *redirect* permet de rediriger les paquets concernés par les paramètres définis et la commande *permit* de laisser les paquets transiter normalement. L'ordre de configuration est important puisque l'action est réalisée lorsque la première condition est vérifiée. Il faut donc spécifier la condition " *permit ip any any"* qui laisse passer tous les paquets en dernier.

La configuration interne du BAS, et notamment les redirections par les access-lists, peut être schématisée de la manière suivante (Figure 4. 7) :

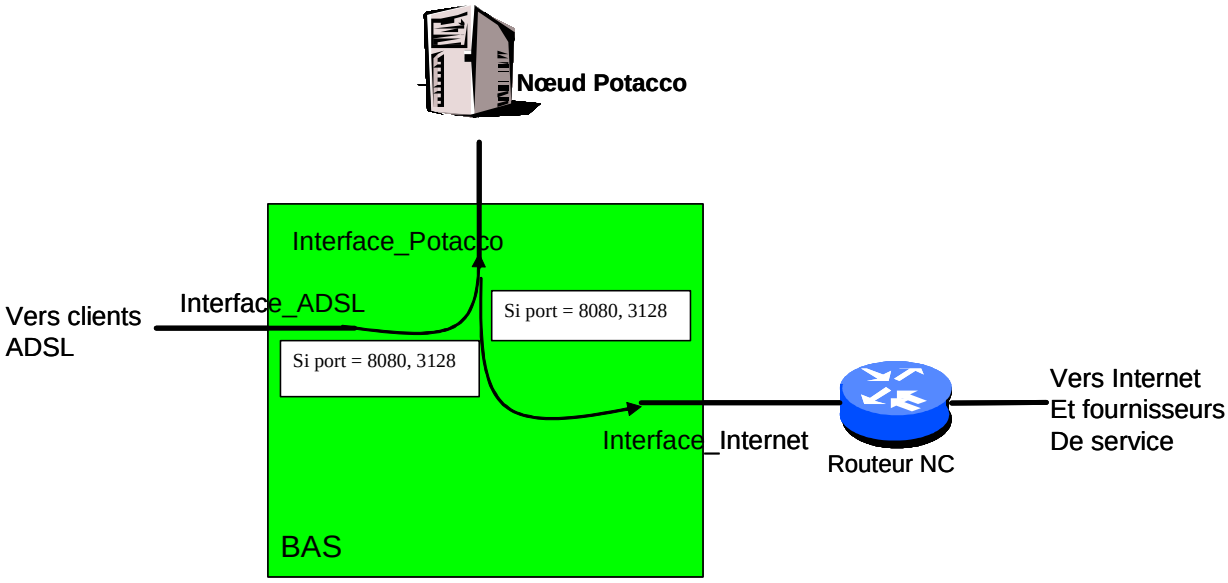

Figure 4. 7: Configuration interne du BAS

La fonctionnalité de configuration du BAS par *access-list* est intéressante mais limitée puisqu'elle n'est pas dynamique (c'est défini manuellement par l'utilisateur en ligne de commande) et ne concerne que les ports HTTP définis.

Le scénario a illustré le cas où des utilisateurs n'avaient pas souscrit au service et pour lesquels il ne fallait pas insérer le contexte. Dans ce cas, le module MIP était configuré pour retransmettre directement les paquets. Cependant, au vu de la Figure 4. 6, et de l'intégration du Potacco dans l'architecture ADSL, il est plus judicieux de transférer directement ces paquets au niveau BAS plutôt que de les transférer au Potacco, qui n'en fera rien, si ce n'est les retransmettre. Ainsi, dans un tel cas de figure, lorsque le nœud Potacco aura récupéré le contexte de l'utilisateur et détecté qu'une insertion n'est pas nécessaire, il configurera dynamiquement le BAS en définissant une *access-list* (utilisant la fonction *permit*) en fonction du couple adresse source/adresse destination pour l'utilisateur concerné et le fournisseur de service associé.
Le BAS utilisé dans le démonstrateur étant configurable par une commande Telnet, un module de configuration du BAS, écrit en langage Java, permettant de gérer une session Telnet, a été rajouté sur le nœud Potacco pour prendre cette fonctionnalité en charge.

4.2.3.1 Tests & Résultats

Pour ces tests, un simulateur réseau a été utilisé: le NetworkTester, de la société Agilent Technologies.

Cet équipement permet de simuler du trafic IP suivant plusieurs protocoles dont HTTP. L'équipement est une machine possédant plusieurs *blades* (cartes). Dans les tests, une carte a été utilisée pour simuler les clients et une autre pour simuler le serveur. Le système d'exploitation Linux est installé sur ces 2 cartes pour exécuter les scripts de tests.
Le logiciel, fourni avec le simulateur, permettant de définir les configurations et les scripts de tests est *NetPressure*.

La configuration matérielle du nœud Potacco est la suivante :

PC Compaq Evo N610c, Pentium 4 m, 520 Mo RAM
Linux Redhat 9.2 et un Kernel 2.4.20

Dans ces tests, l'objectif était de mesurer le temps induit par le nœud Potacco, connecté à un BAS dans un mode de fonctionnement représentatif de la réalité. C'est-à-dire, que le simulateur n'a pas été configuré pour envoyer le plus possible de requêtes, mais plutôt pour envoyer des requêtes selon une fréquence, qui pourrait être celle d'un utilisateur lambda. De la même manière, les fichiers, correspondants aux pages HTML, retournés par le simulateur de serveur ont deux tailles différentes : 16 kbit et 64 kbit pour représenter des types de pages réelles.
Ainsi, 5 comportements d'utilisateurs ont été définis :

- Comportement 1 : Emission d'une requête « HTTP Get » toutes les minutes, la session dure 10 minutes, soit 10 paquets envoyés en tout. La page demandée est *16.html*.
- Comportement 2 : Emission d'une requête « HTTP Get » toutes les 2 minutes, la session dure 10 minutes, soit 5 paquets envoyés en tout. La page demandée est *64.html*.
- Comportement 3 : Emission d'une requête « HTTP Get » toutes les 20 secondes, la session dure 10 minutes, soit 30 paquets envoyés en tout. La page demandée est *64.html*
- Comportement 4 : Emission d'une requête « HTTP Get » toutes les 40 Secondes, la session dure 10 minutes, soit 15 paquets envoyés en tout. La page demandée est *16.html*
- Comportement 5 : Emission d'une requête « HTTP Get » toutes les 10 Secondes, la session dure 10 minutes, soit 60 paquets envoyés en tout. La page demandée est *16.html*

Dans ces configurations, les tests durent 10 minutes.

Les Figure 4. 8 et Figure 4. 9 présentent les temps de réponse moyens pour les tests d'utilisateurs ayant le comportement 3; la première quand les utilisateurs ont souscrit au service et que l'insertion du contexte est réalisée par le nœud Potacco, la deuxième quand aucune insertion n'est requise.

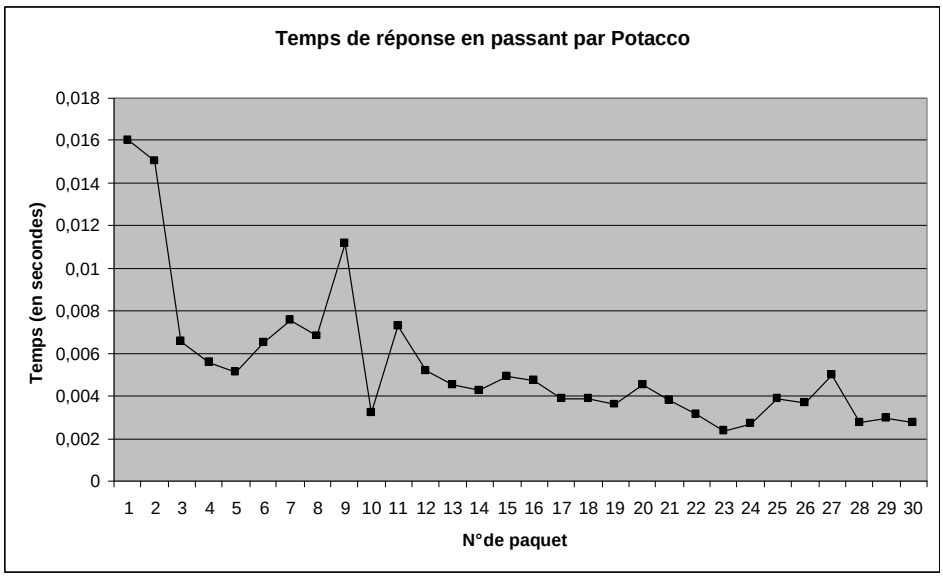

Figure 4. 8: Temps de réponse avec insertion par Potacco

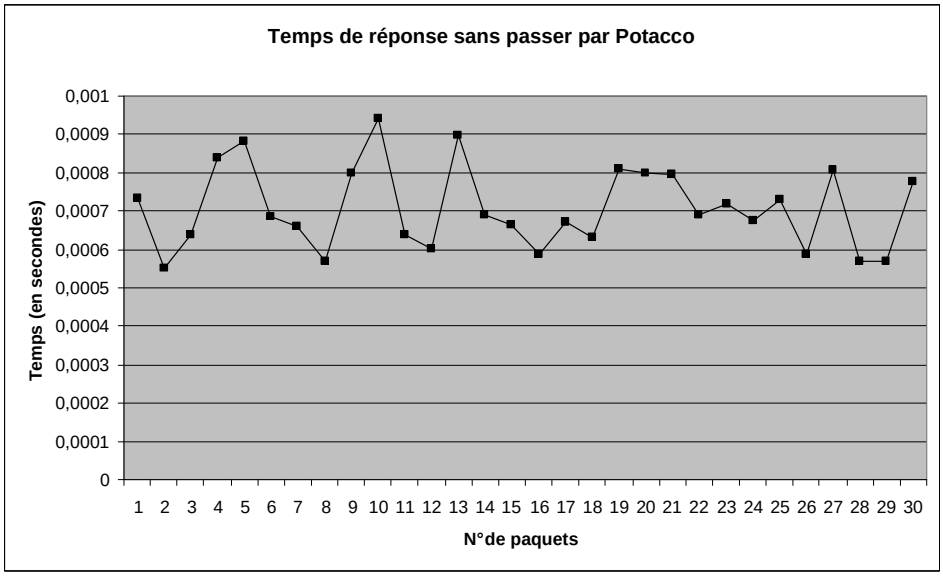

Figure 4. 9: Temps de réponse sans insertion

On remarque ainsi que le temps total (Aller/Retour) de la requête / réponse avec insertion du contexte est de l'ordre 5 à 6 ms et qu'il est inférieur à 1 ms quand il n'y a pas d'insertion. Le temps généré uniquement par l'insertion du contexte par le nœud Potacco est donc négligeable, de l'ordre de 4 à 5 ms.

Les autres tests réalisés présentant des résultats similaires, il a été décidé de ne montrer ici que les résultats correspondant au comportement 3.

Pendant les tests, la mémoire utilisée par le nœud Potacco a été mesurée (Figure 4. 10). On remarque qu'elle augmente au début, le temps de charger les modules et le contexte puis reste quasiment constante pendant la durée du test. Le nœud Potacco gère donc parfaitement les requêtes successives.

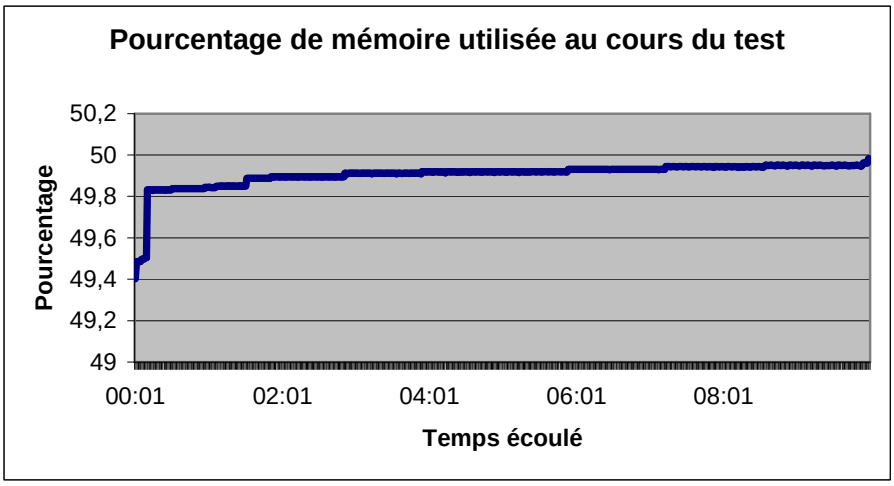

Figure 4. 10: Mémoire Potacco utilisée

Les tests de performance du Potacco montrent donc des résultats satisfaisants. Ceci est encore plus vrai en comparant ces valeurs avec le délai moyen de transport des paquets dans un réseau ADSL (environ 60 ms) et qui est encore moindre que le temps de traitement des requêtes HTTP et des adaptations par les serveurs [Yasu'01]. Ce délai est donc tout à fait acceptable et presque imperceptible pour l'utilisateur en train de naviguer sur le Web.

4.2.4 Conclusion

Ce scénario illustre l'utilisation du nœud Potacco en mode transparent pour insérer les informations de contexte des utilisateurs dans les requêtes HTTP.
Les tests de performance réalisés ont démontré que le délai généré par le nœud Potacco est très faible et imperceptible par l'utilisateur.
De plus, l'intégration du nœud Potacco dans un réseau (comme l'exemple ADSL) est tout à fait réalisable et le traitement applicatif associé n'ajoute pas de délai important.
Puisqu'il est situé dans le réseau, le Potacco pourrait rajouter dans le contexte des informations dynamiques relatives au réseau comme la bande passante réelle, le nombre de paquets perdus...
La possibilité de déployer dynamiquement les modules applicatifs dans le Potacco permet de prendre en charge n'importe quel format et même des formats propriétaires et ainsi de fournir les informations de contexte dans le format souhaité par les fournisseurs de service. Ceci permet d'installer et de supprimer dynamiquement les modules qui ne sont plus utilisés.
En guise de conclusion, cette solution de gestion transparente des connexions TCP par un nœud Potacco est très satisfaisante et peut être utilisée et généralisée à d'autres services en coupure, tels que l'insertion de bandeaux publicitaires, un mécanisme de contrôle parental, un pare-feu configurable...

Chapitre 5

Nœud Potacco en tant que Nœud de réseaux overlay de services

Dans ce chapitre, une utilisation et une adaptation du nœud Potacco dans une architecture réseau différente sont présentées. Contrairement au cas précédent où le nœud est utilisé en tant qu'élément dans le réseau physique, ici, le nœud Potacco est un nœud d'un réseau overlay (réseau de recouvrement au dessus des réseaux physiques) [Mat'07]. Cette adaptation est réalisée en ayant à l'esprit l'évolution des terminaux, qui pourraient offrir de telles fonctionnalités et l'évolution des réseaux, avec notamment le développement d'architectures de réseaux overlays pour la fourniture de services avancés et la virtualisation des réseaux qui est un sujet d'actualité dans le monde de la recherche depuis un peu plus d'un an. La mise en œuvre du Potacco sur un environnement de nœud physique virtualisé n'étant pas encore possible, elle est faite dans le réseau overlay pour avoir une idée de ce qu'il pourrait être dans un tel réseau.
Les nœuds overlays sont des nœuds physiques et peuvent être aussi bien des terminaux utilisateurs que des nœuds du réseau physique. Dans ce cas, la différentiation entre nœuds du réseau et nœuds terminaux utilisateurs n'est plus aussi prononcée qu'avant (les nœuds des utilisateurs devenant eux-mêmes nœuds du réseau et non plus uniquement points finaux). Ceci a déjà été plus ou moins introduit avec les réseaux ad hoc où les nœuds utilisateurs sont utilisés pour router le trafic IP entre nœuds terminaux, et donc jouant le rôle d'un routeur. Avec les réseaux overlays, c'est aussi une notion présente puisque la couche overlay introduit son propre algorithme/mécanisme de routage en passant par des nœuds overlays intermédiaires.
Dans ce chapitre, on retrouve cette notion de nœud intermédiaire, mais à un rôle plus applicatif, puisque chaque nœud peut héberger un composant de service. Le routage des flux des utilisateurs amène à la composition d'un service global fourni à l'utilisateur, réalisé par l'enchainement des traitements applicatifs élémentaires réalisés par les nœuds intermédiaires.
Pour se placer dans ce cas, le nœud Potacco a besoin de certaines adaptations. Typiquement, le mode de fonctionnement en mode transparent n'est plus une préoccupation puisque le routage overlay implique une connaissance des nœuds. Par contre, cela induit aussi la nécessité d'avoir un mécanisme de routage overlay. De plus, la mobilité des nœuds est un argument à prendre en considération et notamment par le fait que le terminal peut changer d'adresse IP; l'identification du nom ne se fait plus au niveau de l'adressage IP mais à un niveau supérieur, permettant de s'abstraire des modifications d'adresses IP du nœud.
Dans ce chapitre, une présentation de ce qu'est un réseau overlay de services et notamment dans le cadre des réseaux ambiants est introduite avant l'adaptation du

nœud Potacco pour fonctionner en nœud overlay. Un cas d'usage de ce réseau overlay pour un service d'IPTV personnalisé est présenté à la fin [Mathie'07].

5.1 Réseaux overlays de services pour les réseaux ambiants

5.1.1 Présentation des réseaux ambiants

Le concept des réseaux ambiants est apparu récemment (2002-2003) [Nie'04] [Abr'05] et peut se résumer comme une architecture ouverte d'interconnexion de réseaux, utilisant des infrastructures différentes, qui se composent dynamiquement permettant la mise en relation dynamique et automatique des équipements relatifs aux utilisateurs. Ainsi pour un utilisateur isolé, possédant un assistant personnel (PDA), un téléphone mobile, un casque bluetooth et un ordinateur portable, un réseau ambiant intégrant ces quatre équipements sera créé. De la même manière, un réseau ambiant peut être créé dans un bus ou dans une gare, reliant les équipements de la gare et offrant éventuellement une interface vers un réseau extérieur (par exemple réseau Internet) et par ce biais un réseau ambiant élargi. Lorsque le premier utilisateur s'approchera de la gare, les 2 réseaux ambiants se détecteront et pourront se composer, permettant ainsi la communication des équipements entre les réseaux ambiants. On voit par ce simple exemple qu'un réseau ambiant peut être très petit, de type réseau personnel PAN, mais aussi à très grande échelle, si le propriétaire du réseau ambiant n'est plus un individu mais une ville ou un opérateur de réseau ou un fournisseur d'accès Internet ou après composition de réseaux ambiants plus petits (la composition de deux réseaux ambiants aboutissant à la création d'un réseau ambiant plus grand).
L'objectif des réseaux ambiants est donc d'assurer la coopération entre réseaux hétérogènes et de permettre la composition/décomposition des réseaux. Une des attentes sous-jacentes est aussi de faciliter la convergence fixe-mobile tant attendue pour les services.
Le projet européen "Ambient Networks" [AN] vise à étudier les réseaux ambiants sous diverses perspectives : interfaces réseau, composition des réseaux, gestion du contexte, gestion de la mobilité, gestion du réseau, fourniture de services. J'ai contribué dans ce projet dans le sous-projet relatif à la fourniture de services adaptés au contexte utilisateur. Dans la suite de ce chapitre, ce sont les architectures relatives à ce sous-projet qui sont présentées.

Les caractéristiques principales des réseaux ambiants sont :

- Basé sur une solution de réseaux IP
- Composition/Décomposition : Les réseaux ambiants peuvent se composer/décomposer automatiquement après détection. Des accords de composition doivent bien évidemment être définis et échangés pour définir les rôles de chacun après la composition (rôle partagé, un maitre, réseaux fusionnés ou simple communication…). Cette composition est essentielle pour permettre la fourniture de services avancés, n'importe où, dans n'importe quelle situation.

- Gestion transparente de la mobilité : aussi bien pour la mobilité des utilisateurs dans un réseau ambiant que des réseaux ambiants eux-mêmes (notamment par la composition/décomposition)
- Les infrastructures physiques peuvent être différentes
- Gestion autonome et automatique par des mécanismes de contrôle, configuration et d'échanges d'information.
- Les entités d'administration peuvent être différentes puisque les réseaux ambiants peuvent englober différents domaines.
- Gestion du contexte, aussi bien au niveau réseau ambiant qu'au niveau utilisateur: cela permet aux applications de pouvoir adapter le service fourni au contexte de l'utilisateur à chaque instant.
- Des interfaces définies permettant aux services de bénéficier pleinement de cette solution et des avantages inhérents (gestion transparente de la mobilité, sécurité, découverte du voisinage et éventuellement composition…) : déploiement de modules de services, interrelation entre modules, création de réseau overlay spécifique au service…
- Des services inaccessibles à un utilisateur peuvent le devenir en cas de composition de réseaux ambiants.

Le schéma suivant (Figure 5. 1) récapitule les fonctionnalités qu'un réseau ambiant peut fournir, gérées par une entité globale (ACS: Ambient Control Space) ainsi que les interfaces définies pour contrôler les équipements des réseaux hétérogènes (ARI : Ambient Resource Interface), pour communiquer entre réseaux ambiants et éventuellement permettre la composition (ANI : Ambient Network Interface) et pour permettre aux services d'être déployés sur ces réseaux ambiants (ASI : Ambient Service Interface).

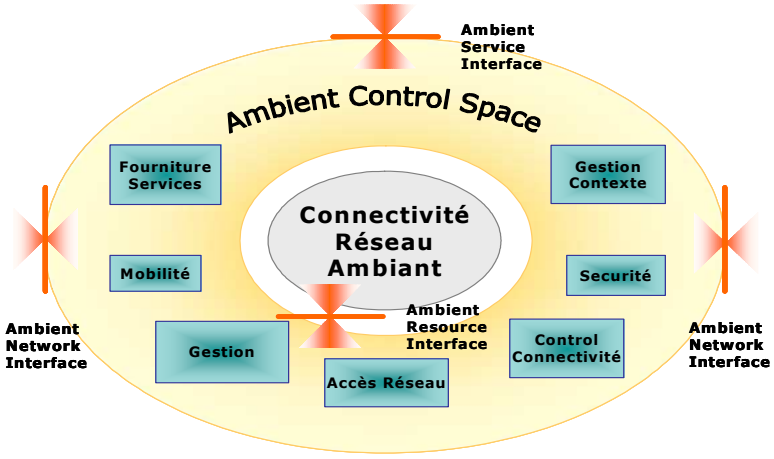

Figure 5. 1: Fonctionnalités et interfaces d'un réseau ambiant

5.1.2 Présentation des réseaux overlays

Les réseaux overlays sont apparus comme une solution pour s'affranchir des caractéristiques et des contraintes des réseaux physiques. Un réseau overlay est un réseau virtuel (logique) au-dessus des réseaux physiques, qui implémente ses propres mécanismes de contrôle et qui définit et maintient un protocole de routage au

niveau overlay, indépendant du routage physique sous-jacent. Cette solution permet de définir son propre réseau applicatif et de le faire évoluer indépendamment des réseaux physiques, d'optimiser le routage en s'adaptant en cas de problèmes de réseaux physiques, tels que les congestions de réseau, l'arrivée ou départ de nœuds mobiles…

Les exemples les plus connus d'utilisation de réseaux overlays sont les réseaux Pair à Pair (P2P : Peer-to-Peer) [Lua'04]. Les protocoles P2P tels que Chord [Sto'01], CAN [Rat'01], Tapestry [Zha'04], Pastry [Row'01] définissent ainsi leur propre réseau virtuel, connectant leurs utilisateurs, quel que soit le réseau physique sous-jacent et maintiennent leur topologie virtuelle par leur protocole de routage spécifique. Le cas le plus répandu d'applications P2P est le transfert de fichiers, mais d'autres types de services sont aussi possibles tels que le stockage d'information, la diffusion de contenu vidéo, des jeux….Dans le chapitre suivant, un cas d'utilisation de réseau overlay pour la diffusion de streaming vidéo en P2P est réalisé.

D'autres études utilisent les réseaux overlays pour optimiser le réseau. Dans RON [And'01] un réseau overlay pour rendre le transport plus fiable et résilient, ce qui permet de compenser rapidement en cas de problèmes (panne, réseau bloqué, ou surchargé, etc); dans QRON [Li'03], qui est une extension de RON, l'objectif est de fournir une architecture de base de réseau overlay pour améliorer la qualité du réseau Internet, principalement en utilisant un routage logique, évitant les nœuds critiques des réseaux physiques. Dans un objectif semblable, OverQoS [Sub'04] vise à étudier la Qualité de Service dans les réseaux overlays. Dans MBONE [Kum'95], le réseau overlay est utilisé pour transporter des paquets multicast tout en optimisant le transport et l'efficacité du réseau pour le multicast. XBone, lui, fournit à l'utilisateur une interface graphique, qui permet de créer un réseau overlay IP au-dessus du réseau IP [Tou'05]. L'objectif de ce type de réseau overlay est de permettre la construction d'un testbed aussi facile que possible.

D'autres types de réseau overlay apparaissent tels que les réseaux overlays sémantiques dans lesquels les réseaux sont constitués en fonction de la sémantique des informations mises à disposition par exemple. Parmi les exemples les plus connus de réseau overlay sémantique figurent Edutella [Nej'02], psearch [Tan'03] …

L'avantage principal des réseaux overlays est la capacité de déployer une topologie spécifique, un routage spécifique et l'inclusion de nœuds spécifiques dans ce réseau overlay en fonction des besoins et des objectifs. Ainsi, des réseaux overlays différents peuvent être créés pour différents services.

5.1.3 Réseaux overlays de services pour les réseaux ambiants

La brève introduction des réseaux ambiants a permis de mettre en évidence les notions de mobilité, de contexte, et de composition qui sont cruciales dans de tels réseaux et que doivent prendre en considération les services pour fournir un service adapté. Les réseaux overlays offrant une architecture flexible, leur utilisation pour la

fourniture de service dans des réseaux ambiants apparaît comme une solution intéressante.
Cette solution a été définie dans le contexte du projet sous les noms de SSON (Service Specific Overlay Networks) [Amb'05] [Rey'05] puis de SATO (Service-aware Adaptive Transport Overlay networks) [Amb'07] permettant de créer un réseau overlay spécifique pour un service donné pour délivrer un service optimisé pour l'utilisateur, où qu'il soit, quel que soit son réseau physique, quel que soit son terminal. SATO peut donc être utile pour la personnalisation ou l'adaptation du contenu au contexte des utilisateurs mais il peut également permettre de fournir des services à valeur ajoutée comme la détection de virus, la limitation de messages téléphoniques indésirables, Spits (Spam for VoIP), des services P2P.

- Architecture de réseau

La Figure 5. 2 présente la notion de réseau overlays, réseau logique au dessus des réseaux physiques réels. Dans l'exemple ci-dessous, certains nœuds font partie des deux réseaux alors que certains ne font pas partie du réseau overlay.
Les nœuds faisant partie de ce réseau overlay sont les nœuds terminaux (serveurs et clients) mais aussi des nœuds intermédiaires, appelés SatoPorts (SP), qui sont chargés de réaliser des traitements applicatifs selon les requis du service et les besoins du contexte.
Ce réseau overlay est créé par service (ou par type de service) et sur demande, mais la topologie de ce réseau peut évoluer en fonction des utilisateurs se connectant ou se déconnectant au service ou en fonction de l'évolution du réseau ambiant (mobilité de l'utilisateur, composition avec d'autres réseaux ambiants…). Ainsi, le réseau overlay doit pouvoir être étendu (rajout de nœuds offrant le traitement adéquat) ou diminué (suppression de nœuds devenus inutiles) en fonction du contexte. De la même manière, les composants de service (SP) doivent pouvoir être facilement et dynamiquement activés ou déployés sur les nœuds.
Un nœud overlay peut héberger plusieurs SPs et ainsi offrir plusieurs modules de services. De la même manière, un SP peut aussi appartenir à plusieurs réseaux overlays et fournir le traitement pour plusieurs services, puisque les SATOs sont spécifiques à un service ou un type de service. Dans ce cas, la composition des éléments de service est différente selon le service à fournir. C'est l'algorithme de routage overlay qui a pour rôle de chainer les composants de service (SPs) requis dans le bon ordre pour fournir le service final adapté à l'utilisateur.
Un réseau overlay définit ses propres mécanismes de communication et de maintenance du réseau. Pour la communication, les données overlays sont encapsulées dans un en-tête overlay spécifique avant d'être encapsulées dans l'en-tête IP traditionnel pour pouvoir être véhiculées. Un nœud overlay doit donc posséder la couche OSL (Overlay Support Layer) qui assure cette fonction.
La gestion des réseaux overlays SATO est assurée par une entité fonctionnelle dédiée, faisant partie du module de contrôle du réseau ambiant. Cette entité est ainsi en relation avec les autres entités fonctionnelles telles que la mobilité, la gestion du contexte ou encore la sécurité pour permettre d'adapter les réseaux overlays SATO dynamiquement, automatiquement et de manière transparente pour les services déployés dans le réseau ambiant.

Ces services s'interfacent avec le module de contrôle du réseau (ACS) ambiant via l'interface ASI (Ambient Service Interface). Cette interface permet aux composants de service localisés en dehors de l'ACS d'accéder aux composants de l'ACS et aux composants de l'ACS d'émettre des notifications aux composants de services.
Via cette interface, les services peuvent demander la création d'un réseau overlay spécifique SATO, en définissant les requis du service et la qualité de service souhaitée. L'entité de gestion du réseau overlay analyse la requête et cherche à établir un tel SATO en intégrant les nœuds permettant de fournir la qualité demandée en fonction des requis spécifiés. Via l'interface ASI, les services sont aussi en mesure de demander des modifications du SATO pour, par exemple, ajouter de nouveaux clients, de nouvelles contraintes. Enfin, les services peuvent demander la terminaison du réseau overlay et donc la libération des ressources.

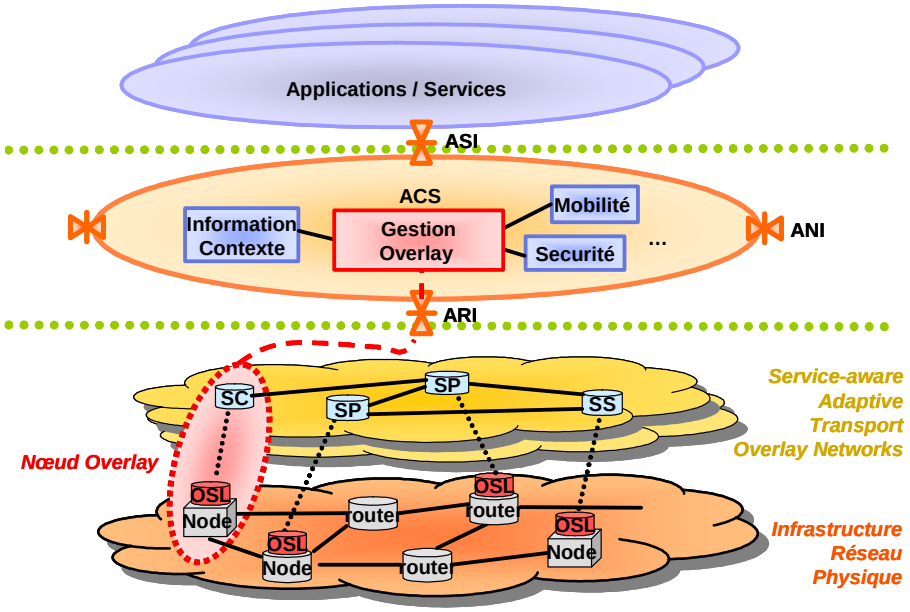

Figure 5. 2: Architecture du réseau overlay de service

- Composition de services

La section précédente décrit l'architecture du réseau overlay et les SPs le composant. Nous voyons ainsi que le service fourni aux utilisateurs sera réalisé par la succession de traitement par des modules de services, les SatoPorts. Cette façon de fournir un service diffère des architectures traditionnelles de type Client / Serveur et donc la conception des services devra être prévue dans cet esprit. La notion de chaine de service est ainsi introduite pour décrire comment un service final peut être rendu en utilisant les modules de services intermédiaires (SPs) et principalement leurs enchainements. Cette chaine est bien évidemment spécifique au service et aux fonctions à réaliser, mais aussi dépendante de la réalisation des modules de traitements.
En effet, en fonction de la granularité des modules de services intermédiaires, certaines fonctions peuvent varier. Par exemple, la fonction de transcodage audio/vidéo peut être fournie par un seul SP ou bien 2 SPs, l'un réalisant le transcodage audio, l'autre le transcodage vidéo.

Le schéma suivant (Figure 5. 3) présente un exemple de chaine de service, compose de 5 modules de services intermédiaires:

Figure 5. 3: Chaine de service initiale pour un service défini

Une fois la chaine de service finalisée et les nœuds overlays, hébergeant des SPs offrant les fonctionnalités requises, trouvés, un réseau overlay SATO intégrant ces SPs est créé. Ce réseau SATO ne contient pas uniquement une instance de chaque module de service mais plusieurs. Ceci permet au routage overlay de sélectionner le meilleur SP pour chaque type de composant lorsque le service sera utilisé par les clients. Ainsi, pour un même SATO pour un service donné, un flux d'un client pourra passer par un SP pour une fonction spécifique et le flux d'un autre client pourra passer par un autre SP offrant la même fonction. Le SATO pour ce service particulier peut donc être représenté par la Figure 5. 4 et les chemins des flux utilisateurs en rouge et noir pour les deux clients :

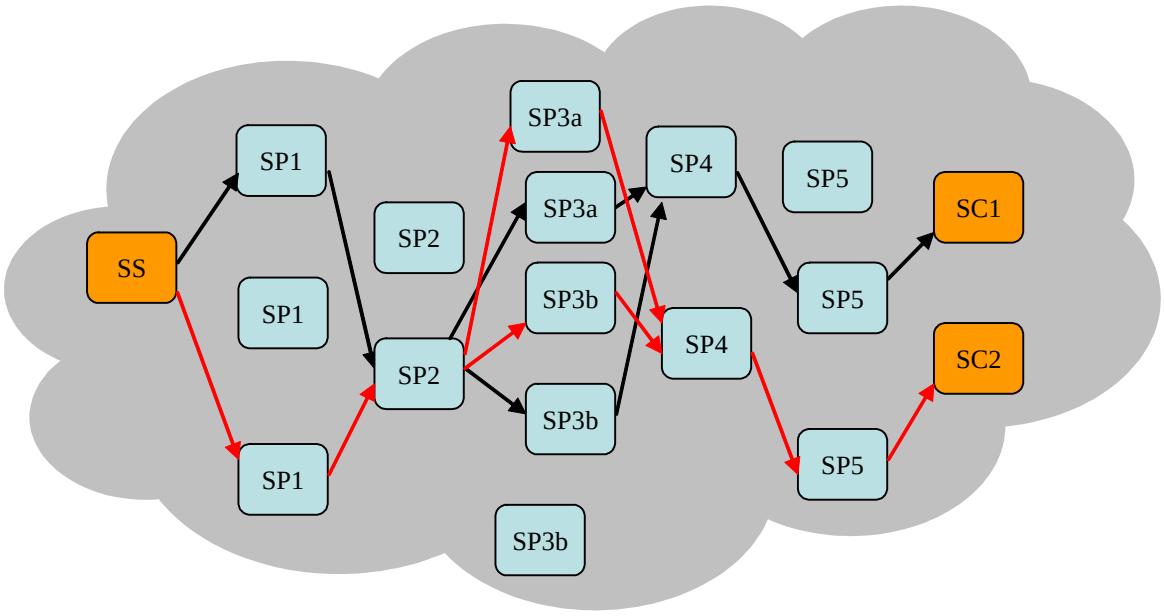

Figure 5. 4: Routage dans le réseau overlay de service

Le choix du SP est dynamique et fait en temps réel en fonction du contexte courant, par le protocole de routage overlay. En début de section, il a été dit que la taille de réseaux ambiants est très variable et peut être petite ou très grande. Ainsi, un protocole de routage unique ne conviendra pas à toutes les situations. Chaque réseau overlay SATO pourra utiliser un protocole de routage adapté et donc plusieurs protocoles de routage overlays peuvent être implémentés sur un même nœud faisant partie de plusieurs SATO. Des protocoles de routage basé sur les protocoles réseau courants tels que RIP ou OSPF peuvent être utilisés, mais les protocoles de routage P2P sont aussi étudiés pour les réseaux à large échelle et très dynamiques. De la

même manière, pour les réseaux overlays requérant une qualité de service précise, des protocoles de routage overlays avec QoS doivent être mis en œuvre.

Pour conclure, il est possible de résumer les avantages majeurs d'une telle solution par :

- La composition dynamique des éléments de services : les modules ne sont pas liés entre eux de manière fixe, mais peuvent évoluer en fonction du contexte et de l'évolution du réseau ambiant. Ainsi, une QoS requise par un service peut influer sur les choix des modules de service via le protocole de routage mis en œuvre.
- La mutualisation des éléments : un même composant de service (SP) peut être utilisé par plusieurs services ayant le requis d'un tel module. Cela réduit le temps de mise en œuvre des services, le nombre d'instances d'un même module…
- La fiabilité et tolérance aux pannes des services : plusieurs instances des modules de la chaine de service sont disponibles dans le SATO et le choix d'utilisation de l'un est défini en temps réel par le protocole de routage overlay. Ainsi, la qualité de service peut toujours être assurée. En cas d'échec d'un module, un autre est utilisé et l'entité de gestion du réseau overlay peut adapter le SATO pour inclure un nouveau SP assurant cette fonction.

5.2 Architecture du Nœud Overlay

Après avoir détaillé l'architecture du réseau overlay et la composition des modules de services (SPs) dans le réseau par le protocole de routage, cette section décrit l'architecture du nœud overlay lui-même (Figure 5. 5). Issu de la description précédente, un nœud overlay doit offrir plusieurs fonctionnalités et modules nécessaires :

- Un environnement d'exécution dans lequel les composants de services (SP) devront pouvoir être exécutés.
- Un mécanisme de déploiement dynamique dans le cas où le composant requis n'est pas disponible sur un nœud du réseau ambiant actuel
- Des composants de détection du contexte courant : soit eux-mêmes intégrés dans le nœud (comme les sondes, dénommées senseurs), soit des composants permettant de récupérer des informations d'autres entités (par exemple, l'entité de contexte ou de mobilité).
- Un ou des protocoles de routage overlay pour l'enchainement des SPs.
- Un module de gestion de vie du réseau overlay : participation à la création, maintenance, terminaison du réseau overlay de services en fonction d'événements : mobilité de l'utilisateur, changement de terminal, changement de réseau d'accès.

Les trois premiers points sont présents dans le nœud Potacco précédemment décrit et sont instanciés dans ce nœud overlay suivant les mêmes principes. Le protocole de routage overlay lui remplacera le module de routage du nœud Potacco. Le module d'interception de paquets (MIP) et le gestionnaire de connexion réseau (GCR) n'ont plus lieu d'être dans cette architecture puisqu'au niveau overlay, les connexions ne

sont pas transparentes mais bien identifiées. Enfin, un nouveau module pour la gestion de vie du réseau overlay est apporté, associé à l'interface ASI pour recevoir les requêtes.
La Figure 5. 5 suivante présente l'architecture du nœud overlay intégrant ces fonctionnalités. Le nœud a été défini en fonction des discussions au sein du projet, mais les travaux de cette thèse se retrouvent dans les parties concernant l'environnement d'exécution, le module de déploiement de code, les senseurs et les modules de détection d'information de contexte et la communication interne. D'autres modules sont intégrés dans ce nœud, notamment ceux relatifs à la gestion de vie (création, adaptation et terminaison) des réseaux overlays, la pile OSL, le protocole de routage… Ces blocs fonctionnels ont été investigués par d'autres partenaires du projet mais cités ici, car ce sont des éléments indispensables au réseau overlay.

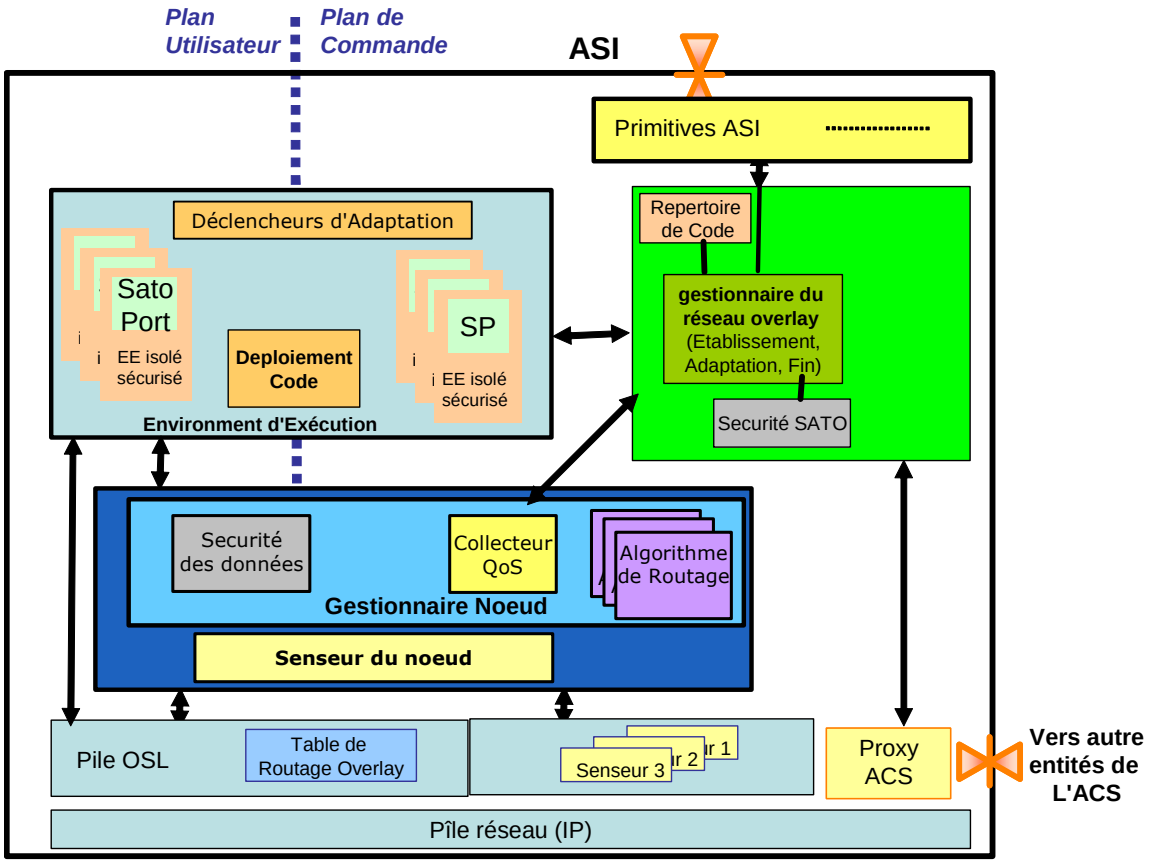

Figure 5. 5: Architecture du nœud SATO

Le *gestionnaire du réseau overlay* est le composant chargé de la gestion de vie du SATO en fonction des requêtes ASI (Application Service Interface) reçues des fournisseurs de service. Du fait de la nature très variable et dynamique des réseaux ambiants et des réseaux overlays, il a été décidé de distribuer cette fonctionnalité sur les nœuds plutôt que d'avoir une entité centralisée. Les nœuds présents dans le réseau ambiant communiquent entre eux pour gérer les réseaux SATO.
Pour établir un SATO, ce composant analyse les composants de service (SatoPorts) nécessaires pour fournir le service demandé par le fournisseur (via l'interface ASI) selon la qualité de service requise. Cette analyse conduit à la mise en place d'une

"chaine de service" (cf. section 5.1.3), qui représente l'ordre dans lequel exécuter les modules de services. Une fois établie cette chaine, il recherche les nœuds offrant les fonctionnalités requises, par une recherche sur un serveur centralisé ou de manière distribuée.

Le chainage des bons SPs pour fournir le service global est assuré par le protocole de routage overlay.
Dans le cadre du projet, un *algorithme de routage* a été défini, basé sur l'algorithme de Dijsktra [Dij'71]. Il permet de calculer le meilleur chemin en fonction de critères concernant la qualité des liens mais aussi la qualité des SPs:

$$Cost(Service\_Path) = \sum_{i=0}^{n} \alpha_i * (\chi * L_i) + \sum_{j=0}^{m} \beta_j * (\delta * SP_j)$$

La valeur Li correspond à la qualité des liens, évalués par les données *délai, bande passante, taux de perte de paquets et fiabilité.*
La valeur SPi correspond à la qualité des SatoPorts, évalués par les données *délai, mémoire, disponibilité, fiabilité.*
Pour différencier le chemin en fonction des applications et de l'importance de Li par rapport à Spi, des poids sont affectés à chacun.

Après sélection des nœuds et des meilleurs chemins possible (ou du moins répondant aux critères de QoS pour le service concerné), la *table de routage du réseau overlay* est configurée par l'algorithme de routage. Ceci est fait à l'établissement du réseau SATO mais aussi lors des adaptations éventuelles, en fonction des changements de contexte. La table de routage associe les identifiants des SPs et des nœuds overlays avec l'adresse physique (IP) de ces nœuds pour pouvoir les joindre.

La *pile OSL* est la couche qui permet d'encapsuler/décapsuler les données du réseau overlay dans les paquets réseau (IP par exemple). Associée à la table de routage overlay, la pile transfère les paquets aux modules applicatifs activés sur le nœud si besoin ou alors route au niveau overlay les paquets vers le nœud suivant, qui héberge le SP qui doit effectuer le traitement à la suite selon le chainage précédemment établi. La table de routage OSL tient donc à jour une correspondance entre les identifiants des SatoPorts impliqués (identifiant présent dans les paquets overlays) et les numéros de processus locaux des SPs.

Le réseau overlay SATO doit dynamiquement pouvoir s'auto-adapter aux changements des conditions de réseau, à la mobilité des utilisateurs, au changement de terminal…. Pour cela, une communication est établie avec une autre entité fonctionnelle de l'ACS: celui qui gère le contexte. Ainsi, la mobilité de l'utilisateur et le changement de terminal sont typiquement des exemples où l'information est fournie par une autre entité. La communication se fait par le composant dénommé *proxy ACS* sur le dessin.

Le SATO implémente aussi ses propres *senseurs* (Le terme senseur a été employé dans le cadre du projet, correspondant au terme "sonde" précédemment utilisé). Les senseurs sont prévus pour être de petite taille et flexible pour être dynamiquement déployés sur des nœuds de capacités variées. Il est prévu un senseur par type de

ressource supervisée. Les informations récupérées des senseurs sont stockées dans une base d'information, le module information de contexte, présent dans la figure 6.1. Un senseur "nœud" supervise les ressources internes du nœud overlay, telles que l'utilisation de la CPU ou de la mémoire. Un senseur réseau supervise la présence de connectivité et le délai entre les nœuds. Ceci est réalisé en envoyant des messages ICMP comme le fait un "ping". Un senseur "Bande Passante" permet d'évaluer la capacité et la bande passante disponible pour un chemin entre deux nœuds overlay. La bande passante disponible est évaluée par une approche active.

Dans le nœud, le composant *collecteur de QoS* est chargé de récupérer la valeur de certains senseurs, de les analyser et en cas de non-respect de la QoS, d'informer le module en charge de l'algorithme de routage afin d'identifier de nouvelles routes, assurant la qualité de service requise dans ce nouveau contexte.

Les *SATOPort* (SP) sont les modules applicatifs qui font des traitements sur les paquets transitant par le nœud dans le réseau overlay SATO. Ils peuvent opérer dans le plan de données ou dans le plan de contrôle.
Dans le plan de données, les SPs peuvent, par exemple, être des modules réalisant des fonctions de contrôle de virus, d'intégrité de données, de transcodage, de synchronisation de flux... Cela peut aussi être des moteurs d'adaptation tels que décrits dans la partie 5 de MPEG-21 [DIA]...
Dans le plan de contrôle, ils peuvent être des modules de type proxy SIP (Session Initiation Protocol), utilisés par exemple pour des communications en P2P SIP [SIPP2P] ou être des proxys RTSP (Real Time Streaming Protocol) pour Controller une session multimédia ou encore faire des détections de spam ou de spit (Spam over IP Telephony)…
Un SP peut donc fournir une fonction atomique, qui agrégée avec les autres, forme le service de bout en bout.

Un nœud peut héberger plusieurs SPs, instanciés dans un *environnement d'exécution* sécurisé. Si lors de l'établissement d'un SATO, un nœud offrant un SP requis n'est pas trouvé, un mécanisme de *déploiement de code* est activé pour déployer le code du bon SP sur un nœud. Le déploiement et l'activation d'un SP peut aussi être initié par des événements extérieurs, *les déclencheurs d'adaptation,* relatifs au contexte qui aurait changé.
Typiquement, ce mécanisme reprend la solution décrite dans le chapitre précédent, en l'instanciant pour un usage dans le contexte des réseaux mobiles. Ceci explique certains choix décrits dans le chapitre précédent tel que l'indépendance vis-à-vis de l'adressage IP, pour prendre en compte la mobilité des nœuds et notamment l'usage de HIP.
La Figure 5. 6 montre comment ce mécanisme de déploiement sécurisé de code, utilisant notamment un algorithme de Diffie-Helman, et une connexion IPSEC, associé à un environnement d'exécution de type OSGi a été défini dans ce nœud overlay.

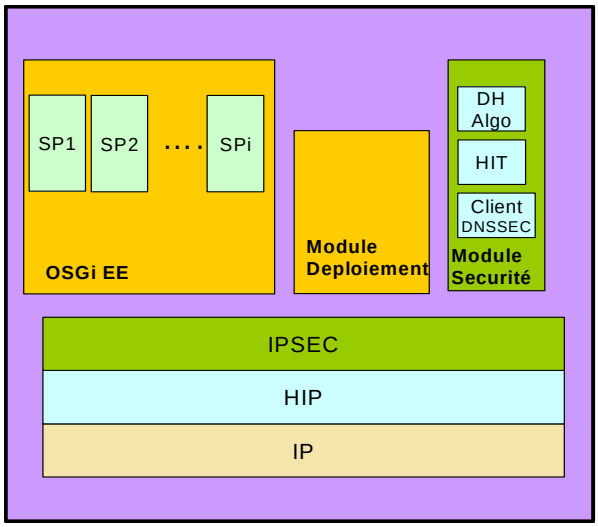

Figure 5. 6: Module de déploiement du SATO

Enfin, la sécurité est impliquée à deux endroits: pendant la phase de déploiement de code, dans le gestionnaire de réseau overlay avec un module de *sécurité SATO* et lorsqu'un lien doit être sécurisé en établissant une connexion de type TLS (Transport Layer Security) plutôt que du TCP pour la *sécurité des données.*

5.3 Démonstrateur d'un service IP TV personnalisé

La télévision diffusée sur le réseau Internet avec le protocole IP (IPTV) est un nouveau service qui est actuellement déployé dans le monde entier. Dans le passé, du fait de la faible bande passante des réseaux accès des utilisateurs (par exemple, liaison RTC ou RNIS, le service d'IPTV ne pouvait pas être fourni. Aujourd'hui du fait de la croissance du nombre d'utilisateurs ayant une connexion haut débit, ce service peut être déployé par plusieurs opérateurs de réseaux. Comparé à la télévision traditionnelle, le service IPTV va augmenter les fonctionnalités de la télévision. En effet, des fonctions comme la télévision interactive, la vidéo sur demande, la vidéoconférence, le partage vidéo, la publicité et la diffusion de contenu personnalisé pourraient être déployées dans un scénario IPTV. Cependant, actuellement, le contenu diffusé est unique, indépendamment du contexte de l'utilisateur. La diffusion vers des terminaux à capacités variées en utilisant des réseaux d'accès différents et en prenant en compte les préférences des utilisateurs représente encore un challenge à mettre en œuvre. Ceci est d'autant plus vrai pour les utilisateurs qui peuvent bouger et où les terminaux peuvent se connecter à différents réseaux d'accès, comme c'est le cas dans les réseaux ambiants. Dans cette section, un démonstrateur illustrant l'intérêt de la mise en œuvre d'un réseau SATO pour un service IPTV est présenté. Ce démonstrateur suit un scénario où un utilisateur accède au service depuis différents réseaux d'accès et où divers modules d'adaptation sont activés dans le réseau overlay pour offrir le service personnalisé.

5.3.1 Scénario

Ce scénario se situe dans un contexte où l'utilisateur dispose d'un réseau personnel PAN (Personal Area Network), composé d'un ordinateur portable, équipé d'une carte sans-fil WLAN et d'un téléphone mobile de type UMTS. Le réseau PAN est connecté à Internet par l'intermédiaire d'une gateway domestique, connectée à un réseau haut débit (type ADSL).
L'utilisateur veut assister à un cours d'enseignement à distance diffusé sur Internet en IP TV.
Préalablement à la diffusion du flux vidéo, le fournisseur de contenu IPTV demande la création d'un réseau overlay spécifique, SATO, pour fournir le service d'IPTV. Cette requête est transmise en utilisant l'interface ASI, en spécifiant les requis en terme de QoS et de fonctionnalités que doit apport le réseau SATO. L'ACS établit ensuite un réseau overlay pour diffuser le contenu vidéo. La source du flux est insérée dans le réseau overlay en tant que SatoServer. Durant la phase de création, comme expliquée précédemment, l'entité de gestion du réseau overlay définit la chaine de service nécessaire pour fournir le flux et recherche et instancie les modules de traitements (SPs) nécessaires.
Ensuite en fonction du contexte du client et des besoins de traitement intermédiaires, de nouveaux SPs peuvent être activés dans le réseau SATO. Dans ce scénario, 4 types de SPs sont déployés: un SP de transcodage, un SP de personnalisation, un SP pour séparer les flux audio et vidéo et un autre pour les réassembler (dans le cas où il faut transcoder le flux vidéo mais le flux audio par exemple).

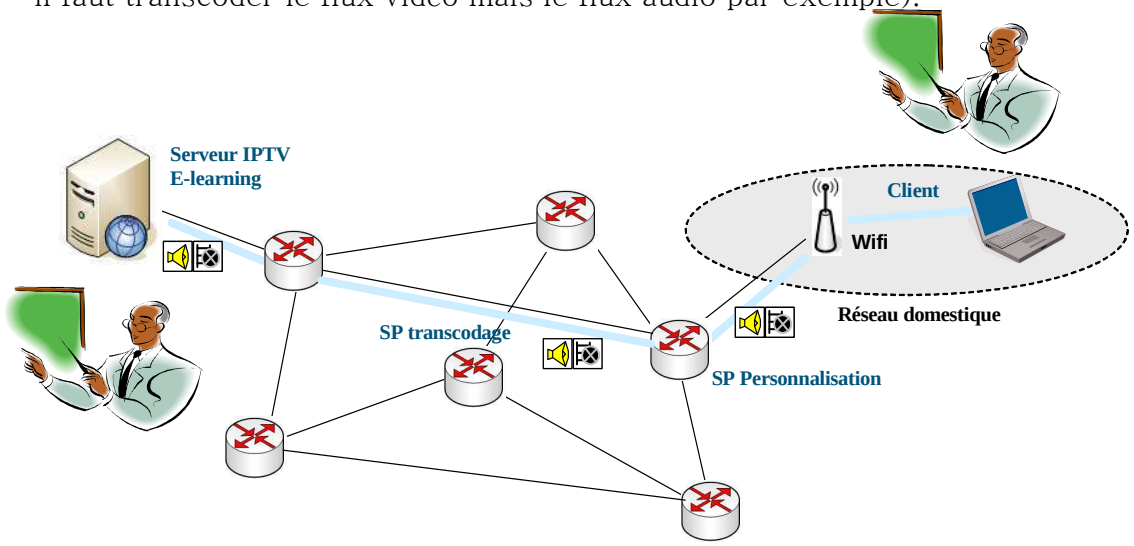

Figure 5. 7: Scénario IPTV : Service e-Learning

Au début, l'utilisateur est chez lui, et le flux vidéo d'enseignement à distance (e-learning) est diffusé sur son PC portable avec la qualité initiale, fournie par le fournisseur du flux. Dans ce cas là, il n'est pas nécessaire d'inclure les SP dans le chemin overlay. Cependant, comme illustré sur la Figure 5. 7, les données passent par un nœud offrant le service de personnalisation, mais le module applicatif de personnalisation n'est pas encore activé dans ce cas. Les données sont donc simplement routées par ce nœud.

Ensuite, l'utilisateur souhaite voir une chaine d'information (news TV). Cependant, il ne veut pas simplement zapper car il veut conserver un œil sur le service d'enseignement actuel. Pour cela, le SP de personnalisation est activé et celui-ci réalise la fonction de "PictureInPicture (PiP), autrement dit la faculté d'intégrer le flux de la chaine initiale en incrustation dans un coin de la nouvelle chaine. Ceci implique un traitement sur les données des flux, typiquement, une modification de la résolution spatiale pour la chaine initiale et la modification du contenu de la nouvelle chaine en insérant la chaine initiale dans un coin de l'image. Ceci pourrait être éventuellement être fait sur le terminal utilisateur (s'il a les capacités de traitements nécessaires) mais le faire dans un nœud intermédiaire permet de limiter le trafic diffusé jusqu'à l'utilisateur puisque cela revient à diffuser le débit d'une seule chaine au lieu de deux, résultant à un gain de bande passante dans le réseau d'accès de l'utilisateur. L'entité de gestion overlay va donc activer le module de personnalisation pour effectuer le traitement adéquat (Figure 5. 8).

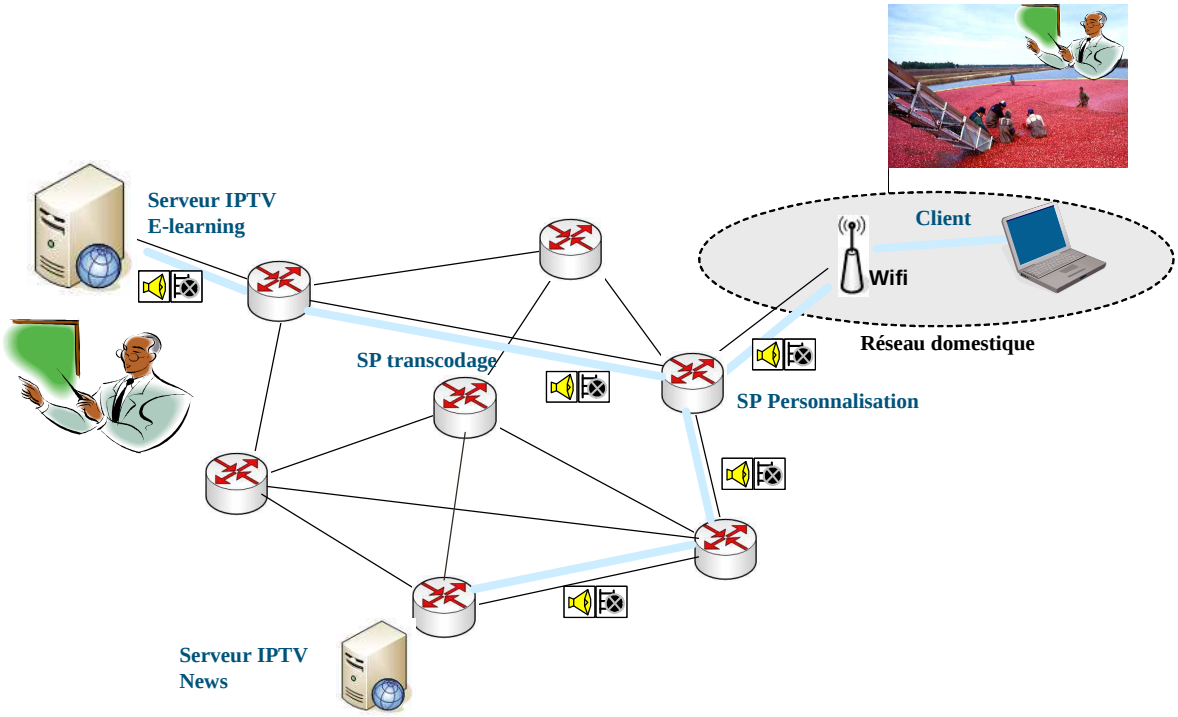

Figure 5. 8: Scénario IPTV : Fonction PiP : News + e-learning

Puis, une promotion ponctuelle limitée dans le temps est proposée par l'agence commerciale d'Orange à Lannion, comme par exemple : "Exceptionnel: Pendant 2 heures, réduction de 50 % sur les terminaux UMTS/Wifi". Ce message publicitaire est envoyé par le serveur de publicité au SP de personnalisation, qui a pour rôle d'insérer le message (au format "textbox") dans le flux vidéo actuellement transmis (Figure 5. 9). On peut aussi imaginer d'autres insertions, comme, sur réception d'un SMS, l'entité en charge d'envoyer le message l'envoie au SP de personnalisation qui l'insérera de la même manière dans le flux vidéo, plutôt que de l'envoyer sur le téléphone portable de l'utilisateur qui n'est pas forcement à côté de lui (ou l'envoyer au deux, pour que l'utilisateur puisse conserver une copie sur son portable) …

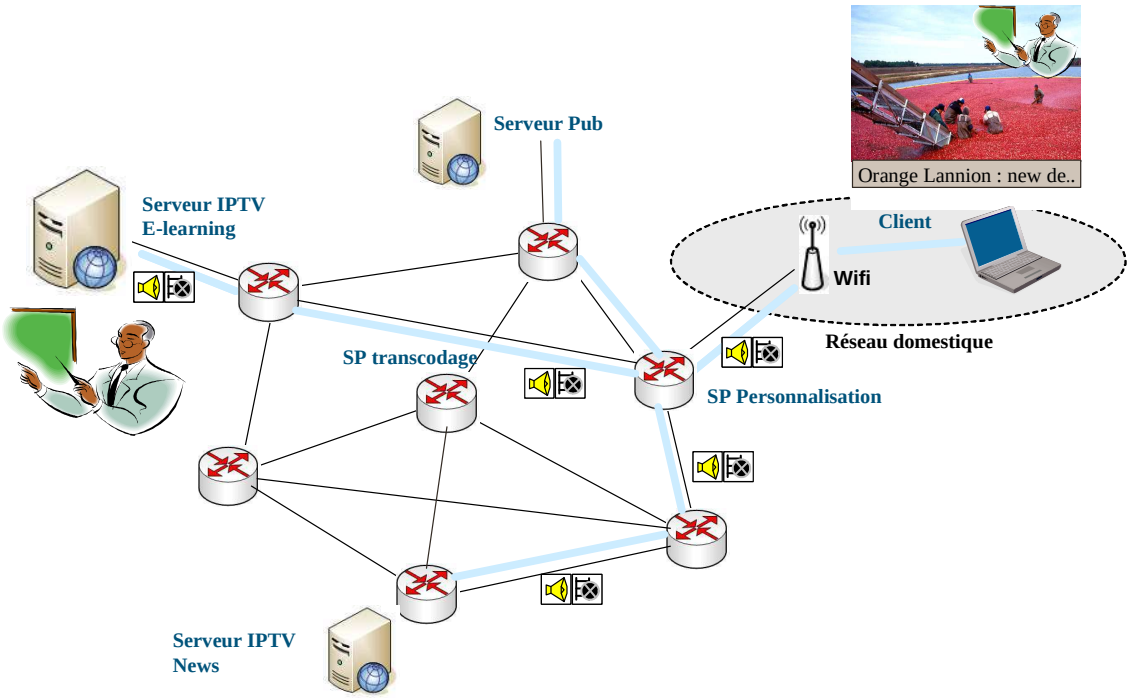

Figure 5. 9: Scénario IPTV : Insertion messages (textbox)

Finalement, l'utilisateur doit quitter son domicile et donc un transfert de la session vers son téléphone portable, qui est connecté au réseau UMTS, doit se dérouler. Le changement de contexte de l'utilisateur est détecté (nouveau terminal, nouveau réseau d'accès) par une entité de l'ACS qui en informe l'entité de gestion du réseau SATO. Celui-ci enclenche une procédure d'adaptation du SATO pour fournir un contenu adapté à ce nouveau contexte. Cela résulte en l'activation d'un module de transcodage vidéo pour adapter le format vidéo aux capacités du terminal mobile (par exemple réduction de la résolution spatiale et du bitrate de la vidéo) et en l'ajout des SP de séparation puis de réassemblage des flux audio et vidéo. Le chemin overlay est ainsi modifié pour passer via les SPs nécessaires (Figure 5. 10).

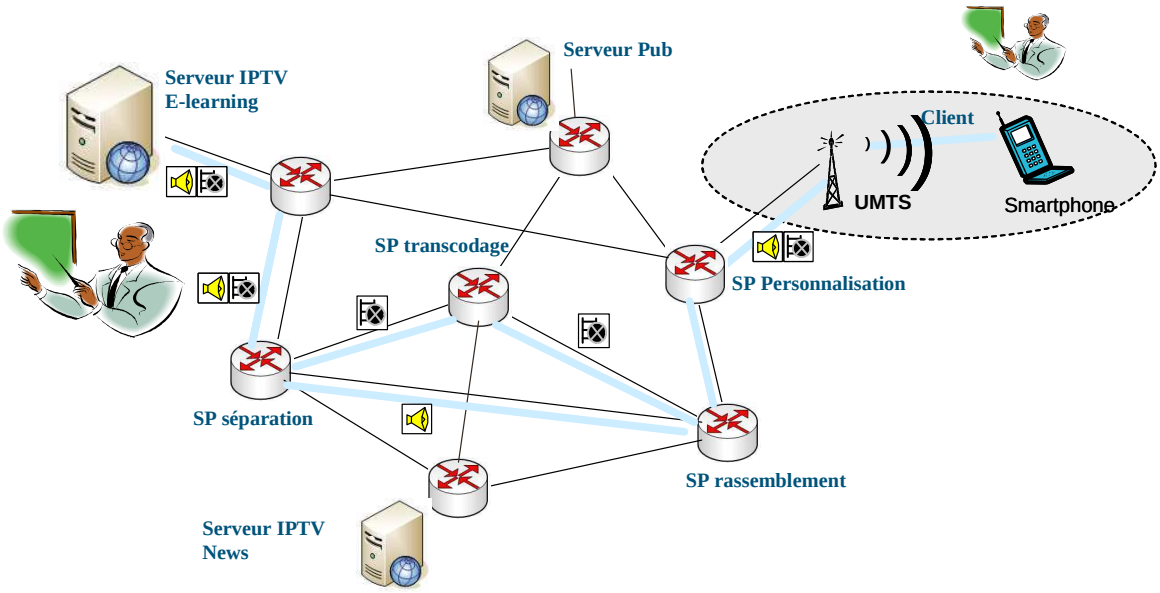

Figure 5. 10: Scénario IPTV : Basculement sur téléphone UMTS

5.3.2 Tests du scénario mis en œuvre

Le scénario présenté ci-dessus a été mis en œuvre dans le cadre du projet Ambient Networks, notamment par Ericsson, pour valider le concept et la faisabilité d'une solution de réseau overlay de services pour la fourniture de contenu adapté aux utilisateurs. Par rapport au scénario présenté, seules les phases de séparation et rassemblement de flux n'ont pas été implémentées.

Le nœud overlay repose sur un système *FreeBSD* et l'entité de gestion de réseau overlay et les modules de traitements ont été développés en *C/C++* et en *Java*. Pour le développement des SPs de traitements multimédias, *FFMPEG*, une librairie open source de codec audio/vidéo et *liblive555*, une librairie pour la diffusion de contenu multimédia, ont été utilisées. Le serveur de streaming a été développé à partir de la librairie liblive555 et le client vidéo est *VLC*. Pour la communication entre les entités fonctionnelles de l'ACS (voir section 5.1.1), le mécanisme de communication est basé sur les "*Web services*". En effet, cela permet une indépendance des langages de programmation et présente une approche standardisée des interfaces offertes par les entités de l'ACS.
Le testbed est composé de 5 PC portables, d'un téléphone portable et de 2 points d'accès Wifi. Dans le cadre du démonstrateur, les 2 flux IPTV ont été encodés en format CIF (Common Image Format) à 1500 kbit/s sur un format de transport MPEG2. Le changement de réseau d'accès est mis en œuvre par le basculement d'un point d'accès Wifi à un autre, ce qui entraine la reconfiguration automatique du SATO et l'activation des SPs.

Plusieurs mesures ont été réalisées pour avoir une idée de la performance du système, notamment du temps nécessaire pour la configuration des SPs et du temps de réponse perçu par l'utilisateur dans différents cas.

Le temps d'activation des SPs est calculé à partir du moment où le message de configuration est reçu jusqu'au moment où le composant est initialisé et prêt à fonctionner. Le tableau suivant indique le temps d'activation pour le SP de transcodage et le SP de personnalisation, avec la valeur moyenne et la valeur minimale et maximale entre parenthèses.

SP	Temps d'activation (ms)
Transcodage	12 (4 – 29)
Personnalisation: Picture in picture avec textbox	32 (12 – 40)

Le temps de réponse totale, perçue par l'utilisateur est calculé de la manière suivante :

$$T_{\text{reponse total}} = T_{\text{sync}} + T_{\text{reconfig}} + T_{\text{client}}$$

T_{sync} est le temps nécessaire pour stopper les tâches courantes. Il faut noter que pendant ce temps, les SPs continuent à effectuer le traitement sur les paquets selon

la configuration courante. D'après les évaluations, le temps moyen pour ce paramètre est entre 30 et 40 ms.

$T_{reconfig}$ est le temps nécessaire pour la reconfiguration des SP. Il faut noter que le SP est en pause pendant cet événement.

Le tableau suivant présente le temps de reconfiguration moyen (minimum et maximum entre parenthèses) en fonction de l'action à effectuer.

Action de reconfiguration	Temps de reconfiguration (ms)
Transcodage	1.6 (0.7 – 2)
Ajouter la textbox	0.7 (<1)
Changer la police de la textbox	0.7 (<1)
Changer la position de la textbox	0.2 (<1)
Supprimer la textbox	0.5 (<1)
Changer la position du PiP	0.4 (<1)
Supprimer les images de faible résolution	0.3 (<1)
Basculer entre résolution faible et principale	18 (11 – 27)
Ajouter les images de faible résolution	292 (154 – 447)

T_{client} est le temps nécessaire pour que le client vidéo (VLC) détecte les changements de format du flux. Cela est indépendant des SPs et du SATO lui-même. Cette évaluation met en évidence qu'à part pour l'action d'ajouter les images de faible résolution, les actions de reconfiguration prennent un temps court, et n'ont pas d'influence sur le temps de réponse perçu par l'utilisateur. Ceci démontre qu'avoir de tels composants dynamiques et reconfigurables entre le client et le serveur sur un réseau overlay de services est possible sans surcoût de traitement excessif.

5.4 Conclusion

Ce chapitre a montré l'utilisation du noeud Potacco dans un réseau overlay de services, dénommé SATO, facilitant le déploiement de services ambiants en ayant pour objectif la fourniture de services personnalisés, adaptés au contexte de l'utilisateur. Le réseau overlay SATO s'auto-adapte dynamiquement en fonction des changements de contexte en changeant la topologie du réseau (ajout/suppression de modules de traitement, re-routage des flux…). Ceci est un des aspects importants pour les réseaux B3G où les services sont structurés à partir des utilisateurs et où les notions de continuité de service sans couture sont de rigueur, quels que soient le réseau d'accès et le terminal utilisé.

Dans ce cas d'usage, le nœud Potacco offre les fonctionnalités d'hébergement de modules de traitements applicatifs, dynamiquement activables et les notions de détection de contexte, deux composants nécessaires notamment dans le cadre défini, celui des réseaux ambiants. Ce cas d'usage a montré le besoin d'un protocole de routage overlay pour l'enchainement des modules de traitement pour fournir le

service global et le nommage des nœuds et des composants indépendamment des réseaux physiques sous-jacents.
Un scénario et un démonstrateur relatif à un service d'IP TV (avec différents modules de traitement applicatif multimédia intermédiaires) ont été présentés pour illustrer l'intérêt de cette solution et les mesures réalisées sur le démonstrateur mis en œuvre ont indiqué des temps de reconfiguration et de traitement faibles, n'engendrant pas une dégradation du temps de réponse et de la qualité du flux fourni à l'utilisateur.

Chapitre 6

Nœud Potacco pour adaptation de contenu multimédia diffusé en P2P en fonction du contexte du réseau et des utilisateurs

Dans ce chapitre, le nœud Potacco fonctionne dans un réseau overlay en tant que nœud appartenant à un réseau P2P (Peer-to-Peer). Ceci implique que le nœud est à la fois client (consommateur du contenu) et serveur (fournisseur du contenu) pour un autre nœud du réseau P2P. De ce fait, l'adaptation est appliquée dans le réseau P2P lui-même et relative aux deux rôles d'un nœud P2P; le nœud souhaite lui-même recevoir un contenu adapté à son contexte, mais doit aussi pouvoir fournir un contenu adapté au contexte du nœud qu'il sert.
Le nœud Potacco doit toujours offrir un environnement d'exécution dans lequel on peut activer des mécanismes d'adaptation mais par rapport aux autres cas d'usage précédents, des modules de détection de QoE (Quality of Experience) sont ajoutés en tant que sondes applicatives. En effet, il faut pouvoir évaluer la qualité du contenu reçue/perçue par l'utilisateur pour savoir si cela satisfait ses requis et sinon, choisir un autre pair pour lui fournir le contenu qui lui convient.
De plus, il faut un module gérant le contexte des pairs et la sélection des pairs en fonction de différentes caractéristiques. Non seulement, les informations relatives au contexte du pair sont à prendre en considération (les caractéristiques du terminal utilisé, les préférences utilisateurs, le type de réseau d'accès) mais aussi des informations relatives au réseau lui-même pour optimiser les ressources réseau.
Après un descriptif des architectures et solutions actuelles de diffusion de contenu vidéo sur des réseaux P2P et leurs limitations en terme d'adaptation de contenu, ce chapitre présente l'architecture du nœud permettant l'adaptation dynamique de contenu multimédia diffusé sur un réseau P2P, ainsi qu'une entité de gestion du réseau P2P en charge de la sélection des pairs, en fonction du contexte des nœuds et de caractéristiques réseau [Mathi'07].
Finalement, le démonstrateur, réalisé en tant que preuve de faisabilité, et les tests d'évaluation réalisés sur le réseau PlanetLab, sont présentés.

6.1 Diffusion multimédia sur un réseau P2P

6.1.1 Architectures de réseau P2P pour la diffusion multimédia

L'architecture de réseau P2P de streaming inclut à la fois la manière de transférer le contenu multimédia entre les nœuds et les entités impliquées. L'entité, dénommée nœud ou pair, est un élément qui peut jouer soit le rôle de source, quand elle initie la

diffusion des contenus multimédia et a l'intention de les partager avec d'autres pairs, soit le rôle de client, quand elle souhaite visualiser un flux, soit le rôle de relai, quand elle reçoit le contenu et le transmet ensuite à un (des) autre(s) pair(s), assurant ainsi un "routage" overlay pour le streaming.
Pour la distribution du contenu, principalement deux modèles existent ; soit un arbre de diffusion avec une communication directe entre deux pairs (appelé modèle mono-source), soit un réseau entremêlé où le pair client reçoit le flux de plusieurs pairs (appelé modèle multi-sources).

- Architecture mono-source

L'architecture de réseau mono-source est utilisée lorsque le contenu multimédia est diffusé par un seul pair vers un ou des pair(s) client(s). Ainsi, un pair client reçoit le flux entier depuis un seul pair fournisseur. Cependant, comme pour tout réseau P2P, puisque le P2P repose sur la collaboration entre les pairs, le pair client, recevant le flux est lui aussi pair fournisseur pour un autre pair. Pour un service de streaming multimédia, cela signifie que le pair client bufférise une partie du contenu et le retransmet ensuite à la demande d'un pair client.

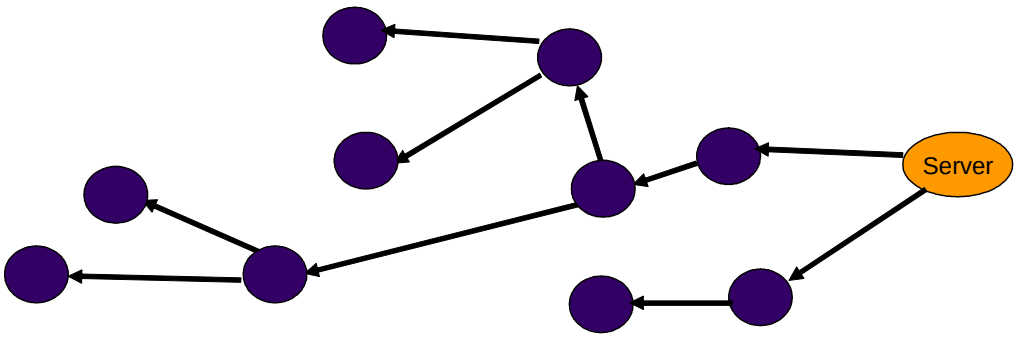

Figure 6. 1: Architecture de diffusion mono-source

Cette architecture a été initialement proposée par Peercast ; elle est ainsi parfois nommée "peercasting". Au début, Peercast a été utilisé pour la diffusion de flux audio (musique, radio) car les débits sont faibles et ainsi chaque pair recevant le flux peut lui aussi le retransmettre. En effet, avec l'asymétrie des réseaux ADSL, il fallait que le débit soit inférieur au flux montant des utilisateurs (et non pas seulement adapté au flux descendant).
Cependant, avec l'évolution des réseaux, l'extension à l'ADSL2+ et le déploiement à venir des liens FTTH (Fiber To The Home), le lien montant devient important (et avec le FTTH, les liens montant et descendant deviennent symétriques), laissant prévoir qu'une solution mono-source peut être utilisé dans le contexte de la diffusion de streaming vidéo avec maintenant des débits montants suffisants.
De plus, les usages évoluent et les utilisateurs commencent à utiliser leur terminal mobile (téléphone portable ou PDA) pour regarder des vidéos. Dans ce cas, ils sont connectés à des réseaux mobiles à débits limités. Il est donc possible qu'un pair sur le réseau fixe puisse fournir le contenu vers des pairs mobiles, son lien montant étant supérieur au débit descendant de l'utilisateur mobile.

A titre d'exemple de solution de streaming vidéo diffusé en P2P mono-source, on peut citer la solution PeerTV, qui se repose sur la solution Peercast.

- Architecture multi-sources

Une architecture réseau multi-sources est utilisée lorsque le contenu multimédia est transmis depuis plusieurs pairs fournisseurs vers un pair. Dans ce cas, le contenu multimédia est divisé en plusieurs morceaux (appelés aussi segments ou "chunks"); ce sont alors ces morceaux qui sont transmis dans le réseau. En d'autres termes, un pair client obtient certains segments du contenu multimédia d'un pair fournisseur et d'autres segments d'autres pairs. Et de la même manière, ce pair pourra lui aussi retransmettre les segments dont il dispose à d'autres pairs.
Cette architecture avec plusieurs sources de segments implique une phase de traitement et d'analyse préalable, afin de savoir quels morceaux obtenir à partir de quels pairs. Ceci nécessite alors un trafic de signalisation entre pairs pour échanger les tables d'informations indiquant les segments qu'ils possèdent et ceux qu'ils recherchent (par exemple la "buffer map" utilisée par PPLive).

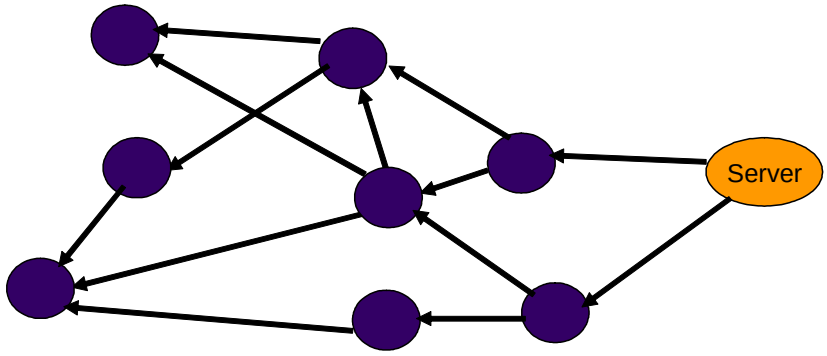

Figure 6. 2: Architecture de diffusion multi-sources

Cette architecture est principalement utilisée pour les flux vidéo où le débit est plus important que pour les flux audio. Ceci était particulièrement vrai avec les liaisons ADSL asymétriques où le débit montant était limité. Avoir ainsi plusieurs sources permettait de recevoir un flux d'une qualité correcte, ce qui n'aurait pas pu être possible avec une seule source ayant un débit montant inférieur au débit vidéo requis. Ce type d'architecture est utilisé par exemple par les applications actuelles telles que PPlive, PPStream, Tvants…

Indépendamment de l'architecture retenue, d'une manière générale, on peut dire que pour la gestion du réseau P2P, il est important que la solution prenne en considération les critères suivants :

- Puisqu'un réseau P2P est un réseau virtuel, la construction du réseau overlay doit être optimisée et ne pas interférer ou être en opposition avec les réseaux physiques sous-jacents. De plus, la maintenance du réseau overlay ne doit pas générer un surcoût important en termes de messages échangés ou de bande passante utilisée.

- Le réseau doit pouvoir gérer de manière efficace l'hétérogénéité des pairs et leur gestion de vie; les pairs pouvant arriver ou quitter le réseau fréquemment. Cela doit être pris en compte pour réagir rapidement à ces événements.
- Superviser et adapter le réseau logique P2P en fonction des conditions du réseau ou de la qualité. En effet, les réseaux mobiles sont pris en compte dans cette étude et la variabilité de la qualité sur ces réseaux doit être prise en compte.
- Enfin, la qualité vidéo doit être adaptée à ces environnements et pouvoir offrir aux utilisateurs un flux selon une qualité satisfaisante.

6.1.2 Description de solutions

Dans cette section sont décrites quelques solutions de streaming multimédia. Les trois premières sont des solutions "industrielles", actuellement déployées. Il en existe beaucoup d'autres telles que PPStream, TVants, Sopcast, UUSee, Babelgum etc, mais la présentation est limitée aux 3 ci-dessous puisqu'elles permettent de présenter les trois cas différents (mono-source, multi-sources et architecture mixte avec P2P dynamique et serveurs fixes) et que les autres sont plus ou moins similaires à l'une des trois.
Ensuite sont présentées des solutions académiques apportant certains arguments intéressants à considérer.

PeerTV est une application de streaming vidéo, créée en 2006.
Techniquement, PeerTV est un exemple d'implémentation de solution de distribution en P2P de type mono-source, puisqu'il s'appuie sur Peercast. Peercast était le premier protocole P2P permettant une mise en œuvre de diffusion multimédia en P2P. Le protocole de recherche de pairs utilisé par peercast utilise en partie gnutella [Rip'01]. Une implémentation peercast est disponible en logiciel libre et a une empreinte mémoire faible (#10 M) ce qui permet de l'installer sur divers environnements.
Peercast organise son réseau de distribution en arbre, chaque pair pouvant fournir le flux à un certain nombre de pairs (2 par défaut) et ainsi de suite. Après connexion par l'utilisateur à un site Web pour sélectionner le flux souhaité, le protocole peercast télécharge un fichier avec une extension "pls" depuis un serveur d'annuaire fixe lui indiquant le fournisseur du flux. Ensuite, via l'utilisation de Gnutella, peercast recherche un pair pouvant servir de relais.
En réception, les données multimédias sont bufférisées dans un fichier temporaire dans lequel l'application vidéo vient les récupérer pour afficher le flux. Ce fichier est aussi lu afin de pouvoir les retransmettre à d'autres pairs.

PPlive [Sil'06][Liu'06][Hei'07] est une application créée fin 2004 et aujourd'hui largement déployée, principalement en Asie et aux Etats-Unis.
Il utilise un protocole propriétaire, cependant après des analyses réseaux et comportementales, on peut dire que PPlive utilise des serveurs de chaines qui gèrent les pairs connectés à une chaine et fournissant ainsi une liste de ces pairs à un nouveau pair se connectant sur le flux concerné. Ensuite, le nouveau pair initie des connexions vers les pairs potentiellement fournisseurs et des messages relatifs aux

segments vidéos disponibles et manquants entre les pairs sont échangés. En effet, PPLive fonctionne selon une approche multi-sources et donc les contenus vidéo sont encodés et divisés en segments. Les pairs bufférisent les segments un certain moment selon un mécanisme de fenêtre glissante et permettant l'échange de segments entre pairs. La Figure 6. 3 présente la "buffer map" de PPlive.

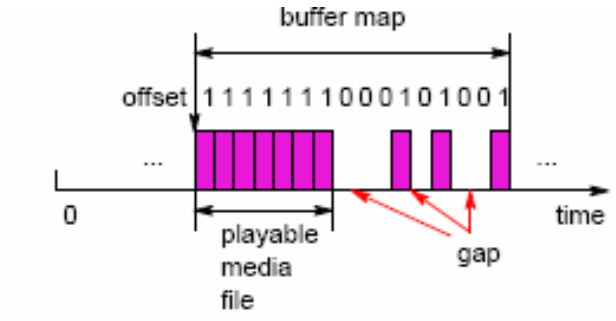

Figure 6. 3: Table des segments PPLive

Il a été remarqué par expérience qu'un pair PPlive connecté à haut débit pouvait fournir le flux à plusieurs pairs, le débit montant pouvant atteindre 8 Mbit/s. Au contraire, les pairs avec une bande passante montante faible ne fournissent aucun autre pair, de même que les pairs derrière un NAT (Network Address Translation) puisque PPLive ne gère pas le NAT.

Joost est une solution développée fin 2006 par les créateurs de Kazaa et Skype. Joost est une application de streaming uniquement et non pas de téléchargement progressif (condition "sine qua non" imposée par les industries pour autoriser Joost à diffuser leurs contenus). L'objectif initial de Joost était de fournir un contenu haute qualité et incluant les fonctionnalités de lecteur (avance rapide, pause, reprise…).
Techniquement, Joost repose sur une solution propriétaire, mais des analyses ont permis d'identifier des serveurs fixes d'authentification, de publicité et de gestion des chaines. De plus, des serveurs fixes servent aussi de pairs fournisseurs. Joost est donc un mélange entre une architecture P2P et une solution de serveurs. Peu d'informations sont fournies quant aux modules techniques, mais selon GigaOm [Gig'07], le codec vidéo pourrait être un codec H264, de la société CoreCodec. Les échanges sont cryptés et les contenus contiennent apparemment des DRM (Digital Right Management).

PULSE [Pia'06] est un système P2P conçu pour fonctionner dans les scénarios où la bande passante des nœuds peut être très hétérogène et variable dans le temps, comme aujourd'hui Internet. Il n'y a pas de contraintes structurelles imposées sur le processus de distribution : les données sont librement échangées entre les nœuds qui s'associent à cet effet, créant un réseau maillé (mesh). Le choix d'un réseau maillé a été fait pour permettre aux nœuds de changer de position dans la distribution en fonction des arrivées/départs de nœuds et en fonction de la bande passante disponible ; même si cela engendre un surcoût à cause des messages de signalisation. Le réseau P2P n'est pas structuré, une approche de diffusion épidémiologique (gossip) [Gan'03] a été préférée. L'association des nœuds est basée sur des incitations fondées sur des informations fournies, telles que la quantité de données échangées et

le délai de réception du flux, proche de la solution "tit-for-tat" de bittorent [Leg'05], permettant d'associer les pairs le mieux possible et améliorant les performances de téléchargement.
Pulse est composé principalement de 3 modules : le buffer de données, où les segments vidéo sont stockés avant lecture, une historisation de connaissance, qui stocke les informations sur les autres pairs, telles que la présence, les contenus, les relations passées, les associations courantes, etc, et une logique d'échange, qui a pour objectif la négociation des segments entre pairs et la planification des échanges.

PROMISE [Hef'03] est un système qui réalise plusieurs optimisations de sorte que le récepteur observe le minimum de fluctuation de qualité du flux multimédia reçu. En effet, PROMISE définit des mécanismes permettant la sélection des meilleurs pairs fournisseurs, la supervision des caractéristiques du réseau sous-jacent et la commutation dynamique des pairs fournisseurs. Trois approches sont proposées pour la sélection des meilleurs pairs: la sélection aléatoire de pairs qui peuvent répondre aux exigences du débit global, la sélection de bout-en-bout, qui évalue la qualité des liens overlays entre chaque pair candidat et le récepteur et la sélection en fonction de la topologie qui a connaissance du réseau sous-jacent et qui évalue la qualité de chaque segment du chemin.
L'évaluation réalisée par simulation montre que le choix avec connaissance de la topologie permet un choix judicieux en évitant les pairs connectés sur un lien à faible débit et ne permettant pas la fourniture du flux selon une bonne qualité.

P2VoD [Do'04] profite de pairs intermédiaires qui diffusent le contenu multimédia en cachant le contenu le plus récent du flux vidéo qu'il reçoit. Les clients dans P2VoD peuvent transférer le flux vidéo à un nouveau client s'ils ont suffisamment de bande passante et s'ils détiennent le premier bloc du fichier vidéo dans le buffer. Un système de cache est utilisé pour permettre à un groupe de clients, arrivant dans le système à des moments différents, de stocker le même contenu vidéo dans leur buffer. Un mécanisme de gestion efficace, pour faciliter l'arrivée et le départ des nœuds, ainsi que les réparations, basé sur un arbre de diffusion multicast est également proposé.

ZIGZAG [Duc'03] traite le problème de l'émission d'une source vers plusieurs destinations en prenant en compte l'état du réseau. Les objectifs sont de réduire au minimum le délai de bout en bout, de gérer la dynamicité de l'utilisateur et de limiter le surcout de trafic pour permettre le passage à l'échelle. Pour atteindre cet objectif, ZIGZAG organise les pairs clients en groupes (clusters) de taille limitée et construit un arbre multicast sur les clusters. Un ensemble de règles assure la connectivité de cet arbre et garantit que l'arbre a toujours une hauteur de $O(\log_k N)$ et un degré $O(k^2)$, où N est le nombre de clients et k une constante.

GnuStream [Jia'03] est construit au-dessus de Gnutella et conçu pour prendre en considération la dynamicité des réseaux P2P et son hétérogénéité. Il a pour objectif de gérer l'agrégation de bande passante, la défaillance de pair, la détection de dégradation et la maintenance de la qualité de streaming. Les changements de statut des pairs sont détectés à l'aide de sondes (envoi périodique dans le réseau). La

récupération de l'échec d'un pair ou de la dégradation de la qualité est assurée par la sélection des meilleurs pairs.

PALS [Rej'03] est un framework pour le streaming P2P avec un encodage adaptatif, défini en plusieurs couches. Le pair client coordonne la livraison du flux encodé en couches provenant de plusieurs émetteurs. Au démarrage, les pairs sont sélectionnés aléatoirement. Après cette première étape, la sélection de pairs est effectuée par un processus itératif. Un nouveau pair est conservé comme pair expéditeur seulement s'il améliore le débit global sinon il est rejeté. Pour l'adaptation de la qualité, le récepteur gère son buffer sur la base des paquets utilisés (affichés) et envoie l'état de son buffer à chaque pair émetteur régulièrement. Le mécanisme d'adaptation de la qualité de PALS est défini pour une période de temps plutôt que par paquet.

6.1.3 Limitations des solutions actuelles

Les solutions de streaming P2P vidéo commencent à prendre leur essor. Cependant, plusieurs limitations ou manques existent encore.

D'abord, le format du contenu diffusé est géré par le fournisseur initial et est dédié aux clients de réseau fixe (e.g. ADSL) et destiné à être visualisé sur un poste fixe (écran de PC). Les réseaux mobiles de type UMTS, Wifi et les terminaux légers de type Smart Phone, PDA par exemple ne sont pas pris en compte. Il n'y a pour l'instant aucune prise en compte du contexte de l'utilisateur qui permettrait de choisir dans quel format transmettre pour fournir un contenu adapté. Ceci implique aussi qu'il n'y a, pour l'instant, pas de convergence des services pour être multi-réseaux (accessible depuis plusieurs réseaux) et qu'il n'y a pas non plus de mécanisme de continuité de service entre réseaux et/ou terminaux.

De plus, bien que destinés à des écrans de PC (et donc ayant potentiellement une taille d'écran convenable), les contenus n'utilisent pas pleinement cette possibilité du fait de la bande passante disponible pour joindre l'utilisateur. En effet, les pairs peuvent être connectés via un réseau haut débit mais aussi via un réseau de plus faible débit (par exemple ADSL 512 kbit/s). De ce fait, pour pouvoir joindre un nombre plus important d'utilisateurs potentiels, les fournisseurs de contenus encodent le format avec un débit assez faible (256 ou 512 kbit/s), afin de permettre au plus grand nombre de recevoir le flux. Même si dans l'esprit, cela peut se concevoir, malheureusement, cela prive les utilisateurs ayant des capacités réseau plus importantes (ADSL2 ou FTTH) de bénéficier d'une meilleure qualité.
Eventuellement, on peut prévoir d'encoder la vidéo dans différents formats, pouvant être diffusés à destination de terminaux différents. C'est par exemple le cas du service TV d'Orange qui gère des contenus et des moyens de diffusion différents selon que l'utilisateur est chez lui, connecté à l'ADSL ou en déplacement sur son téléphone portable connecté au réseau UMTS (même si le cas d'orange TV ne fonctionne pas en mode P2P, il est cité à titre d'exemple pour la diffusion du même contenu encodé selon différents formats). Même si cette manière d'opérer peut être compréhensible pour quelques contenus et pour certains gros fournisseurs, elle n'est évidemment pas adaptée à l'ensemble des contenus multimédias qui peuvent être

visualisés. Les fournisseurs de contenus ne voulant peut-être pas s'intéresser au format dans lequel il doit être diffusé et les hébergeurs n'ayant peut-être pas envie de stocker x fois le même contenu, enregistré en x formats différents. Ceci est d'autant plus vrai maintenant avec la mise à disposition de contenus autoproduits. De plus, autant en mode client-serveur il est possible de gérer différents formats, autant cela se complique en modèle P2P, puisque le contenu passe par des nœuds intermédiaires, eux-mêmes consommateurs du contenu mais aussi re-diffuseur. La seule manière actuelle de faire est de créer autant de réseaux d'utilisateurs P2P que de formats connus (2 utilisateurs regardant le même contenu mais encodé différemment n'étant pas dans le même réseau de consommateurs). Cela conduit à une séparation des utilisateurs du réseau et donc à une possible isolation.

Ces solutions ne prennent pas (ou peu) en compte des informations sur les réseaux physiques. Par exemple, des informations, plutôt statiques sur le réseau, telles que la localisation des pairs (plaque ADSL, région, pays), sur l'ISP des pairs (Orange, Free …) ou sur les coûts de peering éventuels ne sont pas intégrées. De même, une analyse de l'utilisation du réseau en fonction des heures de la journée (heures creuses, pleines, week-end, semaine…) pour choisir les pairs n'est pas prise en compte. En effet, il peut être intéressant pour un opérateur réseau comme Orange de récupérer le flux multimédia depuis un pair fournisseur connecté sur la même plaque ADSL, ou depuis un pair chez Orange plutôt que chez Free (évitant ainsi le transfert de données via le point de peering) ou à l'inverse la journée d'aller chercher le contenu aux US plutôt que sur la plaque locale….
Les conditions du réseau, plutôt dynamiques, sont un peu plus prises en compte par les solutions actuelles (principalement le délai) mais insuffisamment et cela est important pour pouvoir fournir un contenu en essayant de proposer la meilleure qualité possible. En effet, nous avons vu que le format est défini à l'avance et donc le débit réseau nécessaire aussi. Ainsi, en cas de congestion ou de problèmes réseau, le service de P2P vidéo peut être inutilisable ou de mauvaise qualité pour l'utilisateur.

Finalement, aucun mécanisme de supervision ou détection de la qualité fournie à l'utilisateur n'est mis en œuvre dans les solutions actuelles. En effet, le contenu est transmis dans son format unique à destination de tous les clients, en "best-effort". Ainsi, il se peut que la qualité soit dégradée à cause de congestion réseau, de pairs relais surchargés… Les solutions actuelles n'offrent pas de moyen de détecter que la qualité perçue par l'utilisateur (QoE : Quality of Experience) n'est pas satisfaisante et ainsi pouvoir réorganiser le réseau P2P de distribution en fonction.

La solution présentée dans ce chapitre, basée sur une architecture où le nœud Potacco est utilisé en tant que nœud pair d'un réseau P2P, a pour objectif de prendre en considération ces manques, principalement la prise en compte du contexte utilisateur (et des informations réseaux) lors de la sélection des pairs, la possibilité de fournir un contenu adapté aux utilisateurs et la supervision en permanence la qualité du flux fournie pour pouvoir réorganiser la distribution du flux P2P en cas de mauvaise qualité.

6.2 Description de la solution

L'idée à la base de la solution est de structurer un arbre de diffusion du flux multimédia en fonction du contexte de l'utilisateur et entre autres de son réseau d'accès. En effet, avec la diversité des réseaux d'accès aujourd'hui, chacun ayant ses propres caractéristiques de débit, il est possible qu'un utilisateur connecté à un certain type de réseau d'accès puisse retransmettre le flux à un utilisateur ayant un réseau d'accès similaire ou avec des caractéristiques inférieures en débits. Ceci est présenté sur la Figure 6. 4, où les utilisateurs ayant des réseaux d'accès haut débit sont en haut de l'arbre et ceux ayant les plus faibles débits en bas de l'arbre.

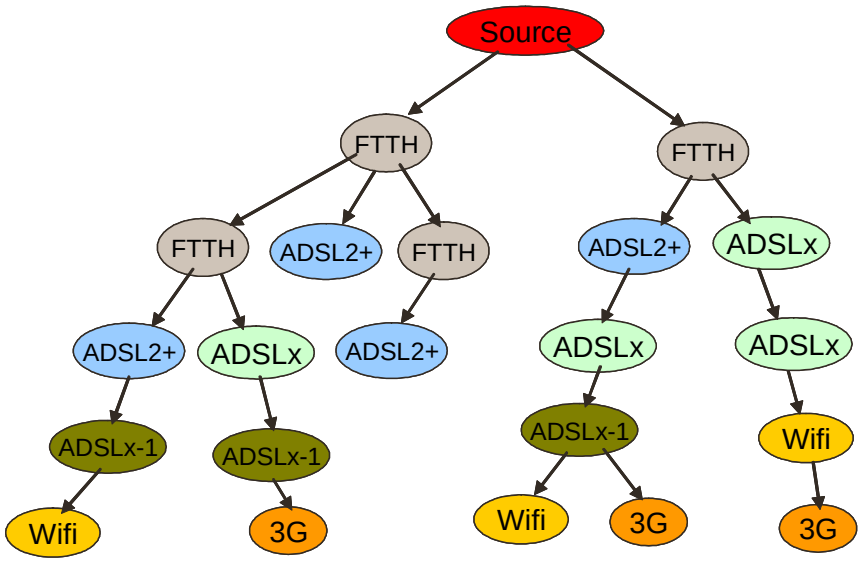

Figure 6. 4: Arbre de diffusion prenant en compte le réseau physique

Cette structuration implique que le flux multimédia sera délivré sur des liens réseaux à capacités de plus en plus faibles. C'est ici qu'intervient la notion d'adaptation dynamique par les pairs, qui pourront ainsi adapter le flux multimédia en fonction des caractéristiques des pairs clients qui sont connectés à lui. Par exemple, sur la Figure 6. 4, un utilisateur ADSL2+ pourra recevoir un flux de bonne qualité, mais devra l'adapter avant de le remettre vers un utilisateur ayant un réseau d'accès ADSLx-1 par exemple.
D'autres critères réseau, tels que l'ISP du pair, par exemple, pourront être pris en compte pour connecter le pair client à un pair fournisseur appartenant au même ISP: ce qui est présenté sur la figure avec des sous-arbres différents : par exemple à gauche les utilisateurs Orange (qui eux-mêmes se redivisent en deux en fonction de leur localisation par exemple), et à droite, ceux de Free….
Il est aussi possible de structurer l'arbre de diffusion en fonction d'informations de contexte telles que la qualité souhaitée, le type de terminal…

Cette structuration fait aussi apparaitre que le débit vidéo ne pourra pas être le même à tous les endroits de l'arbre (à moins d'encoder le flux au débit le plus bas, compatible avec les terminaux mobiles, ce qui serait dommage pour les utilisateurs haut débit).

Les solutions d'adaptation vidéo basées sur le codage hiérarchique apparaissent comme une solution intéressante à utiliser conjointement avec cette organisation pour pouvoir adresser différents niveaux de qualité. Ainsi les utilisateurs FTTH, recevront le flux avec la meilleure qualité possible. Ensuite, ces pairs adapteront le flux hiérarchique (en supprimant quelques données d'amélioration) avant de le transférer aux utilisateurs ADSL2+ et ainsi de suite. Grâce à l'utilisation des codages hiérarchiques et l'adaptation dynamique, le débit réseau nécessaire sera moindre et le flux pourra ainsi être délivré sur les différents réseaux. Si le codage hiérarchique ne permet pas d'adapter le flux sur toute la profondeur de l'arbre, on peut imaginer d'inclure des transcodeurs sur certains pairs pour réduire le débit. De la même manière, on peut aussi penser à adapter le contenu au type de terminal en faisant une adaptation de format avant de transmettre le flux (par exemple aux utilisateurs mobiles ayant un PDA).

Des modules de détection de la qualité reçue (et perçue par l'utilisateur) sont rajoutés sur les pairs pour mesurer si la qualité est satisfaisante et si elle ne l'est pas, cela déclenche la réorganisation du réseau, c'est-à-dire la recherche d'un autre pair fournisseur dans le réseau, soit directement, soit réalisant une adaptation.

Cette utilisation conjointe de la connaissance du contexte utilisateur (terminal, réseau d'accès) avec les capacités d'adaptation de codage hiérarchique ou de transcodeur représente l'idée à la base de cette solution. Il est ainsi possible de représenter l'arbre de diffusion du flux vidéo en P2P adapté au contexte comme sur la Figure 6. 5 où les nœuds de couleurs différentes représentent un contenu différent.

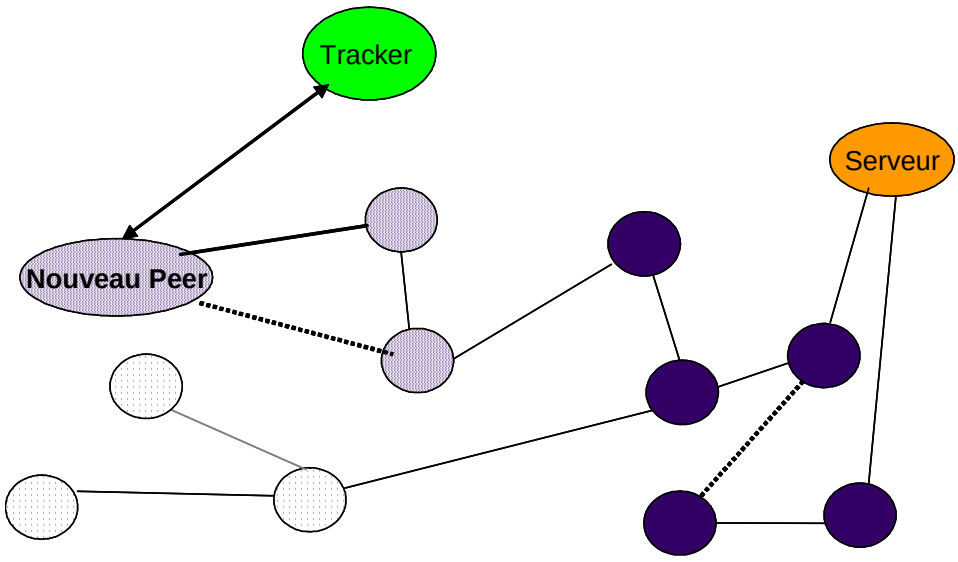

Figure 6. 5: Arbre de diffusion P2P avec contenu vidéo adapté

6.2.1 Architecture fonctionnelle de la solution

Dans l'architecture, il y a fonctionnellement 4 entités distinctes :

- 1 pair client, recevant le flux
- 1 pair serveur, fournissant le flux
- 1 pair serveur/adaptateur, fournissant le flux adapté
- 1 entité de gestion du réseau P2P (gestion des contextes et sélection des pairs fournisseurs)

Les 3 premières entités, bien que distinctes fonctionnellement, peuvent être physiquement activées sur un même nœud Potacco, jouant ainsi différents rôles. Fonctionnellement, il est possible de différencier les pairs serveurs "simples" des nœuds serveurs/adaptateurs en disant que les nœuds adaptateur soient des nœuds ayant des capacités supérieures aux nœuds simplement serveurs par exemple. Le nœud Potacco est ainsi utilisé dans un réseau P2P et peut faire de l'adaptation dynamique en fonction du contexte des pairs (autres nœuds Potacco) qu'il fournira.
L'entité de gestion du réseau P2P a en charge la gestion des contextes des utilisateurs, les pairs connectés à un flux et la responsabilité de la sélection des pairs fournisseurs lors d'une requête d'un pair client. Cette entité peut être une entité centralisée (par exemple comme un tracker Bittorrent) ou décentralisée, des parties de la fonction étant résidentes sur les nœuds pairs du réseau.
Dans le démonstrateur réalisé pour illustrer le fonctionnement de cette solution, le choix a été fait d'avoir une entité centralisée. En effet, cela permet au fournisseur du service d'apporter des informations supplémentaires dans les contextes utilisateurs, issues de bases de données externes (comme des informations réseau par exemple). Cela permet aussi d'avoir un meilleur contrôle sur le réseau, puisque c'est cette entité qui est en charge de la sélection des pairs et de la mise en relation des pairs, et de pouvoir facilement modifier l'algorithme de sélection ou de le rendre plus souple. Finalement, cela évite les messages de signalisation entre les pairs pour s'échanger les informations de chacun et sélectionner les fournisseurs. Dans ce cas, seule une communication entre le pair client et l'entité de gestion P2P est établie.

La figure suivante présente les blocs fonctionnels majeurs de cette solution.

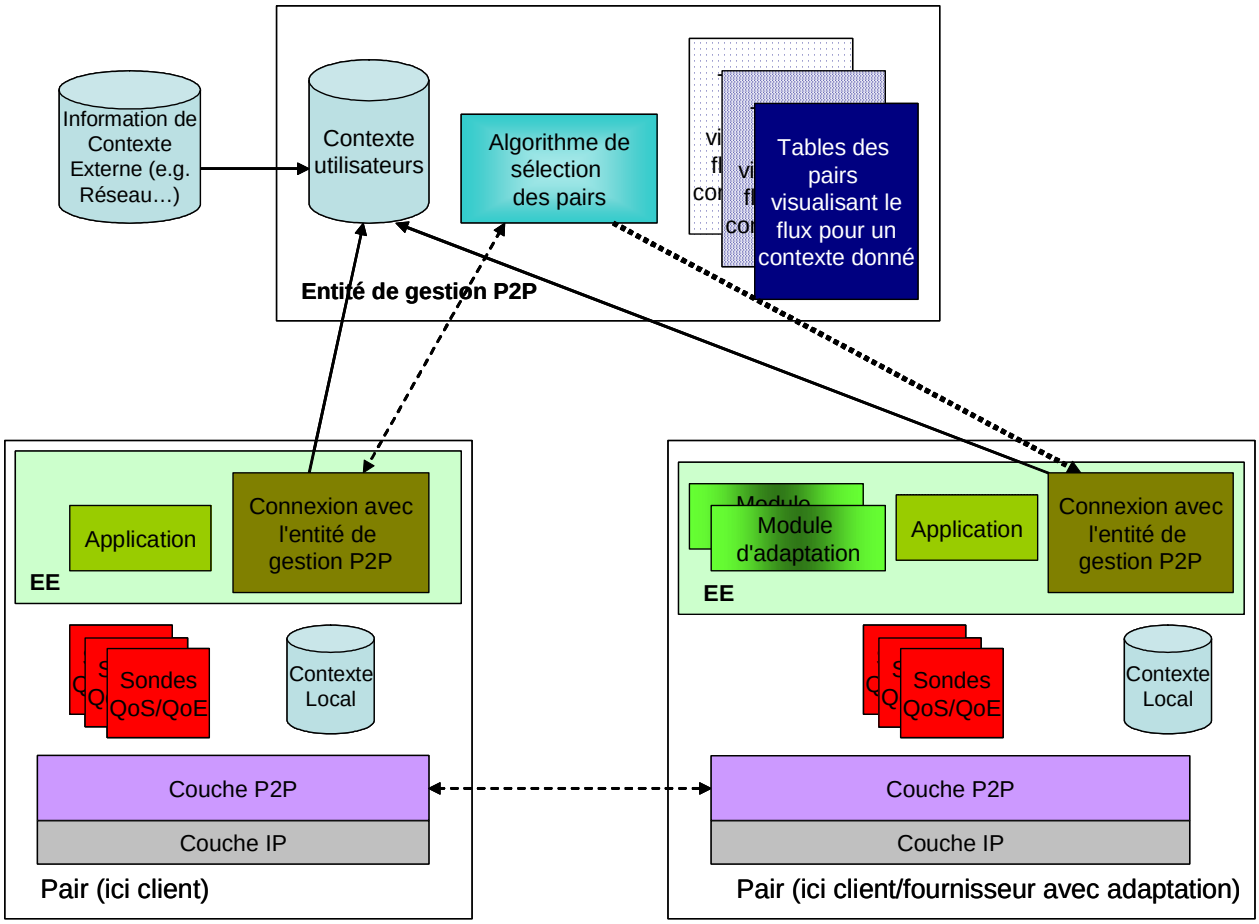

Figure 6. 6: Architecture fonctionnelle de la solution

Sur les pairs, 6 composants principaux existent :

- Le module "*couche P2P*" qui gère les connexions entre les pairs (pour la signalisation et pour le transfert des données multimédia). Cette connexion est simple et directe dans le cas du mode de diffusion mono-source.
- Le module "*contexte local*" qui recense les caractéristiques et paramètres locaux à donner à l'entité de gestion du réseau P2P lorsque le pair veut se connecter.
- Le module "*connexion avec l'entité de gestion P2P*" pour l'échange des messages pour se connecter à un flux ou pour demander un autre pair lorsque la qualité n'est pas bonne ou pour quitter…
- Le module "*Application*" qui est l'application elle-même qui permet de visualiser le flux. Par exemple dans le démonstrateur réalisé, cela peut être VLC ou Mplayer.
- Le ou les module(s) "*Analyse QoS/QoE*" qui analyse(nt) en temps réel et en tâche de fond les valeurs de QoS/QoE pour détecter si elles sont valides (par rapport aux seuils définis). Ces valeurs sont par exemple le nombre de paquets perdus, le débit moyen reçu (réseau et applicatif), le délai et la gigue entre paquets successifs, la similarité entre images…
- Le ou les module(s) "*d'adaptation*" qui ont pour charge de réaliser les adaptations nécessaires pour fournir un contenu adapté au contexte du pair client, en fonction des commandes reçues de l'entité de gestion du réseau P2P.

Sur l'entité de gestion du réseau P2P, il y a 3 modules principaux.

- Un composant gérant les *contextes utilisateurs.* Ce module récupère les informations de contexte depuis les requêtes des pairs clients et éventuellement d'autres modules (ou bases de données) externes de contexte ou d'informations utilisateurs.
- Un module "*algorithme de sélection des pairs*" qui a pour rôle de sélectionner les pairs fournisseurs du flux au pair client faisant sa requête. Ce module utilise les informations de contexte pour sélectionner les pairs adéquats et les tables de pairs visualisant les flux par contexte pour connaître les pairs fournisseurs potentiels
- Un module de "*tables de pairs visualisant les flux par contexte*". Ce module gère des tables de pairs visualisant le même flux (selon le nom) et dans le même contexte (format, qualité…). Il peut y avoir donc plusieurs tables pour un même flux vidéo. Dans ces tables figurent les pairs clients (juste receveur) mais aussi les pairs fournisseurs. Pour chaque nœud pouvant faire fournisseur (relais), il y a un champ qui indique combien de pairs clients, le pair concerné peut fournir. Cette valeur est variable et dépendante des informations de contexte. Par exemple, il est possible de configurer cette valeur en disant qu'un pair FTTH peut fournir 10 pairs suivant la qualité 1 ou 15 pairs suivant la qualité 2 et qu'un pair ADSL2 peut fournir 4 pairs pour la qualité 1 et 6 pairs avec la qualité 2… Ces valeurs sont configurables en fonction du contexte et du type de flux.

Ainsi, lorsqu'un nouveau pair se connecte, l'entité de gestion du réseau P2P cherche dans la bonne table le ou les pair(s) qui pourraient fournir le flux au pair client en fonction du contexte du pair client. Bien évidemment, il peut se produire le cas où il y a un pair avec le bon contexte mais qui fournit déjà le flux pour x pairs (x étant le maximum) et ne peut pas en fournir plus. Il faut donc choisir un autre pair dans la même table et s'il n'y en a pas en choisir un autre dans une table de qualité supérieure et lui commander de faire de l'adaptation.

Le schéma suivant résume ainsi le mode de fonctionnement de connexion d'un nouveau pair au réseau P2P pour recevoir le flux adapté :

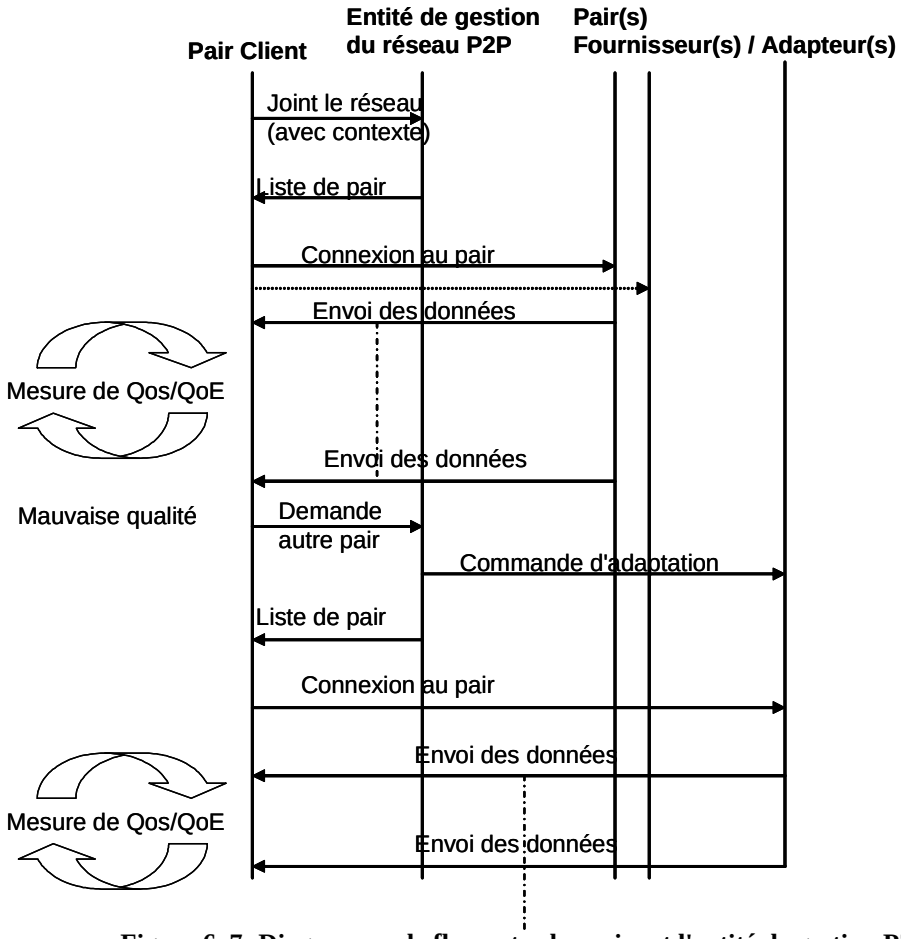

Figure 6. 7: Diagramme de flux entre les pairs et l'entité de gestion P2P

6.3 Démonstrateur

6.3.1 Choix technologiques

Un démonstrateur a été réalisé pour illustrer la solution décrite ci-dessus.

Dans ce démonstrateur, le mécanisme de diffusion *mono-source* a été choisi, en rapprochement avec l'arbre des réseaux d'accès introduit dans la description de la solution.
Le contenu diffusé est de type vidéo. Bien entendu, l'implémentation fonctionne aussi pour les flux audio. Seuls les modules d'adaptation de contenus et de détection de QoS/QoE pouvant différer.

Dans cette implémentation, l'*entité de gestion du réseau P2P est une entité centralisée* (de type tracker). Cette option semble intéressante pour un service de diffusion P2P proposé par un opérateur réseau comme Orange car il permet d'avoir un contrôle sur le réseau P2P. L'entité de gestion du réseau P2P est un programme écrit en *C* et fonctionnant sur une machine dédiée, montrant ainsi l'isolation entre cette fonctionnalité et les pairs du réseau P2P.

Les pairs (clients ou serveurs) sont écrits en langage *C* et le protocole P2P mis en œuvre est basé sur le protocole *Peercast,* qui a été modifié pour prendre en compte les objectifs annoncés. En effet, peercast implémente le protocole Gnutella pour la recherche et la communication de pairs. Dans cette solution, Gnutella n'est pas utile puisque c'est l'entité de gestion du réseau P2P qui se charge de mettre en relation les pairs. De plus, la communication entre les pairs et le serveur d'annuaire de Peercast (et les mécanismes sous-jacents) ont été inhibés.
Les paquets émis sur le réseau entre les pairs sont des paquets IP mais un paquet ne correspond pas forcément à une image du flux. Il faut donc un module qui puisse réassembler une image entière à partir de plusieurs paquets IP transmis sur le réseau. Pour cela, la librairie *FFMPEG* est utilisée.

Sur chaque pair, des modules de détection de qualité (QoS et QoE) sont implémentés.
La QoE (Quality of Experience) est généralement évaluée en donnant un résultat entre 0 et 5, 0 représentant une qualité inacceptable et 5 une très bonne qualité. Dans ce démonstrateur, trois sondes de mesures de QoE ont été développées pour évaluer la qualité des services de streaming audio et vidéo fournies.
La première sonde, une *Sonde MDI*, est chargée d'évaluer le MDI (Media Delivery Index) [Agi'06] pour des connexions multimédias surtout. Le MDI est estimé en fonction de deux paramètres : le DF (Delay Factor) et le MLR (Media Loss Rate). Le DF est basé sur la gigue entre les réceptions de paquets successifs. Pour être acceptable le DF doit être compris entre 9 et 50 ms selon [Agi'06]. Le MLR est le nombre de paquets média perdus ou mal-ordonnés.
Une *sonde MPQM* (Moving Pictures Quality Metric) [Bran'96] est basée sur la métrique définie au MPEG forum, visant à évaluer la qualité d'une vidéo. La formule MPQM inclut dans son calcul un paramètre de complexité de l'image. Ce paramètre est donc certainement performant pour la mesure de deux vidéos proches après transcodage ou autres pour mesurer la qualité mais dans notre cas, via le nœud intermédiaire, cela est plus compliqué. Cependant [Net'03] a défini une simplification pour ce paramètre et l'a spécifié à 3 pour les images dites complexes et 2 pour les images plus simples. [Spi'06] présente une comparaison entre MPQM et MDI.
Enfin, une *sonde SSIM* (Structural Similarity) [Wan'04] a été implémentée. SSIM a été définie pour identifier la similarité structurelle entre 2 images, notamment pour évaluer la qualité de transcodeurs. Ici, nous mesurons la similarité entre images successives pour évaluer la variation d'une image à l'autre. Cette valeur seule peut ne pas être représentative car les images peuvent être différentes dans la séquence vidéo. Seulement, ici nous couplons la valeur du SSIM avec la valeur du MDI pour évaluer la QoE. Les deux sondes sont donc utilisées en association.
Pour chacun de ces modules, la comparaison des valeurs calculées en temps réel avec des seuils permet de savoir si la qualité est satisfaisante ou pas. Par exemple, si le taux de MLR est supérieure à 0.004., si la valeur du taux de perte de paquets est supérieur à 0,5% ou la valeur de délai est supérieure à 50ms ou le MPQM est inférieure à 3.8 [Agi'06] alors on considère que la qualité n'est pas bonne.
Ces modules évaluent la qualité de l'image qui a été reconstruite grâce à FFMPEG.
Finalement, sur les pairs, l'application *Mplayer* ou *VLC* peut être utilisée pour visualiser les flux multimédias.

La requête émise par le pair désirant visualiser un contenu vidéo vers l'entité de gestion P2P, contenant les informations du contexte local de l'utilisateur peut être transmise selon différentes manières : cela peut être une requête HTTP de type GET, incluant le contexte au format CC/PP, ou bien de type SDPng sur une signalisation SIP ou bien encore simplement à l'aide de socket destinée directement à l'entité de gestion du réseau P2P avec les informations de contexte définies selon un mode textuel…. Dans l'implémentation mise en œuvre, cette requête est réalisée à l'aide de *sockets* réseau.

La requête d'adaptation issue de l'entité de gestion vers les pairs adaptateurs, peut être implémentée de plusieurs façons et que ce soit une socket entre les 2 entités, une méthode classique RPC (Remote Procedure Call) de type RMI, Corba ou encore des requêtes SOAP (Simple Object Access Protocol) importe peu. Il faut simplement que cette méthode prenne comme paramètres quel type d'adaptation est à réaliser (par exemple Transcodage de MPEG2 vers H264, ou suppression de données hiérarchiques de niveau 3 ou adaptation de la taille de CIF en QCIF …) et vers quel pair client (adresse IP). Dans l'implémentation réalisée, cette interface se base sur les sockets réseau.

6.3.2 Information de contexte

La description a montré que la solution est valide pour une complétude d'information de contexte (locale et/ou récupérée depuis des bases externes).
A titre d'exemple, nous pouvons citer :

- le Système autonome (AS) auquel l'utilisateur est rattaché (par exemple Orange France, Free…)
- la zone dans laquelle il se situe (cette notion de zone pouvant être variable et pouvant être une région, un pays, la zone couverte par un POP ADSL…)
- le réseau d'accès utilisé par l'utilisateur (ADSL2+ , FTTH, Wifi, UMTS…)
- Le type de terminal de l'utilisateur
- La langue préférée de l'utilisateur
- la qualité du flux souhaitée par l'utilisateur ou à laquelle il a souscrit en cas de service payant par exemple (débit vidéo à 2 Mbit/s, 1 Mbit/s, 512 kbits/s… ainsi que la taille de l'image de type CIF, QCIF…)
- le nombre de clients que le pair peut fournir; cette valeur pouvant être une valeur issue d'études statistiques en fonction du débit du flux multimédia et du type de réseaux d'accès du pair serveur par exemple.
- …

Il est aussi possible d'imaginer de rajouter des informations de contexte plus dynamiques tels que:

- le délai entre les pairs (par exemple via des mécanismes de détection utilisant des nœuds repères) [Lan'04]
- l'estimation en temps réel de la bande passante; par exemple avec des mécanismes tels que STAB (Spatio-Temporal Available Bandwidth Estimation) [Rib'04] [Ribe'04] ou pathLoad [Jai'02] …

- la charge courante du terminal
- …

Dans le démonstrateur réalisé, les informations de contexte "*AS, région, réseau d'accès, terminal, qualité*" ont été utilisées pour la sélection des pairs.

6.3.3 Algorithme de sélection des pairs

Dans ce démonstrateur, réalisé en tant que preuve de faisabilité, le mécanisme de sélection des pairs, instancié sur l'entité de gestion P2P, est simple puisqu'il se charge de trouver les pairs ayant des caractéristiques semblables au pair demandeur en priorisant certains attributs:

- On cherche d'abord les pairs qui sont disponibles dans le même AS (opérateur) ayant un réseau d'accès équivalent ou supérieur.
- Ensuite parmi les pairs disponibles, on recherche ceux qui sont dans la même région ou on étend aux régions voisines si aucun pair n'est trouvé.
- Puis on choisit les pairs qui visualisent le flux dans le même format (qualité) pour n'avoir pas de processus d'adaptation à réaliser. Si aucun n'est trouvé, on choisit un pair avec des capacités suffisantes pour effectuer le traitement d'adaptation.

6.3.4 Fonctionnalités des pairs

Dans ce démonstrateur, l'idée n'est pas de faire une application complète déployable immédiatement, mais d'illustrer le concept. Ainsi, toutes les fonctionnalités que doit avoir une telle solution ne sont pas toutes implémentées, mais seulement celles qui sont nécessaires pour le fonctionnement (ou pour aider à comprendre le fonctionnement) et pour les tests d'évaluation.
Voici les fonctions qui ont été implémentées sur les pairs pour la communication avec l'entité de gestion du réseau P2P.

- "server" : cette fonction indique qu'un pair veut émettre un flux multimédia. Les paramètres sont le nom du flux, le flux concerné (nom du fichier local ou interface d'entrée si caméra par exemple…), le format du flux, le débit, et un commentaire éventuel
- "client" : cette fonction est la requête initiale lorsqu'un client veut se connecter au réseau P2P. Le paramètre est le nom du flux qu'il veut recevoir.
- "zap" : cette fonction permet au client de dire qu'il souhaite se connecter à un autre flux. Les paramètres sont le nom du flux actuel et le nom du flux souhaité.
- "backup" : cette fonction permet au client de demander de se connecter à un autre pair (si la qualité reçue est mauvaise par exemple)
- "leave", cette fonction permet au client de dire que le pair se déconnecte du réseau pour le flux concerné (donc n'est plus client) et n'est plus utilisable en relais (fournisseur)

Il y a d'autres fonctions implémentées, mais à usage local uniquement (par exemple "status" qui permet de connaître l'état du pair, "relay" pour savoir vers qui ce pair envoie le flux et de qui il le reçoit, "keep" qui signifie que l'application visuelle n'est pas activée (l'utilisateur ne regarde pas le flux) mais que le nœud sert toujours de relais …

6.4 Tests sur Planetlab

6.4.1 Environnement

Pour les tests de ce démonstrateur, des machines locales ont été utilisées mais aussi des nœuds du réseau PlanetLab pour avoir plus de nœuds à disposition.
Les nœuds PlanetLab ont été choisis pour être répartis sur la planète (des nœuds aux USA, en Europe, en Asie…).

L'architecture finale servant pour les tests est donc la suivante :

- 3 PC locaux
- De 1 à 20 nœuds PlanetLab

La configuration des PC locaux est la suivante :

Machine	CPU	RAM	OS
DELL D420	Intel Core Duo U2500/ 1,20 GHz	1 Go	Linux Kernel 2.6.17 Ubuntu 6.10 Edgy Eft
DELL D620	Intel Core Duo T2500 / 2 GHz	2 Go	Linux Kernel 2.6.17 Ubuntu 6.10 Edgy Eft
DELL D620	Intel Core Duo T2500 / 2 GHz	2 Go	Linux Kernel 2.6.17 Ubuntu 6.10 Edgy Eft

Les 3 machines sont connectées à un Hub 100 Mbit/s et reliées à l'Internet ouvert (pour se connecter aux machines PlanetLab).

Pour 2007, la configuration des nœuds PlanetLab devrait être, selon les recommandations matérielles définies par l'équipe de gestion du réseau PlanetLab:

Machine	CPU	RAM	OS
PlanetLab	Intel Pentium 2.4Ghz AMD 2400+	1 Gbyte	Linux Kernel 2.6

Cependant, de nombreuses machines ont été installées avant et donc n'ont pas les capacités citées (mais on peut estimer qu'elles ont quand même des capacités correctes). De plus et malheureusement, fréquemment, les machines installées ne sont pas conformes aux recommandations et donc peuvent présenter des capacités inférieures.

6.4.2 Tests de Performance

Les tests ont été réalisés avec pour objectif de vérifier que le démonstrateur avait un comportement et des performances correctes par rapport aux autres solutions actuelles. Typiquement, les mesures qui ont été faites concernent :

- Le temps de connexion au réseau P2P (pour un nouveau client) jusqu'à réception du flux vidéo: sans et avec le lancement de l'application de visualisation
- Le temps de recherche du pair fournisseur par l'entité de gestion P2P
- Le temps de zapping pour un client : pour évaluer le temps de basculer d'un flux à un autre
- Le temps de détection de disparition d'un pair fournisseur (celui qui fournit le stream au client) et de connexion à un autre pair actif.
- Le nombre de clients qu'une machine peut fournir, en capacité de traitement

Les autres avantages de la solution tels que le choix des pairs en fonction de leur ISP, en fonction de leur réseau d'accès (et intrinsèquement du nombre de clients qu'ils peuvent fournir), l'adaptation de flux, ne sont pas illustrés dans cette partie, car ils font partie des tests fonctionnels, mais pas de performance.

Temps de connexion au réseau P2P

Dans ce test est mesuré le temps nécessaire pour un nouveau pair pour se connecter au réseau P2P et commencer à recevoir le flux vidéo depuis un pair fournisseur. Ce temps est donc :

$$T_{connexion} = T_{sig\ avec\ Tracker} + T_{sig\ avec\ pair\ fournisseur}$$

Plusieurs tests ont été réalisés suivant la localisation des pairs.

Client à	Tracker à	Serveur à	Tconnexion
Lannion	Lannion	Lannion	100-120 ms
Lannion	Boston	Lannion	280-330 ms
Lannion	Boston	Londres	550-600 ms

Lorsque le client est à Lannion et le tracker à Boston :

$$T_{sig\ avec\ Tracker} = 120\text{-}130\ ms$$

A ceci, pour un perçu utilisateur, il convient de prendre en compte le temps de lancement de l'application vidéo.
Sur les PC du testbed, ce temps de lancement a été mesuré :

Application	Temps de lancement
MPlayer	1 – 1,5 sec
VLC	2,5 – 3,5 sec

Temps de recherche du pair fournisseur

Temps de connexion en fonction du nombre d'entrées dans le tracker (peers: serveur ou clients/relai)
Dans ce test il est mesuré le temps nécessaire pour l'entité de gestion du réseau P2P de trouver un pair fournisseur à la réception d'une requête client.
Dans ce test les tables de pairs du tracker sont remplies "artificiellement" (ce ne sont pas des nœuds réels) avec différentes caractéristiques.
Le temps indiqué est calculé entre le temps de réception de la requête du client et l'émission de la réponse.

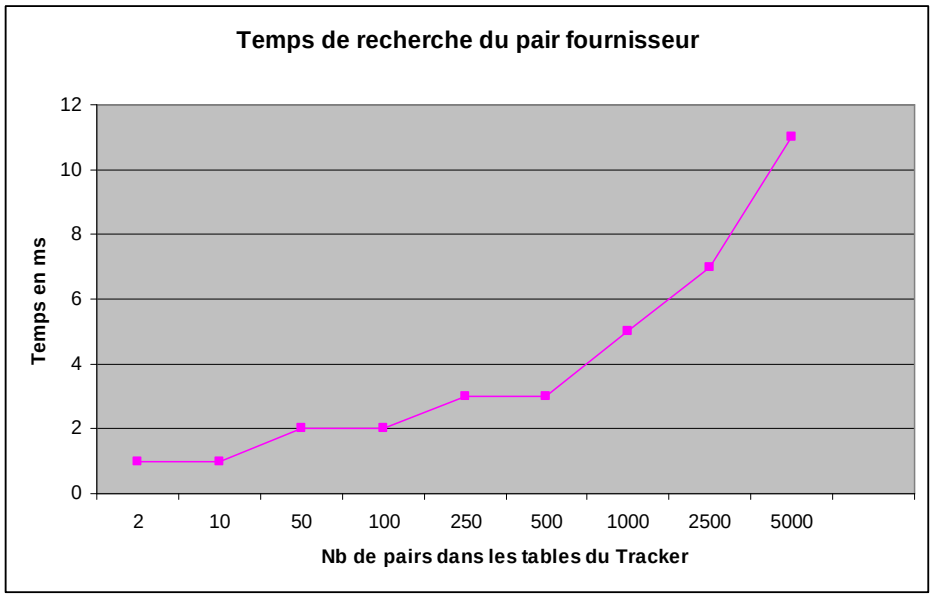

Figure 6. 8: Temps de recherche du pair fournisseur

Ces deux tests font ressortir des résultats intéressants, le temps très court de réception du flux vidéo à la connexion au réseau P2P (c'est le lancement de l'application vidéo qui nécessite le plus de temps).
En effet, le choix d'avoir une entité centralisée, seule responsable du choix des pairs, facilite le traitement de décision car toutes les informations sont locales. De plus, il n'y a qu'un échange réseau entre le pair client et l'entité de gestion P2P. Ceci est beaucoup plus complexe et couteux en messages de signalisation dans le cas d'une architecture complètement distribuée.
Ces résultats montrent aussi que le tracker répond rapidement aux requêtes utilisateurs, indiquant que l'algorithme de sélection mis en place est satisfaisant. Il faut modérer ces résultats en rappelant que l'algorithme est toutefois assez simpliste et ne prend pas en compte de nombreux critères.
De plus, dans l'implémentation mise en œuvre (se basant sur Peercast), les données reçues des pairs fournisseurs sont stockées dans un fichier circulaire temporaire, spécifique pour le flux. L'application de vidéo vient lire ce fichier directement pour afficher le flux, sans forcement attendre qu'il soit rempli, ce qui explique pourquoi le premier flux est visualisé rapidement.

Temps de zapping

Ce test a pour objectif d'évaluer le temps de zapping pour un client, c'est-à-dire quitter un flux vidéo émis par un serveur et se reconnecter à un autre émis par un autre serveur.
Les temps mesurés sont :

- Le temps entre l'envoi de la requête de zapping du client et la réception de la réponse du Tracker indiquant le nouveau pair à contacter: 250-280 ms
- Le temps entre l'envoi de la requête de zapping du client et la réception des premières données du pair fournisseur 1 situé à Lannion: 360-410 ms
- Le temps entre l'envoi de la requête de zapping du client et la réception des premières données du pair fournisseur 1 situé à Londres: 630-670 ms

Client à	Tracker à	Serveur1 à	Serveur2 à	Tconnexion
Lannion	Boston			250-280 ms
Lannion		Lannion		360-410 ms
Lannion			Londres	630-670 ms

Ce temps de signalisation avec le Tracker localisé à Boston est 2 fois plus important que pour la connexion. Ceci s'explique par le fait que l'implémentation de la fonctionnalité et réalisée par une requête indiquant que le pair souhaite quitter le flux puis une requête de connexion à un flux. Ceci engendre ainsi 2 types de messages, donc 2 Allers/Retours entre le client et le tracker.

Comme dans le cas de la première connexion au réseau P2P, en cas de zapping, le temps de réception du nouveau flux est très court. Ceci est dû à l'entité de gestion centralisée (comme dans le cas précédent) et au fait qu'un nouveau fichier est créé dès que l'action de zapping est confirmée par l'entité de gestion. Ainsi, la lecture du flux peut commencer rapidement, contrairement au cas où les nouvelles données seraient stockées dans le même buffer que les anciennes, nécessitant d'attendre la lecture complète.

Des tests ont été faits pour évaluer si le temps de zapping est influencé par le nombre de pairs actifs sur le réseau P2P (pour voir si la taille des tables de pairs par contexte sur l'entité de gestion P2P influence le temps de traitement, comme dans le cas précédent). Puisque le processus est similaire, les résultats montrent que cela est négligeable et que le temps de zapping est sensiblement toujours le même.
Concernant le temps de zapping, pour comparer les résultats du démonstrateur par rapport à l'existant, les valeurs des applications actuelles telles que PPlive ou Joost sont plutôt de l'ordre de 15-20 secondes (voir plus pour certains), donc le démonstrateur présente des résultats intéressants.

Temps de connexion à un autre pair actif

Ce test visait à évaluer le temps nécessaire pour un qu'un pair client se reconnecte à un autre pair fournisseur lorsque le pair qui le fournissait jusqu'alors quittait le réseau.

Pour ce test, la configuration retenue a été la suivante :

- Le tracker était localisé à Boston
- Le client à Lannion
- Le serveur du flux à Londres
- 1 pair à Helsinki, 1 pair à Lannion et 1 pair à Berlin.

Les nœuds, dits relais, sont des clients du flux et qui vont rediffuser le flux vers le client à Lannion.
A la connexion du client, le tracker choisit le nœud à Helsinki comme pair fournisseur pour le client.
Ensuite, le nœud à Helsinki quitte le réseau, amenant ainsi le tracker à signaler au client de se connecter à un autre pair fournisseur, celui de Lannion dans le 1er cas et celui de Berlin dans le 2ème cas.

Les temps mesurés sont :

- Le temps entre le départ du client relai à Helsinki et l'indication du tracker au client de se connecter à un autre pair : # 180-200 ms
- Le temps entre l'envoi de la connexion du client vers l'autre pair fournisseur (celui de Lannion) et la réception des premières données : # 150-190 ms
- Le temps entre l'envoi de la connexion du client vers l'autre pair fournisseur (celui de Berlin) et la réception des premières données : # 320-350 ms

Client à	Relai1 à	Relai2 à	Relai3 à	Tconnexion
Lannion	Helsinki (départ)			180-200 ms
Lannion		Lannion		150-190 ms
Lannion			Berlin	320-350 ms

Lorsqu'un pair quitte le réseau P2P, un message de type "leave" est envoyé par le pair à l'entité de gestion. Celle-ci a ensuite pour rôle de rediriger les pairs clients, que le pair en partance fournissait, vers d'autres pairs avec des contextes similaires.
Dans ce cas, le temps pour se reconnecter est assez court, puisque tout le traitement est centralisé sur l'entité de gestion et les pairs récepteurs n'ont qu'à se reconnecter sur les nouveaux pairs indiqués par l'entité de gestion, en remplacement du pair en partance. Pour l'utilisateur, cela est imperceptible.

Nombre de pairs clients

Ce test a pour objectif d'évaluer si le fait de relayer un flux vidéo vers plusieurs pairs a une influence sur les capacités du nœud relayeur.
Dans le test réalisé, le nœud relai reçoit le flux vidéo et l'affiche (comme un client ordinaire) et le rediffuse vers x autres pairs (x allant de 1 à 10).
Dans tous les cas, la valeur de CPU oscille entre 22-27 %.

Machine Pair	CPU
DELL D420	22-27 %

Les résultats sont exprimés en intervalle car les valeurs fluctuent pendant les tests. Ceci est dû aux réceptions des données (qui ne suivent pas obligatoirement un débit constant) et au décodage des images (les images étant différentes, la puissance pour les décoder peut différer).
Les résultats indiquent que le nombre de pairs a très peu d'influence sur les capacités du nœud en terme de CPU; le processus utilisant le plus de CPU étant l'application vidéo visualisant le flux. Les flux renvoyés n'ont donc qu'une petite influence sur le nœud, ce qui confirme que la participation d'un nœud dans un réseau P2P de diffusion vidéo est plutôt limitée par la bande passante disponible du nœud que par ses capacités de traitements elles-mêmes.

6.5 Conclusion

Dans ce chapitre a été présentée une solution permettant d'adapter dynamiquement les flux multimédias diffusés sur un réseau P2P où le nœud Potacco est le nœud pair du réseau P2P. Contrairement à un mode client/serveur, en P2P, les nœuds sont à la fois clients (receveurs) des flux multimédias et serveurs (fournisseurs) pour d'autres pairs. Dans cette solution, le rôle des pairs est élargi puisqu'ils peuvent adapter le flux multimédia via des modules d'adaptation activés dynamiquement en fonction du contexte des pairs vers lesquels ils envoient le flux. Pour ce faire, une entité de gestion du réseau P2P est définie et a en charge d'analyser le contexte des pairs (lorsque ceux-ci se connectent au réseau) et de sélectionner les pairs fournisseurs pour ce nouveau pair client en fonction de son contexte. Si une adaptation de contenu est requise, l'entité de gestion commande les pairs fournisseurs pour activer le module d'adaptation adéquat.
Le contexte inclut des informations spécifiques à l'utilisateur et à l'application, mais aussi des informations relatives aux réseaux (type de réseau d'accès, opérateur ou ISP, zone de localisation…). Ainsi, dans cette solution, une organisation du réseau P2P prenant en compte les caractéristiques des réseaux physiques (FTTH, ADSL, UMTS..) est mise en œuvre pour avoir une adéquation des capacités du réseau d'accès et de la qualité fournie aux pairs. L'adaptation dynamique peut être mise en œuvre pour transmettre le flux multimédia entre deux pairs de réseaux ayant des capacités différentes.
Enfin, des modules de QoS/QoE sont instanciés sur les pairs pour évaluer en temps réel la qualité du flux multimédia reçu par les clients. En cas de qualité insuffisante, le pair client va demander à l'entité de gestion de reconfigurer le réseau P2P pour lui fournir une meilleure qualité. Cela peut se faire par le choix d'autres pairs fournisseurs ou par une adaptation de contenu par un pair dans le réseau.
La réalisation d'un démonstrateur prouve la faisabilité de la solution et les évaluations indiquent des résultats satisfaisants pour la recherche de pairs de fournisseurs, pour le changement de pairs en cas de départ du pair fournisseur ou en cas de zapping.

Chapitre 7

Conclusion et Perspectives

Dans cette thèse, l'adaptation de contenu adapté au contexte par un nœud intermédiaire a été étudiée. L'objectif était de définir un nœud permettant l'adaptation de contenu divers dans des environnements différents. Ce nœud a été dénommé nœud polymorphique, et plus précisément Potacco (*nœud **PO**lymorphique **T**ransparent pour l'**A**daptation de **C**ontenu adapté au **CO**ntexte*) car il peut offrir différentes formes vis-à-vis des services et vis-à-vis des architectures réseaux.

Au point de vue des services, le nœud offre un environnement d'exécution dans lequel des composants de services peuvent être dynamiquement déployés, activés ou au contraire, arrêtés et supprimés, en fonction des requis des fournisseurs de service. Cela a été illustré par des démonstrateurs, notamment celui concernant l'insertion de contexte utilisateur suivant deux formats, les deux modules étant activés en parallèle sur le nœud Potacco, et celui du réseau overlay de services où des modules de traitement applicatif déployés sur des nœuds intermédiaires permettent de fournir un service d'IPTV personnalisé. Les modules peuvent aussi être dynamiquement activés en fonction du contexte courant, comme cela a été illustré par le cas d'usage où le nœud Potacco est utilisé en tant que passerelle filaire/sans-fil. Pour connaître le contexte, des collecteurs (des sondes) ont été mis(es) en œuvre aussi bien au niveau réseau (bande passante, connectivité, délai..), qu'au niveau traitement (CPU, mémoire..) et qu'au niveau applicatif (critères de QoE). L'association des informations de contexte locales et distantes permet de configurer les nœuds Potacco pour adapter le contenu. Pour la communication des informations, la coordination des différentes entités composant le Potacco et la gestion des événements qui peuvent se produire, une entité centrale, le Gestionnaire Potacco, a été définie. Cette entité sert d'interface entre les composants applicatifs et les modules de détection de contexte.

Au point de vue des réseaux, ce nœud peut avoir différentes formes selon son utilisation et son emplacement dans les réseaux. Il peut agir en tant qu'élément constitutif d'un réseau physique en tant que nœud transparent, réalisant de l'adaptation de contenu de manière transparente (par rapport aux connexions réseau) vis-à-vis des utilisateurs finaux et des fournisseurs de contenus. Ce mode de fonctionnement transparent est possible par l'utilisation conjointe d'un module d'interception des paquets IP et par la mise à jour dynamique des en-têtes et autres champs de contrôle des paquets IP en fonction de l'adaptation des données applicatives. Ce fonctionnement a été illustré par la mise en place de ce nœud dans une chaine de collecte ADSL pour l'insertion du contexte utilisateur dans des requêtes HTTP. Un autre cas d'usage du Potacco en tant qu'élément physique a été démontré quand ce nœud est utilisé en tant que passerelle sans-fil/filaire et réalisant de l'adaptation dynamique de contenu multimédia en fonction de la bande passante du réseau sans-fil.

Ce nœud polymorphique a été ensuite adapté pour fonctionner dans un réseau overlay, réseau virtuel au dessus des réseaux physiques. Le Potacco devient membre du réseau overlay et voit son champ d'utilisation étendu puisqu'il n'est plus uniquement un nœud physique du réseau mais peut aussi être un terminal utilisateur. Un cas d'usage a été réalisé par l'établissement de réseaux overlays spécifiques à des services, créés sur demande et en fonction des besoins. Le réseau overlay comprend dans ce cas des nœuds Potacco hébergeant des composants de services de base, qui enchainés, fournissent un service global adapté au contexte de l'utilisateur. Là encore, le contexte pouvant varier très rapidement, des sondes d'évaluation et de détection de changement du contexte sont mises en œuvre et qui peuvent déclencher l'adaptation du réseau overlay pour modifier les nœuds impliqués ou les traitements à réaliser. Le démonstrateur mis en place pour illustrer ce fonctionnement est basé sur un service personnalisé d'IP TV. Un autre cas d'usage du Potacco dans un réseau overlay a été présenté lorsque le nœud polymorphique est membre d'un réseau P2P (Peer-to-Peer) de diffusion de streaming multimédia en ayant pour objectif d'adapter dynamiquement le flux multimédia en fonction des contextes des réseaux et des utilisateurs. Dans ce cas, le nœud Potacco réalise des traitements d'adaptation intermédiaire pour les clients qu'il sert, mais est aussi lui-même client d'un autre nœud Potacco et receveur d'un flux adapté. Des sondes applicatives de QoE ont été intégrées dans le nœud pour évaluer la qualité du flux multimédia reçue et perçue par l'utilisateur pour déclencher une réorganisation du réseau de diffusion P2P en cas de qualité insuffisante.
Ce nœud peut ainsi être instancié en tant que nœud de réseau physique et/ou nœud terminal et permettre une coopération entre réseau et terminal, plutôt qu'une séparation des entités, pour offrir une qualité satisfaisant les requis des utilisateurs en offrant un contenu adapté à leur contexte. De la même manière, le Potacco permet l'interaction entre les réseaux et les services pour aboutir à la fourniture de services personnalisés.

Concernant les évolutions et études possibles, ce nœud Potacco pourrait être étendu, notamment dans le cadre de la virtualisation des réseaux. En effet, cette thématique récente est de plus en plus d'actualité et vise à virtualiser les réseaux physiques pour que des opérateurs de réseau virtuels puissent offrir des services différents et une architecture différente de l'opérateur propriétaire de l'infrastructure physique. Le nœud Potacco offre une virtualisation des services, puisque des composants différents peuvent être déployés dans des environnements isolés et n'ayant pas d'interférences entre eux et pourrait être étendu en ajoutant la virtualisation des accès au réseau. Les études menées dans cette thèse sur les réseaux overlays sont ainsi les prémices de ce que pourrait être la virtualisation réseau puisque des réseaux overlays spécifiques sont créés, avec un nommage spécifique et un routage propre par réseau overlay. L'évolution du nœud Potacco pour en faire un nœud supportant la virtualisation des réseaux pourrait consister en l'intégration d'un hyperviseur sur le Potacco et à « redescendre d'un cran » les modules réalisés pour les réseaux overlays pour les appliquer aux réseaux physiques.
La virtualisation physique du Potacco pourrait avoir des impacts sur la qualité de service et notamment, comment l'assurer, en virtualisant les ressources physiques,

en définissant le routage virtuel avec QoS, par chaînage des composants de service ou par routage overlay spécifique… Cela peut demander à être étudié plus en détail.

Le nœud Potacco a été adapté pour fonctionner en réseau overlay, notamment pour être déployé sur des nœuds mobiles dans des réseaux sans-fils. Les réseaux sans-fils sont par nature instables, variables et avec une connectivité intermittente. Des travaux de recherche apparaissent concernant le routage DTN (Delay Tolerant Networking) où les nœuds routent les données, mais sont aussi capables de conserver aussi une copie au cas où le routage ne serait pas possible (pas de connectivité courante) : ce mécanisme est connu sous l'appellation "Store and Forward". Une évolution pourrait consister en l'étude du nœud Potacco comme nœud dans un réseau DTN, où les données pourraient être stockées (et éventuellement adaptées) pour être transmises et où les collecteurs de contexte (principalement réseau dans ce cas-ci) aideraient à la décision de routage.

La solution d'adaptation dynamique de contenus dans un réseau de diffusion P2P, en prenant en compte le contexte des utilisateurs et les réseaux physiques supportant le réseau P2P, présente des avantages intéressants pour les opérateurs réseau pour optimiser les ressources, ainsi que pour les fournisseurs de contenus, puisqu'elle leur permet de cibler une large audience. Une piste d'étude future pourrait être l'analyse précise du trafic de streaming en P2P, sur les différents paramètres de la chaine de distribution (ISP, opérateur, pays, charge du réseau en fonction du jour et de l'heure…) pour pouvoir définir un algorithme optimal pour la sélection des pairs en ayant pour objectif d'offrir une meilleure qualité tout en optimisant les coûts réseau (coût d'utilisation des réseaux d'accès, des coûts de peerings….). De la même manière, cette analyse pourrait amener à des choix de déploiement dynamique de modules d'adaptation spécifiques ou en fonction de critères précis. Ces aspects ne sont pas pris en compte par les fournisseurs de telles solutions, car ils n'ont pas la maitrise sur les réseaux physiques, mais un opérateur pourrait le réaliser.

References

[Abr'05] H. Abramovicz, al, “AN Framework Architecture”, IST-2002-507134-AN/WP1-D05, Ambient Network Project, 2005.

[Agh'03] P. Aghera, A. Dixit, R. Oliveira, V. Samanta, "Wireless Middleware: Dynamic Video Transcoding", CS211 Project Report, 2003

[Agi'06] Agilent Technologies, "IPTV QoE: Understanding and interpreting MDI values", White paper, 30 Aout 2006

[Ahm'03] T. Ahmed, Diffusion Adaptative des Paquets Vidéo sur IP: Une Approche Cognitive, thèse, Université de Versailles Saint-Quentin en Yvelines, 25 Novembre 2003

[Ahm'06] T Ahmed, I Djama, "Delivering audiovisual content with MPEG-21-enabled cross-layer QoS adaptation", journal of Zhejiang University-Science A, pages 784-793, 2006

[Amb'05] Ambient Networks Work Package 5, D5-3, "SMART: Final Architecture Design", IST-2002-507134-AN/WP5/D03, www.ambient-network.org, 2005

[Amb'07] Ambient Networks Work Package F, DF-2, " Final System Design of SATO and ASI", FP6-CALL4-027662-AN P2/DF2, www.ambient-network.org, 2007

[AN] The Ambient Networks (ANs) Project, http://www.ambient-networks.org

[Anag'02] M. E. Anagnostou1, A. Juhola2, E. D. Sykas1, ""Context Aware Services as a Step to Pervasive Computing", LOBSTER Workshop, Mykonos, Greece, 3-5 October 2002

[And'01] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris “Resilient overlay networks”, Proc. 18th ACM SOSP, Banff, Canada, 21-24 Octobre, 2001.

[Ard'01] S. Ardon, P. Gunningberg, Y. Ismailov, B. Landfeldt, M. Portmann, A. Seneviratne, B. Thai, "Mobile aware server architecture: A distributed proxy architecture for content adaptation", In proc. of INET2001, Stockholm, Sweden, June 2001

[AVC'05] ISO/IEC standard, 14996-10, "Information technology — Coding of audio-visual objects — Part 10: Advanced Video Coding", Second edition, 2005-12-15

[Bell'03] P. Bellavista, A. Corradi, C. Stefanelli, “Application-level QoS Control and Adaptation for Video on Demand”, IEEE Internet Computing, Vol. 7, No. 6, Nov. Déc. 2003.

[Ben'01] I. Ben-Shaul , O. Holder , B. Lavva, "Dynamic Adaptation and Deployment of Distributed Components In Hadas", IEEE Transactions on Software Engineering, v.27 n.9, p.769-787, September 2001

[Bes'02] B.Bessette, R.Salami, R.Lefevre, M.Jelinek, JR.Pukkila, J.Vaino, H.Mikkola, K.Järvinen."The Adaptive Multi-rate Wideband Speech Codec (AMR-WB)". IEEE Trans. On Speech and Audio Processing, Vol.10 n°8, November 2002.

[Bha'98] H. Bharadvaj , A. Joshi , S. Auephanwiriyakul, "An Active Transcoding Proxy to Support Mobile Web Access", Proceedings of the The 17th IEEE Symposium on Reliable Distributed Systems, p.118, 20-23 Octobre 1998

[Bosz'07] L. Böszörmenyi, H. Hellwagner, P. Schojer, " Metadata-driven optimal transcoding in a multimedia proxy", Multimedia Systems, Volume 13, Number 1, Mars 2007

[Bran'06] J. Brandt, L. Wolf, " A Gateway Architecture for Mobile Multimedia Streaming", in Proceedings Eumob Workshop, Alghero, Sardaigne, Italie, 20 Septembre 20, 2006

[Bran'96] C. J. Branden Lambrecht, O. Verscheure, "Perceptual Quality Measure using a Spatio-Temporal Model of the Human Visual System", in Proc. SPIE Vol. 2668, pages 450-461, Mars 1996.

[Buch'04] S. Buchholz, T. Hamann, G. Hübsch, "Comprehensive Structured Context Profiles (CSCP): Design and Experiences", Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, p.43, March 14-17, 2004

[Camp'99] A. T. Campbell, H. G. De Meer, M. E. Kounavis, K. Miki, J. B. Vicente, D. Villela, "A Survey of Programmable Networks", ACM Computer Communications Review, April 1999

[Car'05] A.R. Cardoso, A.Serhrouchni, B. Mathieu, M. Salaün, "Service Deployment in Active Networks Based on a P2P System", Proc MATA (Mobility Aware Technologies and Applications) 2005, Montreal, Canada, 17-19 Octobre 2005

[CCPP1'04] G. Klyne, F. Reynolds, C. Woodrow, H.Ohto, J. Hjelm, M. H. Butler, L. Tran, "Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0", W3C Recommendation, 15 Janvier 2004

[CCPP2'07] C. Kiss, "Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 2.0", W3C Working Draft 30 Avril 2007

[CCPP-Ex'99] H. Ohto, J. Hjelm, "CC/PP Exchange protocol based on HTTP Extension Framework", W3C Note, 24 Juin 1999

[Cha'96] M. Chatel, “Classical versus transparent IP proxies”, RFC 1919, IETF, Mars 1996.

[Chan'05] I.C. Chang, S.W. Hsieh, "ATF: an Adaptive Three-layer Framework for inter-stream synchronization of SMIL multimedia presentations", Journal of Systems and Software, Adaptive Multimedia Computing, Volume 75, Issue 3, Pages 283-303, March 2005

[Coh'99] A. Cohen, S. Rangarajan, N. Singh, "Supporting transparent caching with standard proxy caches", In Proc. of the 4th International Web Caching Workshop, Mars 1999

[Cou'04] S. Coulombe , G. Grassel, “Multimedia adaptation for the multimedia messaging service,” IEEE Commun. Mag., vol. 42, no. 7, pp. 120–126, Juillet 2004.

[CSS2'98] B. Bos, H. W. Lie, C. Lilley, I. Jacobs, "Cascading Style Sheets, level 2 CSS2 Specification", W3C Recommendation, 12 Mai 1998

[Curr'05] K. Curran , S. Annesley, "Transcoding media for bandwidth constrained mobile devices", International Journal of Network Management, vol.15 n.2, p.75-88, 2005

[Dey'00] A.K. Dey, G. Abowd, “Towards a Better Understanding of Context and Context-Awareness.” In Proceedings of CHI workshop on the What, Who, Where, When and How of Context-Awareness, The Hague, Netherlands April, 2000

[Dey'01] A.K. Dey, "Understanding and using context", Personal and Ubiquitous Computing, Special issue on Situated Interaction and Ubiquitous Computing 5, 2001

[DH'76] W. Diffie and M. Helman. New directions in cryptography. IEEE Transactions on Information Society, 22(6), pages 644–654, Novembre 1976.

[DIA] A. Vetro, C. Timmerer, "Text of ISO/IEC 21000-7 FCD – Part 7: Digital Item Adaptation", ISO/IEC JTC 1/SC 29/WG 11/N5845, July 2003, Trondheim, Norway

[Dij'71] E. W. Dijkstra, "*A short introduction to the art of programming* »", pages 67-73, Aout 1971

[Do'04] T.Do, KA. Hua, M. Tantaoui, "P2VoD: Providing Fault Tolerant Video-on-Demand Streaming in Peer-to-Peer Environment.", to appear in the Proc. of the IEEE International Conference on Communications (ICC 2004), Paris, France, 20-24 Juin 2004

[Dro'03] S. Drossopoulou, G. Lagorio, and S. Eisenbach. "Flexible Models for Dynamic Linking", In P. Degano, editor, Proceedings of the 12th European Symposium on Programming (ESOP 2003), volume 2618 of LNCS, pages 38--53. Springer-Verlag, Avril 2003

[Duc'03] Duc A. Tran, Kien A. Hua, Tai T. Do, "ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming.", In Proceedings of IEEE INFOCOM 2003, San Francisco, CA, USA, 30 Mars – 03 Avril 2003

[Fal'03] B. Falchuk, J.Chiang, A.Hafid, Y.-H.Cheng, N.Natarajan, F.J.Lin, H.Cheng "An Open Service Platform for Deploying and Managing Services at Network Edges," in IEEE Conference Open Architectures and Network Programming (OpenArch) 2003, pp. 77-86, San Francisco, California, USA, 4-5 April 2003

[FFMPEG] http://ffmpeg.mplayerhq.hu/in

[For'06] M. Forte, W. Souza, A. Prado, "A content classification and filtering server for the Interne", In Proceedings of the 21st Annual ACM Symposium on Applied Computing, Vol. 2, pages 1166--1171, Dijon, France, 2006.

[Fry'99] M. Fry, A. Ghosh, "Application level active networking", Computer Networks, 31 (7) pages 655-667, 1999

[G727] "G.727 – Modulation par impulsions et codage différentile adaptatif (MICDA) imbriqué à 5,4,3,2 bits par échantillon", norme ITU-T, Genève, 1990

[Gal'00] A. Galis and al "A Flexible IP Active Networks Architecture", IWAN 2000, Tokyo, Japan, 16 – 18 Octobre 2000; FAIN: Future Active IP Network

[Gal'04] A. Galis, S. Denazis, C. Brou, C. Klein (ed) – "Programmable Networks for IP Service Deployment" ISBN 1-58053-745-6; March 2004 - Artech House Books, 46 Gillingham Street, London SW1V 1AH, UK

[Gan'03] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié, "Peer-to-peer membership management for gossip-based protocols.", in IEEE Transactions on Computers, Vol. 52, No. 2, February 2003

[Gig'07] http://www.neteco.com/68306-web-p2p-venice-project-joost.html

[Gil'03] R. Gil, R. García, J. Delgado, "Delivery context negotiated by mobile agents using CC/PP", *in* 'Mobile Agents for Telecom Applications, (MATA'03)', Springer-Verlag, pp. 99-110.

[Glas'01] J. Glasmann, W. Kellerer, H. Muller, "Service Development and Deployment in H.323 and SIP," *iscc*, p. 0378, Sixth IEEE Symposium on Computers and Communications (ISCC'01), 2001

[Gou'04] Y. Gourhant, Y. Carlinet, B. Mathieu, D.E. Meddour, "Retour d'expérience sur différentes applications de réseau actif", In Annales des télécommunications (Annals of telecommunications), Special issue on "Active networks : architecture and flexible applications", Vol. 59, n°5-6, May-June, 2004

[Grill'97] Grill. "Implementation of a scalable audio coder based on AAC Modules and a core model". ISO/IEC JTC1/SC29/WG11 MPEG97/1816. February 97.

[Gros'04] J. Gross J Klaue, H Karl, A Wolisz, "Cross-Layer Optimization of OFDM Transmission Systems for MPEG-4 Video Streaming," Computer Communication, vol. 27, Applications and Services in Wireless Networks, pp. 1044–55, Février 2004

[Guen'05] T. Guenkova-Luy et al., Harmonization of Session and Capability Descriptions between SDPng and MPEG- 21 Digital Item Adaptation, IETF Work-in-progress, draft-guenkova-mmusic-mpeg21-sdpng-00, Feb. 2005

[H264AVC'03] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," IEEE Transaction Circuits Syst. Video Technol., vol. 13, no. 7, pp. 560-576, Jul. 2003.

[Ha'04] S. Ha, I.G.. Chun, "Adaptation of the Internet Product Information for Mobile Clients," In Proc. of International Conf. on Internet Computing, 2004.

[Han'98] R. Han, P. Bhagwat, R. LaMaire, T. Mummert, V. Perret, J. Rubas, "Dynamic adaptation in an image transcoding proxy for mobile WWW browsing. IEEE Personal Communication, 5(6), Dec 1998

[Hash'03] K. Hashimoto, Y. Shibata, "Design of a middleware system for scalable multimedia streaming", in Proceedings of 23rd International Conference on Distributed Computing Systems Workshops, pages 660- 665, 19-22 May 2003

[Hass'03] R. Haas, C. Jeffries, L. Kencl, A. Kind, B. Metzler, R. Pletka, M. Waldvogel, L. Freléchoux, P. Droz, "Creating Advanced Functions on Network Processors: Experience and Perspectives," IEEE Network, Juillet 2003

[Hef'03] M. Hefeeda, A. Habib, B. Botev, D. Xu, B. Bhargava, "PROMISE: Peer-to-Peer Media Streaming Using CollectCast.", In Proc. of ACM Multimedia 2003, pages 45--54, Berkeley, CA, Novembre 2003

[Hei'07] X. Hei, C. Liang, J. Liang, Y. Liu and K. W. Ross, "A Measurement Study of a Large-Scale P2P IPTV System.", to appear in IEEE Transactions on Multimedia, November, 2007, available as Technical Report.

[Held'02] A. Held, S. Buchholz, A. Schill, "Modeling of Context Information for Pervasive Computing Applications", In Proc. of the World Multiconference on Systemics, Cybernetics and Informatics (SCI) 2002, Orlando, USA, 15-18 Juillet 2002

[Hick'98] M. Hicks, P. Kakkar, J. T. Moore, C. A. Gunter, S. Nettles, "PLAN: A Packet Language for Active Networks," in Proceedings of the Third ACM SIGPLAN International Conference on Func tional Programming Languages, pp. 86–93, 1998

[Ho'01] Jiann-Min Ho, Jia-Cheng Hu, P. Steenkiste, "A conference gateway supporting interoperability between SIP and H.323", Proceedings of the ninth ACM international conference on Multimedia, Ottawa, Canada, 2001

[Hutt'05] A. Hutter, P. Amon, G. Panis, E. Delfosse, M. Ransburg, and H. Hellwagner, “Automatic Adaptation of Streaming Multimedia Content in a Dynamic and Distributed Environment”, IEEE Int’l. Conf. on Image Processing 2005, Gênes, Italie, Septembre 2005.

[ICAP'01] ICAP White paper, "Internet Content Adaptation Protocol (ICAP)", Network Appliance, Version 1.01, 30 Juillet 2001

[ICAP'03] J. Elson, A. Cerpa, " Internet Content Adaptation Protocol (ICAP)", IETF RFC 3507, Avril 2003

[Ind'03] J. Indulska, R. Robinsona, A. Rakotinirainy, K. Henricksen, "Experiences in using cc/pp in context-aware systems", In Proceedings of the 4th International Conference on Mobile Data Management (MDM2003), Melbourne, Australia, Janvier 2003

[Jai'02] M. Jain and C. Dovrolis, "Pathload: A measurement tool for end-to-end available bandwidth", In Proceedings of Passive and Active Measurement Workshop (PAM), Fort Collins, CO, Mars 2002

[Jean'05] K. Jean , N. Vardalach, A. Galis," Towards Programmable Context-aware VoIP Services (CaVoIP)", International Conference on Intelligence in Communication Systems (IntellComm) 2005, Montréal, Canada, 17-19 Octobre 2005

[Jia'03] X. Jiang, Y. Dong, D. Xu, B. Bhargava, "GnuStream: a P2P Media Streaming System Prototype.", *IEEE International Conference on Multimedia and Expo* Baltimore, MD, Juillet 2003

[JMF] http://java.sun.com/products/java-media/jmf/

[Kan'04] D. Kanjo, Y. Kawai, K. Tanaka, "A3: framework for user adaptation using xslt", Proceedings of the 13th international World Wide Web conference, Poster Session, New York, NY, USA, 17 Mai 2004

[Kaw'05] V. Kawadia and P. R. Kumar, "A cautionary perspective on cross-layer design", IEEE Wireless Communication Magazine, pages 3- 11, Février 2005

[Kazi'04] Z Kazi-Aoul, I Demeure, JC Moissinac, "Une architecture générique pour la fourniture de services multimédia adaptables - illustration par un scénario", Mobilité & Ubiquité 2004, Nice, France, 1-3 Juin 2004

[Kel'02] R. Keller, L. Ruf, A. Guindehi, B. Plattner, "PromethOS: A Dynamically Extensible Router Architecture Supporting Explicit Routing", Proceedings of the Fourth Annual International Working Conference on Active Networks (IWAN 2002), Zurich, Switzerland , 4-6 Décembre 2002

[Khu'07] A. Khurri, "Performance of Host Identity Protocol on Nokia Internet Tablet", IP Research Group, IETF 68 Prague, 23 Mars 2007

[Kim'03] Y. Kim, B.J. Jeong, J. Chung, C.S. Hwang, J.S. Ryu, K.H. Kim, Y.K. Kim, Beyound 3G: vision, requirements, and enabling technologies,” IEEE Communications Magazine, pages 120-124, Mars 2003

[Kov'04] B. Kövesi, D. Massaloux and A. Sollaud, "A scalable speech and audio coding scheme with continuous bitrate flexibility", ICASSP2004, Montréal, May 2004

[Kum'95] V. Kumar, “Mbone: Interactive Multimedia on the Internet”, livre ISBN 156205937, 1995

[Lag'06] J. Laganier, L. Eggert, "Host Identity Protocol (HIP) Rendezvous Extension", draft-ietf-hip-rvs-05, 7 Juin 2006

[Lan'04] S. van Langen, X. Zhou, P. van Mieghem, "On the estimation of Internet distances using landmarks", In Proc. of the International Conference on Next Generation Teletrafic and Wired/Wireless Advanced Networking, NEW2AN'04, St.-Petersburg, Russia, 2 – 6 Février, 2004

[Lee'03] Y.W. Lee, G. Chandranmenon, S.C. Miller, "GAMMA: A Content-adaptation Server for Wireless Multimedia Applications", Bell Laboratories.

[Lee'06] I. W. Lee, H-J. Park, K-R Park, Y. Choi, S-H. Kim, "A scheme using OPES for real-time translation services in digital home networks based on delivery management systems", 8th International Conference Advanced Communication Technology (ICACT) 2006, Phoenix Park, Korea, 20-22 Février 2006

[Leg'05] A. Legout, G. Urvoy-Keller, and P. Michiardi, "Understanding Bit-Torrent: An Experimental Perspective.", Tech. Report inria-00000156, INRIA, Sophia Antipolis, November 2005

[Lem'03] T. Lemlouma, N. Layaïda, "SMIL Content Adaptation for Embedded Devices", SMIL Europe 2003, Paris, 12-14 Février, 2003.

[Lem'04] T. Lemlouma, "Architecture de Négociation et d'Adaptation de Services Multimédia dans des Environnements Hétérogènes", thèse, Institut National Polytechnique de Grenoble, 09 juin 2004

[Leml'04] T. Lemlouma, N. Layaïda, Context-aware adaptation for mobile devices. Proceedings of the IEEE International conference on Mobile Data Management, Berkeley, CA, USA, 19-22 Janvier 2004

[Leml'03] T. Lemlouma , N. Layaïda, "Adapted Content Delivery for Different Contexts", Proceedings of the 2003 Symposium on Applications and the Internet, p.190, 27-31 Janvier 2003

[Leo'04] K. Leopold, D. Jannach, H. Hellwagner, "A Knowledge and Component Based Multimedia Adaptation Framework", in Proceedings of the IEEE Sixth International Symposium on Multimedia Software Engineering (ISMSE'04), Miami, FL, USA, 13-15 Décembre 2004

[Li'03] Z. Li, P. Mohapatra, "QRON: QoS-aware Routing in Overlay Networks", IEEE JSAC, 2003.

[Liu'06] Y. Liu, X. Hei, C. Liang and K. W. ROSS. "Insight into pplive: A measurement study of a largescale p2p iptv system". 2006.

[Live555] http://www.live555.com/liveMedia/dex.html

[Lua'04] E.K. Lua, J. Crowcroft, M. Pias, R. Sharma, S. Lim, "A survey and comparison of Peer-to-Peer overlay networks schemes", IEEE Communications Survey, March 2004

[Mah'02] A. Maheshwari, A. Sharma, K. Ramamritham, P. Shenoy, "TranSquid: Transcoding and caching proxy for heterogenous e-commerce environments", In Proc. 12th IEEE Workshop on Research Issues in Data Engineering (RIDE '02), San Jose, CA, Février 2002

[Mat'04] B. Mathieu, Y. Gourhant, Y. Carlinet, "A Programmable Network enabling Content Adaptation", Proc MATA (Mobility Aware Technologies and Applications) 2004, Florianopolis, Brazil, October 20-22, 2004

[Mat'07] B. Mathieu, "Réseaux overlays de services pour les réseaux ambiants", chapitre du livre "Des réseaux intelligents à la nouvelle génération de services", sous la direction de N. Simoni, Hermes Science Books, ISBN 978-2-7462-1218-3, pp292, Février 2007

[Math'05] B. Mathieu, Y. Carlinet, D. Massaloux, B. Kövesi, D. Deleam, "A Network-Aware Truncating Module for Scalable Streams Saving Bandwidth for Overused Networks", Proc MATA (Mobility Aware Technologies and Applications) 2005, Montreal, Canada, October 17-19, 2005

[Math'07] B. Mathieu, "A Context-Aware Network Equipment for Dynamic Adaptation of Multimedia Services in Wireless Networks", special issue "Serving and Managing Users in a Heterogeneous B3G Wireless World: Requirements, New Research Challenges, Emerging Solutions" of the "Wireless Personal Communications" Springer Journal, Mars 2007

[Mathi'07] B. Mathieu, "A Context and Network Aware Adaptive P2P Video Streaming Solution", 21ème congrès DNAC, Paris, France, 15-16 Novembre 2007

[Mathie'07] B. Mathieu, A. Galis, K. Jean, R. Ocampo, Z. Lai, M. Stiemerling, M. Kampmann, M. A. Tariq, M. Cano Soveri, K. Balos, K Ahmed, B. Busropan, M. Prins, Teemu Rinta-aho, "Dynamic Adaptable Overlay Networks for Personalised Service Delivery", M2NM 2007, Sydney, Australia, October 16-19, 2007

[Mod'06] R. Moskowitz, P. Nikander, "Host Identity Protocol (HIP) Architecture", RFC 4423, Mai 2006

[Mok'05] A. Mokhtari, "ARFANet : Une nouvelle approche pour les réseaux actifs", thèse, Université de Versailles Saint-Quentin en Yvelines, 30 Septembre 2005

[Mor'04] R. G. Moro, R.de Matos Galante, C.A. Heuser, "A version model for supporting adaptation of web pages", in Proceedings of the 6th annual ACM international workshop on Web information and data management 2004, Washington DC, USA, 12-13 Novembre 2004

[MPEG] http://www.chiariglione.org/mpeg/

[MPEG-21] I. S. Burnett, F. Pereira, R. Van de Walle, R. Koenen, "The MPEG-21 Book", Wiley editions, ISBN 0-470-01011-8, 2006

[Nej'02] W. Nejdl , B. Wolf , C. Qu , S. Decker , M. Sintek , A. Naeve , M. Nilsson , M. Palmér , T. Risch, "EDUTELLA: a P2P networking infrastructure based on RDF", Proceedings of the 11th international conference on World Wide Web, Honolulu, Hawaii, USA, 07-11 Mai, 2002

[Neo'03] R. Neogi, K. Lee, K. Panesar, J. Zhou.,"Design and performance of a network-processor-based intelligent DSLAM", IEEE Network, Juillet 2003

[Net'03] NetPredict, Inc., "Performance Analysis for Video Streams across Networks", White Paper, Décembre 2003

[Nguy'03] H.-B. Nguyen, A. DUDA, "ProAN : an Active Node for Proactive Services in Pervasive Environments", in procedings of the 2nd International Workshop on Active Network Technologies and Applications (ANTA 2003), Osaka, Japan, May 2003,

[Nguy'04] H.-B. Nguyen, "Services Actifs et Passerelles Programmable", thèse Institut National Polytechnique de Grenoble, 16 janvier 2004

[Nguye'03] H.-B. Nguyen, A. DUDA, " GateScript : A Scripting Language for Generic Active Gateways", poster Fifth Annual International Working Conference on Active Networks (IWAN2003), Kyoto, Japan, 10-12 Décembre 2003

[Nie'04] N. Niebert, A. Schieder, H. Abramowicz, G. Malmgren, J. Sachs, U. Horn, C. Prehofer, H. Karl, "Ambient Networks: An Architecture for Communication Networks Beyond 3G", IEEE Wireless Communications, Volume: 11, Issue: 2 pages 14- 22, Avril 2004.

[Ocam'05] R. Ocampo, L. Cheng, Z. Lai, A. Galis, "ContextWare Support for Network and Service Composition and Self-adaptation", International Workshop on Mobility Aware Technologies and Applications (MATA), Montréal, Canada, 17-19 Octobre 2005

[OCPCore] A. Rousskov, " Open Pluggable Edge Services (OPES) Callout Protocol (OCP) Core", IETF RFC 4037, Mars 2005

[OCPHTTP] A. Rousskov, M. Stecher, " HTTP Adaptation with Open Pluggable Edge Services (OPES)", IETF RFC 4236, Novembre 2005

[OCPReq] A. Beck, M. Hofmann, H. Orman, R. Penno, A. Terzis, "Requirements for Open Pluggable Edge Services (OPES) Callout Protocols", IETF RFC 3836, Aout 2004

[OCPSMTP] M. Stecher, A. Barbir, "Open Pluggable Edge Services (OPES) SMTP Use Cases", IETF RFC 4496, Mai 2006

[Ohm'05] J. Ohm, "Advances in scalable video coding," *Proc. IEEE*, vol. 93, no. 1, pp. 42-56, Janvier 2005.

[OPES] A. Barbir, R. Penno, R. Chen, M. Hofmann, H. Orman, "An Architecture for Open Pluggable Edge Services (OPES)", IETF RFC 3835, Aout 2004

[OSGI] OSGI Alliance : www.osgi.org

[OSGI_Spec] OSGi Alliance, " OSGi Service Platform Core Specification", Release 4, Revision 4.1, Avril 2007

[OSGI_WP]OSGi Alliance, "About the OSGi Service Platform", Technical White Paper, Revision 4.1, 7 Juin 2007

[Pai'03] V. Pai, A. Cox, V. Pai, W. Zwaenepoel, ' A Flexible and Efficient Application Programming Interface (API) for a Customizable Proxy Cache', in USENIX Symposium on Internet Technologies and Systems, Seattle, WA, March 2003.

[Pap'02] C. Papachristos, E. Markatos, "A CC/PP Aware Apache Web Server", 7th CabertNet Radicals Workshop, Bologna, Italy, 13-16 Oct 2002

[Pas'99] J. Pascoe, N.S. Ryan, D.R. Morse, "Issues in Developing Context-Aware Computing", in Proceedings of the International Symposium on Handheld and Ubiquitous Computing, pages 208-221, Karlsruhe, Germany, Septembre 1999

[Pia'06] F. Pianese, J. Keller, and E. W. Biersack, "PULSE, a Flexible P2P Live Streaming System", in Proc. of 9th IEEE Global Internet Symposium 2006, Barcelona, Spain, 28 & 29 April 2006

[Rat'01] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable and Content-Adressable Network",. In Proc. ACM SIGCOMM, San Diego, CA, USA, Aout2001

[Raz'00] D. Raz, Y. Shavitt, "Active Networks for Efficient Distributed Network Management'', IEEE Communications Magazine, 38(3, pages 138--143, Mars 2000.

[Raz'99] D. Raz, Y. Shavitt, "An Active Network Approach for Efficient Network Management'', IWAN'99, Berlin, Germany , Juillet 1999

[Rej'03] R. Rejaie, A. Ortega, "PALS: Peer-to-Peer Adaptive Layered Streaming.", in Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video, Monterey, California, Juin 2003

[Rey'05] J. Rey, B. Mathieu, D. Lozano, S. Herborn, K. Ahmed, S. Schmid, S. Goebbels, F. Hartung, M. Kampmann, "Media Aware Overlay Routing in Ambient Networks", Proc. PIMRC 2005, Berlin, Germany, September 2005

[Rib'04] V. Ribeiro, R. Riedi, R. Baraniuk, "Locating Available Bandwidth Bottlenecks", IEEE Internet Computing, pp. 34-41, September-October 2004

[Ribe'04] V. Ribeiro, R. Riedi, R. Baraniuk, "Spatio-temporal available bandwidth estimation with STAB", ACM SIGMETRICS Performance Evaluation Review Volume 32, Issue 1, Juin 2004

[Rip'01] M. Ripeanu, "*Peer-to-Peer Architecture Case Study: Gnutella Network",* 1st International Conference on Peer-to-Peer Computing (P2P'01), Linköpings, Suède, 27-29 août 2001

[Rod'01] P. Rodriguez, S. Sibal, O. Spatscheck, "TPOT: Translucent Proxying of TCP, Computer Communication", 24(2), pages 249-255, 2001

[Row'01] A. Rowstron, P. Druschel, « Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems », Proc. of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001). Heidelberg, Germany, Novembre 2001.

[Schm'00] S. Schmid, "LARA++ design specification", Technical report MPG-00-03, Computing Department, Lancaster University, Lancaster, UK, 2000

[Scho'06] P. Schojer, L. Böszörmenyi, and H. Hellwagner, QBIX-G – A Transcoding Multimedia Proxy, Technical Reports of the Institute of Information Technology, University Klagenfurt, TR/ITEC/05/2.11, accepted for Multimedia Computing and Networking 2006

[Schu'05] C; Schuba, M. Hefeeda, J. Goldschmidt, M. Speer, "Scaling Network Services Using Programmable Network Devices", IEEE Computer, Avril 2005

[SDP] M. Handley, V. Jacobson, "SDP: Session Description Protocol", IETF RFC 2327, Avril 1998

[SDPng] Kutscher, Ott, Bormann, "Session Description and Capability Negotiation", draft-ietf-mmusic-sdpng-08.txt, 20 Février 2005

[SDPngTrans] J. Ott/C. Perkins, SDPng Transition, draft-ietf-mmusic-sdpng-trans-04.txt, 15 Mai 2003

[SHA1'93] "Secure Hash Standard", United States of American, National Institute of Science and Technology, Federal Information Processing Standard (FIPS) 180-1, April 1993.

[SHA1'01] D. Eastlake, P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, September 2001

[Shah'03] N. Shah,W. Plishker, K. Keutzer "NP-Click: A programming model for the Intel IXP1200", In Network Processor Design, vol 2., Novembre 2003.

[Shan'02] Y. Shan and A. Zakhor, "Cross Layer Techniques for Adaptive Video Streaming over Wireless Networks," IEEE ICME, vol. 1, pp. 277-80, 2002

[Shin'05] Y.S. Shin, J.P. Hong, T.I. Kim, K.S. Cho, " Design and implementation of open pluggable edge services", 7th International Conference Advanced Communication Technology (ICACT) 2005, Phoenix Park, Korea, 21-23 Février 2005,

[Sik'05] T. Sikora, “Trends and perspectives in image and video coding”, Proceedings of the IEEE, vol. 93, no. 1, pp. 6–17, 2005.

[Sil'06] T. Silverston and O. Fourmaux, “P2P iptv measurement: A comparison study.”, http://www.arxiv.org/abs/cs.NI/0610133, 2006.

[Singh'00] K. Singh, H. Schulzrinne, "Interworking between SIP/SDP and H.323", In Proceedings of the 1st IPTelephony Workshop (IPtel 2000), Berlin, Germany, Avril 2000.

[SIP] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "SIP: Session Initiation Protocol". RFC 3261 (Proposed Standard), Juin 2002. Mis-a-jour par RFCs 3265, 3853.

[SIPP2P] D. Bryan, P. Matthews, E. Shim, and D. Willis, "Concepts and Terminology for Peer to Peer SIP", draft-ietf-p2psip-concepts-01, Novembre 2007

[SMIL2.1] D. Bulterman, G. Grassel, J. Jansen, A. Koivisto, N. Layaïda, T. Michel, S. Mullender, D. Zucker, "Synchronized Multimedia Integration Language (SMIL 2.1)", W3C Recommendation, 13 Décembre 2005

[SMIL2] Jeff Ayars et al., "Synchronized Multimedia Integration Language (SMIL 2.0) Specification - [Second Edition]", W3C Recommendation, 07 Janvier 2005

[Sol'02] M. Solarski, M. Bossardt, T. Becker, “Component-Based Deployment and Management of Services in Active Networks”, IWAN 2002, Zurich, Switzerland, Dec. 2002

[Spi'06] Spirent Communications, "MPQM vs. Media Delivery Index, Toward a comparison framework for delivered video quality metrics", white Paper, Fevrier 2006

[Stee'04] R. Steele, M. Lubonski, Y. Ventsov, E. Lawrence,"Accessing SMIL-based Dynamically Adaptable Multimedia Presentations from Mobile Devices", Proceeding of the International Conference on Information Technology: Coding and Computing, ITCC04, 2004

[Sto'01] I. Stoica, R. Morris, D. Karger, M. Kaashoek, H. Balakrishnan, “Chord: A scalable peer-to-peer lookup service for internet applications”, In Proceedings ACM SIGCOMM 2001.

[Sub'04] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz, "OverQoS: An Overlay Based Architecture for Enhancing Internet QoS", Proc. 1st NSDI, San Francisco, CA, Mars 2004.

[Sur'01] L. Suryanarayana and J. Hjelm, “CC/PP for Content Negotiation and Contextualization,” in Mobile Data Management: Second International Conference, MDM 2001, vol. 1987 of LNCS, p. 239, Springer Verlag, Jan. 2001.

[Tad'99] H.Taddei, D.Massaloux, A.Le Guyader. "A Scalable Three Bitrate (8, 14.2 and 24 kbit/s) Audio Coder". The 107th Convention AES, New York, Septembre 1999.

[Tan'03] C. Tang , Z. Xu , M. Mahalingam, "pSearch: information retrieval in structured overlays", ACM SIGCOMM Computer Communication Review, v.33 n.1, p.89-94, Janvier 2003

[Tenn'96] D. Tennenhouse, D. Wetherall – "Towards an active network architecture" Computer Communications Review, 26, 2, pages 5-18, 1996

[Tenn'97] D.L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, G. J. Minden, "A survey of active network research", IEEE Communications Magazine, 35(1), pages 80-86, Janvier 1997.

[TINA] TINA-C Spécifications : http://www.tinac.com/specifications/specifications.htm

[Tou'05] J. D.Touch, Y.-S. Wang, V. Pingali, L. Eggert, R. Zhou, G. Finn "A Global X-Bone for network Experiments", First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities (TRIDENTCOM'05), Trento, Italie, 2005.

[UAProf] WAP Forum, "WAG UAProf - Wireless Application Protocol - WAP-248-UAPROF-20011020-a" , 20-Oct-2001

[VdS'05] M. van der Schaar, S. Shankar N., "Cross-layer wireless multimedia transmission: challenges, principles, and new paradigms," IEEE Wireless Communications, pp. 50—58, Aout 2005

[Vetro'05] A. Vetro and C. Timmerer, "Digital Item Adaptation: Overview of Standardization and Research Activities", IEEE Transaction on Multimedia, vol. 7, no. 3, Juin 2005.

[Vit'05] A. Vitali, M. Fumagalli, "Standard-compatible Multiple-Description Coding (MDC) and Layered Coding (LC) of Audio/Video Streams", Internet Draft-Network Working Group. Juillet 2005.

[Wan'04] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," IEEE Transactions on Image Processing, vol. 13, no. 4, pages 600-612, Avril 2004.

[Wet'98] D. J. Wetherall, J. Guttag, D. L. Tennenhouse, "ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols". In IEEE OPENARCH'98, San Francisco, CA, USA, Avril 1998

[Wie'03] T. Wiegand, G.J. Sullivan, G. Bjntegaard, A. Luthra, "Overview of the H.264/AVC video coding standard", Special Issue on the H.264/AVC video coding standard, IEEE Trans. on Circuits and Systems for Video Technology, Vol. 13, number 7, pp 557- 559, July 2003

[Witt'96] R. Wittig, P. Chow, "OneChip: An FPGA Processor with Reconfigurable Logic", IEEE Symposium on FPGAs for Custont Computing Machines, 1996.

[WML2'02] WAP Forum, "Wireless Markup Language Version 2 – Wireless Application Protocol WAP-238-WML-20010626-p", 26 Juin 2001

[WML'99] WAP Forum, "WAP WLM - Wireless Application Protocol Wireless Markup Language Specification Version 1.1", 3 Fevrier 1999

[Wolf'04] I. Wolf, B. Feiten, Guenkova-Luy, A. Shoor, F. Hauck, AJ.J Kassler, "MPEG-21 DIA based delivery using SDNgp and RTP", MPEG2004/M10996, Redmond, Washington, USA, Juillet 2004

[XHTML'02] S. Pemberton and al., "XHTML 1.0 The Extensible HyperText Markup Language (Second Edition), A Reformulation of HTML 4 in XML 1.0", W3C Recommendation 26 Janvier 2000, revised 1 Aout 2002

[XHTMLBasic'07] M. Baker, M. Ishikawa, S. Matsui, P. Stark, T. Wugofski, T. Yamakami, "XHTML Basic 1.1", W3C Candidate Recommendation, 13 Juillet 2007

[Xie'02] R. Xie, J. Liu, X. Wang, Zhejiang , " Efficient MPEG-2 to MPEG-4 compressed video transcoding ", Image Processing. 2002, San Jose, USA, 21-23 Janvier 2002

[XML'06] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fourth Edition)" W3C Recommendation, 16 Aout 2006

[XPath'07] A. Berglund (XSL WG), S. Boag D. Chamberlin, M. F. Fernández M. Kay, J. Robie, J. Siméon, "XML Path Language (XPath) 2.0", W3C Recommendation, 23 Janvier 2007

[XSLT'07] M. Kay, "XSL Transformations (XSLT) Version 2.0", W3C Recommendation, 23 Janvier 2007

[Xu'06] M. Xu, J. Li, L.-T. Chia, Y. Hu, B.-S. Lee, D. Rajan, J. S. Jin, "Event on demand with MPEG-21 video adaptation system", in Proceedings of the 14th ACM International Conference on Multimedia, 23-27 Octobre, 2006, Santa Barbara, CA, USA

[Xyn'04] S. Xynogalas, M. Chantzara, I. Sygkouna, S. Vrontis, S. Roussaki, M. Anagnostou, "Context Management for the Provision of Adaptive Services to Roaming Users. IEEE Wireless Communications, Vol. 11. No. 2. pages 40-47, Avril 2004

[Yasu'01] K. Yasuda, T. Asada, T. Hagino, "Effects and Performance of Content Negotiation Based on CC/PP", Second conference, MDM 2001, Hong Kong, China, January 2001

[You'06] B Yousef, DB Hoang, G Rogers, " Serviter: A service-oriented programmable network platform for shared infrastructure", Computer Communications, Elsevier, Volume 29, Issue 5, Pages 642-659, Mars 2006

[Zha'04] B. Y. Zhao et al., « Tapestry: A Resilient Global-Scale Overlay for Service Deployment », IEEE Journal on Selected Areas in. Communications (JSAC), Vol 22, No 1, Janvier 2004