ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2012-ENST-021

EDITE ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

Télécom ParisTech

présentée et soutenue publiquement par

Houda ALAOUI SOULIMANI

le 18 juin 2012

Pilotage Dynamique de la Qualité de Service de Bout en Bout pour une Session «User Centric»

Directeur de thèse : **Noëmie SIMONI**

Jury

Mme. Michelle SIBILLA, Professeur, Université Paul SEBATIER	Président du Jury
M. Ghislain DU CHENE, Directeur Recherche&Développement, SFR	Examinateur
Mme. Nadia BOUKHATEM, Professeur, Télécom ParisTech	Examinateur
Mme. Dominique GAITI, Professeur, Université de technologie de Troyes	Rapporteur
M. Jean-Pierre CLAUDE, Professeur, Université de Versailles	Rapporteur
Mme. Evelyne LE GALL, Responsable Ingénierie Réseaux Cœur /Voix, SFR	Invité

T H È S E

Télécom ParisTech
Ecole de l'Institut Télécom – membre de ParisTech
46, rue Barrault – 75634 Paris Cedex 13 – Tél. + 33 (0)1 45 81 77 77 – www.telecom-paristech.fr

Résumé

Aujourd'hui, le marché des services est devenu de plus en plus concurrentiel. Les exigences des clients pour des offres de service en adéquation avec leurs usages et leurs préférences conduisent les fournisseurs à proposer de nouveaux services qui répondent à ce nouveau besoin pour se démarquer des concurrents et attirer de nouveaux clients.

Avec la convergence des réseaux et celle des services de nouvelle génération (NGN/NGS), de nouveaux services sont apparus. Les utilisateurs sont nomades et veulent utiliser leurs services de différentes manières n'importe où, n'importe quand et par n'importe quel type de terminal, et cela avec une continuité de service et une qualité de service de bout en bout. Ainsi, fournir des services personnalisés aux clients dans un environnement hétérogène et mobile devient un challenge pour les opérateurs et les fournisseurs de service pour améliorer le retour sur investissement (ROI) et le délai de mise sur le marché (TTM).

Nos réflexions à propos de la fourniture des services personnalisés selon les besoins fonctionnels et non-fonctionnels (QoS) des usagers, nous ont conduits à identifier les besoins du nouveau contexte NGN/NGS défini par l'intersection de ces trois éléments «user-centric, mobilité et QoS». Comment piloter dynamiquement la QoS de bout en bout pour une session unique «user-centric»? Comment assurer le « service Delivery» dans un contexte de mobilité et d'ubiquité? Ces nouveaux besoins, nous ont motivé à proposer des solutions à travers trois contributions principales qui prennent en considération la vision utilisateur et opérateur.

Notre première contribution porte sur le modèle organisationnel. Nous proposons une nouvelle organisation avec un maximum de flexibilité, d'adaptabilité et d'autogestion, qui permet de piloter la QoS à chaque niveau de l'architecture (équipement, réseau et service). Dans cette organisation nous avons défini des acteurs et le rôle que joue chacun d'eux par rapport à la prise de décision au cours de la session de l'utilisateur, et cela pour maintenir la QoS de bout en bout dans un environnement qui est totalement hétérogène et mobile.

Notre deuxième contribution traite du composant de service autonomique. Avec la complexité de la personnalisation des services dans un contexte hétérogène et mobile et le besoin de satisfaire la QoS de bout en bout, les ressources services doivent être prises en compte au même titre que les ressources réseaux. Donc, un degré élevé d'autosuffisance, d'autogestion et d'automatisation est demandé dans la ressource service (composant de service) pour améliorer le service delivery. Pour cela, nous proposons un composant de service autonomique «ASC: Autonomic Service Component» basé sur un agent de QoS intégré qui s'autocontrôle et s'autogère pour adapter dynamiquement ses ressources en réponse à un changement de situations au cours de la session de l'utilisateur.

Notre troisième proposition couvre le modèle protocolaire. La session de services personnalisés nécessite des interactions plus flexibles au niveau service pour avoir une session unique avec une continuité de service. Nous proposons un protocole de signalisation SIP+ qui permet la négociation de la QOS des services personnalisés dès la phase d'initialisation de la session et de la renégociation de la QoS pendant l'usage, pour maintenir le service avec la QoS requise à travers une session unique.

De façon plus concrète, nous présentons nos expérimentations à travers un scenario et une plate-forme de démonstration qui nous permet de tester la faisabilité et la performance de nos contributions. Les apports et les perspectives de cette thèse sont consignés en conclusion.

Tables des matières

Table des figures

Liste des Abréviations

NGN	Next Generation Network
NGS	Next Generation Service
ROI	Return On Investment
TTM	Time to Market
UBIS	User-Centric uBiquité et Intégration de Services
SOA	Service Oriented Architecture
SIP	Session Initiation Protocol
NLR	Nœud Lien Réseau
HO	Handover Horizontal
HV	Handover Verticale
ARB	Autonomic Ressource Broker
QB	QoS Broker
MOSE	Model-Based Self-adaptation of SOA System
3GPP	3rd Generation Partnership Project
TISPAN	Telecoms & Internet converged Services & Protocols for Advanced Networks
IMS	IP Multimedia Subsystem
GSM	Global System for Mobile communications
GPRS	General Packet Radio Service
LTE	Long Term Evolution
WLAN	Wireless Local Area Network

xDSL	x Digital Subscriber Line
UMTS	Universal Mobile Telecommunications Systems
RTC	Réseau téléphonique commuté
ATM	Asynchronous Transfer Mode
MPLS	Multiprotocol Label Switching
RSVP	Resource Reservation Setup Protocol
I-CSCF	Interrogation-Call Session Control Function
S-CSCF	Serving- Call Session Control Function
P-CSCF	Proxy- Call Session Control Function
HSS	Home Subscriber Server
SGSN	Serving GPRS Support Node
GGSN	Gateway GPRS Support Node
PCRF	Policy charging and rules function
PCEF	Policy and Charging Enfoncement Function
GTP	GPRS Tunnelling Protocol
NASS	Network Attachment Subsystem
RACS	Resource and Admission Control
NAT	Network address translation
BGF	Border Gateway Function
C-BGF	Core Border Gateway Function
I-BGF	Interconnection Border Gateway Function
AF	Application Function
SPDF	Service-based Policy Decision Function
A-RACF	Access - Resource and Admission Control Function
CN	Core Network

PS	Packet Switched
CS	Circuit Switched
TINA	Telecommunication Information Networking Architecture
AS	Access Session
SS	Service Session
SC	Switched session
RTP	Real Time Protocol
RTCP	Real Time Control Protocol
NM	Mobile Node
NC	Correspondent Node
SDP	Session Description Protocol
VPxN	Virtual Private (x=Service, User, Connectivity, Equipment) Network
VSC	Virtual Service Community
VQC	Virtual Queue Community
MSC	Managment Service Component
BSC	Basic Service Component
ASC	Autonomic Service Component
IT	Information Technology
E2E	End-to-End
PC	Personal Computer
PDA	Personal Digital Assistent
QoS	Quality of Service
VOD	Vidéo à la demande
SLA	Service Level Agreement

Chapitre I Introduction Générale

I.1 Contexte

Aujourd'hui, avec l'avènement des réseaux et services de nouvelle génération (NGN/NGS), l'approche «User-Centric» a évolué. Les besoins des utilisateurs sont toujours en croissance parce qu'ils deviennent de plus en plus nomades et ils désirent l'accès à leur services sans aucune barrière technique, temporelle, économique ou géographique. Dans de nombreuses situations, les besoins d'un utilisateur nomade ne peuvent pas être satisfaits par un seul service, il faut donc lui offrir la possibilité de personnaliser ses services selon ses besoins fonctionnels, non-fonctionnels (QoS) et ses préférences. Les préférences d'un utilisateur sont liées aux choix du terminal (PDA, PC, etc.), du réseau d'accès (WI-FI, 3G, Ethernet), du service (VoD, WEB, Email) et aussi au prix qu'il va payer.

Dans de nombreuses situations, l'utilisateur veut accéder à ses services personnalisés proposés par différents fournisseurs de services lorsqu'il change de terminal ou de réseaux d'accès. Nous distinguons une mobilité spatiale qui inclut la mobilité de l'utilisateur, la mobilité du terminal et la mobilité de service d'une mobilité temporelle qui se traduit par un changement dans la session de bout-en-bout en raison de la mobilité spatiale. Un des défis majeur imposé par la mobilité spatiale auquel les fournisseurs de services doivent faire face aujourd'hui est d'offrir, d'une façon continue et conformément aux exigences de l'utilisateur et de ses préférences, ses services personnalisés, en dépit de l'hétérogénéité des terminaux, des réseaux d'accès, et des plates-formes de services.

Ce nouvel environnement hétérogène et mobile de l'utilisateur qui est devenu plus complexe, impacte l'architecture en générale et la session en particulier. La session, qui est une mise en relation temporelle entre l'utilisateur durant tout son déplacement et l'ensemble du système depuis son terminal jusqu'à la plate-forme de service, en passant par les composants des réseaux d'accès et du réseau cœur, doit donc faire face à ce nouveau contexte, entre autre à la

mobilité, pour garantir à l'utilisateur un service personnalisé avec une qualité de service de bout en bout qui doit être parfaitement adaptée à toute variation dans son contexte ambiant.

I.2 Problématiques

Face aux besoins imposés par ce nouveau contexte (user-centric, NGN/NGS et mobilité), il nous faut repenser les architectures de services d'aujourd'hui pour avoir une organisation avec un maximum d'autosuffisance, d'automatisation et d'autogestion, afin de satisfaire la QoS de bout en bout. Actuellement, il existe un ensemble de mécanismes utilisés au niveau des réseaux d'accès et de transport pour surveiller et contrôler les ressources réseaux, ce qui répond aux exigences spécifiques de la QoS des applications au niveau réseau afin d'assurer le transport d'un flux media «Media Delivery» d'un point A à un point B. Par contre si le point A est changé en raison de la mobilité du terminal ou bien de l'utilisateur cela peut provoquer une éventuelle interruption de la délivrance du service, la QoS du réseau doit être redéfinie et celle des services applicatifs doit être prise en compte. Les solutions de gestion et de contrôle de QoS qui opèrent uniquement sur les ressources réseaux ne sont plus suffisantes car la QoS perçue par un utilisateur peut être affectée par d'autres facteurs, notamment le comportement des applications dans les plates-formes de service. Pour cela, Il faut améliorer la prestation de service «Service Delivery», pour fournir à l'utilisateur des services personnalisés selon ses exigences et ses préférences de la façon la plus transparente possible.

En effet, Il faut prendre en considération la gestion de la QoS au niveau applicatif de l'architecture pour lui assurer une continuité de service lorsque l'un des points d'extrémité A ou B change, c'est à dire quand on change de plate-forme de services ou de terminal, ou encore de lieu.

Nous identifions un premier verrou : Quelle organisation et donc quelle coopération pour contrôler et gérer la QoS de bout en bout? Qui pourrait piloter cette organisation de façon efficace? Un modèle centralisé pénaliserait-il les temps de réponse? Quel modèle organisationnel pour avoir le maximum d'autosuffisance, d'automatisation et d'autogestion ?

Grâce à l'architecture Service Oriented Architecture (SOA), les fournisseurs de service gagnent en termes de flexibilité car elle permet une création rapide des applications en se

basant sur la composition des composants de service qui sont déjà existants. Aujourd'hui, la plupart des applications sont fournies aux consommateurs exactement de la même manière. Cependant, les besoins et les préférences des utilisateurs varient selon leur contexte parce qu'ils deviennent nomades et les applications telles qu'elles sont proposées aujourd'hui sont trop monolithiques.

Nous devons répondre à ce deuxième verrou: Comment offrir aux consommateurs un service personnalisé en fonction de leurs besoins (fonctionnels, non-fonctionnels(QoS)), leurs préférences et leur contexte ambiant? Comment un composant de service peut améliorer le service delivery? Comment repenser les composants de service pour avoir une bonne composition et une meilleure performance de QoS?

Durant l'usage, la session de l'utilisateur doit être maintenue pour lui permettre d'accéder à ses services personnalisés n'importe où, n'importe quand et par n'importe quel terminal. Elle doit être adaptée aux changements des préférences de l'utilisateur lors de ses déplacements pour lui assurer une continuité de service et une QoS de bout-en-bout.

C'est là que se situe notre troisième verrou à lever : Quel protocole de signalisation permettrait la création et le maintien de la session de service dans un contexte hétérogène et mobile ? Comment assurer la continuité de service Anywhere, Anytime and Anyhow, quel que soit le prestataire de service concerné? Comment assurer la QoS de bout en bout d'une session mobile?

Pour cela, il nous faut un protocole de signalisation qui prend en considération la négociation de la QoS au niveau service pendant la phase de la création de la session de service, et la renégociation de la QoS pendant la phase d'exploitation pour assurer la continuité de service d'une session continue et mobile.

I.3 Contributions de la thèse.

L'objectif principal des travaux de cette thèse est de pouvoir lever les verrous que nous venons d'identifier. Le périmètre de cette thèse se situe autour d'un ensemble de propositions concernant le modèle organisationnel (Qui fait Quoi?) (§I.3.1), l'acteur de la dynamicité (§I.3.2) et le modèle protocolaire (§I.3.3) pour assurer un pilotage dynamique de la qualité de service de bout-en-bout d'une session «user-centric».

I.3.1 Modèle organisationnel pour le contexte NGN/NGS

Les besoins de notre contexte NGN/NGS nous conduit à résoudre les problèmes des architectures contemporaines qui nécessitent plus de dynamicité, flexibilité et réactivité en temps réel. Nous devons automatiser et contrôler la gestion de la QoS dans chaque composant de service, impliqué dans le «Service Delivery», afin de garantir le contrat SLA établi entre les clients et les fournisseurs de service au cours de la session de l'utilisateur.

Nous répondons au premier verrou par notre première contribution qui traite le modèle organisationnel «Qui fait quoi?» et qui a pour objectif le pilotage dynamique de la QoS de bout en bout. Dans ce modèle nous définissons les acteurs et le rôle que joue chacun d'eux par rapport à la prise de décisions lors des différents changements de QoS, qui se produisent au cours de la session de l'utilisateur et qui peuvent être liés à la mobilité ou à un dysfonctionnement d'un service.

I.3.2 Agent de QoS pour un pilotage dynamique

Ce modèle organisationnel s'appuie sur notre deuxième contribution qui propose un agent de QoS intégré dans chaque composant de service de l'architecture. Il joue le rôle d'un pilote organisationnel car il vérifie en temps réel pendant l'exécution la conformité du contrat de QoS, et il réagit dynamiquement et d'une manière autonome lorsqu'un changement dans le contrat de QoS se produit au cours de la session de l'utilisateur. L'agent QoS offre un contrôle et une gestion distribués de la QoS, et il joue un rôle important dans le pilotage dynamique de

la QoS de bout en bout car il contribue au maintien de service demandé par l'utilisateur au cours d'une session mobile.

Le composant de service est autonomique grâce à son agent de QoS qui le rend à la fois autonome et auto-gérable. Cette caractéristique favorise une composition de service plus flexible dans un contexte hétérogène et mobile de l'utilisateur. Elle permet le changement, d'une manière transparente, d'un composant de service par un autre composant ubiquitaire en cas de dégradation de la QoS au cours de la session.

I.3.3 Modèle protocolaire pour une session «user-centric»

Une session «user-centric» est une session qui se veut mobile, unique et sans couture dans un contexte, qui est complètement hétérogène et mobile (utilisateur, terminal, réseau et service). L'établissement, la modification et la maintenance de cette session sont des points importants pour répondre aux besoins fonctionnels et non-fonctionnels (QoS) de l'utilisateur ainsi qu'à ses préférences.

Les procédures existantes permettent la négociation de la QoS media entre les participants, via le protocole de signalisation SIP, pour assurer le «Media Delivery» d'une seule application pendant la session de l'utilisateur. Ce protocole permet la création d'une session pour chaque application que l'utilisateur demande. Mais le besoin de l'utilisateur impose de plus en plus la nécessité d'avoir accès à tous les services qu'ils demandent, à travers une seule et même session. Pour cela, il faut étendre la négociation de la QoS au niveau service de l'architecture pour lui garantir le «Service Delivery» de son service personnalisé, qui est composé de plusieurs services, dans son environnement ambiant.

C'est pourquoi, notre troisième contribution, qui répond à notre 3ème verrou, vise à proposer une modification de la portée du protocole SIP existant nommé SIP+, et cela pour créer et maintenir ce type de session qui nécessite de fortes interactions plus flexibles qui agissent sur la couche de service, par analogie à celles de la couche réseau. Ce protocole servira à la négociation de la QoS des services à la phase d'initialisation de la session, et également à la renégociation de la QoS lors de la phase d'exploitation, pour maintenir la session et résoudre

les problèmes de dégradation de la QoS qui surgissent au cours de la session, et qui sont dus à la mobilité ou bien aux changements de préférences de l'utilisateur.

I.4 Organisation du rapport

Pour répondre aux différents objectifs de cette thèse, le présent rapport décrit nos recherches bibliographiques, nos travaux et nos résultats à travers sept chapitres:

Le Chapitre I introduit d'une manière générale et synthétique le contexte, les problématiques et le périmètre d'étude de cette thèse. Après avoir décrit les besoins imposés par le nouveau contexte «user-centric», des verrous ont été identifiés pour faire face aux différentes problématiques existantes. Nous présentons d'une manière succincte les différentes solutions qui constituent les contributions de cette thèse.

Le Chapitre II présente le contexte de nos travaux de recherche qui définit les limites de notre périmètre d'étude, ayant pour objectifs principaux, d'offrir à l'utilisateur une session unique et sans coupure correspondant à son usage, et de maintenir le «Service Delivery» avec une QoS de bout en bout.

Le Chapitre III analyse des travaux existants qui traitent des problématiques des domaines qui font partie de notre périmètre de thèse, à savoir: les architectures autonomiques, la prise en compte de la qualité de service et la session. Ces travaux nous ont servi à déterminer les limitations des solutions actuelles, sur lesquelles nous nous sommes basées pour proposer des solutions qui prennent en considération les besoins du nouveau contexte.

Le Chapitre IV consigne quelques travaux effectués par notre groupe de travail, parce que la thèse s'inscrit dans le cadre du projet UBIS, et repose sur un certain nombre de concepts qui nous ont servi d'outils de base pour nos différentes propositions. Dans cette partie, nous présentons l'architecture UBIS structurée en quatre niveaux de visibilités (utilisateur, équipement, connectivité et service), le modèle «NLR» utilisé pour modéliser ces niveaux de visibilité et le modèle informationnel qui décrit les différents profils de la base de connaissance.

Le Chapitre V consigne l'ensemble de nos propositions qui lèvent les verrous identifiés et répondent aux besoins et problématiques décrits précédemment. La première proposition (§V.1) est relative à la dimension organisationnelle, où nous proposons une nouvelle organisation de l'architecture UBIS offrant un pilotage distribué de la QoS avec le maximum de flexibilité et de dynamicité, pour une continuité de service dans un environnement ubiquitaire et mobile. La seconde proposition est relative au composant de service autonomique (§V.2), cette partie traite comment repenser les composants de services pour qu'ils soient le plus autonome, le plus auto-gérable et le plus auto-adaptable possible lors de la session de l'utilisateur, et cela pour répondre à la QoS globale demandée par l'utilisateur (§V.3). La dernière proposition concerne la dimension protocolaire qui traite principalement les différentes interactions de signalisation de la session UBIS qui sont guidées par les besoins et les préférences de l'utilisateur.

Le Chapitre VI démontre la faisabilité et la performance de nos propositions à travers un scénario qui illustre un cas d'usage, et une plate-forme UBIS composée d'un réseau de transports virtuels (Routeurs VIRTUOR), d'une architecture de contrôle Open IMS Core, d'un serveur d'application (Glassfish V3/SailFin) et d'une base de données ORACLE.

Le dernier chapitre présente une conclusion de ce rapport de thèse et consigne nos perspectives de recherche.

Chapitre II Le contexte

II.1 Introduction

L'orientation qui nous intéresse dans cette thèse est celle d'une approche « User-Centric » dans un contexte NGN/ NGS (mobile et hétérogène). L'utilisateur d'aujourd'hui est nomade. Il souhaite avoir la possibilité de se connecter à ses services personnalisés sans coupure et sans couture avec une meilleure performance de QoS et cela n'importe où, n'importe quand et par n'importe quel type de terminal. La convergence des réseaux NGN favorise la mobilité de l'utilisateur parce qu'elle lui permet d'être facilement connecté à ses services pendant ses déplacements, par n'importe quel réseau de sa préférence dont il possède les droits d'accès et à partir de n'importe quel terminal. La convergence des services NGS permet à l'utilisateur d'avoir une composition des services (IT, Web, Telecom) fournis par différents fournisseurs de service. Dans ce nouveau paysage, la délivrance des services personnalisés à l'utilisateur est étroitement liée à une session unique et continue pendant laquelle le contrat SLA établi entre l'utilisateur et le fournisseur de service doit être maintenu durant la mobilité et les changements de préférences de l'utilisateur.

Nous présentons dans ce chapitre notre contexte qui est défini par l'intersection des trois domaines : User-Centric (§II.1.1), Mobilité (§II.1.2), et Qualité de service de bout en bout (§II.1.3). Notre périmètre (§II.1.4) d'étude qui a pour objectif principal une session unique sans couture et sans coupure, correspond à l'usage de l'utilisateur, pour qui on veut garantir une QoS de bout en bout et assurer le « service Delivery ».

II.1.1 User-centric

Au cours des années, le contexte technologique a évolué du «Hardware-centric» vers l'«user-centric» tout en passant par l'«application-centric» et le «network-centric». Auparavant, la création et l'usage des services étaient fortement liés aux capacités de traitement Hardware «Hardware-centric», ou bien aux infrastructures de transport réseaux qui conditionnent toutes

demandes de services. Actuellement, la plupart des solutions qui existent pour répondre aux besoins des utilisateurs sont centrés sur l'application «application-centric» dans une architecture client-serveur.

Mais de nos jours, l'utilisateur est devenu le point central de l'architecture. L'utilisateur «user-centric » a des exigences en termes de personnalisation de services, mais pour satisfaire ce nouveau besoin, les architectures contemporaines client-serveur qui fournissent des services à options ne sont plus suffisantes. Il nous faut repenser les architectures d'aujourd'hui et les mécanismes de gestion de la QoS. Il nous faut une architecture avec plus de dynamicité et de flexibilité pour répondre aux préférences de l'utilisateur et à son nouveau contexte qui est totalement hétérogène et mobile.

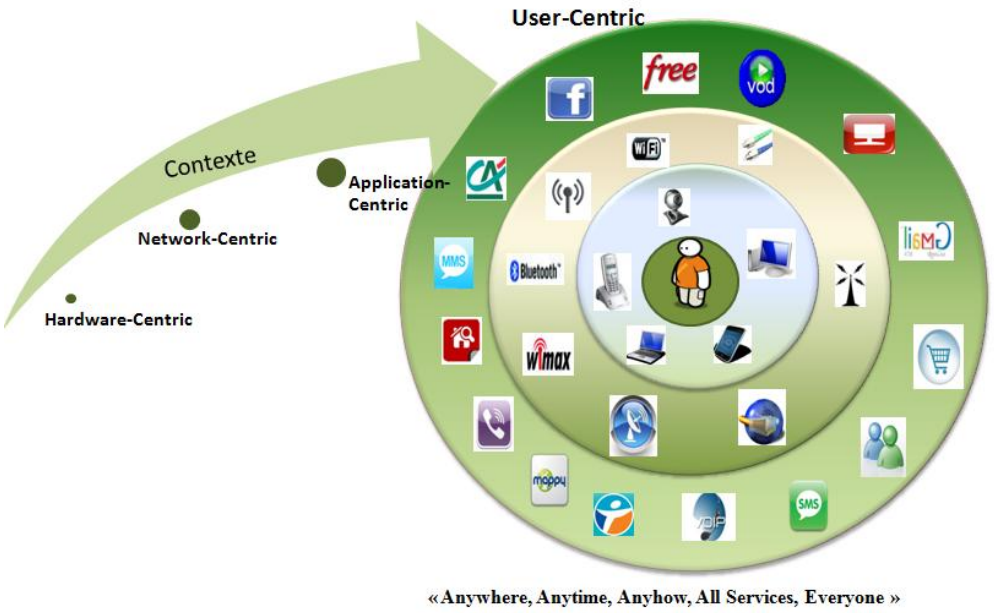

Figure 1: Contexte User-centric

II.1.2 Mobilité

Dans le contexte de nouvelle génération des réseaux et services, la mobilité offre aux utilisateurs la possibilité de se déplacer et d'accéder à leurs services au gré de leurs besoins et de leurs préférences. La localisation ne représente plus une contrainte pour les utilisateurs, ils peuvent accéder à leur services n'importe où et cela grâce à une grande couverture et à la diversité des réseaux d'accès. Dans le contexte NGN/NGS, l'hétérogénéité des terminaux, des

réseaux et des services est omniprésente. Elle impacte la continuité de service durant la mobilité et elle rend la continuité de session de plus en plus complexe. Dans ce qui suit, nous analysons les différents types de mobilité qui peuvent survenir au cours de la session de l'utilisateur : la mobilité de l'utilisateur, la mobilité du terminal, et la mobilité du service qui représentent les mobilités spatiales et la mobilité de la session qui est temporelle.

a) Mobilité de l'utilisateur

Les utilisateurs d'aujourd'hui disposent de plusieurs terminaux (PC, PDA, tablette,…). La mobilité de l'utilisateur signifie la capacité d'un utilisateur à changer de terminal au cours de la session tout en gardant ses services personnalisés. Durant ce changement, il faut appliquer les adaptations nécessaires pour assurer la continuité de ses services sur son nouveau terminal utilisé.

b) Mobilité du terminal

La mobilité du terminal se rapporte à un terminal qui change de localisation. Le terminal se déplace à travers différents réseaux d'accès ou bien à travers différents points d'accès d'un réseau tout en assurant la continuité de service. Deux types de solutions existent qui permettent le déplacement de terminal sans coupure de la session: le Handover et le Roaming.

Le Handover est une technique qui permet à un terminal mobile de changer son point d'attachement d'une manière transparente tout en maintenant la communication. Ce passage entre deux points d'attachement peut entrainer, dans certains cas, une déconnexion temporaire du terminal ou bien une dégradation de qualité des communications au cours de la session. Les solutions de Handover existantes proposent le Handover horizontal (HHO) et le handover verticale (VHO).

Le Handover horizontale offre à un terminal la possibilité de changer le domaine d'accès, dans un même niveau architectural, tout en gardant la même technologie d'accès. Quant au Handover verticale, il permet le changement de réseau d'accès (par exemple de la 3G au Wifi). Le traitement de ce type de Handover se fait dans plusieurs niveaux architecturaux à cause de l'hétérogénéité des environnements.

Le Roaming décrit la possibilité d'appeler ou d'être appelé quelle que soit la localisation. Ce principe est souvent utilisé lorsqu'un utilisateur se trouve dans un pays étranger. Il permet au terminal d'un abonné de passer, selon les accords bilatéraux, d'un réseau d'accès mobile à un autre réseau d'un opérateur étranger. Ce changement peut entrainer des perturbations dans l'accès au service et provoquer une interruption de service pour l'utilisateur.

c) Mobilité de service

La mobilité de service offre à l'utilisateur une continuité de ses services indépendamment du terminal, du réseau d'accès ou bien de sa localisation. Dans ce cas, il existe plusieurs services, soit ubiquitaires, soit dupliqués, qui sont déployés dans différentes plates-formes et qui sont offerts par différents fournisseurs. Lors de la demande de l'utilisateur, le service sera sélectionné en fonction de la stratégie du fournisseur.

Dans nos propositions, le choix du service se fera suivant le besoin fonctionnel et non-fonctionnel (QoS) et selon les préférences de l'utilisateur. On a plutôt choisi des services ubiquitaires qui vont nous servir à assurer le remplacement d'un service dégradé par un autre équivalent pour maintenir le contrat de QoS de bout en bout pendant les différentes mobilités.

d) Mobilité de session

La session de l'utilisateur doit faire face à tous ces types de mobilité (utilisateur, terminal, et service) pour assurer la continuité de service avec une qualité de service de bout en bout. Lorsque l'utilisateur change de préférences, du réseau d'accès ou bien de terminal, c'est la session qui fait la mise en relation des nouveaux éléments. C'est cette mise en relation qui définit la mobilité de session.

Dans nos propositions nous avons des mécanismes qui font que les changements soient transparents pour l'utilisateur et qu'ils n'impactent pas la fourniture du service. La mobilité de session sera un support pour la continuité de service.

II.1.3 Qualité de service de bout en bout

L'approche «qualité de service de bout en bout» se définit comme étant le degré de satisfaction de l'utilisateur et elle couvre l'ensemble de la chaine depuis le terminal jusqu'à la plate-forme de service. La qualité de service est souvent liée au transport des données dans le réseau (media delivery), beaucoup d'autres facteurs sont à prendre en considération pour satisfaire le bout en bout comme la surcharge ou bien la panne des plates-formes de service. Ces problèmes, qui sont liés à la plate-forme de service, sont aussi pénalisants que la défaillance au niveau du réseau de transport, et sont la cause de la mauvaise qualité de service perçue par l'utilisateur.

La gestion de la QoS, et donc sa garantie, ne peut en aucun cas se limiter à une partie de la chaine de bout en bout ou bien à une technologie. Elle doit être traitée dans sa totalité, parce que la QoS finale perçue par l'utilisateur est impactée par l'élément le moins performant de la chaine de distribution de service.

II.1.4 Périmètre de la thèse

L'analyse du contexte a mis en exergue les exigences de l'utilisateur « user-centric » en termes de mobilité (utilisateur, terminal, réseau et service) et de QoS de bout en bout, qui nous dirigent vers une session mobile qui gère l'ensemble de ses services qu'il a personnalisé. Ces services, qui sont fournis par différents fournisseurs, doivent être accessibles à tout moment par n'importe quel terminal à travers la meilleure connectivité.

Dans ce contexte se pose, d'une part, le problème de la continuité et de l'unicité de la session mobile de l'utilisateur qui lui permet l'accès à l'ensemble de ses services personnalisés, et d'autre part, le problème du «Service Delivery» pour garantir une qualité de service de bout en bout. L'intérêt du «Service Delivery» est d'assurer une continuité de service en tenant compte non seulement de la qualité de service du niveau transport, mais aussi de celle du niveau service.

Notre périmètre d'étude (Figure 2) se positionne à l'intersection des trois domaines, que nous avons décrits précédemment (user-centric, mobilité et qualité de service de bout en bout), pour garantir le «service Delivery» et assurer la continuité et l'unicité de la session mobile afin de répondre aux besoins des utilisateurs:

- Au niveau de l'organisation, il nous faut une architecture décentralisée avec plus de flexibilité et de dynamicité pour permettre à l'utilisateur de faire sa propre composition de service avec des changements dynamiques dans un contexte hétérogène;
- Il nous faut repenser les services applicatifs pour intégrer la gestion de la qualité de service au niveau service de l'architecture, et cela pour améliorer le «Service Delivery»qui tient compte en plus de la gestion de la QoS au niveau du réseau de transport, de la gestion de la QoS au niveau des services applicatifs;
- Et il nous faut aussi un protocole de signalisation qui assure une interaction plus flexible entre les différents acteurs de l'architecture, pour faciliter la création et la modification d'une session mobile afin de garantir la continuité de service.

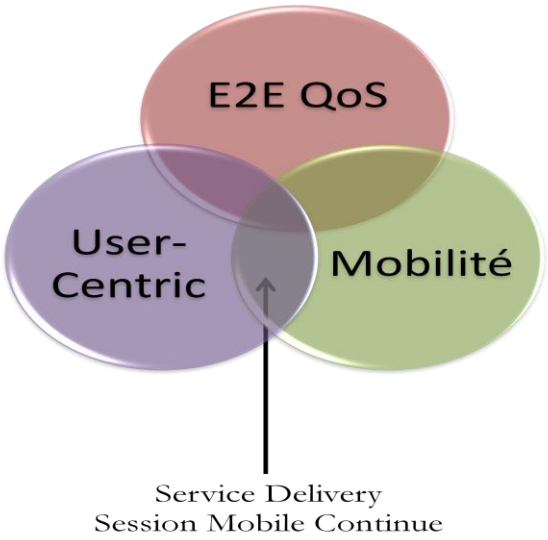

Figure 2:Périmètre d'étude de la thèse

Chapitre III L'état de l'art

III.1 Introduction de l'état de l'art

Dans le chapitre précédent, nous avons présenté les besoins imposés par le nouveau contexte (User-centric, E2E QoS et Mobilité) dans lequel se situe notre recherche. L'objectif de cette thèse est de piloter dynamiquement la qualité de service de bout en bout pour assurer le «Service Delivery» et la continuité d'une session mobile «user-centric». Les besoins imposés par ce nouveau contexte, nous ont conduits à repenser les architectures d'aujourd'hui pour avoir une organisation avec un maximum *d'autosuffisance, d'automatisation et d'autogestion* pour satisfaire le besoin de la *qualité de service*, et également à modifier le protocole de *signalisation SIP* existant pour fournir à l'utilisateur la possibilité d'avoir une session de services personnalisés unique et continue. C'est pourquoi nous étudions d'abord les travaux existants qui traitent des architectures autonomiques (§III.2), puis ceux qui traitent de la QoS (§III.3), et finalement nous nous intéressons à la session (§III.4) afin d'identifier les limites des solutions existantes (§III.5) pour prendre en compte les nouveaux besoins.

III.2 Architectures autonomiques:

Actuellement, ils existent plusieurs solutions qui visent à automatiser les mécanismes de gestion et de traitement dans les systèmes. Dans cette partie, nous analysons tout d'abord les travaux existants sur l'autonomique computing (§III.2.1), ensuite ceux qui traitent les architectures autonomiques (§III.2.2), et finalement nous listons quelques recherches sur les composants de services autonomiques (§III.2.3).

III.2.1 Autonomique Computing

Ces dernières années, plusieurs projets ont traité les problèmes d'autogestion dans les architectures autonomiques. Le premier projet qui a travaillé sur les architectures autonomiques est IBM [1]. IBM a développé des systèmes informatiques auto-gérables qui fonctionnent sans une intervention humaine. La vision d'IBM s'appuie essentiellement sur des applications autonomiques qui s'auto-adaptent en fonction de l'évolution de leur environnent d'exécution pour assurer la délivrance des services IT. Pour cela, une application autonomique doit contenir un élément qui gère l'adaptation de l'application au regard de l'environnement et des événements qui s'y produisent. Ce concept est basé sur un élément clé: le gestionnaire autonomique qui est responsable de la gestion des ressources des applications. Il supervise le comportement des ressources qu'il gère selon les règles prédéfinies par l'administrateur, et il prend les décisions sur d'éventuelles actions à effectuer pour maintenir l'application dans un état acceptable. Le gestionnaire autonomique fonctionne selon une chaine de contrôle représentée dans Figure 3:

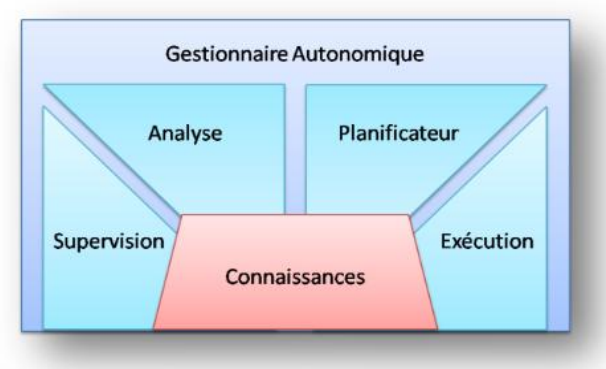

Figure 3: Gestionnaire autonomique D'IBM

- La fonction de gestion**:** Cette fonction fournit les mécanismes qui servent à la collecte des données de surveillance des ressources gérées, et de les agréger. Ces données comprennent des informations sur la configuration, la capacité offerte, le statut. Cette fonction fait l'interface entre l'élément géré et le gestionnaire autonomique. Elle reçoit les informations qui proviennent de différentes ressources, les agrège, les filtre jusqu'à

ce qu'elle détermine un symptôme, relatif à une combinaison d'événements particuliers, qui doit être analysée.

- La fonction d'analyse: elle fournit les mécanismes qui permettent d'observer et d'analyser les situations afin de déterminer si un changement doit être fait. Elle récupère les données fournies par la fonction de gestion et elle les analyse. Dans le cas d'un dysfonctionnement, la fonction d'analyse génère une demande qui décrit les modifications à appliquer et l'envoie à la fonction Planification.

- La fonction de planification: cette fonction a pour rôle la détermination des actions à effectuer en fonctions des problèmes identifiés par la fonction d'analyse. Elle fournit un plan d'action à effectuer pour appliquer les modifications nécessaires à la ressource gérée.

- La fonction d'exécution: elle fournit les mécanismes qui effectuent les changements nécessaires pour la ressource gérée en tenant en considération les mises à jour dynamiques, et cela lorsque la fonction de planification génère un plan d'action qui correspond à un plan de changement. Cette fonction assure le lien entre le gestionnaire autonomique et la ressource gérée.

- La base de connaissance: elle contient les informations nécessaires à la gestion des ressources de l'élément. Elle comprend les données partagées entre les quatre fonctions du gestionnaire autonomique (surveillance, analyse, planification et exécution). Elle contient par exemple des informations sur les politiques de gestion, l'historique d'utilisation de l'élément, l'état de ressource, les mesures et les informations de topologie.

IBM a proposé une architecture autonomique (auto-gérable) avec les propriétés suivantes: auto-configuration (self-configuring), auto-optimisation (self-optimizing), autoréparation (self-healing), and autoprotection (self-protecting):

- Self-configuring: cette propriété représente la capacité d'un système de s'auto-configurer pour s'adapter aux différents changements dans son environnement. Elle rend la configuration et la reconfiguration d'un système et de ses composants

autonome et dynamique. Elle est nécessaire lorsqu'un nouveau composant s'introduit dans l'environnement du système. Ce nouveau composant doit être capable de s'auto-incorporer d'une manière transparente et il doit détecter les règles et les politiques en vigueur. Le système doit s'auto-adapter pour tenir compte de ce nouveau composant dans l'environnement.

- Self-optimizing: un système autonomique doit être capable d'auto-optimiser son fonctionnement dans des environnements imprévisibles, pour que les équipements et les réseaux fonctionnent efficacement sans une intervention humaine. Pour cela, il faut automatiser et évaluer, d'une façon continue, les performances du système. Le système mesure continuellement son rendement et il le compare par rapport à sa performance idéale pour effectuer des stratégies d'amélioration dans son environnement.

- Self-healing: cette propriété définit la capacité à découvrir, analyser des problèmes potentiels dans un système, et de les réparer pour qu'il fonctionne correctement. L'objectif est de prévenir les problèmes et de prendre des mesures proactives, pour éviter les défaillances ou bien pour réduire leur impact.
- Self-protecting: désigne la capacité du système à s'auto-protéger et à s'auto-sécuriser. Elle implique la protection contre les attaques malveillantes, les tentatives d'intrusion et elle interdit les opérations non-autorisées.

IBM décrit une application autonomique par une composition d'éléments autonomiques en interaction. L'objectif est que chacun des éléments du système optimise son fonctionnement pour qu'il s'auto-adapte aux éventuelles évolutions de son environnement d'exécution, ainsi il a la capacité de proposer aux autres éléments de l'environnement de changer leur mode de fonctionnement. IBM préconise la construction des applications autonomiques en fonction des entités autonomes qui sont capables de s'auto-adapter à leur environnement. Cette architecture représente une référence de plate-forme générique à laquelle les développeurs d'applications et de plates-formes autonomiques peuvent se référencer. Cependant l'architecture ne propose pas de solution pour la réalisation d'applications autonomiques basées sur la qualité de service. Il y a aussi un problème, le système autonomique est composé

d'un ensemble d'éléments autonomiques en interaction, par contre les méthodes pour assurer la communication entres les gestionnaires autonomiques ne sont pas encore précisés.

Il existe plusieurs projets qui mettent en place l'architecture proposée par IBM de différentes façons pour simplifier et améliorer la gestion des applications autonomiques:

– Le projet Autonomia développé à l'université d'Arizona [2] qui a pour but de gérer les ressources et les services mis en réseaux. Il fournit aux administrateurs l'ensemble des outils qui leur permettent de spécifier le comportement autonomique d'une ressource. Il propose l'ensemble des propriétés de configuration qui sont nécessaires à l'automatisation du déploiement et de l'activation d'une ressource, il définit les opérations de supervision et les différentes actions à effectuer sur une ressource, et il spécifie aussi les politiques de gestion du comportement interne des ressources, ainsi que les politiques d'interaction entre les autres parties du système.

– Le projet AutoMate [3] de l'université Rutgers a pour objectif de spécifier l'ensemble des technologies clés y compris les modèles de programmation, les Framework et les middlewares de services, permettant le développement d'applications autonomiques pour les grilles de calcul, et cela pour gérer les différents problèmes liés à la complexité, le dynamisme, l'hétérogénéité et l'incertitude dans les environnements d'exécution des grilles. L'objectif global de ce projet est de développer des modèles conceptuels et de mise en œuvre d'architectures qui permettent le développement, l'exécution et l'autogestion de chaque application de la grille. AutoMate définit une infrastructure qui supporte l'exécution des éléments autonomiques, et il définit un système qui fournit des applications dynamiques et autonomiques composées d'éléments autonomiques. Cette composition est conditionnée par les politiques et les contraintes qui sont définies, déployées et exécutées en temps réel, et cela selon les ressources disponibles (système, services, stockage), les exigences, les capacités et l'état actuel des éléments. Ce projet utilise une plate-forme basée sur des composants pour séparer les aspects fonctionnels et les aspects non-fonctionnels, c'est-à-dire son objectif est de séparer les aspects règles de composition automatique et de gestion autonomique du code fonctionnel de l'application. Il propose un gestionnaire autonomique qui est affecté à chaque composant du système

pour gérer les applications. De plus, il propose un processus d'expression des politiques au niveau de chaque application. Le gestionnaire de composition décompose les règles exprimées au niveau de chaque application en règles à injecter dans les différents gestionnaires autonomiques. En résumé, cette solution permet de définir les règles globales de gestion d'une application et de les gérer d'une façon distribuée.

– Le projet RainBow [4] est développé par l'université Carnegie Mellon. Il propose une architecture qui définit un ensemble d'éléments pour la création d'applications autonomiques. Il propose le composant Runtime manager qui définit une interface permettant de gérer plusieurs types d'applications implémentées avec différents langages de programmation. Cette interface permet d'accéder et de modifier l'architecture de l'application, elle assure une séparation entre les aspects relatifs au langage d'implémentation de l'application et les aspects relatifs à son adaptation et sa gestion. Les politiques d'administration sont en charge de la détection des défaillances dans l'architecture du système et également d'adapter le système par rapport aux problèmes détectés. Les possibilités d'adaptation, proposées par le composant Runtime manager, sont contraintes par l'infrastructure d'exécution de l'application. Ce projet définit un système pour la gestion autonomique de l'application, il propose une gestion centralisée de l'architecture du système.

– Le Projet JADE [5] développé par l'université de Grenoble propose une plate-forme pour la création et la gestion des applications autonomiques. Cette plate-forme est basée sur un modèle à composants pour le développement des applications autonomiques. JADE propose une gestion centralisée basée sur un seul gestionnaire qui gère l'ensemble du système. Ce gestionnaire prend en charge plusieurs fonctions telles que l'autoréparation, l'auto-configuration et l'auto-optimisation, et permet aussi de maintenir automatiquement des informations de l'architecture actuelle du système géré dans la base de connaissances. Le système offre une séparation entre les aspects fonctionnels et les aspects non-fonctionnels tels que la gestion autonomique des composants. Ce projet propose d'encapsuler les logiciels préexistant dans des composants ayant une interface d'administration unifiée, dans laquelle les politiques d'adaptation de l'application

appliquées par le gestionnaire autonomique sont déjà prédéfinies. L'objectif de ce projet est de fournir une infrastructure logicielle générique pour la gestion autonomique d'applications réparties qui s'exécutent sur des grappes de calcul.

Discussion

L'intérêt du computing autonomique est de minimiser les interventions des administrateurs dans les systèmes. Il propose des applications qui sont capables de s'auto-administrer pendant le fonctionnement. Dans ce cas, l'administrateur spécifie des politiques en termes des objectifs pour contrôler le système. Actuellement, ce domaine connaît un essor considérable car il représente un enjeu économique important pour les industriels. Les solutions actuelles traitent les systèmes auto-organisés pour réduire les coûts de maintenance des applications dans les systèmes. Il n'y a pas de consensus sur les techniques à appliquer pour construire une plate-forme autonomique, l'architecture de gestion du système est différente dans chaque projet ainsi que les politiques appliquées pendant la phase d'exécution des applications autonomiques. Ce domaine est particulièrement intéressant mais il n'intègre pas la gestion de la QoS au niveau des composants de service pour contrôler leur comportement pendant la phase d'exécution. Pour cette raison, il faut s'intéresser aux différents problèmes liés à un changement de comportement pendant la phase d'exécution causé par la dégradation de la QoS pendant la session. Il faut des composants de service qui s'autogèrent et s'autocontrôlent pour obtenir plus de flexibilité et de réactivité au cours de la session afin de satisfaire le SLA contracté.

III.2.2 Les solutions existantes pour les architectures autonomiques

Ces dernières années, de nombreux chercheurs ont concentré leurs efforts sur les problématiques d'autogestion dans les architectures autonomiques, et cela pour satisfaire les besoins des consommateurs en termes de fourniture de QoS dans un environnement dynamique. Nous présentons quelques travaux qui traitent les architectures autonomiques.

F. Farha et al [6] présentent dans leur papier une architecture de service autonomique (ASA) pour délivrer des applications et des services via une infrastructure tout IP. Leur solution est

basée sur les concepts: SOA, la virtualisation et le service delivery. Ils proposent un broker ARB (Autonomic ressource Broker) qui représente une entité d'autogestion pour automatiser le service delivery des services au niveau transport. ARB gère d'une manière automatique l'activation, la réservation, la gestion et la libération des ressources réseaux. Le broker interagit avec les ressources du niveau transport et avec d'autres ARBs pour fournir les services, et il applique les politiques pour garantir le SLA contracté entre les utilisateurs et les fournisseurs de service. Dans les architectures de service autonomiques, les mécanismes d'autogestion et d'auto-adaptation sont centralisés. Les solutions actuelles agissent essentiellement sur les ressources des réseaux de transport pour assurer la QoS du media Delivery, et cela pour améliorer la délivrance des données pour une application par session

John Strassner et al. [7] proposent une architecture autonomique pour gérer les applications et les réseaux ubiquitaires. Elle fournit un Framework évolutif pour gérer les services de l'internet du futur, il génère dynamiquement un code pour faire des changements de configuration. Ce Framework réduit les coûts opérationnels en automatisant les taches de configuration. Leur approche présente une orchestration des connaissances et des règles, et cela pour gérer d'une manière transparente le comportement des ressources physiques, virtuelles en cas de changement de contexte. Leur solution s'appuie sur un middleware intelligent qui combine et harmonise les diverses données pour résoudre les problèmes liées au langage et à la sémantique associée qui est utilisée par les différents fournisseurs de service.

Muhammad Agnu Catur Bhakti et al dans [8] présentent une architecture SOA autonomique. Ils proposent une extension de l'architecture SOA existante en intégrant le paradigme du computing autonomique proposé par IBM. Ils intègrent un gestionnaire autonomique qui effectue les fonctions décrites par IBM (la supervision, l'analyse, la planification et l'exécution). Ils utilisent aussi une base de connaissance qui stocke les profils de service tels que le type de service (atomique, composite), les fournisseurs de service et les services atomiques qui composent le service. Ils proposent une architecture autonomique SOA structurée en 3 couches : une couche de présentation qui fournit une interface pour les différents utilisateurs, une couche de traitement qui inclut le gestionnaire autonomique et une couche de services qui contient les fournisseurs de services. La caractéristique autonomique

permet d'adapter le système pour remédier aux différents changements et problèmes qui peuvent se produire, et cela en utilisant les informations stockées dans la base de connaissance.

Yan Liu et al proposent, dans leur article [9] une architecture basée sur un middleware autonomique pour assurer la coordination entre plusieurs applications, qui sont capables d'adapter et d'autogérer leur comportement à travers des composants qui mesurent leur comportement et qui fournissent un rapport de contrôle. Leur solution s'appuie sur une approche autonomique qui a pour objectif de faire une coordination entre plusieurs entités de contrôle pour gérer la qualité de service, et cela en utilisant des modèles de contrôle exécutables. La flexibilité de cette approche réside dans le fait qu'elle permet de composer de nouveaux composants de contrôle à partir des composants existants dans le middleware. En outre, le processus des modèles de contrôle peut être modifié, composé, et déployé par le middleware autonomique sans affecter la logique du processus des services.

Discussion:

L'intérêt d'avoir des architectures autonomiques est d'automatiser le «service Delivery» au niveau transport dans les réseaux de nouvelle génération. Ces solutions sont basées sur un orchestrateur qui contrôle et gère les ressources réseaux selon les besoins applicatifs. Dans les solutions actuelles, la réactivité et la flexibilité du système peuvent être mises en cause. En effet, en utilisant des architectures avec des gestionnaires centralisés, les données de surveillance doivent remonter jusqu'à l'entité de gestion et de contrôle qui doit ensuite traiter l'ensemble des données provenant des différentes applications, et par conséquent prendre plusieurs décisions simultanément et envoyer des ordres de réadaptations . Si plusieurs demandes d'adaptations doivent être exécutées au même temps, il est fort probable que le gestionnaire ne va pas pouvoir traiter l'ensemble des adaptations en parallèle, ce qui peut provoquer une baisse de la réactivité et impacter le service rendu.

III.2.3 Les solutions existantes pour les composants de service autonomiques

Être totalement dépendant des interventions de maintenance manuelles a des impacts sur le degré de rapidité et d'adaptabilité dans les systèmes d'aujourd'hui. C'est la raison pour

laquelle plusieurs chercheurs se sont intéressés aux domaines qui traitent les composants de service autonomiques pour avoir des systèmes automatisés. Ces systèmes intègrent les capacités d'adaptation et d'autogestion dans les composants de service pour répondre aux besoins fonctionnels en cas de changements d'environnement.

F.Zambonelli et al.[10] proposent des composants de service autonomiques qui sont en mesure d'adapter dynamiquement leur comportement en réponse aux différents changements de situation. Ils présentent des mécanismes qui permettent aux composants de services d'acquérir un certain degré d'autosuffisance pour appliquer les adaptations les plus appropriées en cas de changement.

Hua Liu et Manish Parashar [11] présentent un Framework qui permet le développement d'éléments autonomiques et la composition dynamique des applications autonomiques en utilisant les éléments autonomiques. Ils proposent des règles et des mécanismes pour la composition dynamique de composants autonomiques, de telle sorte que le comportement des éléments ainsi que leurs compositions et leurs interactions peuvent être gérés pendant la phase d'exécution, et cela en utilisant des règles qui sont dynamiquement injectées.

Ces approches évoquent les paradigmes de la création d'applications autonomique et l'adaptation du comportement des composants de service. Toutefois, ces solutions actuelles mettent l'accent sur l'auto-adaptation et l'autosuffisance du composant de service autonomique, mais elles ne prennent pas en compte l'aspect dynamique de la session au niveau du service.

III.3 Les solutions existantes de prise en compte de la QoS

Dans cette partie nous exposons quelques travaux relatifs à la prise en compte de la QoS d'une part au niveau applicatif, et d'autre part au niveau réseau. Nous commençons par lister quelques recherches sur les mécanismes de gestion de la QoS dans les architectures SOA (§III.3.1), après nous analysons quelques travaux existants sur la sélection de service basée sur la QoS (III.3.2) et également quelques-uns sur le multimédia service Delivery (§III.3.3), puis nous détaillons les mécanismes de gestion de QoS dans les architectures NGN (§III.3.4).

III.3.1 La gestion de la QoS dans les architectures SOA

L'architecture orientée service (SOA) [12] fait partie d'un domaine de recherche qui est en pleine croissance. Cette architecture permet une composition de services distribués à couplage lâche. Elle permet la création rapide d'applications à faible coût, car elle offre la possibilité de réutiliser des composants de service qui sont déjà développés. Plusieurs travaux ont traité le problème de la gestion de la QoS au niveau applicatif, on en cite dans ce qui suit quelques-uns:

Elarbi Badidi et al [13] proposent une solution pour la gestion de la QoS au niveau SOA. Ils présentent un Framework basé sur les règles de gestion de QoS dans un environnement SOA pour garantir des services aux utilisateurs ayant divers terminaux. Le Framework s'appuie sur une fédération de brokers qui sont en charge d'assurer, d'une part, la médiation entre les demandeurs et les fournisseurs de services, et d'autre part la réalisation de diverses opérations de gestion de la QoS. Ils présentent également des règles de sélection des services selon les attentes des utilisateurs en termes de QoS. Leur approche s'appuie sur un algorithme qui classe les services qui sont fonctionnellement équivalents, en fonction de leur capacité à accomplir le service demandé par l'utilisateur selon ses exigences de QoS.

Daniel.A et al présentent dans [14] la gestion de la QoS dans les architectures orientées services (SOA). Leur solution est basée sur un QoS broker (QB) pour une composition de service dans un système distribué, il permet aussi une reconfiguration dynamique des composants de service pour supporter non seulement les évolutions des besoins fonctionnels et de performance de QoS des utilisateurs, mais aussi pour remédier aux différents changements qui sont causés par l'environnement ambiant de l'utilisateur (réseaux , hardware et software). Leur méthode utilise des mécanismes de réservation de ressources au niveau des composants pour fournir les exigences de QoS. Cette solution ne garantit pas la QoS en temps réel. Le QB joue un rôle important dans la gestion de la QoS, il se charge de gérer et de réserver les ressources des composants de services fournies afin de garantir la QoS requises par les clients. Il assure les fonctions suivantes :

- l'enregistrement des services basé sur la QoS: le QB offre aux fournisseurs de service la possibilité d'enregistrer les caractéristiques de performances de leurs services.

- La découverte des services basée sur la QoS: le QB trouve les services requis en se basant sur les fonctionnalités et la performance de QoS
- La capacité de négociation: la QB assure la négociation de QoS des services qui peuvent se trouver dans une même plate-forme ou bien dans des plates-formes différentes.
- La réservation des ressources: le QB assure la réservation des ressources pour satisfaire les demandes des clients.

Lorsqu'un utilisateur demande un service avec un certain niveau de QoS, le QB détermine si le fournisseur de service peut lui offrir ce service, sinon il négocie le niveau de QoS avec le client pour lui rendre un autre niveau de QOS.

Joseph Loyall et al. présentent dans leur article [15] une approche pour gérer la QoS dans des architectures SOA. Ils décrivent les fonctions clés qui sont requises dans les middlewares SOA pour qu'ils supportent la gestion de la QoS des applications nécessitant des performances prévisibles. Ils proposent une approche axée sur la politique de gestion de la QoS dans les systèmes SOA en se basant sur le principe d'évaluer d'une manière empirique la capacité des systèmes à fournir la qualité de service en cas de surcharge CPU et de limitation de bande passante .

Valeria Cardellini et al. proposent dans [16] un service broker qui assure une adaptation de QOS en temps réel des applications SOA qui sont composées de plusieurs composants de service. Ils décrivent les fonctionnalités qui sont fournies par le broker et ils présentent aussi le design du prototype MOSE (Model-based Self-adaptation of SOA system), qui représente un Framework du service broker. L'objectif principal de cette solution est d'assurer une auto-adaptation dans un système SOA pour répondre aux besoins non-fonctionnels (QoS) d'un service composé, comme le temps de réponse, la disponibilité et le coût, et cela pour respecter le SLA contracté entre les clients et les fournisseurs de service.

Discussion

Actuellement, la majorité des recherches qui traitent la gestion de la QoS au niveau des architectures de service (SOA), proposent des solutions de gestion qui sont centralisées. Ils s'appuient sur un orchestrateur qui gère les ressources des services applicatifs dans le but d'assurer la QoS durant l'usage des services. Cependant, les mécanismes proposés

aujourd'hui n'offrent pas une gestion flexible, dynamique et adaptable au cours de la session et selon le contexte de l'utilisateur qui ne cesse pas de changer. Dans le contexte NGS, nous trouvons des services ubiquitaires hébergés dans des plates-formes trans-organisationnelles, c'est pourquoi il faut un gestionnaire de QoS décentralisé, qui soit intégré dans chaque service pour qu'il soit le plus autonome, le plus auto-suffisant, et le plus auto-gérable possible. Par ailleurs, ces caractéristiques nous sont utiles dans un environnement multifournisseurs dans l'objectif d'assurer la continuité de service, et cela avec une possibilité de changer de service, d'une manière transparente, en cas de dysfonctionnement ou bien de dégradation de la QoS au cours de la session de service. C'est pourquoi nous proposons une nouvelle organisation basée sur un acteur principal, qui est le composant de service autonomique, et qui joue un rôle important dans le maintien du service rendu à l'utilisateur sans impacter la QoS de bout en bout. Ce composant participe aux résolutions des différents problèmes qui surgissent au cours de la session de service, en interagissant avec d'autres composants sur la même plate-forme de service ou bien sur d'autres plates-formes.

III.3.2 La sélection de service basée sur la QoS

Plusieurs services peuvent fournir des fonctionnalités similaires mais avec des propriétés non fonctionnelles différentes comme le délai de délivrance de service. Lors de la phase de sélection des services, il est important de considérer aussi bien les propriétés fonctionnelles que les propriétés non-fonctionnelles (QoS) pour répondre aux besoins des utilisateurs. Par ailleurs, l'intérêt des critères de QoS est de fournir une différenciation entre les services concurrents pour que le choix soit conditionné par les exigences relatives à la mobilité et les besoins fonctionnels et non-fonctionnels de l'utilisateur. Récemment, la sélection de service basée sur la QoS dans le cadre de la composition des services a attiré l'attention de nombreux chercheurs.

Di Marco et al. [17] proposent un framework pour réaliser une composition de service dynamique dirigée par la QoS. Le framework contrôle les services composés pour vérifier si le SLA est satisfait. Dans le cas où le SLA n'est pas satisfait, le framework propose à l'utilisateur d'autres services avec des QoS différentes et dans le cas où l'utilisateur n'accepte aucun de ces services, la requête est supprimée. L'inconvénient de cette approche réside dans

le fait que l'utilisateur peut demander un service avec un certain niveau de QoS, mais le fournisseur ne peut pas lui satisfaire sa demande. En outre, leur travail ne couvre pas le cas d'une architecture trans-organisationnelle où de nombreux fournisseurs peuvent fournir les mêmes services avec les mêmes caractéristiques fonctionnelles et non–fonctionnelles (QoS) pour garantir toujours le SLA contacté et satisfaire les besoins de leurs clients.

S.Neelavathi et al. [18] présentent le modèle SIBE utilisé pour la sélection des services qui est intégré dans un orchestrateur SOA. Ce modèle s'appuie sur une estimation de disponibilité des services en utilisant plusieurs contraintes de QoS. Il prend en considération l'historique de performance des services durant la phase d'usage, pour répondre aux attentes des nouveaux utilisateurs. La méthode proposée fournit une solution optimale de sélection de services qui s'appuie sur l'évaluation de la QoS pendant l'utilisation du service. La méthode de gestion de QoS proposée est une gestion statique et centralisée dans l'orchestrateur SOA car elle est basée sur les statistiques d'utilisation d'un service par les utilisateurs.

Hamid et al. [19] proposent une solution pour satisfaire les besoins et les préférences des utilisateurs, et cela lorsqu'ils veulent changer de terminal durant la mobilité de session. Ils proposent un modèle de préférences qualitatives qui est traduit en un modèle de préférences quantitatives pour choisir le terminal le plus approprié pour l'utilisateur. Cette approche vise à maintenir la continuité de la session de l'utilisateur dans le cas d'un changement de terminal, mais elle ne couvre pas les autres cas de la mobilité ou bien de changement des préférences de l'utilisateur.

III.3.3 Le multimédia service Delivery et la QoS

Quelques articles de recherche discutent les sujets liés au multimédia service delivery et à la personnalisation des services. Dans leur papier, Chitra et al. [20] s'intéressent au multimédia service delivery pour les réseaux de nouvelle génération NGN, ils présentent un framework basé sur une architecture IMS pour les applications multimédias afin d'assurer un service delivery qui est contrôlable et adaptable. Ce framework est désigné pour évaluer le provisionnement des services multimédias dans les réseaux de transport avec un plan de contrôle IMS. Il analyse la performance QoS à l'égard du délai induit par le niveau réseau durant la session et la mobilité du terminal, et cela dans l'objectif de délivrer des services avec

une QoS de bout en bout aux utilisateurs. D'autre part, Jose et al.[21] s'intéressent à la personnalisation des services, ils présentent une architecture CUMDA qui permet aux utilisateurs de concevoir leurs services. CUMDA permet la création des services ubiquitaires à travers un environnement IMS. Leur approche se focalise sur le paradigme de la création des services qui prend en considération la demande des utilisateurs pour fournir des services personnalisés qui sont adaptés au contexte. Maamar et al.[22] présentent une approche intéressante, qui a pour objectif d'assurer la personnalisation de la composition des web services et le provisioning en tenant en compte du contexte. Ce dernier représente les informations qui caractérisent les interactions entre les utilisateurs, les applications et l'environnement ambiant. Leur solution vise à satisfaire la demande des utilisateurs dans le cas où elle ne peut pas être exaucée par aucun service existant, elle requiert une combinaison de plusieurs web services. Il stocke les détails reliés à la personnalisation comme le temps d'exécution et la localisation des services web préférés pour assurer le changement d'un service web à cause des préférences des utilisateurs et de la disponibilité des ressources.

La majorité des recherches s'est focalisée sur deux aspects. Ils ont traité d'une part le multimédia service delivery pour gérer la QoS relative à une application par session, et d'autre part, ils ont étudié la personnalisation des services selon les besoins fonctionnels des utilisateurs et leur contexte ambiant. A ce jour, il n'existe pas de travaux qui permettent aux utilisateurs de personnaliser leurs services en composant des services qui proviennent des domaines hétérogènes (IT, Telco, WEB) selon leurs besoins fonctionnels et non-fonctionnels. Pour cela, il faut une solution qui permet la gestion distribuée de la qualité de service d'une manière Top-Down. Il est essentiel de remonter la gestion de QoS au niveau service, qui sera rajoutée à la gestion de la QoS assurée par le niveau transport, et cela afin de garantir le «service Delivery» durant la session «user-centric» dans un environnement qui est totalement hétérogène et mobile.

III.3.4 Les mécanismes de gestion de QoS dans les réseaux NGN

La nouvelle génération des réseaux (NGN) se concrétise avec l'apparition de l'architecture IMS, standardisée par la 3GPP et TISPAN à l'issue de la convergence de la Release 7 (3GPP) et la Release 2 (TISPAN). Cette architecture définit un plan de contrôle commun aux réseaux fixes et mobiles de future génération tels que les réseaux mobiles GSM, GPRS, 3G, LTE, les réseaux WLAN et les réseaux à commutation de circuits. L'architecture IMS [23] est basée sur un cœur de réseau tout IP et permet de fournir une qualité de service de bout en bout pour chaque session. Son objectif principal est de garantir la qualité de service selon les besoins des applications multimédias au niveau transport. L'IMS permet un accès agnostique, ce qui signifie que les services fournis via l'architecture IMS sont indépendants de la technologie d'accès au réseau. Grâce à l'architecture IMS, la gestion de la qualité de service évolue vers un contrôle de ressources allant au-delà du réseau d'accès, car la qualité de service globale fournie au cours d'une session est impactée par les niveaux de qualité de service à chaque segment du réseau traversé. Le principe d'IMS est de séparer la couche transport de la couche services et intègre des fonctionnalités de gestion de la QoS pour assurer la qualité de service souhaitée au niveau transport pour l'application demandée.

L'architecture IMS (Figure 4) est organisée en couches qui scindent les fonctions et les équipements responsables du transport du trafic et du contrôle.

- La couche d'accès regroupe les technologies qui permettent l'accès des terminaux des utilisateurs au réseau y compris l'accès mobile et l'accès fixe tels que GSM, GPRS, UMTS, WIFI, xDSL.

- La couche de Transport représente un réseau cœur de transport tout IP responsable de l'acheminement du trafic voix ou données. Le réseau cœur intègre des mécanismes et les protocoles de gestion de la QoS tels que MPLS, RSVP, Diffserv. A ce niveau, on applique les adaptations des protocoles de transports aux différents types de réseaux physiques disponibles: (RTC, IP, ATM, Ethernet….).

- La couche de contrôle gère l'ensemble des fonctions de contrôle de session appliquées par l'équipement CSCF (Call state Control Function), qui est responsable

d'appliquer les mécanismes de contrôle lors de l'invocation des services. Il assure également le routage des messages de signalisation entre les terminaux des usagers et les services applicatifs

- La couche d'application consiste en des serveurs d'application (AS) dans lesquels sont déployés les différents services et applications qui sont susceptibles d'être offerts aux utilisateurs.

Les principales entités de l'architecture IMS sont les CSCFs (Call Session Control Function) qui assurent la gestion des flux de signalisation. Ils garantissent la fonction d'établissement de la session entre les deux extrémités et ils fournissent le service correspondant aux caractéristiques de la session requise. La Figure 4 montre les briques CSCFs qui sont scindés en 3 entités fonctionnelles : S-CSCF (Serving-CSCF, Interrogating-CSCF, Proxy-CSCF)

Le S-CSCF est un élément central qui gère les sessions multimédias de l'utilisateur, le routage vers les réseaux externes et assure les services de contrôle de session pour le mobile. C'est lui enfin qui gère les mises à jours des contextes clients au niveau de la base de données HSS.

Il contient aussi deux éléments de bordure: le P-CSCF et l'I-CSCF (respectivement le Proxy et l'Interrogating Call Session Control Function). Ces deux éléments représentent des briques essentielles dans les dispositifs de sécurité et de qualité de service mis en place par 3GPP.

Le P-CSCF est le premier point de contact dans le domaine IMS et il effectue les fonctions suivantes:

- Garantie de la qualité de service par contrôle d'admission
- Gestion des appels d'urgence,
- Sécurité du réseau visité,
- Génération de CDRs (tickets d'appels).

L'I-CSCF se situe en bordure du réseau d'accès (ou Visited network) et du réseau opérateur (ou Home network). C'est le point de contact au sein du Home Network pour toutes les sessions destinées à un utilisateur de cet opérateur.

Les fonctions réalisées par cette entité sont :

- L'obtention de l'adresse du S-CSCF auprès de la base de données HSS.
- L'assignation d'un S-CSCF à un utilisateur s'enregistrant.
- L'acheminement des méthodes SIP reçues depuis un autre réseau au S-CSCF.
- la garantie de sécurité entre le Visited network et le Home network.
- Point d'entrée unique du home network.
- Masquage de la topologie des différents réseaux.
- La fonction du firewall.

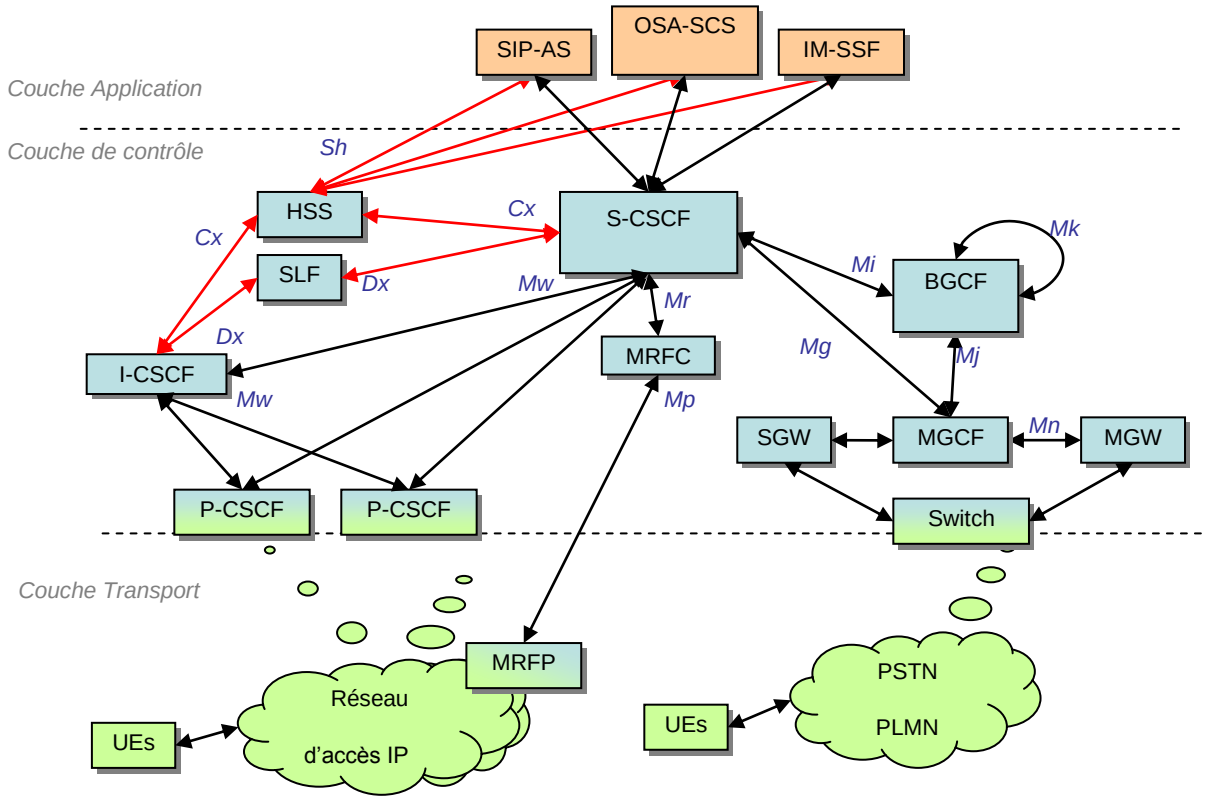

Figure 4: Architecture IMS

(a)Mécanismes de gestion de la QoS spécifiés par la 3GPP

Pour assurer la qualité de service de bout en bout, la 3GPP favorise la gestion et le contrôle de l'utilisation des ressources medias au niveau transport pour satisfaire le besoin du client selon son profil et ses droits. Dans cette partie, on décrit les entités nécessaires au traitement et au

contrôle de la qualité de service définies par la 3GPP ainsi que les différents protocoles utilisés pour assurer les échanges des informations de QoS entre ces différentes entités.

L'établissement de la session pour les services multimédias nécessite une coordination entre le support de transport et la session pour assurer une QOS de bout en bout. Pour cela, l'architecture définie par la 3GPP fournit une séparation claire entre les différents niveaux de l'architecture à savoir: le niveau support (SGSN, GGSN), le niveau contrôle de session d'IMS (P-CSCF, S-CSCF, I-CSCF) et le niveau applicatif. A travers l'IMS, le terminal négocie ses capacités et exprime ses exigences de QoS durant la phase d'établissement de la session avec le protocole SIP. En parallèle le terminal réserve les ressources nécessaires dans le réseau d'accès (UMTS/GPRS) en utilisant le protocole GTP.

Nous présentons les entités qui ont accès aux informations qui décrivent le contenu des flux échangés et qui peuvent en déduire le type de ressources et le niveau de QOS requis pour l'établissement de la session. Tout d'abord, le P-CSCF qui représente le premier point de contact dans IMS indique au PCRF les ressources requises par la session, ce dernier vérifie les droits de l'utilisateur concerné et aussi la disponibilité des ressources nécessaires et suffisantes au niveau transport pour la session. Une fois que le contrôle d'admission par le PCRF est réussi, il envoie les règles de contrôle et de gestion au PCEF(GGSN) via l'interface Rx+ [24], par le biais du protocole Diameter, ce qui déclenche les procédures de QoS dans le GGSN.

L'entité PCEF dans le GGSN associe la correspondance entre les paramètres QOS session et ceux du média, elle assure la fonction de mappage entre les paramètres de QOS IP autorisés reçus via l'interface Gx+ [25] et les paramètres de QOS spécifiques à l'accès pour traduire la QOS demandée par l'utilisateur en fonction des paramètres QOS disponibles et supportés par le réseau.

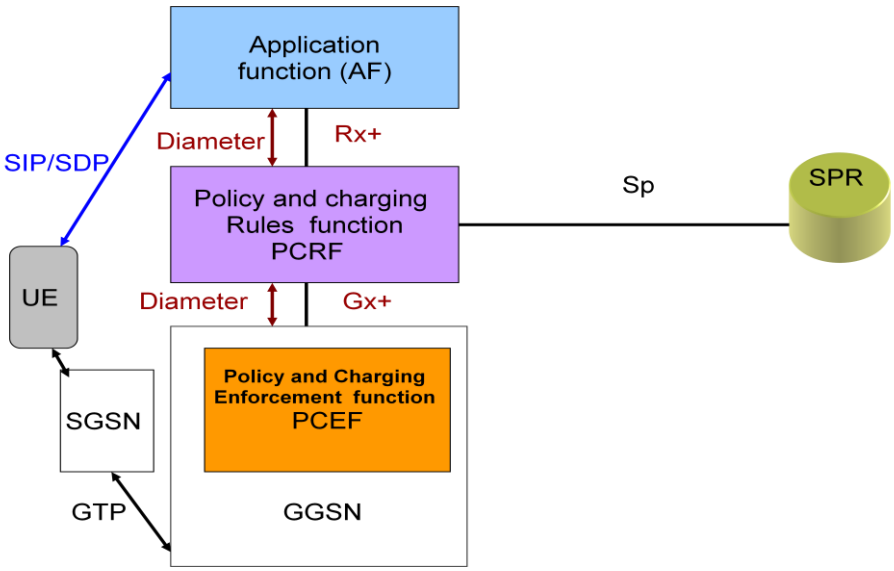

Figure 5: Entités fonctionnelles de gestion de QoS

(b) Les mécanismes de gestion de la QoS spécifiés par TISPAN

Le groupe TISPAN de l'ETSI a spécifié une architecture NGN qui est basée sur l'architecture de contrôle de service IMS. L'architecture NGN [26] est essentiellement composée d'une couche de service qui contient les serveurs d'application (AS) et d'une couche de transport sous-jacente basée sur IP. Le réseau de transport est supervisé par deux fonctions de contrôle: le NASS [27] «Network Attachment Subsystem» et le RACS [28] « Resource and Admission Control Subsystem», qui assurent la séparation entre la couche de service et la couche de transport. Dans ce qui suit, on va détailler les fonctionnalités du RACS et du NASS, et leur rôle dans le contrôle et le maintien de la QoS au niveau réseau. En effet, le groupe TISPAN a défini les mécanismes de réservation de ressources basés sur la QoS afin de garantir le service demandé par l'utilisateur.

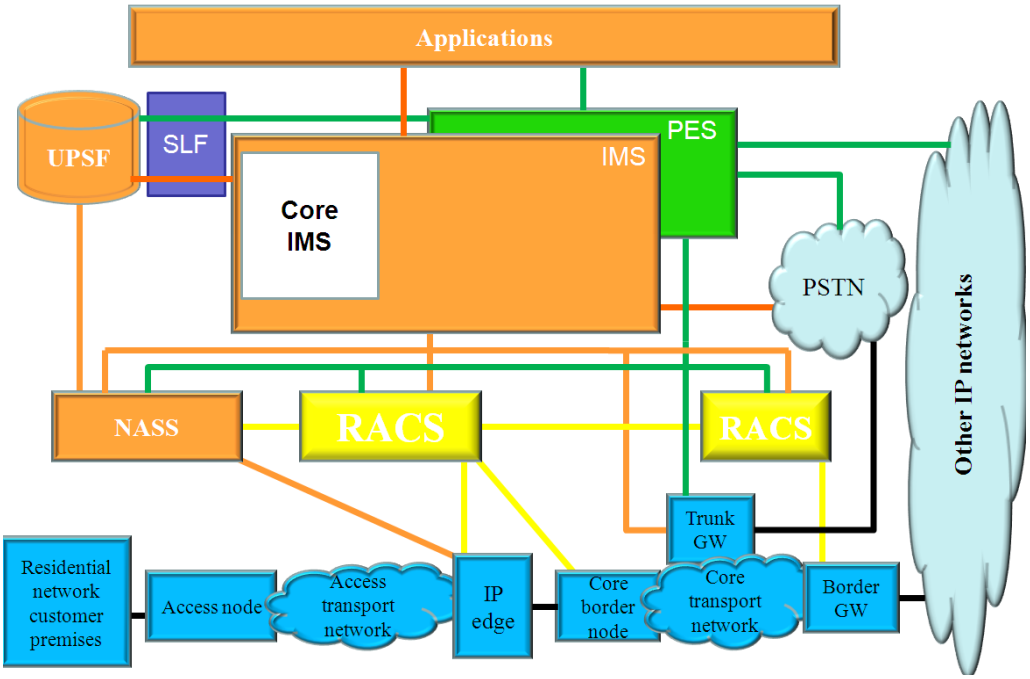

Figure 6: Architecture TISPAN NGN

- **NASS (Network Attachment Subsystem):**

Le NASS définit le premier point de contact entre l'utilisateur NGN et le réseau. Cette entité est responsable de l'enregistrement des terminaux et de l'attribution des paramètres au moment de son attachement au réseau. Il assure l'identification et l'authentification de niveau réseau, il gère également les plages d'adresses IP du réseau d'accès et il authentifie les sessions d'accès. Donc, en résumé, les fonctionnalités du NASS comprennent:

- Le provisionnement dynamique d'adresse IP et d'autres paramètres de configuration qui sont liés au terminal de l'utilisateur.
- L'authentification de l'utilisateur, avant ou pendant la procédure d'allocation d'adresses IP.
- L'autorisation d'accès au réseau qui est basée sur le profil de l'utilisateur.
- La Gestion de la localisation.

- **RACS (Resource and Admission Control Subsystem):**

Le RACS est responsable de l'application des procédures de contrôle du transport de données, qui s'appuie sur des politiques bien définies. En fait, il utilise des mécanismes qui sont responsables du contrôle d'admission et de la réservation des ressources à la fois dans le réseau d'accès et dans le réseau cœur, et cela aussi bien pour le trafic unicast que le trafic multicast. Le RACS assure les fonctionnalités suivantes:

- Contrôle d'admission: le RACS vérifie si la QoS demandée peut être satisfaite par les ressources qui sont disponibles au niveau du réseau d'accès concerné.
- Réservation de ressources: le RACS met en œuvre les mécanismes pour demander les ressources nécessaires, au niveau accès pour le transport des données applicatives.
- Politiques de contrôle: le RACS applique un ensemble de règles, basées sur le profil de l'utilisateur, pour vérifier si les demandes de réservation de ressources peuvent être autorisées.
- NAT/Filtrage: le RACS contrôle les fonctionnalités du NAT et exerce des fonctions de filtrage, et cela au niveau des IP Edge qui se situent à la limite entre le réseau d'accès et le réseau cœur.

RACS a deux sous fonctions :

- RACF (Resource and Admisssion Control Function) contrôle la réservation des ressources au niveau transport.
- S-PDF (Serving Policy Decision Function) applique les décisions de politiques en se basant sur le profil de l'utilisateur. Cette entité reçoit les requêtes en provenance de la fonction d'application (AF), par exemple le P-CSCF, et envoie des demandes de ressources au RACF.

Au niveau réseau cœur qui est responsable du transport de données, Les entités « IP edge node » et « Border Gateway Function (BGF)» appliquent les décisions du RACS. Il existe deux sous-fonctions du BGF :

- C-BGF (Core-BGF) qui se situe à la limite entre le réseau d'accès et le réseau cœur

- I-BGF (Interconnection-BGF) qui se situe à la frontière entre deux cœurs de réseaux.

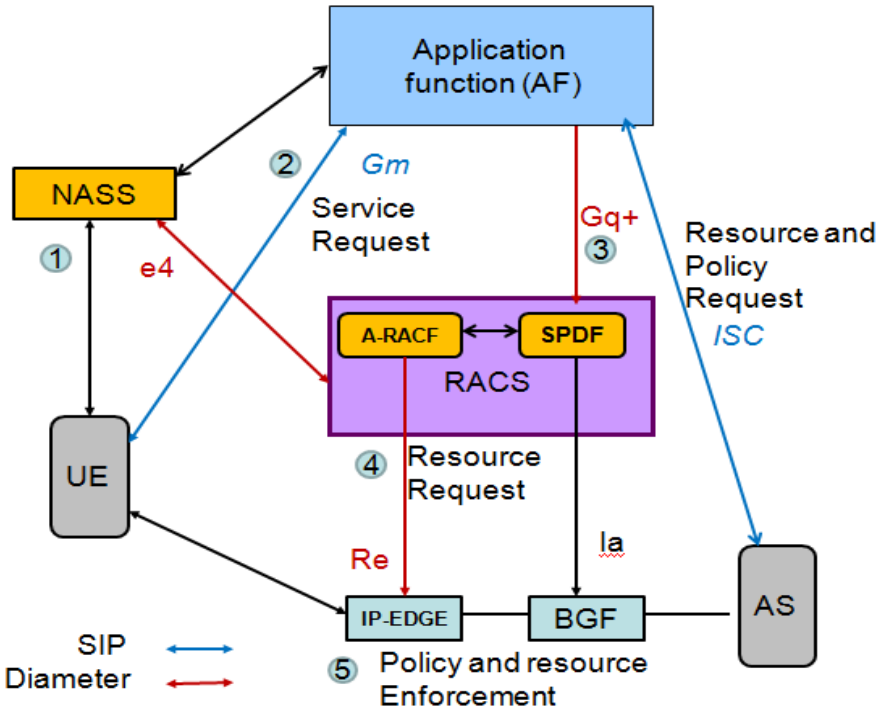

Figure 7: les ménacismes de contrôle de QoS de TISPAN

Ce schéma (Figure 7) illustre les différentes étapes de contrôle de QoS définies par TISPAN. Tout d'abord, le terminal envoie une demande d'authentification, d'autorisation et de configuration d'un service demandé par l'utilisateur au NASS (étape 1). Ensuite, le terminal envoie un message de signalisation SIP à l'AF (par exemple P-CSCF) (étape 2). Une fois cette demande est reçue par l'AF, une demande d'autorisation de QoS et de réservation des ressources est envoyée vers la fonction SPDF du RACS. Cette fonction applique le contrôle d'admission en s'appuyant sur les besoins des applications, l'état des ressources et les politiques des réseaux d'accès (étape 3). A-RACF prend ensuite la décision d'admission en se basant sur l'état des ressources du réseau d'accès (étape 4), et décide si des politiques de trafic doivent être installés. A ce moment, il envoie une demande d'installation de ces politiques aux entités du réseau de transport (étape 5).

Comparaison

Contrôle de QoS	**3GPP**	**ETSI TISPAN**
Policy Decision Function	PCRF- Policy and Charging Rules Function	SPDF- Service based Policy Decision Function A-RACF - Access Resources & Admission Control Function
Transport Resource Control Function	GGSN/ SGSN	A-RACF
Policy Enforcement Function	PCEF- Policy & Charging Enforcement Function(GGSN)	BGF- Border Gateway Function (e.g. Core Border node) RCEF- Resource Control Enforcement Function (e.g. IP edge)

Le tableau 3GPP et TISPAN résume les différentes méthodes de contrôle de QoS qui ont été adoptées dans les différents réseaux NGN. Ces méthodes ont été standardisées par les instances de normalisation 3GPP, TISPAN, dans les domaines de télécommunications concernant les architectures avec des réseaux d'accès fixes et mobiles. L'objectif de ces mécanismes est de fournir un contrôle de QoS au niveau transport pour une application donnée. A ce jour, ces mécanismes garantissent le transport des données avec la QoS exigée par le niveau applicatif, mais ils ne gèrent pas la QoS de l'application elle-même, qui peut dans certains cas impacter le niveau de QoS rendu à l'utilisateur, et causer une non-satisfaction du SLA contracté.

III.4 La session

La session définit la mise en relation temporelle entre deux extrémités de transport, dans (§III.4.1) nous recensons les définitions de la session dans le monde de télécommunications, puis nous définissons le protocole de signalisation SIP (III.4.2) conçu pour établir, modifier et terminer des sessions multimédia et finalement nous détaillons (III.4.3) les différentes types de mobilité de la session.

III.4.1 Les différents types de sessions

Dans le monde des télécommunications, la session est la mise en relation temporelle entre les deux extrémités de transport qui permet une synchronisation des communications entre l'émetteur et le récepteur au cours de la transmission des données. Le transport des données se définit par une connexion point à point avec deux interlocuteurs ou bien par une communication multipoints avec plusieurs interlocuteurs. Le modèle OSI définit deux genres de communication multipoints: les communications multipoints en étoile où une session est un ensemble de communications point à point avec un interlocuteur engagé dans tous les échanges, et les communications multipoints par diffusion où tous les interlocuteurs reçoivent les messages. On trouve plusieurs types de session qui sont définies dans les normes:

Dans le domaine d'IP multimédia, une session spécifiée par la 3GPP [29] est définie comme une connexion entre un ensemble d'émetteurs et de récepteurs pour transmission des données. Ce type de session est supporté par le sous-système CN IP (Core Network) multimédia et elle est supportée par des supports de connectivité IP par exemple des supports de type GPRS. La session IP CAN [30] est une association entre le terminal et le réseau IP. Cette association est identifiée par une adresse IP de l'UE et également par les informations d'identité du terminal dans le cas où elles sont disponibles. La session IP CAN supporte un ou plusieurs supports de transport. Cette session est maintenue tant que l'adresse IP UE est établie et déclarée au niveau du réseau IP.

On trouve d'autre part, la session de commutation de paquets (PS: Packet switched) [31] se définit par une connexion logique établit par un contexte PDP entre un terminal et un domaine PS pour la délivrance des paquets de données. Pour supporter la gestion de la QoS au cours de

la session, la session QoS est spécifiée par la 3GPP [32] pour échanger les PDP contextes entre l'ouverture et la fermeture de la connexion réseau. Elle représente la durée de vie d'un PDP contexte qui indique les profiles QoS au GGSN. Pour gérer la mobilité du Terminal, l'ETSI [33] définit une session transparente (Seamless session) qui est maintenue pendant un changement au niveau du réseau d'accès, et cela sans aucune interruption perceptible par l'utilisateur, tout en s'adaptant aux capacités de chaque système d'accès. L'architecture IMS [34] définit une session multimédia, en utilisant un réseau IP, qui permet d'établir des communications entre plusieurs terminaux, et elle permet d'ajouter des services en temps réel au cours d'une même communication. Avec IMS on passe du mode de transmission à commutation de circuits (CS) au mode à commutation de paquets (PS) pour transporter aussi bien des données en temps réel qu'en non temps réel.

Session basée service [35]: il faut une signalisation au niveau applicatif pour fournir le service demandé par l'utilisateur, qui est séparée du service rendu. Et la session user-centric [36] définit une période de communication entre un ou plusieurs utilisateurs ou serveurs applicatifs. Elle inclut la mise en relation des équipements de l'utilisateur, du réseau d'accès, du réseau cœur et des services.

TINA (Telecommunications Information Networking Architecture) [37] introduit le concept de la session TINA, et définit trois types de session: la session d'accès, la session de communication et la session de service.

La session d'accès (AS) correspond à la mise en place des modalités et des conditions de la session (par exemple l'authentification, la sélection du profile de service) pendant la phase de connexion d'un utilisateur au système. Ce type de session permet à l'utilisateur d'établir des sessions de service, de combiner des sessions et d'utiliser plusieurs services.

La session de service (SS) représente les informations et les fonctionnalités liées aux capacités d'exécution, de contrôle et de gestion des services. Ces services incluent les services primaires (par exemple la conférence multimédia) et les services auxiliaires (par exemple la souscription en ligne). Une session de service est une instance d'un type de service et elle inclut les informations nécessaires à la négociation de QoS, la sécurité, l'utilisation du service, la communication des ressources et aussi les informations pour le contrôle des relations entre les participants de la session de service.

La session de communication (SC) représente le service des flux de connexion entre le terminal et les technologies réseaux qui fournissent les ressources de communication requises pour établir des connexions de bout en bout. Une session de communication peut supporter plusieurs connexions qui peuvent être multipoints et multimédias. Une session de communication peut gérer la QoS, établir, modifier et terminer plusieurs connexions.

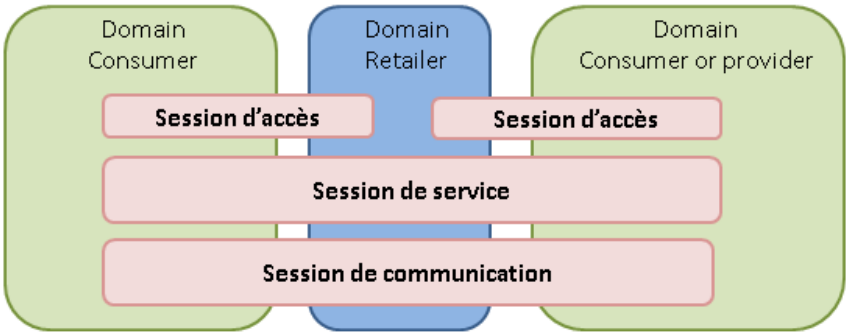

Figure 8: Architecture TINA

En conclusion, il existe plusieurs définitions de la session à plusieurs niveaux de l'architecture. Mais on ne trouve pas une définition d'une session avec une QoS de bout en bout qui supporte la mobilité au niveau service pour répondre au nouveau contexte de l'utilisateur « user-centric ». Nous voulions à travers cette recherche définir une session unique et continue pour l'utilisateur qui peut supporter une connexion multipoints entre plusieurs terminaux, réseaux d'accès et services pour satisfaire ses besoins. Cette session «user-centric» sera maintenue pour garantir à l'utilisateur une continuité de communication non seulement lorsqu'il change de réseaux d'accès mais aussi lorsqu'il change de services applicatifs. L'objectif est de fournir à l'utilisateur une session qui s'adapterait à ses besoins fonctionnels et non-fonctionnels, et aussi au changement de ses préférences. Plus précisément, cette session doit permettre une négociation de la QoS au niveau service d'un service personnalisé qui se rajoute à la négociation de la QoS au niveau transport. Elle doit aussi être capable de supporter la renégociation de la QoS dans le cas d'une mobilité de service.

III.4.2 Le protocole de signalisation SIP

Le protocole de signalisation SIP est un protocole standardisé par l'IETF [38], qui a été conçu pour établir, modifier et terminer des sessions multimédia. Il assure la négociation des types medias entre les différents participants en encapsulant des messages SDP (Session Description Protocol). Le protocole de signalisation SIP a été retenu par l'architecture IMS (IP Multimedia Subsystem) pour le contrôle des sessions multimédias. SIP se charge uniquement de la signalisation et il ne fait pas de réservation de ressources.

Une Transaction SIP représente un ensemble de messages SIP (requêtes /réponses) échangés entre les différents éléments du SIP (client/serveur). Les requêtes et les réponses sont textuelles et comportent un en-tête SIP composé d'une méthode du message, d'un en-tête du message et également d'une ligne vide qui sépare l'en-tête SIP du corps de message SIP. SIP est un protocole de signalisation, il ne prend pas en charge la qualité de service média du réseau. En effet, il s'appuie sur le protocole SDP [39] qui permet de décrire la session média.

Le protocole SIP peut faire appel au protocole RTCP, qui fonctionne parallèlement avec le RTP, pour garantir la transmission des flux de données, en prenant en considération les informations relatives à la qualité du service (QoS). RTCP offre au récepteur la possibilité de renvoyer des informations à l'émetteur sur le nombre de paquets perdus, le délai aller-retour et la gigue.

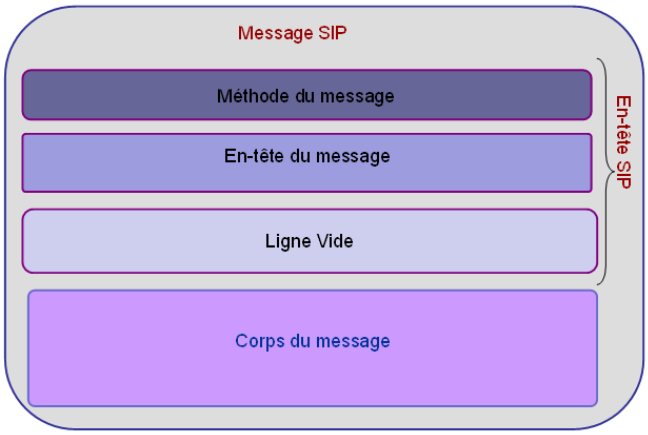

Figure 9: Format du message SIP

On trouve six méthodes de base SIP: REGISTER, OPTIONS, INVITE ACK, CANCEL, et BYE. De nouvelles méthodes telles que SUBSCRIBE, NOTIFY, INFO ont été aussi définies par l'IETF pour étendre les méthodes du protocole SIP. Pour les différentes méthodes, on a 6 types de réponses : 1xx, 2xx, 3xx, 4xx, 5xx, 6xx.

- Les méthodes du protocole SIP:
  - **INVITE**: Invitation à participer à une session. Le corps du message décrit la session. Une étape de négociation permet de définir les média utilisés.
  - **UPDATE**: Mise à jour d'une session (offre SDP) initiée par une transaction INVITE.
  - **ACK** : Confirmation de réponse.
  - **BYE**: Demande de libération d'une session.
  - **CANCEL**: Annulation d'une requête.
  - **INFO:** Transport des informations complémentaires.
  - **MESSAGE**: Envoi des messages instantanés.
  - **REFER**: Transférer une session.
  - **REGISTER**: Enregistrement d'un client auprès d'un serveur.
  - **PUBLISH**: Publication d'un état d'événements.
  - **SUBSCRIBE**: Demander l'état actuel et les mises à jour d'une ressource distante.
  - **NOTIFY** : Informe d'un changement d'état lié au SLA.
  - **RE-INVITE**: Modifier une session.

- **Les réponses SIP**:
  - **1xx Provisoire**: la requête a été reçue, et est en train d'être traitée.
  - **2xx Réussite**: la demande a été reçue avec succès, comprise et acceptée.
  - **3xx Redirection:** des informations supplémentaires sont nécessaires afin de compléter la requête.
  - **4xx Erreur client:** la requête ne peut pas être interprétée par le serveur, la requête doit être modifiée avant d'être renvoyée.
  - **5xx Erreur serveur:** le serveur a échoué à l'accomplissement d'une requête apparemment valide.

- **6xx Echec global:** la requête ne peut être accomplie par aucun serveur.

- **L'entête du protocole SIP:**

Les messages SIP (requêtes et réponses) contiennent un ensemble de champs d'en-tête. Certains champs sont obligatoires et doivent apparaitre dans chaque message et d'autres sont optionnels et apparaissent uniquement lorsque c'est nécessaire.

- **Call-ID**: Représente un identificateur global et unique d'une session.
- **Cseq**: Contient un numéro de séquence. Il est utilisé pour faire correspondre les demandes et les réponses.
- **From**: Identifie l'appelant.
- **To:** Indique l'adresse de destination.
- **Via**: Enregistre la route d'une requête, de manière à permettre aux serveurs SIP intermédiaires de faire suivre aux réponses un chemin exactement inverse.
- **Max-Forward**: Sert à limiter le nombre de sauts que peut faire une demande sur son chemin vers sa destination. Il consiste en un entier qui est décrémenté à chaque saut.
- **Encryption**: Spécifie que le corps du message et éventuellement certains en-têtes ont été chiffrés.
- **Content-Type**: Décrit le type de contenu du corps de message.
- **Content-Length**: Représente le nombre d'octets du corps de message.
- **Route:** Contient les URI SIP des serveurs d'application dans l'ordre dans lequel ils doivent être invoqués.

Corps du message SIP:

Comme la Figure 9 le montre, le corps du message est séparé des champs d'en-tête par une ligne vide. Le protocole SIP peut transporter tout type de corps tels que : Application /SDP, Text/ XML. Le type du corps de message doit être mentionné par le champ content-type de l'en-tête SIP et sa longueur par le champ content-length. La longueur de l'ensemble des champs ne doit pas dépasser le MTU (maximum transmission unit). Si la MTU est inconnue, elle est de 1500 octets par défaut. Cette taille permet l'encapsulation des datagrammes UDP ou segments TCP dans des paquets IP sans fragmentation.

III.4.3 La mobilité de la session

La mobilité de la session [40] permet à un utilisateur de maintenir sa session active lorsqu'il change de terminal ou bien du réseau d'accès.

- Changement de terminal:

La mobilité de la session SIP garantit le transfert d'une session vers un ou plusieurs terminaux avec un changement de l'adresse SIP. Cela permet à un utilisateur qui a initié le transfert de sa session, au cours de l'établissement de la session SIP ou bien lors de la phase de transport des flux de données, de la récupérer. Selon le besoin de l'utilisateur, il existe deux modes de transfert de la session: le mode de transfert avec contrôle et le mode de transfert sans contrôle:

Mode de transfert de session avec contrôle (RFC 3725) [41] : dans ce mode, le Mobile Node (NM) a pour rôle d'établir une session SIP avec chacun des deux terminaux et de mettre à jour la session avec le NC (Correspondent Node) à travers les paramètres du protocole SDP (Session Description Protocol). Dans ce mode de transfert, le NM doit rester actif pour maintenir et contrôler la session.

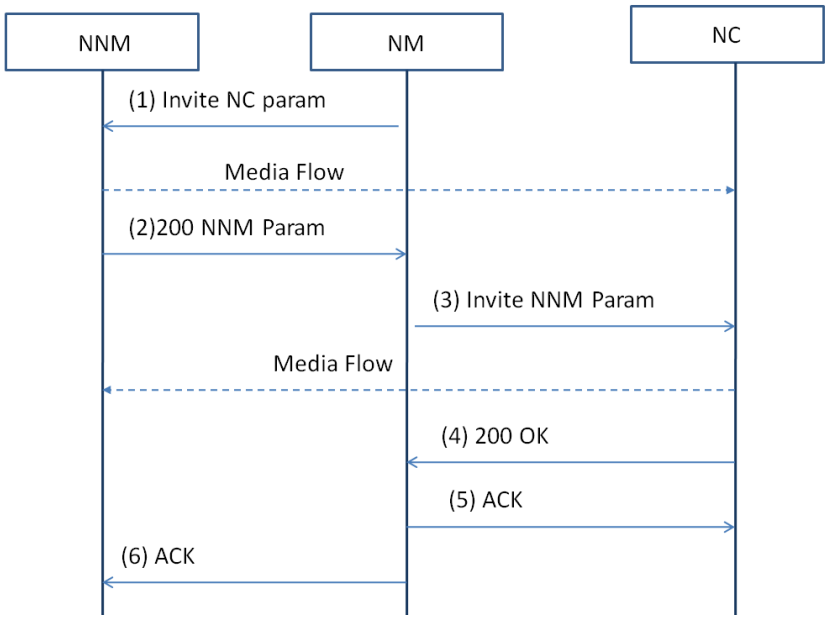

Figure 10: Transfert de session en mode contrôle

Le diagramme de séquence (Figure 10) illustre les différents messages échangés au cours d'une mobilité de session avec contrôle. Le NM, ayant déjà une session active avec le NC, transfert le flux media vers un nouveau nœud NMN (New Mobile Node). Pour réaliser ce transfert, le MN établit une session avec NMN en envoyant un message INVITE qui contient les paramètres SDP du NC. Suite à la réception de ce message, Le NMN répond par un message 200 OK qui inclut ses propres paramètres. Ensuite, le NM retransmet les paramètres du NMN au NC pour qu'il mette à jour la session SDP.

Mode de transfert de session sans contrôle (Figure 11): dans ce cas, la session est transférée complètement d'un terminal vers un autre tout en garantissant la continuité de la communication. Dans ce mode, la méthode REFER est utilisée [42] pour initier le transfert de la session du NM au NMN. Ce message contient l'adresse SIP du NC à contacter et les paramètres de la session en cours. Les informations relatives à la session en cours sont introduites dans l'entête «REPLACES » et doivent être incluses dans la requête INVITE (message 3). Lorsque la connexion est établie entre le NMN et le NC, le NMN informe le NM par le biais d'une requête NOTIFY. Et finalement, le NM envoie un message BYE au NC pour libérer la connexion.

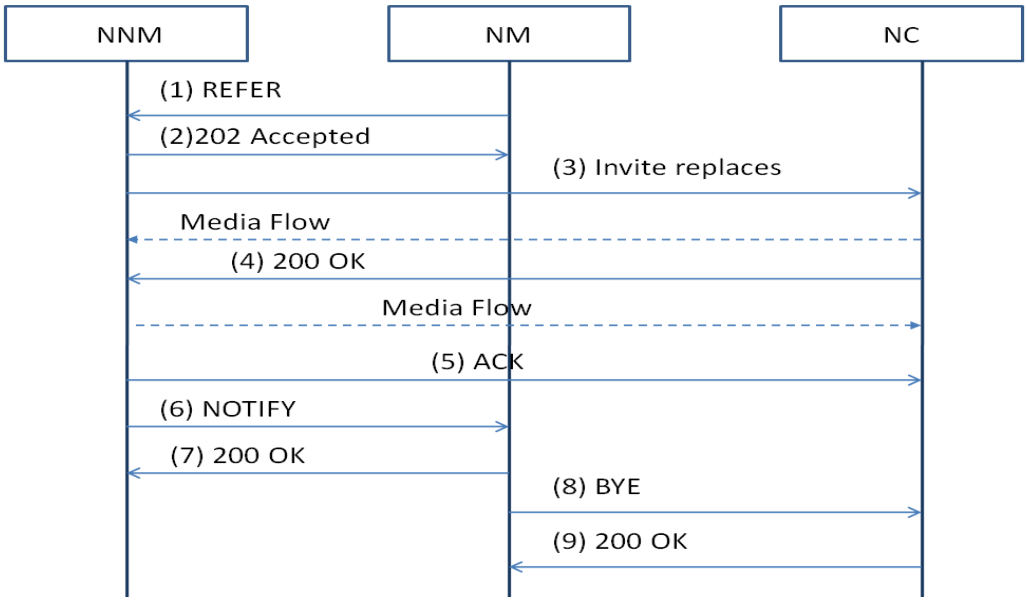

Figure 11: Transfert de session en mode sans contrôle

- Changement de réseaux d'accès:

Dans la mobilité de la session, nous trouvons aussi la mobilité de l'utilisateur qui permet de maintenir une communication en cours lors d'un changement du réseau d'accès. Pour maintenir ce type de session, le message re-INVITE [38] est utilisé. Le NM qui est responsable du changement de réseau d'accès envoie un message re-INVITE qui a exactement le même format qu'un message INVITE classique avec le même Call-ID ce qui permet de modifier la session en cours.

Discussion:

Nous avons décrit la mobilité de la session dans le cas de la mobilité du terminal et celle de l'utilisateur. Pour répondre au besoin de la session «user-centric» en termes de personnalisation des services, nous incluons en plus des deux types de mobilité, la mobilité de service relative au changement de composants de service dans différents plates-formes tout en maintenant le service à l'utilisateur.

III.5 Conclusion: Limite des solutions actuelles

	Les nouveaux besoins	Etat de l'art	Limitation des solutions actuelles	Défis/ Verrous
Organisation	Une autogestion de la QoS de bout en bout	-Autonomique computing: systèmes auto-organisés pour réduire les coûts de maintenance des applications. -Architectures autonomiques : middleware autonomique pour assurer la coordination entre les applications -Composant de service autonomique : adaptation des systèmes selon les besoins fonctionnels	-Pas de gestion de la QoS pour contrôler le comportement des composants de service pendant l'exécution - Gestion centralisée par un orchestrateur dans les architectures autonomiques - Pas d'autogestion selon les besoins non-fonctionnels	- Une organisation avec le maximum *d'autosuffisance,d'automatisation,*et *d'auto-gestion* - Une gestion *décentralisée de la QoS pour avoir plus d'adaptabilité lors de la session de l'utilisateur* en cas de mobilité ou de changement de préférences
Service	Services personnalisés selon les besoins fonctionnels, non-fonctionnels (QoS) et les préférences de l'utilisateur	La prise en compte de la QOS dans: - Les architectures SOA - La sélection de service basée sur la QOS	- Gestion centralisée de la QoS par un orchestrateur qui gère les ressources des applications pendant l'usage. - les applications sont fournies à l'utilisateur de la même manière. - Pas de gestion de la QoS pendant la mobilité de service.	- Gestionnaire de QoS décentralisé. - Composition des services appartenant à des plates-formes trans-organisationnlles.
Session et Protocoles	- Session unique et continue - Le Service delivery.	- Le multimédia service delivery. - SIP : négociation de la QoS media.	- Gestion de la QoS au niveau transport -Une session par application	- Une session par une composition de services personnalisés -Continuité de service durant les mobilités et les changements de préférences de l'utilisateur. - Un nouveau protocole pour la négociation de la QoS applicative d'un service personnalisé

Chapitre IV Du Background vers les propositions

Avant d'exposer nos propositions, nous allons tout d'abord présenter les travaux de notre groupe de travail sur lesquels reposent nos contributions.

Pour offrir aux usagers de terminaux fixes ou mobiles des services personnalisés sensibles aux contextes, il faut une architecture capable de garantir la fourniture de ce type de service. Ce sujet a été l'objet de nombreuses recherches effectuées par le groupe AIRS (Architecture et Ingénierie de Réseaux et Services). Une architecture UBIS (§IV.1) a été proposée pour répondre aux besoins de l'utilisateur «user-centric». La mise en œuvre d'une telle solution architecturale a nécessité une prise en compte de tous les niveaux de visibilité: utilisateur, équipement, connectivité et service. Pour modéliser les quatre niveaux de visibilité, un modèle <NLR> (Nœud, Lien, Réseau) a été proposé (§IV.2), qui représente de manière simple et générique tous les éléments qui composent le réseau de chaque niveau de visibilité. Pour décrire les entités du monde réel à savoir le nœud, le lien et le réseau, un modèle informationnel (§IV.3) a été proposé pour représenter l'architecture matérielle, logicielle et les services qu'offre chaque entité.

IV.1 Architecture UBIS

L'architecture UBIS (Figure 12) fournit une séparation claire entre les niveaux de visibilité, qui se décline en quatre niveaux, celui de l'utilisateur, celui des services, puis celui des réseaux y compris les réseaux d'accès et cœur et pour finir celui des équipements dans lequel nous trouvons entre autres les terminaux de l'utilisateur.

Chaque niveau de visibilité est constitué d'un enchaînement de composants de même nature dénommés VPxN (virtual private - x=Service, Connectivity, Equi)pment, User-Network):

- VPSN: ce niveau est constitué d'un enchainement des composants de service élémentaires. Le lien représente les interactions entre les services au niveau logique, les entités de ce niveau rendent un service applicatif.
- VPCN: représente le réseau de point de vue logique. Il regroupe les services de couches 1, 2, 3 du modèle OSI nécessaires à la réalisation de la fonction de transport.
- VPEN: Les nœuds représentent les équipements physiques qui sont les terminaux, et les serveurs d'application. Le lien est le câblage physique entre eux.
- VPUN: représente les usagers, les clients, les fournisseurs.

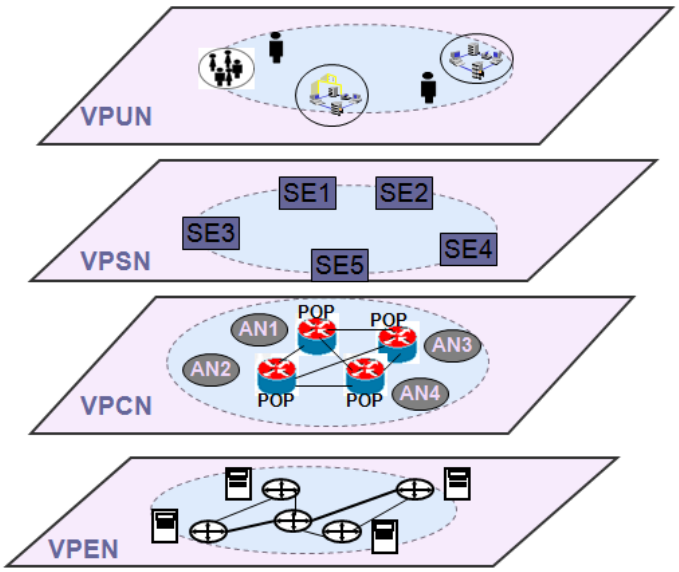

Figure 12: Architecture UBIS

IV.2 Modèle NLR

Le modèle «NLR» permet de modéliser le monde réel qui est complètement hétérogène à travers des réseaux virtuels, qui sont constitués des nœuds et des liens de même nature. Cette modélisation décrit d'une façon claire toutes les ressources impliquées dans la délivrance d'un service de bout en bout. Le méta-modèle «Nœud-Lien-Réseau» est composé de trois objets abstraits:

- **Nœud**: défini l'élément responsable d'un traitement spécifique. Dans le monde réel, il peut être représenté par un terminal, un routeur, ou un composant de service, etc.
- **Lien**: représente la liaison virtuelle de communication qui permet l'interaction entre les différents nœuds. Dans le monde réel, il peut être représenté par une liaison physique (câble), une liaison logique etc.
- **Réseau**: représente l'ensemble de nœuds et de liens de même nature qui offrent, d'une manière transparente, le service global. Il assure la coopération entre les nœuds à travers les liens pour fournir un certain service. Chaque réseau est défini par son niveau de visibilité (Utilisateur, terminal, réseau et services).

IV.3 Modèle informationnel

Dans l'objectif d'avoir une représentation efficace du monde réel qui est hétérogène et mobile, un modèle informationnel a été proposé [43] qui représente une structuration d'informations uniforme de n'importe quelle entité du monde réel (NLR) sur un niveau de visibilité donné. Ce modèle regroupe les informations importantes qui permettent de prendre les décisions au cours de la session de l'utilisateur au bon moment et au bon endroit, par rapport aux besoins de l'utilisateur et de l'évolution de son contexte d'utilisation. Ces informations décrivent toutes les ressources et surtout les connaissances des aspects comportementaux, c'est-à-dire, tout ce qui est relatif à la qualité de service.

Ce modèle informationnel permet la description de chaque entité du monde réel à savoir le nœud, le lien et le réseau, sur un niveau de visibilité donné « V » suivant le concept abstrait et générique « Network Element (V) (NE (V)) [43]» à savoir le nœud, le lien et le réseau sur un niveau de visibilité donné. Pour chaque « NE » (Figure 13), le modèle décrit son architecture matérielle, logicielle ainsi que les services qu'il offre.

Un NE (Figure 13) se compose d'une partie architecture et d'une partie service. La partie architecture se compose d'éléments logiciels qui exécutent et qui rendent le service demandé. Il se compose aussi d'éléments hardwares qui matérialisent les contraintes du composant. Un élément logiciel se compose de : L'entité (classe entité) qui représente les fonctions de base du NE, de sa propre gestion (classe gestion) permettant de gérer le service rendu par l'entité, de sa table de connexion (classe connexion) qui matérialise les relations entre les composants,

et des adresses (classe SAP) par lesquels les services du composant peuvent être fournis ou demandés. Et la partie service représente les interfaces. Ils permettent aux NE de répondre aux demandes de services ou d'envoyer des messages aux entités du monde extérieur. Selon la nature des opérations invoquées et les services fournis, l'interface peut être de type usage, contrôle ou gestion.

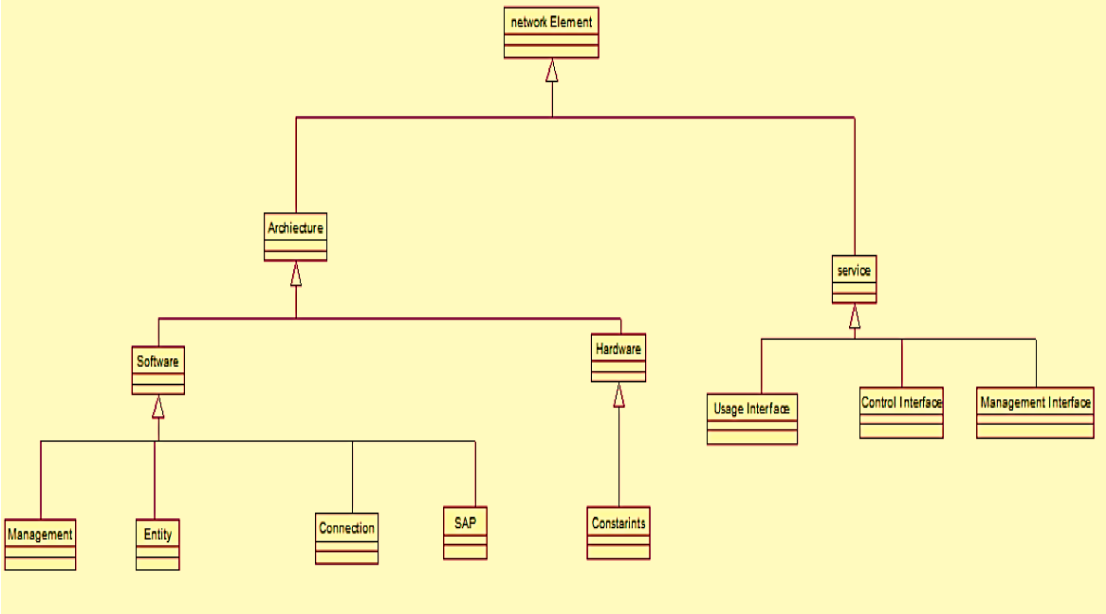

Figure 13: Modèle informationnel

Dans les futurs services à concevoir, il faut que les traitements et les données soient indépendants de l'application qui les utilise. C'est pour cette raison, qu'il est important d'avoir un modèle informationnel qui inclut en plus de la partie traitement, la partie gestion. Plusieurs profils pour la gestion de ressources ont été définis et ils sont sollicités pendant les différentes phases du cycle de vie d'un composant de service : la phase de conception, la phase de déploiement et la phase d'exploitation.

- Le profil de ressource est utilisé pendant la phase de conception d'un composant de service, il contient les valeurs de conception des critères de QoS.
- Le profil d'usage de ressources est utilisé pendant la phase de déploiement pour déployer les composants de service au bon endroit.

- Le profil « Real Time Profile » (RTP) est utilisé pendant la phase d'exploitation d'un service. Le RTP permet de représenter l'image réelle des ressources tout en tenant compte de la QoS.

Chapitre V Propositions

Après avoir analysé le contexte, et rapporté les principaux résultats de l'existant et du background, nous allons consigner dans ce chapitre nos propositions qui visent à offrir à l'utilisateur des services personnalisés, avec une continuité de session et une QoS de bout en bout, et cela dans un contexte qui est totalement hétérogène et mobile. Nos contributions sont d'ordre organisationnel, fonctionnel et protocolaire. Elles se subdivisent en trois principales propositions. La première est relative à la dimension organisationnelle (§V.1), qui adresse notre problématique d'avoir une architecture décentralisée avec le maximum de flexibilité et de dynamicité, afin de garantir une continuité de service dans un environnement hétérogène et avec des changements dynamiques. La seconde proposition est relative au composant de service autonomique (§V.2), dans cette partie nous repensons la notion du « composant de service » et nous proposons un composant de service qui soit auto-gérable et auto-adaptable grâce à un agent-QoS, pour faciliter le remplacement d'un composant de service par un autre composant ubiquitaire sans impacter la QoS du service global. La dernière proposition est la dimension protocolaire (§V.3), dans cette partie nous nous sommes intéressés aux besoins d'une session de service guidée par les préférences de l'utilisateur. Nous proposons un modèle protocolaire pour assurer une interaction plus flexible entre les différents acteurs de l'architecture, et cela pour faciliter la création et la modification d'une session de service unique et continue.

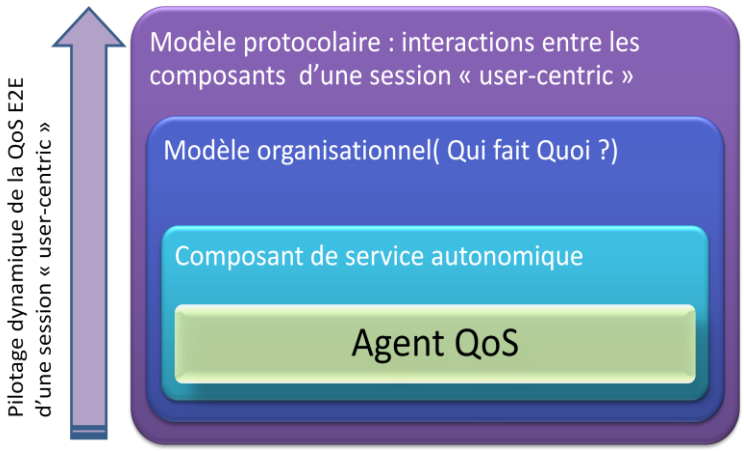

Figure 14: Propositions

V.1 La Dimension organisationnelle

Nous voulions, à travers ce chapitre, faire valoir l'idée que l'identification des acteurs et la séparation des rôles constitue une première solution efficace aux problématiques des architectures d'aujourd'hui, à savoir, la dynamicité, la réactivité, la flexibilité et la QoS de bout en bout. Donc, une nouvelle organisation permet de mieux piloter dynamiquement la qualité de service dans un nouvel environnement de l'utilisateur qui est totalement hétérogène et mobile. Ce chapitre est essentiellement consacré au développement d'un modèle organisationnel qui répond à la question « qui fait quoi ?» dans une architecture de volume et complexité importante. Ce modèle est bâti sur l'architecture UBIS développée par notre groupe de travail qui est structurée en niveaux de visibilité, qui sont des niveaux d'abstraction permettant de différencier les services rendus.

Notre proposition est la répartition des responsabilités de la QoS selon ses niveaux de visibilité de l'architecture UBIS pour faire apparaître les éléments architecturaux impactant la qualité de service de bout en bout:

- Le niveau responsable du service rendu par les équipements physiques est le VPEN, il gère la QoS des ressources physiques tels que la mémoire et la CPU pour garantir un computing delivery.

- Le niveau responsable du service rendu par le réseau de transport est le VPCN, il gère la QOS des couches 1, 2, 3 du modèle OSI responsables de la réalisation des fonctions de transport. Ce niveau est responsable du media delivery, il garantit la QoS du réseau de transport d'un flux média.
- Le niveau responsable de service rendu par le service applicatif est le VPSN, il gère la QoS du composant de service liée au traitement de l'information pour assurer un service Delivery.
- Et le niveau responsable du contrat de service établi entre les utilisateurs et les fournisseurs de service est le VPUN, il gère la QoS demandée par l'utilisateur représenté par son SLA (Service Level Agreement), et il gère aussi ses préférences qui peuvent changer au cours de la session.

Notre préoccupation est de définir une organisation capable de piloter dynamiquement la QoS de bout en bout à travers ces différents niveaux de visibilité. Dans la section suivante, nous allons présenter le modèle organisationnel de l'architecture UBIS (§V.1.1) pour répondre au besoin de l'utilisateur dans ce nouveau contexte qui est totalement hétérogène et mobile. Ce modèle se base sur l'association de deux concepts clés: les composants de service qui représentent les acteurs (§ V.1.2), et les rôles que joue l'agent QoS (§V.1.3) par rapport à la prise de décision lors des différents changements qui se produisent au cours de la session de l'utilisateur, et qui peuvent être liés à la mobilité ou bien à un dysfonctionnement d'un service.

V.1.1 Modèle organisationnel de l'architecture UBIS: Qui fait Quoi?

Dans l'architecture UBIS tout est considéré comme un service. A chaque niveau de visibilité de l'architecture, on peut trouver des composants de service qui rendent un service de type contrôle «CSC», Applicatif «ASC» ou gestion «GSC» qui participent aux différentes phases de la session «user-centric».

Dans une architecture de volume et complexité importantes, il est intéressant d'avoir multiples points de contrôle et de gestion qui participent au maintien du service avec le niveau de la QoS requis au cours de la session de l'utilisateur, et cela suivant une approche Top-

Down du niveau service (VPSN) au niveau utilisateur (VPUN) pour assurer un pilotage dynamique de la qualité de service de bout en bout.

Cet aspect organisationnel nous conduit à prendre comme notion clé l'agent-QoS qui sera intégré dans tout type de services de l'architecture à tous les niveaux de visibilité, c'est-à-dire que chaque composant de chaque niveau architectural sera piloté et invoqué en fonction de son comportement (QoS), qui est contrôlé et géré par son agent de QoS au cours de la session UBIS.

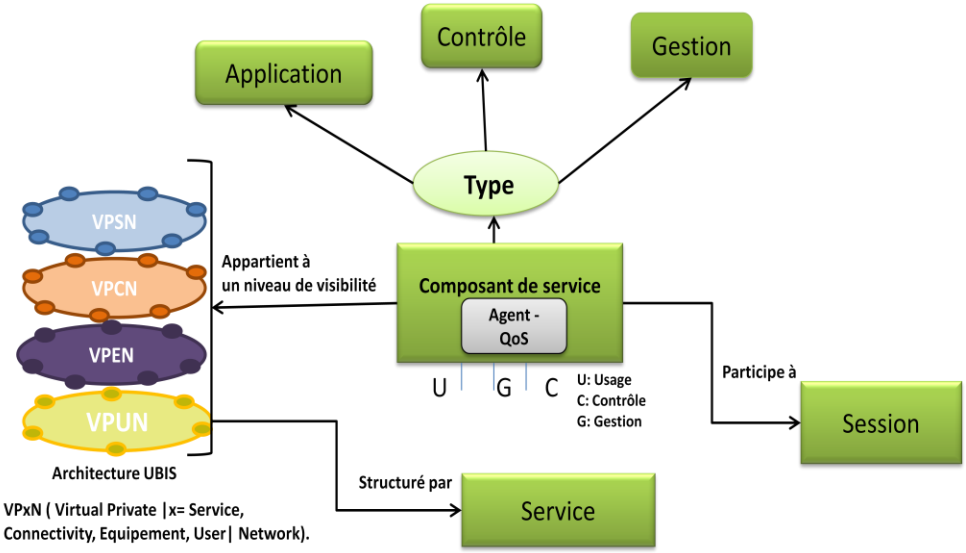

Figure 15: Modèle Organisationnel

.

V.1.2 Acteurs (Composant de service)

Le modèle organisationnel de l'architecture repose sur l'attribution des acteurs et cela pour améliorer la capacité d'actions lorsqu'un changement de QoS apparait suite à la mobilité (utilisateur, terminal, service, session) ou à un dysfonctionnement d'un composant de service. Pour ce faire, chaque acteur doit jouer un rôle organisationnel au sein du niveau de visibilité auquel il appartient (cross-layer) ou bien entre les niveaux de visibilité (intra-layer) pendant la phase de l'exploitation pour participer au processus d'adaptation et du maintient de la QoS de bout en bout pour l'utilisateur final, et il peut être:

- *Initiateur*: représente l'entité qui détecte le changement de QoS grâce à son agent de QoS au cours de la session d'un utilisateur, et se charge d'informer le décideur qui puisse se trouver soit dans le même niveau de visibilité N ou bien dans un niveau de visibilité supérieur (N+1) ou inferieur (N-1), pour qu'il prenne les décisions nécessaires pour maintenir la chaine de QoS de bout en bout.
- *Décideur*: représente l'entité qui se charge de prendre les mesures nécessaires pour alimenter les informations décisionnelles, conformément à ses responsabilités et cela en fonction de son niveau de visibilité.
- *Exécuteur*: représente l'entité qui contrôle le changement de la QoS et effectue les adaptations nécessaires.

V.1.3 Les rôles de l'agent de QoS

L'agent de QoS proposé est un élément générique qui offre une autonomie de traitement de la qualité de service à un composant de service pour avoir une meilleure performance de la QoS au cours de la session de l'utilisateur. Il prend en charge le processus de mesure des ressources internes et également le processus de communication, il assure une coopération au niveau horizontal entre les composants de service quand il s'agit des composants du même niveau de visibilité, et une coordination pour assurer l'agrégation des besoins QoS entre les différents niveaux de visibilité.

L'agent de QoS est une entité stable et selon la stratégie de l'organisation et les besoins au cours de la session des rôles lui sont attachés et qui peuvent être de type : Passif, Actif, Interactif ou Proactif.

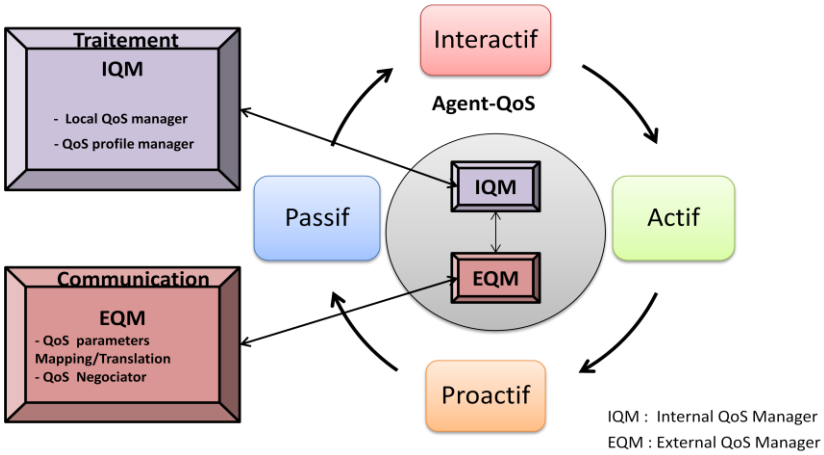

Figure 16: Rôles et fonctionnalités de l'agent QoS

- Le *rôle passif* désigne l'état où l'agent QoS assure uniquement le traitement interne; il mesure la QoS et met à jour les valeurs de QoS courantes. Dans le cas de non-respect du contrat de QoS c.-à-d. le dépassement des valeurs seuils, il ne communique pas son état à son environnement sauf dans le cas où il est sollicité par sa communauté pour l'informer dans quel état il se trouve.
- Le *rôle Actif* désigne l'état où l'agent-QoS joue le rôle de métrologue et de controleur de la QoS, et il notifie régulièrement, à qui a le droit, le statut de QoS de son composant de service, c'est-à-dire s'il respecte toujours son contrat de service ou bien s'il est hors de contrat de service (In contrat/Out contrat).
- Le *rôle Interactif* désigne l'état où l'agent-QoS est doté d'une capacité d'interactions avec d'autres Agent-QoS. Il peut négocier les paramètres QoS à maintenir dans un même niveau de visibilité, ou bien entre les composants de service à activer dans les autres niveaux de visibilité pour assurer le mappage entre les différents paramètres de QoS et par conséquent avoir une solution de bout en bout.
- Le *rôle Proactif* désigne l'état où l'agent QoS possède les connaissances et les règles qui lui permettent de prendre les décisions tout seul, pour remedier à un problème qui lui est propre ou qui relève de son domaine de responsabilité, et il doit envoyer des notifications à qui de droit.

V.1.4 Les fonctionnalités de l'agent QoS

Comme nous l'avons déjà présenté précédemment, L'agent QoS est une entité autonome intégrée dans chaque composant de service qui mesure la QoS en temps réel pour détecter la défaillance au bon moment durant les changements (mobilité, préférences, dégradation). Nous expliquons dans ce qui suit, les éléments fonctionnels de l'agent de QoS qui sont, d'une part, liés au traitement interne pour assurer une réservation des ressources selon la demande de l'utilisateur, et d'autre part, liés à la communication pour assurer une coordination et une coopération entre les différents composants de services appartenant aux différents niveaux de visibilité de l'architecture UBIS. L'agent QoS s'appuie sur deux éléments fonctionnels de gestion: IQM (Internal QoS manager) et EQM (External QoS manager)(Figure 16):

L'IQM est en charge de la gestion et le contrôle local de la qualité de service spécifique pour chaque composant de service. Il surveille les ressources internes du composant de service pour maintenir la QoS. Les fonctions principales d'IQM proposées sont les suivantes:

- *Local QoS Manager* : informe le plan de contrôle des ressources dont il a besoin pour assurer son traitement interne.
- *QoS Profile Manager* : gère le profil de la qualité de service pour chaque élément de service. Il assure en temps réel le traitement nécessaire pour connaître le statut du composant de service en termes de contrat de QoS pendant l'exploitation (In contrat/Out contrat). En effet, l'IQM s'interface avec les bases de données utilisateur et opérateur nommées respectivement Infosphère et Infoware pour l'échange d'informations nécessaires au traitement interne.

La principale fonction de l'EQM est la communication et la coordination entre les ressources QoS des différents niveaux de visibilité et il permet également le mappage des paramètres QoS entre les différents nœuds de l'architecture, ce qui permet une continuité de session en cas de dégradation de la qualité de service et ce quel que soit le type de mobilité. Les fonctions principales d'EQM proposées sont les suivantes :

- QoS Mapping/Translation Parameters : assure le mappage et la translation des mécanismes et paramètres de QoS utilisés. Ce qui permet la déclinaison des

paramètres QoS entre les niveaux de visibilité. L'opération de QoS mapping/translation s'exécute pendant la phase d'approvisionnement lors de l'établissement de la session, et également lorsqu'un changement se produit pendant la phase de l'exploitation lié à la mobilité ou à une dégradation de la qualité de service.

- QoS Negociator : offre une fonction de communication et de coordination verticale entre les niveaux de visibilité pour demander les ressources qui lui sont nécessaires ou les libérer. La communication horizontale assure le signalement d'activité ou de non activité du composant.

Nous avons présenté dans cette partie, d'une manière générale, le modèle organisationnel de l'architecture UBIS. Dans ce qui suit, nous présentons le composant de service autonomique (§V.2) qui est un élément clé du pilotage dynamique de la QoS de l'architecture UBIS car comme nous l'avons précisé précédemment tout est service.

V.2 Composant de service autonomique

V.2.1 Introduction

Dans l'objectif de satisfaire le contrat SLA établi entre les clients et les fournisseurs, et d'améliorer le «Service Delivery» au cours de la mobilité de la session, un pilotage dynamique de la QoS de bout en bout d'un service personnalisé est souhaité. Pour ce faire, nous présentons dans cette partie un composant de service autonomique, qui permet grâce à son agent de QoS de répondre aux besoins d'automatisation, d'autosuffisance et d'autogestion nécessaires pour garantir la continuité de service lors des différents types de mobilité. En effet, sa tâche principale est d'intégrer le contrôle et la gestion de la QoS au niveau de chaque composant de service de l'architecture et cela pour avoir un contrôle et une gestion décentralisé de la QoS. L'avantage de cette solution est de concevoir un composant de service capable de réagir dynamiquement et de manière autonome en temps réel suite à un changement dans le contrat de QoS durant la session de l'utilisateur, telle que la disponibilité. Ce composant jouera un rôle important dans l'obtention d'une composition de service plus flexible et avec une meilleure performance de QoS. Pour expliquer notre proposition, nous détaillons tout d'abord dans la section (§V.2.2) l'aspect architectural et fonctionnel du

composant de service autonomique. Ensuite, nous expliquons dans la section (§V.2.3) le modèle QoS qui est la base de l'autogestion de composant de service. Et finalement, nous expliquons dans la section (§V.2.4) les mécanismes appliqués au cours de chaque étape opérationnelle, qui permettent à un composant de service d'autocontrôler et d'autogérer ses propres ressources, pour avoir une réaction plus flexible durant la session de l'utilisateur, et cela pour maintenir le «service Delivery» avec la QoS requise.

V.2.2 Structure d'un composant de service autonomique

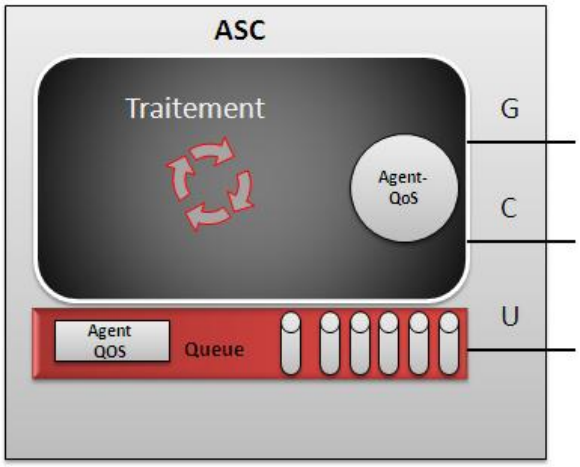

Figure 17: Composant de service autonomique

L'architecture orientée services (SOA) joue un rôle principal dans la création rapide des applications grâce à une composition des composants de service indépendants. Ces composants fournis par différents fournisseurs peuvent offrir des fonctionnalités identiques mais avec des capacités de QoS différentes. Le composant de service autonomique «ASC» que nous proposons est une nouvelle vision de l'architecture SOA. Grâce à son agent de QoS, il a les capacités de contrôle et de gestion de la QoS lors de la session de l'utilisateur pour améliorer le service delivery.

Le composant de service autonomique a les caractéristiques suivantes: Stateless, Mutualisable, Autonome et Auto-gérable.

- *Stateless*: un composant de service est stateless s'il effectue les mêmes traitements (opérations) pour toutes les demandes provenant de différents utilisateurs et cela sans conserver les données ou l'état spécifique à chacun d'eux. Cette caractéristique est importante dans le cas où il est nécessaire de faire un remplacement dynamique d'un composant de service qui a subi une dégradation de la QoS au cours de la session d'un utilisateur, pour que les données et l'état relatifs à une requête précédente ne perturbent pas le traitement de la nouvelle requête.
- *Mutualisable*: le composant de service est mutualisable parce qu'il est conçu pour traiter plusieurs demandes provenant de différents utilisateurs en même temps selon ses capacités. Pour assurer le contrôle et la gestion de ce partage de ressources, on associe une file d'attente (Figure 17) au niveau du plan d'usage du composant de service, cette file d'attente permet de provisionner toutes les requêtes acceptées des utilisateurs. Les ressources relatives à la file d'attente sont contrôlées et gérées par un agent de QoS intégrés dans la file.

Le composant de service est autonomique parce qu'il est à la fois autonome et auto-gérable.

- *Autonome*: Le composant est autonome parce qu'il est fonctionnellement indépendant c'est-à-dire il est auto-suffisant et il n'a pas besoin d'autres composants de service pour accomplir ses fonctionnalités. L'intérêt de cette indépendance fonctionnelle entre les composants de service est de faciliter le changement d'un composant de service par un composant de service ubiquitaire en cas de détérioration de la QoS ou de dysfonctionnement pendant l'exploitation et cela sans impacter le service global demandé par l'utilisateur.
- *Auto-gérable*: Le composant de service est auto-gérable parce qu'il surveille sa propre QoS et il gère aussi ses états liés à l'utilisation des ressources pendant la session utilisateur. Le composant de service a, à un instant t, un de ces quatre états Indisponible, Disponible, Activable, Activé (Figure 18).
  - L'état indisponible signifie que le composant de service est temporairement ou définitivement inaccessible,
  - L'état disponible signifie que le composant de service peut être accessible,
  - L'état activable signifie que le composant de service est prêt à être activé,

- Et l'état activé signifie que les ressources du composant de service sont en cours d'utilisation. Lorsque le composant de service est activé, son agent de QoS, inclus dans son plan de gestion, mesure et contrôle le contrat de QoS établi entre l'utilisateur et le fournisseur de service.

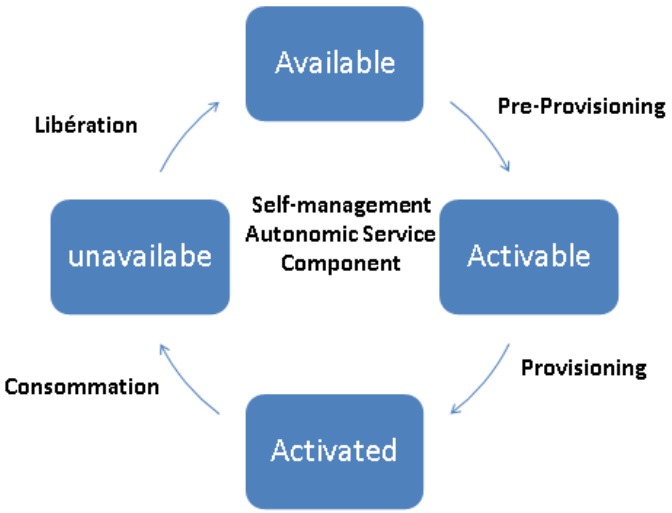

Figure 18: les états d'un composant de service autonomiques

Le composant de service autonomique (ASC) est structuré selon 3 plans: contrôle, usage et gestion:

- Le plan d'usage contient les fonctions principales réalisées par le composant de service pour rendre le service, il comprend tous les mécanismes utilisés pour le traitement d'une demande pendant la phase de consommation.
- Le plan de contrôle contient tous les mécanismes, utilisés d'une manière asynchrone durant la session de l'utilisateur, qui permettent de réserver les ressources du composant de service nécessaires au traitement de la demande de l'utilisateur.
- Le plan de gestion contient les mécanismes appliqués d'une manière asynchrone sur les données, pour gérer et contrôler les ressources du composant de service pendant la phase de consommation.

Un composant de service dispose de trois interfaces : usage, contrôle et gestion. Pour obtenir des processus transverses aux trois plans afin d'avoir l'automatisme nécessaire à la

dynamicité et à la continuité requises par la session, nous avons une interface convergée entre les différents plans du composant de service. Chaque interface convergée définit un jeu d'opérations et d'actions entre deux interfaces élémentaires pour avoir plus de dynamicité, de flexibilité et d'adaptabilité auprès du service à rendre.

- La flexibilité est obtenue grâce à la convergence des deux plans: usage et contrôle, pour prendre en considération les préférences temporelles et spatiales de l'utilisateur au cours de la session. Cette convergence permet d'assurer une surveillance des paramètres de performance pour garantir la QoS pendant la phase d'exploitation avec une réaction en temps réel dans le cas d'une mobilité ou bien d'un changement lié aux préférences de l'utilisateur.

- L'adaptabilité est obtenue grâce à la convergence du plan d'usage et du plan de gestion. Son enjeu majeur est d'assurer l'interopérabilité entre les composants de service dans le cas où un changement se produit au cours de la session. Elle a pour objectif la réservation des ressources en adéquation avec la QoS requise pour le traitement de la demande des utilisateurs. Cette convergence offre une coordination entre les informations du plan d'usage et celles du plan de contrôle pour appliquer les adaptations nécessaires à la continuité de service avec la qualité de service requise.

- Et la dynamicité est obtenue grâce à la convergence du plan de gestion et du plan de contrôle. Elle permet de fusionner les informations de gestion (détection de défaillance, localisation de dysfonctionnement, dégradation QoS) et l'automatisme du plan de contrôle (signalisation) pour réagir le plus rapidement possible pendant la phase de l'exploitation et maintenir la conformité du SLA. Grâce à cette convergence l'agent de QoS intégré dans le plan de gestion joue également un rôle dans le contrôle de la QoS au niveau du plan de contrôle.

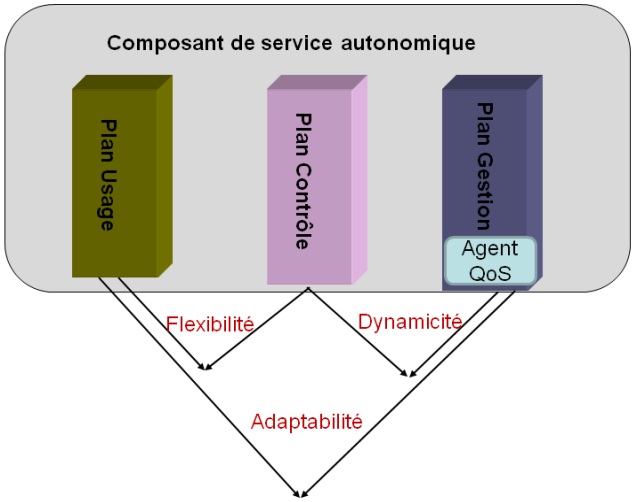

Figure 19: Convergence des plans

V.2.3 Le Modèle de QoS

Pour évaluer le comportement de QoS de bout en bout au cours de la session utilisateur, il est nécessaire d'avoir une expression homogène de la qualité de service d'un composant de service. Pour ce faire, chaque composant de service instancie un modèle de QoS pour gérer les ressources service en temps réel et leurs possibles dégradations. Cette solution semble efficace pour répondre au besoin de l'automatisation et de distribution du contrôle et de gestion de la QoS au niveau service de l'architecture.

Pour cette raison nous avons proposé un modèle de QoS qui décrit la QoS d'un composant de service d'une manière homogène et uniforme. Le comportement de chaque composant de service est reflété par les paramètres de QoS mesurables qui sont classés selon un vecteur de quatre critères: Disponibilité, fiabilité, délai et capacité.

- *Disponibilité* «**A**» indique le taux d'accessibilité d'un composant de service, elle indique le pourcentage de temps qu'un composant est en mesure d'accepter des demandes

$$A = 1 - U / T$$

Avec **U** et **T** qui représentent respectivement le nombre de demandes qui ont été rejetées par un composant de service, et le nombre total de demandes envoyées sur une période de temps.

- *Fiabilité* «**F**» représente la capacité d'un composant de service à fonctionner correctement sans modification de l'information traitée, elle indique le pourcentage des invocations réussies sur une période de mesure.

$$F = 1- R/ T$$

Avec **R** qui indique le nombre de demandes qui ont échoué pendant une période de temps.

- *Délai* **D** représente la durée moyenne pour traiter une demande par un composant de service.
- *Capacité* **C** représente le taux de charge maximal supporté par un composant de service.

Ces critères de QoS sont tous nécessaires et suffisantes pour qu'un composant de service s'autocontrôle et s'autogère. Ces critères sont évalués à travers trois types de valeurs mesurables : les valeurs de conception, les valeurs courantes et les valeurs seuils.

- Les *valeurs de conceptions* sont déterminées au moment de la conception du composant de service, elles définissent les capacités maximales de traitement d'un composant de service.
- Les *valeurs courantes* sont utilisées pendant la phase de l'exploitation pour surveiller le comportement d'un composant de service au cours du traitement des demandes.
- Les *valeurs seuils* indiquent les valeurs limites à ne pas dépasser par un composant de service pour qu'il assure son traitement normalement.

Ces informations de QoS sont utilisées pour prendre les décisions pendant la phase de l'exploitation. En effet, pour prendre les bonnes décisions au cours de la session de l'utilisateur, au bon endroit et au bon moment, il est nécessaire d'avoir une représentation efficace du monde réel.

Pour cette raison, nous avons besoin d'avoir une structure d'informations uniforme qui contient à la fois la description du composant de service (fonctionnel) et aussi la connaissance de son comportement (non-fonctionnel QoS). Cette description est fournie par le modèle

informationnel qui contient les différents profils à solliciter durant les différentes phases opérationnelles d'un composant de service: provisioning et exploitation

- Pour la phase du pré-provisioning, nous avons le profil des ressources, qui contient les valeurs QoS de conceptions définies à la phase conceptuelle du composant de service.
- Durant la phase d'exploitation, le profil Real Time est instancié en temps réel pour avoir une gestion dynamique de la QoS, il contient les valeurs de QoS courantes qui seront comparées aux valeurs de QoS seuils pour contrôler le comportement d'un composant de service pendant l'usage.

V.2.4 Les mécanismes de gestion de la QoS

Dans l'objectif d'améliorer le «service Delivery», nous avons intégré des mécanismes de gestion de la QoS dans le composant de service autonomique. Nous avons proposé des mécanismes pour chaque phase opérationnelle:

Durant la phase du pré-provisioning, il faut sélectionner un composant de service en se basant sur les valeurs de conceptions. Pour cela, au moment de l'initiation de la session, chaque ASC procède par un contrôle d'admission de QoS « QAC» pour vérifier s'il dispose des capacités requises pour traiter une nouvelle demande d'un nouveau utilisateur. Cette étape permet d'identifier et de sélectionner les composants de services qui répondent aux besoins des utilisateurs en terme de QoS, parce qu'il existe plusieurs composants de service ubiquitaires qui offrent les mêmes fonctionnalités mais avec des QoS différentes.

Durant la phase du provisioning, le composant de service réserve les ressources nécessaires pour traiter la demande de l'utilisateur. Comme nous l'avons souligné précédemment, chaque composant de service contient une file d'attente qui stocke les requêtes acceptées de différents utilisateurs parce qu'il est mutualisable. Pour contrôler ce partage de ressources, nous devons appliquer un contrôle d'admission de QoS «QAC» pour accepter une nouvelle demande dans la file d'attente d'un composant de service. Le rôle du «QAC» est de déterminer si une nouvelle requête peut être acceptée dans la file d'attente, sans violer les SLA des requêtes qui sont déjà acceptées.

Durant la phase de consommation, il faut contrôler et gérer la QoS d'une façon dynamique et continue. Notre vision d'autogestion de la QoS est basée sur l'agent de QoS intégré dans le plan de gestion du composant de service, et qui mesure continuellement les ressources QoS relatives à son traitement interne. Chaque composant appartient à une communauté de service virtuel «VSC» qui contient des composants de service qui ont les mêmes fonctionnalités et QOS. En cas de dégradation de la QoS, l'agent QoS envoie un évènement à sa communauté VSC pour qu'elle assure son remplacement par un composant équivalent, et par conséquent adapter la session de l'utilisateur sans causer une détérioration du service fourni.

Chaque file d'attente d'un composant de service appartient à une communauté virtuelle des files d'attente «VQC» qui regroupe des files d'attente qui ont les mêmes ressources, si l'agent QoS de la file d'attente détecte qu'une requête a pris plus temps que prévu pour son traitement (délai), il notifie sa communauté VQC pour réapprovisionner cette requête dans une autre file d'attente pour ne pas impacter le temps de traitement des autres requêtes qui sont déjà provisionnées dans la file.

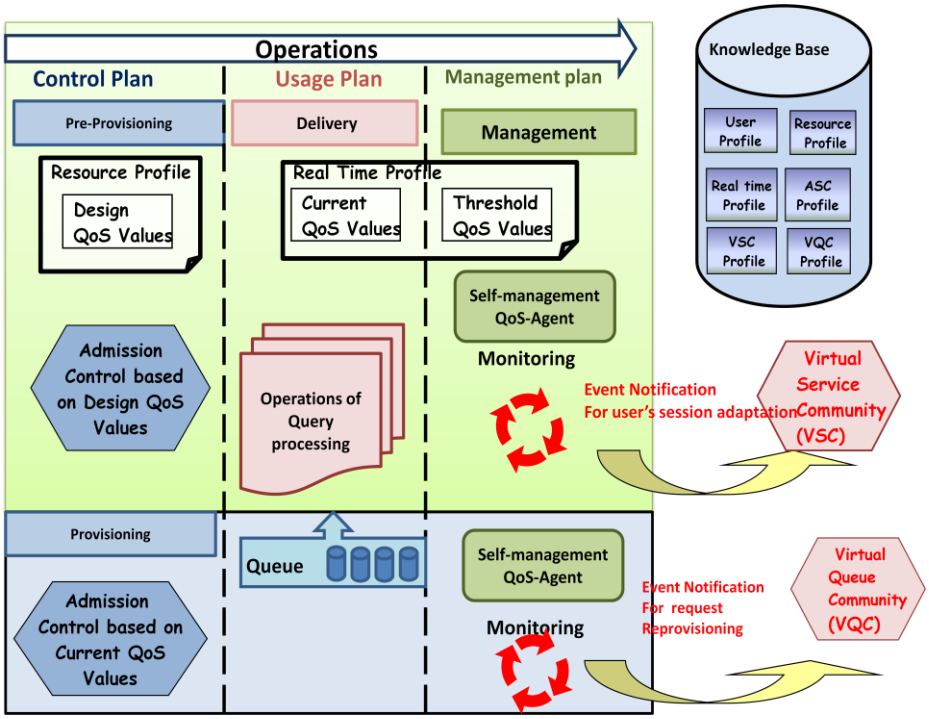

Figure 20: Mécanismes de gestion de la QoS

Nous détaillons dans ce qui suit via un automate, les mécanismes de gestion de la QoS utilisés dans chaque plan du composant de service autonomique au cours des différentes phases opérationnelles: le pré-provisioning, le provisioning, et le delivery.

Pendant la phase du pré-provisioning (Figure 21), chaque ASC procède par un contrôle d'admission pour vérifier s'il a les capacités requises pour traiter une nouvelle demande d'un utilisateur. Tout d'abord, il reçoit dans son plan de contrôle un message de signalisation pour attacher une nouvelle session de service. Puis, il incrémente la variable «Attach» à «Attach + 1» pour garder la traçabilité du nombre de sessions de service qui sont attachées à ce même service. Ensuite, l'agent QoS vérifie la QoS statistique qui est basée sur les valeurs de QoS de conception, afin de contrôler son attachement à une nouvelle session de services, car un composant de service peut être partagé entre plusieurs utilisateurs parce qu'il est mutualisable. Si le résultat est positif "OK", l'ASC change son état à activable dans le cas où il n'est pas précédemment rattachées à une session de service d'un autre utilisateur. Ensuite, il ajoute l'identifiant de la session utilisateur «ID VPSN» dans son profil dans la base de connaissance L'INFOWARE. Si le résultat est négatif "NOK", il faut décrémenter la variable «Attach» à «Attach-1" et ensuite le système se charge de trouver un autre composant de service ayant les mêmes fonctionnalités et une QoS équivalente pour répondre à la demande de l'utilisateur.

Pendant la phase de Provisioning (Figure 21), l'ASC reçoit la transaction dans son plan de contrôle, puis il incrémente la variable J à «J+1». Ensuite, il vérifie la QoS dynamique en se basant sur la QoS courante de la file d'attente afin de contrôler l'acceptation de la transaction de l'utilisateur dans la file d'attente. Si le résultat est positif, il provisionne la requête dans la file d'attente et il change son état de activable à activé dans le cas où il n'y a pas de requêtes provisionnées auparavant dans la file d'attente (c'est-à-dire J = 1), sinon il décrémente la variable de J à J-1.

Pendant la phase de consommation un composant de service peut ne pas continuer à fonctionner normalement ou à répondre aux exigences de l'utilisateur en termes de QoS. C'est pour cette raison que nous proposons des mécanismes pour surveiller et contrôler la QoS au niveau de chaque composant de service. Notre concept repose sur un agent de QoS intégré dans le plan de gestion de chaque composant de service. La (**Erreur ! Source du renvoi**

introuvable. Figure 21) montre les différents mécanismes utilisés lors de la phase de la consommation pour maintenir le service pour l'utilisateur avec le niveau de QoS requis.

Si l'ASC prend plus de temps pour traiter une requête (Timer 4 a expiré), cela peut donc affecter le temps de traitement des autres requêtes qui sont déjà provisionnées dans sa file d'attente. C'est pour cette raison qu'on utilise un agent de QoS pour surveiller et contrôler la QoS de la file d'attente. Lorsque l'agent QoS détecte une violation de contrat de QoS, il doit aviser la VQC (Virtual Queue Community) pour renvoyer la requête dans une autre file d'attente ubiquitaire. Il convient de mentionner que chaque composant de service appartient à un VSC (Virtual Service Community), qui contient un ensemble de composants de service autonomique ayant les mêmes fonctionnalités et une QoS équivalente, et aussi chaque ASC dispose d'une file d'attente faisant partie d'une communauté virtuelle de file d'attente VQC (Virtual Queue Community) qui contient des files d'attente ubiquitaires.

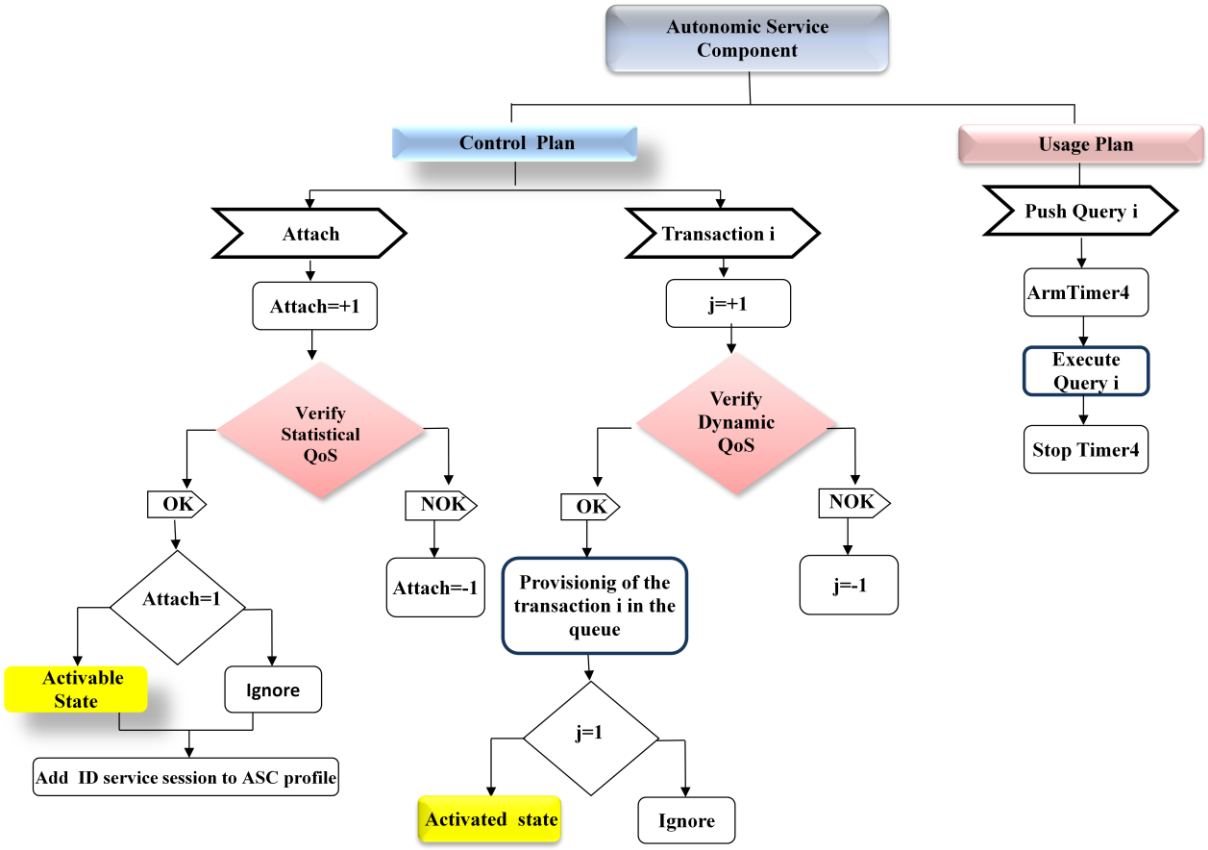

Figure 21: Automate du composant de service (plan d'usage et de contrôle)

Durant la consommation (Figure 22), l'agent QoS de chaque ASC qui compose le service global de l'utilisateur compare la QoS courante avec les valeurs de QoS seuils. Si le résultat est positif, l'agent QoS envoie un événement «IN Contract» à la communauté «VSC» pour notifier qu'il respecte toujours son contrat de QoS. Sinon, dans le cas où l'agent QoS détecte une dégradation de la QoS il envoie un événement «Out contrat» (Armer Timer2 et Timer3) à la communauté VSC afin de lancer l'algorithme de remplacement du composant de service qui est en cours d'utilisation par un composant de service ubiquitaire (Arrêter Timer2). Lorsque le Timer 3 expire, l'ASC change son statut de disponible à indisponible et il quitte sa communauté. Puis il rejoint une nouvelle communauté VSC selon sa QoS courante. Il change par conséquent son état de indisponible à disponible. Toutes ces opérations sont transparentes pour l'utilisateur final et sont effectuées automatiquement par le système lors de la session utilisateur.

Chaque ASC participe à la gestion de sa communauté VSC; il notifie régulièrement les autres membres sur son état de QoS (In contrat / Out contrat). Pour ce faire, si l'ASC reçoit un Out Contract il met son FlagOUT= 1 et il informe sa communauté VSC pour exclure l'ASC qui est hors contrat de la communauté et de procéder par la suite à son remplacement afin de maintenir la communauté VSC. Si l'ASC reçoit un «IN contrat», il arrête leTimer1 qui a été armé en attendant de recevoir un «IN contrat». Dans le cas où le timer 1 expire sans recevoir un IN contrat, l'ASC doit tout d'abord vérifier son FlagOut. Si le FlagOut= 1, cela signifie que l'ASC a déjà reçu un Out Contract. Sinon, il avise la VSC sur une communication défectueuse de l'ASC.

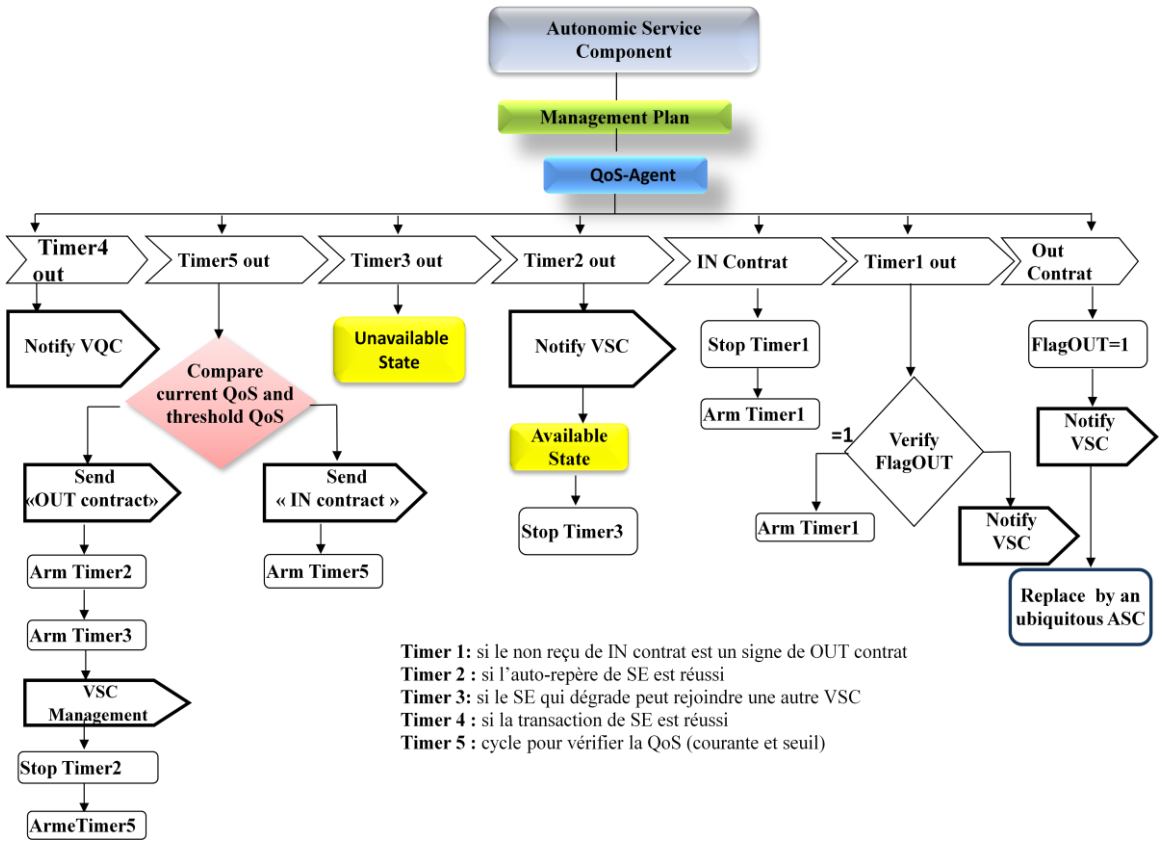

Figure 22:Automate du composant de service (plan de gestion)

V.3 Dimension protocolaire

V.3.1 Introduction

L'établissement, la modification et la maintenance d'une session unique et sans couture dans un contexte hétérogène et quel que soit le type de mobilité (utilisateur, terminal, réseau et service) est un point important pour répondre au besoin de l'utilisateur. Les différents types de mobilité qui peuvent se produire au cours de la session «user-centric», nécessitent l'exécution d'interactions plus complexes entre les éléments de service appartenant aux différents niveaux de visibilité en vue de rendre le service attendu avec la qualité de service requise.

La composition des services s'appuie sur l'agrégation modulaire de plusieurs services élémentaires issus d'environnement et de plate-forme hétérogènes. Cette composition doit être hautement dynamique, c.-à-d. activable à la demande des utilisateurs et elle doit d'une part, prendre en compte les besoins et contraintes multiples et évolutifs dans le temps en fonction du contexte ciblé, et d'autre part traiter dynamiquement les évolutions de ce contexte liées à la mobilité ou à une dégradation de la qualité de service.

Pour ce faire, nous structurons l'échange de données selon la dimension protocolaire qui se définit par l'intersection de trois éléments principaux les interfaces, les protocoles de communication et le plan d'appartenance. Elle inclut tous les mécanismes d'échanges, d'interactions et de coopérations entre les éléments pour assurer d'une part la composition de service de manière efficace et automatisée et d'autre part pour satisfaire le besoin d'une «session unique» sans couture et sans coupure. Le maintien de ce type de session impose de fortes interactions entre les acteurs afin de prendre en compte dynamiquement les changements, tant du côté de l'usager liés à la mobilité ou au changement de ses préférences que du côté de l'environnement d'exécution (QoS du réseau et des composants de services).

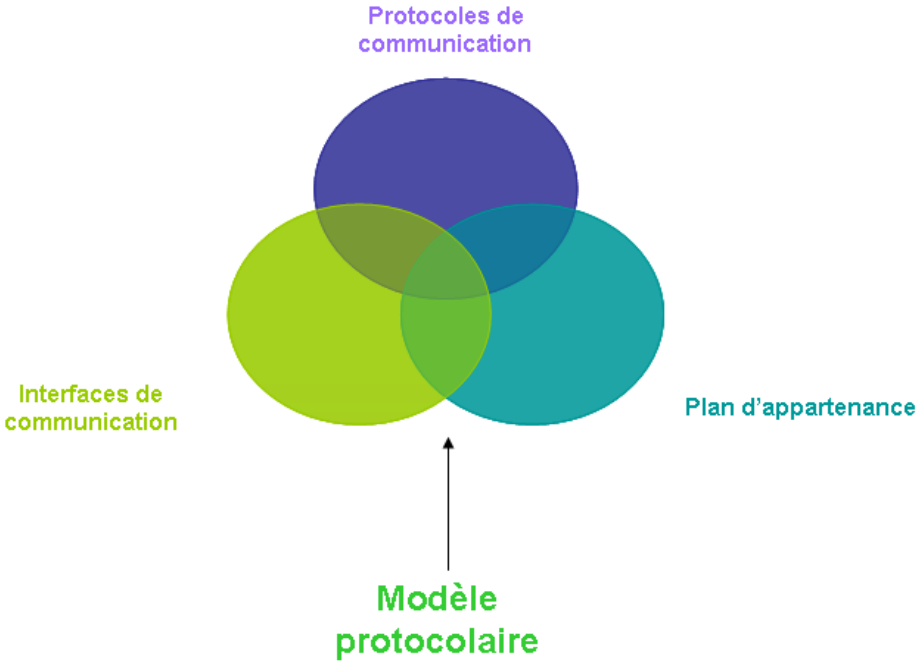

Figure 23: Modèle protocolaire

- ***Les protocoles de communication***

Chaque entité dans l'architecture UBIS rend un service qui peut être de type usager, contrôle ou gestion. Pour assurer la communication entre ces différentes entités, il existe des protocoles spécifiques pour chaque type de service qui permettent de structurer l'échange des informations entre les entités communicantes à travers les interfaces associées (gestion, contrôle et usage).

Les protocoles décrivent le comportement de communication externe des services qui peuvent être établies entre deux éléments de services. Pour assurer une interopérabilité correcte et automatisée, il existe des protocoles relatifs pour acheminer les données entre les composants de service faiblement couplés à travers les interfaces correspondantes de chaque plan (plan d'usager, plan de contrôle et plan de gestion). En outre, la composition de service permet de créer des services personnalisés à l'aide des composants de service issus des environnements et plates-formes hétérogènes. Il faut donc définir les protocoles d'interactions pour accomplir la composition de service et maintenir la session pendant les différents changements qui

peuvent se produire suite à la mobilité ou à une dégradation de la qualité de service d'une manière simplifiée et automatisée.

Les protocoles de communication permettent de structurer les messages échangés, ils représentent l'ensemble des règles permettant de définir un type de communication particulier. Ils adressent les aspects liés au transport des messages de type usage, contrôle ou gestion. Pour cela, il faut spécifier la nature des messages échangés et le mode.

Généralement, ces messages sont des requêtes (confirmées ou non confirmées) ou des notifications. Grâce à ce type de message, les entités permettent de réaliser une tâche en traitant les messages reçus ou en prenant l'initiative lorsqu'elles ne sont pas capables de fournir le service demandé. Pour le plan d'usage, on distingue deux autres natures de message:

- Les Interactions (doublet <question/réponse>) permettent aux entités de l'architecture UBIS de dialoguer entre elles au cours de la session.
- Les transactions (triplet <question, traitement, réponse> sont à la demande, elles répondent aux propriétés «ACID». Une transaction peut correspondre à une procédure SQL utilisée par les bases de données ou à une requête HTTP.

Deux modes de communication peuvent être envisagés:

- Mode connecté : L'entité émettrice et l'entité réceptrice du message doivent être actives en même temps pour vérifier si les données envoyées sont effectivement reçues.
- Mode non connecté: L'entité émettrice et l'entité réceptrice du message ne sont pas actives en même temps. Ce qui n'implique pas d'être connecté en permanence, ni de recevoir immédiatement une réponse. Ce mode correspond à des architectures faiblement couplée, c'est-à-dire que chaque composant de service n'impose pas à l'autre une interface de communication, c'est au message lui-même d'encapsuler les données et le contexte devenant ainsi une entité autonome.

- **Le plan d'appartenance**

L'architecture UBIS est organisée selon une structuration en 3 plans dans laquelle les piles protocolaires sont autonomes. Le plan usager pour les données, le plan de contrôle pour la signalisation et le plan gestion pour la supervision interne.

Nous proposons pour cette dimension protocolaire une articulation autonome des 3 plans grâce à la mise à disposition des informations:

- le plan de gestion pour les informations de surveillance durant l'exploitation,
- le plan de contrôle pour celles du provisioning et
- le plan d'usage pour la consommation des ressources transaction par transaction.

*Plan de gestion***:** C'est dans le plan de gestion que sont décidées, évaluées et modifiées selon le besoin les politiques appliquées par le plan de contrôle. Ce plan regroupe les opérations et fonctions y compris mécanismes et protocoles effectués sur les flux de données de manière asynchrone. Il regroupe tous les flux de gestion entre les entités communicantes à des fins de qualité de service de bout en bout.

Les flux de gestion qui ont pour rôle:

- L'authentification, autorisation, et la configuration du service demandé.
- La notification du statut de la QoS (In/Out contrat)
- Le transfert des règles de politiques QoS.

Plan de contrôle: Le plan de contrôle contient les opérations exécutées de manière synchrone sur les données applicatives émises ou reçues. C'est dans le plan de contrôle qu'est réalisée la composition des mécanismes permettant d'appliquer la politique choisie pour offrir le service attendu. Le plan de contrôle regroupe également l'ensemble des flux de signalisation nécessaires à la réservation de ressources et à la transmission des données relatives à l'établissement, la modification ou la terminaison d'une session entre deux entités.

Les flux de contrôle ont pour rôle:

- Le routage basé sur les informations QoS.
- Le contrôle d'admission basé sur l'état des ressources disponibles et la négociation de service de QoS.

– La signalisation de QoS pour la réservation des ressources et la négociation de service QoS.

*Plan Usage***:** Le plan usage a en charge le transport des informations utilisateurs, il regroupe tous les mécanismes relatifs au transfert des données du service dans le réseau. Ce plan comprend également les traitements de contrôle d'erreurs et de contrôle de flux qui sont appliqués parallèlement au transfert des données.

- **Les interfaces**

L'échange de message s'appuie sur des protocoles de communications selon le type d'informations échangées. Pour chaque plan (plan usager, plan de contrôle et plan de gestion), il y a des protocoles correspondants pour transmettre les informations à travers les interfaces associées.

Notre but est d'utiliser les protocoles existants pour transporter les informations des interfaces convergées. En effet, pour la fusion des actions, à l'interface, il faut identifier les différents protocoles possibles à chaque interface convergée et proposer des protocoles améliorés ou modifiés afin de simplifier les messages des différents protocoles et réduire le temps de traitement.

Notre objectif est donc d'assurer la QoS de bout en bout pour la continuité de session dans le NGN en intégrant les trois plans pour plus de dynamicité, de flexibilité et d'adaptabilité, mais non pas en rajoutant des protocoles dans les interfaces intégrées, mais en faisant converger ceux qui existent à travers l'information comme nous l'avons expliqué ci-dessus.

- **Adaptabilité (convergence plan usage et plan de contrôle)**

Quelle politique mettre en place au niveau des interfaces pour avoir plus d'adaptabilité? Quel est l'intérêt de la convergence du plan usage et du plan de contrôle?

Les capacités d'adaptabilité sont obtenues grâce à la convergence des deux plans usage et contrôle. L'adaptabilité est un enjeu majeur pour assurer l'interopérabilité des sous réseaux hétérogènes et la réservation des ressources en adéquation avec la QoS de chacun des sous réseaux. L'adaptabilité est nécessaire pour prendre en compte les changements induits par les

différentes mobilités qui conduisent à un changement de réseaux d'accès ou réseaux cœur avec des QoS différentes. Donc, elle permet selon les besoins d'affiner la qualité de service. Pour cela, il faut coordonner entre les informations du plan d'usager et celles du plan de contrôle pour appliquer les adaptations nécessaires pour la continuité de connectivité avec la qualité de service requises.

Par exemple, dans l'objectif d'optimiser une qualité de service de bout en bout il faut effectuer une adaptation pour assurer l'interopérabilité de la classification du trafic et du marquage de paquets dans les réseaux hétérogènes dans le NGN.

Dans cette interface convergée, il se trouve les requêtes du plan usage pour le contrôle d'admission et les messages de QoS signalisation. Les protocoles possibles pour cette interface seraient des «protocoles de signalisation» comme NSIS ou SIP, etc.

- **Flexibilité (convergence plan usager et plan de gestion)**

Quelle est la politique à mettre en place au niveau des interfaces pour avoir plus de flexibilité au cours de l'exploitation?

Les capacités de flexibilité sont obtenues suite à la convergence des deux plans gestion et usage pour tenir en considération les préférences spatiales et temporelles de l'utilisateur. Cette convergence permet d'assurer une surveillance des paramètres de performance de service pendant l'exploitation avec une réaction en temps réel suite à un changement lié à la mobilité ou bien aux préférences de l'utilisateur pour maintenir le service.

Dans cette interface convergée se trouve les informations relatives à l'utilisateur et les informations de gestion sur la tarification ce qui permet plus de flexibilité, et l'utilisateur a la meilleure connexion avec la tarification la plus appropriée.

- **Dynamicité (Convergence plan de gestion et plan de contrôle)**

Quelle est la politique à mettre en place au niveau des interfaces pour faire face au dynamisme?

La convergence du plan de gestion et du plan de contrôle répond aux capacités de dynamicité. Elle permet de fusionner les informations de gestion (détection de défaillance, localisation de

dysfonctionnement, dégradation QoS) et l'automatisme du plan de contrôle (signalisation) pour réagir le plus rapidement possible pendant la phase de l'exploitation et maintenir la conformité de SLA.

Grâce à la convergence CG (contrôle-Gestion), on transfert les informations de contrôle liées à la configuration des services et les messages de surveillance du plan de gestion au plan de contrôle. Et puis, le plan de contrôle fait le routage ou re-routage correspondant aux informations de QoS et il renvoie le feed-back au plan de gestion. Les protocoles possibles utilisables pour cette interface sont les protocoles de gestion Diameter ainsi que les protocoles de contrôle SIP/SDP etc.

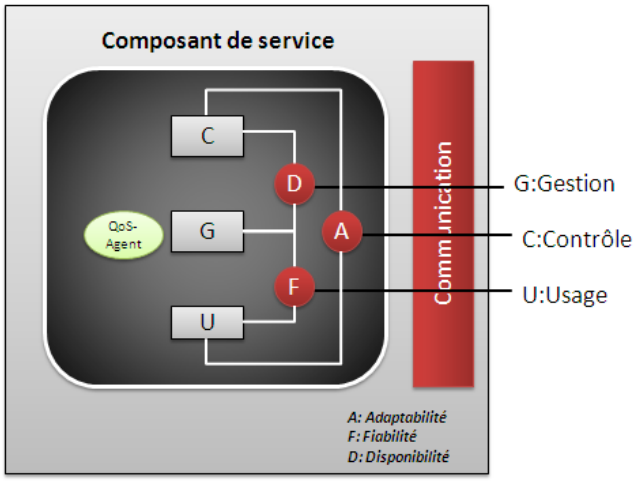

Figure 24: Composant de service (Interfaces convergées)

V.3.2 La session UBIS

(a) Les besoins de la Session UBIS

Actuellement, IMS (IP Multimedia Subsystem) spécifié par la 3GPP (Third Generation Partnership Project) propose une négociation de QoS pour déterminer la meilleure configuration des services et également pour assurer la réservation des ressources réseau, ce qui garantit les performances du réseau pour chaque service multimédia. IMS fournit les mécanismes de gestion de QoS qui visent à offrir aux utilisateurs finaux une meilleure prestation des médias à travers plusieurs terminaux et réseaux d'accès (mobilité d'utilisateur et mobilité du terminal). Les procédures d'IMS pour la négociation de la QoS sont basées sur le

protocole de signalisation SIP/SDP spécifié par l'IETF (Internet Engineering Task Force). En effet, IMS initie le processus de la création des sessions media via les invitations du protocole de signalisation SIP qui transporte le protocole SDP (Session Description Protocol) pour permettre une négociation des descriptions media entre les participants afin d'assurer le Media Delivery au niveau du réseau. Les solutions actuelles du «Media Delivery» offrent un ensemble de mécanismes et protocoles pour permettre uniquement la négociation de la QoS et la réservation des ressources réseau. Donc, le «Service Delivery» ne tient pas compte de la négociation et de la gestion des ressources service au niveau service de l'architecture, alors que des problèmes surgissent durant la session à cause du comportement des services au niveau de la plate-forme de services. Ces solutions actuelles ne sont plus suffisantes car les utilisateurs d'aujourd'hui veulent accéder à leurs services personnalisés dans un contexte hétérogène et mobile. Les fournisseurs devraient donc garantir à l'utilisateur une session unique et continue pour offrir une continuité de service avec une QoS de bout en bout. En fait, le bout en bout suppose que, hormis les ressources réseaux, les ressources services devraient être prises en compte en tant que ressources importantes lors de la session de l'utilisateur. Au niveau de la couche réseau de nombreux protocoles tel que SIP/SDP sont utilisés pour négocier la QoS des ressources réseaux. Créer et maintenir ce type de session nécessite de fortes interactions plus flexibles qui agissent sur la couche de service, par analogie à ceux de la couche réseau. C'est pourquoi nous proposons une extension de la portée du protocole SIP existant à la couche service, nommé SIP +, qui sera utilisé pour négocier la QoS des services personnalisés à la phase d'initialisation de la session de service et aussi de renégocier la QoS lors de la phase d'exploitation pour remédier au problème de dégradation de la QoS causé par la mobilité.

Ce protocole permet de créer, modifier et terminer la session de service d'un utilisateur basée sur la composition des services dans une architecture trans-organisationnelle. Dans cette partie sont exposés les atouts intéressants des spécifications protocolaires attachées au protocole SIP+ qui sont utiles à la composition de services. Le protocole SIP + permet d'assurer le contrôle de la session pendant la phase d'établissement de la session (création du VPSN) et la gestion de la session par le biais du VSC (Virtual Service Community). Pour ce faire, Il faut enrichir les informations véhiculées par les méthodes SIP en rajoutant les informations nécessaires pour le contrôle et la gestion de la QoS au niveau service. Ces

informations permettent de négocier la QoS pendant la phase d'initialisation de la session de service et aussi de renégocier la QoS lors de l'exploitation. Il existe plusieurs méthodes SIP, nous utilisons uniquement la méthode INVITE pour la négociation de SLA, la méthode NOTIFY pour la renégociation de QoS et BYE pour la libération des ressources.

L'établissement d'une session UBIS se déroule en trois étapes : l'ouverture de la session de services pour une personnalisation des services, l'initialisation de la session de services et la création du VPSN.

(b) Les étapes d'une session UBIS

(b.1) Ouverture de la session de services: Personnalisation des services

Lorsque l'utilisateur est autorisé à ouvrir sa session UBIS, il compose à travers une IHM « Interface Homme Machine» des services exposables hétérogènes en fonction de ses besoins (fonctionnel et non-fonctionnel (QoS)) qu'il va utiliser durant sa session. La sélection des services exposables se fait soit à partir du catalogue des services {SExp α, SExp β} stockés dans le profil utilisateur ou bien par la découverte des services existants dans l'environnement de l'utilisateur {SExp γ} (**Erreur ! Source du renvoi introuvable.**Figure 23).

Durant la phase de la découverte des services, on prend en considération les informations de géolocalisation de l'utilisateur (longitude: LO, latitude: LA) pour favoriser les services localisés dans le réseau ambiant de l'utilisateur afin de lui offrir des services avec une valeur ajoutée de son environnement. Les informations de géolocalisation sont obtenues en utilisant un terminal localisé par le biais du GPS, WIFI, GSM, IP ou bien d'autres techniques. Les informations de géolocalisation obtenues sont stockées dans le terminal ou bien dans le serveur de services.

Ensuite, le terminal génère le Workflow avec la composition des services exposables {SExp α @ provider1, SExp β @ provider2, SExp γ @ provider3}, la logique de service et également la localisation de l'utilisateur pour initier la session UBIS de l'utilisateur.

Ces informations sont envoyées via le protocole de signalisation SIP+ en passant par IMS à la plate-forme des services pour leur traitement. Il convient de mentionner qu'avant

l'ouverture de la session de services, il nous faut ouvrir les autres sessions (d'accès et réseaux).

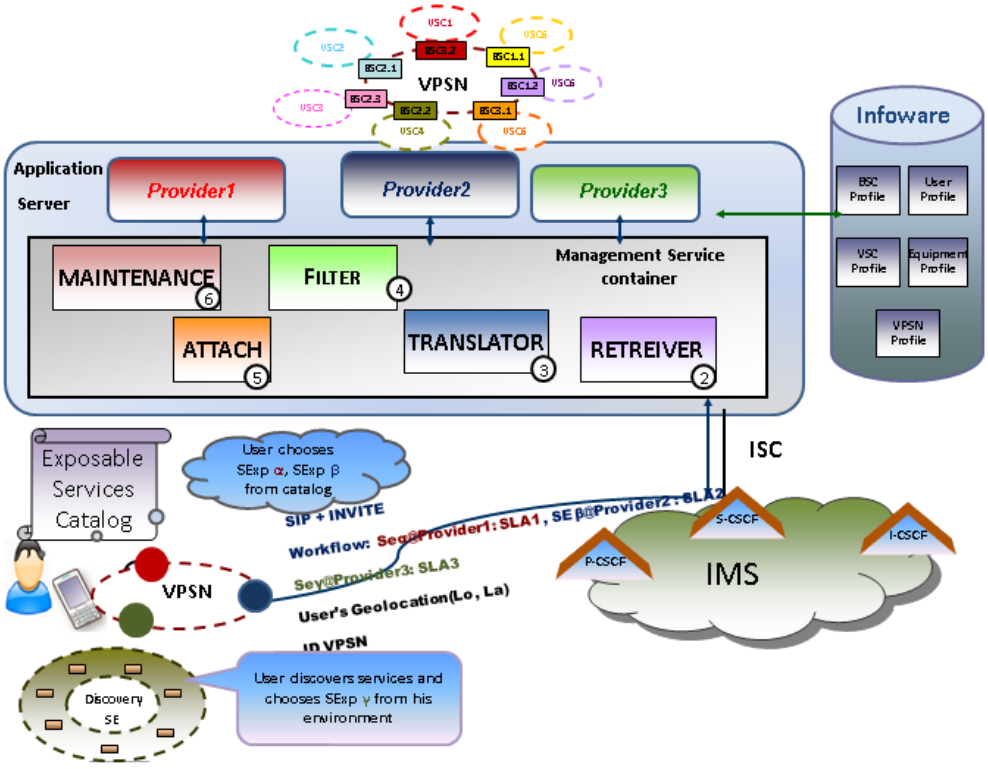

Figure 25: Session de service UBIS

(b.2) Initialisation de la session de services: SIP + INVITE

Au moment de l'initialisation de la session de service (Figure 26), la phase de négociation de la qualité de service de bout en bout est responsable de l'identification des composants de service impliqués dans le service global à rendre. Lorsqu'un l'utilisateur initie sa session, un processus de négociation de la QoS est déclenché pour identifier les composants de service selon le niveau de QoS qu'ils sont capables de fournir. Cette négociation dynamique de la QoS s'appuie sur la méthode INVITE du protocole de signalisation SIP+, qui fournit une description de QoS dans un niveau plus élevé couvrant le service global demandé par l'utilisateur. Ce protocole permet une déclinaison Top-Down de la négociation QoS du niveau service (QoS du composant de service) jusqu'au niveau des équipements (QoS des équipements) en passant par le niveau réseau (La QoS des réseaux d'accès et de transport).

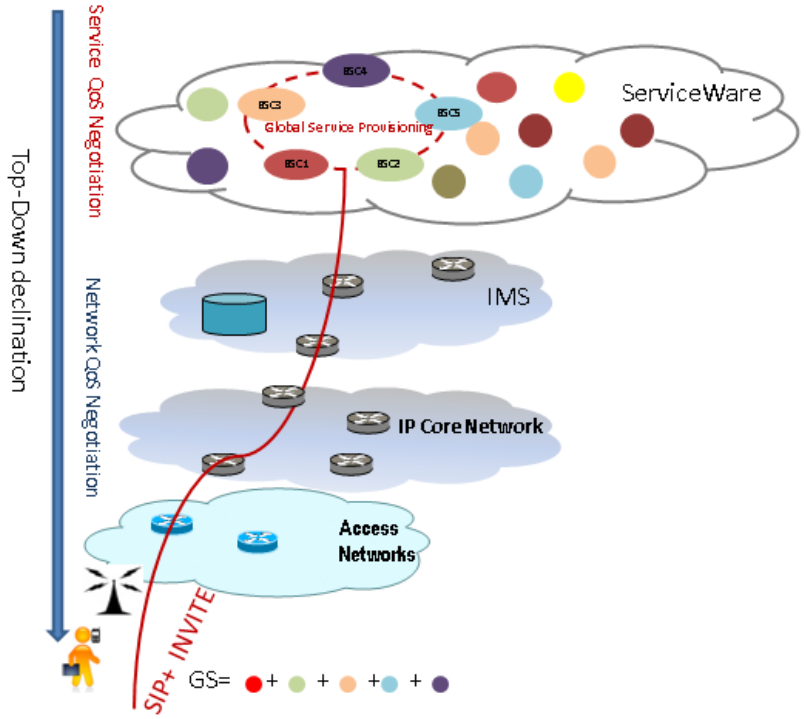

Figure 26: SIP+ INVITE pour une Négociation de la QoS de bout en bout

Le traitement de la demande d'initialisation d'une session de services est lancé par la plate-forme de services suite à la réception du message SIP + INVITE, qui contient toutes les informations concernant la position géographique de l'utilisateur (Lo, La), et également celles relatives à la logique de la composition des services exposables, qui sont sélectionnés par l'utilisateur au moment de la phase de personnalisation des services, et qui vont éventuellement participer à la session «user -centric». Ces informations sont introduites dans le «Body» du message SIP+ INVITE (Figure 27).

Le corps du message SIP+ INVITE contient donc principalement la logique des services, elle définit la liste des transactions des services exposables et leur ordre d'exécution au cours de la session user-centric {SExpα @provider1, SExpβ @ provider2, SExpγ @ provider3}. Il contient également une description de la QoS (SLA) de chaque SExp demandé par l'utilisateur. En particulier, les exigences de l'utilisateur sont décrites pour chaque SExp en fonction du modèle QoS basé sur les quatre critères de QoS (Disponibilité, délai, fiabilité et capacité).

A la réception de ce message, le ServiceWare, va déclencher les mécanismes de négociation de la QoS pour traduire les services exposables en services de base «BSC (Basic Services component)» avec la QoS requise, et va compléter le VPSN.

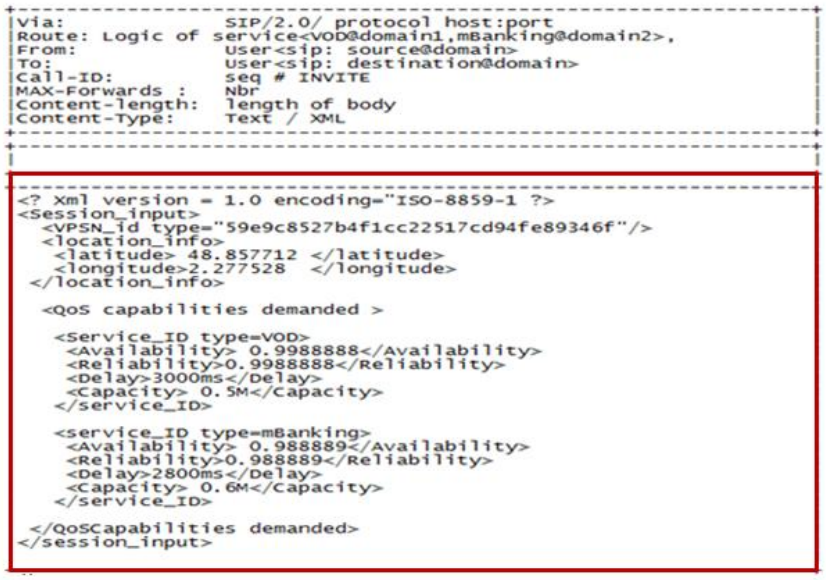

Figure 27: Structure du message SIP+ INVITE

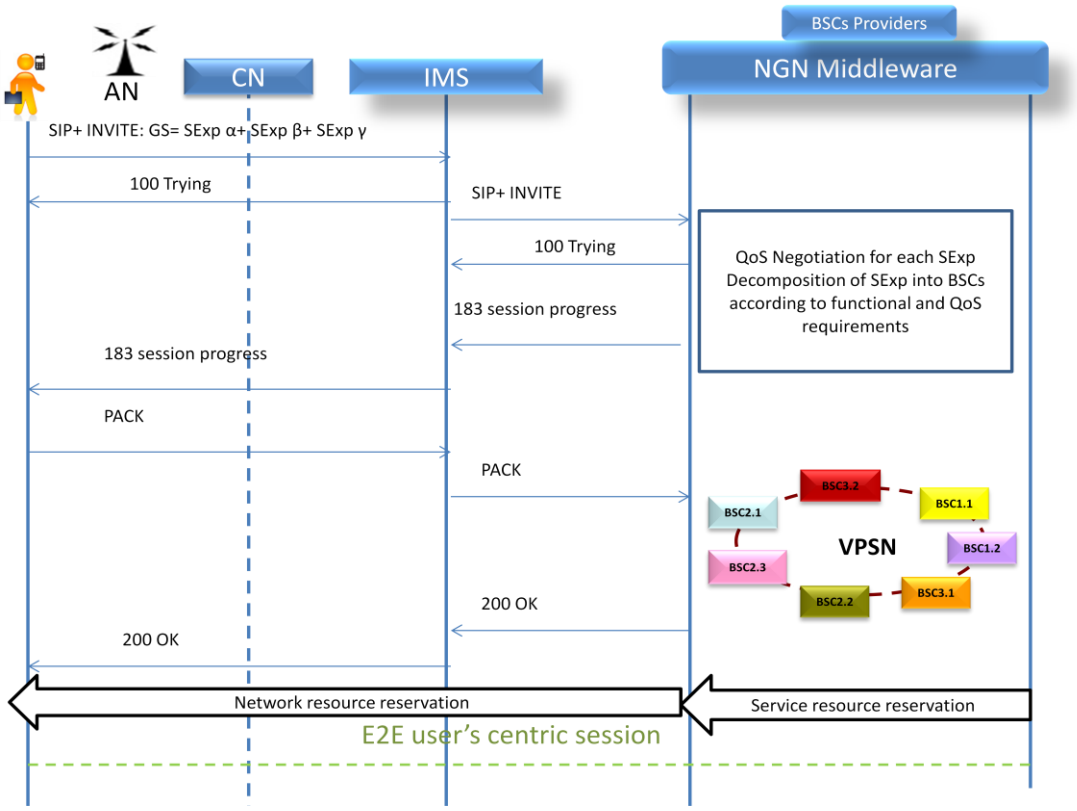

Figure 28: Diagramme de séquence (Initialisation de la session de service)

(b.3) Processus de la Création du VPSN :

Pour atteindre une gestion et un contrôle automatisés et distribués de la qualité de service au niveau service de l'architecture, nous allons appliquer notre vision de SOA basée sur des composants de service génériques, mutualisables, Stateless, et autogérables. Un composant de service peut rendre un service de type usage, contrôle ou gestion.

Dans notre approche, chaque service exposable est composé d'un ensemble de composants de service de base « BSC» de type usage. En outre, la session de service est créée en définissant la chaine logique des composants de services exécutables de type usage « BSC». Cette chaine logique constitue le VPSN (Virtual Private Service Network). Ces BSCs seront classés selon les transactions définies par l'utilisateur au moment de l'ouverture de la session de service.

Le scénario que nous allons décrire se base sur une configuration où nous n'avons qu'une seule plate-forme de service. Dans le cas où nous aurions plusieurs plates-formes de service

(le terminal pourrait en être une), la création du VPSN est partagée, chacune des plates-formes le complétant avec les services dont elle a la responsabilité.

La composition de service permet l'interopérabilité, l'interaction et la coopération entre différentes capacités des composants de service c.-à-d. l'aspect fonctionnel et non fonctionnel du service pour enrichir le service pour l'utilisateur.

Pour créer le VPSN en se basant sur les informations envoyées via le SIP+ INVITE, on utilise les composants de service de type gestion «Management Service Components: MSC» afin d'assurer la création et la maintenance du VPSN dans la plate-forme de service. Notre conception est basée sur un ensemble d'interactions entre des MSC avec des interfaces en couplage lâche facilitant l'interaction et l'interopérabilité pour créer et maintenir le profil VPSN. En effet, le processus de création est basé sur cinq MSCs (**Erreur ! Source du renvoi introuvable.**Figure 29).

- Retriever_SIP «RMSC»: extrait les informations contenues dans le Body SIP+ INVITE
- Translator «TMSC»: traduit le workflow des services exposables et le SLA en composants de service exécutables avec la QoS demandée. On mentionne que la combinaison des fonctions des composants de service et leur QoS permet d'assurer les besoins des utilisateurs.
- Filter «FMSC»: trouve la combinaison (ID-BSC, @ logique) selon la liste des ID-BSC limitée par un cercle avec un rayon Я et un centre {LO, LA} qui correspond à la position géographiques de l'utilisateur au moment de l'établissement de la session pour offrir des services selon le contexte ambiant de l'utilisateur.
- Attach_VPSN «AMSC»: ajoute l'ID-VPSN au profil du composant de service. Un composant de service peut être mutualisé entre plusieurs VPSN. Donc l'AMSC vérifie la QoS statistique en se basant sur les valeurs de conception pour contrôler l'attachement d'un composant de service à un nouveau VPSN.
- Maintenance_VPSN «MMSC»: ajoute les informations du composant de service (ID-BSC, @ logique au profil VPSN. Durant la session de service, le MMSC maintient le VPSN en mettant à jour le profil VPSN dans le cas où la QoS est dégradée. Il assure le

remplacement des informations du composant de service dégradé par celle d'un autre composant qui est fonctionnellement et QoS équivalent.

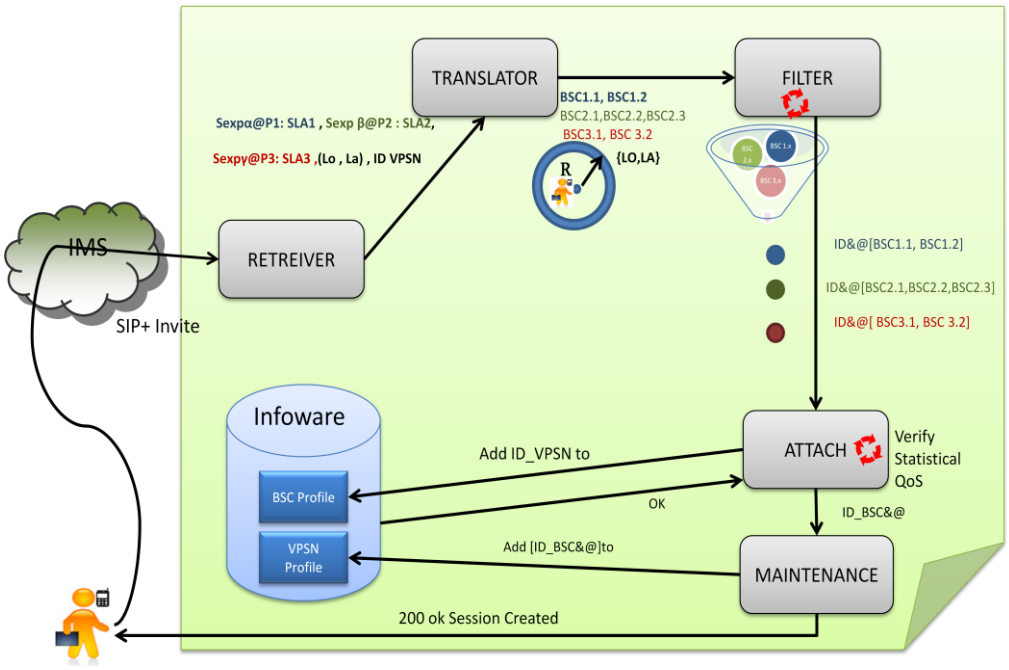

Figure 29: Processus de création de VPSN

La création du VPSN représente les différentes étapes pour créer le «profil VPSN». En effet, le profil VPSN contient la chaine logique de tous les composants de service BSC qui assurent la fonction des services exposables avec la QoS requise. Le processus de la création est lancé par la plate-forme de service après avoir reçu le message SIP+ INVITE à travers IMS.

Premièrement, le RMSC extrait les informations à partir du corps du message SIP+ INVITE. Deuxièmement, le TMSC est invoqué pour traduire le workflow des services exposables en une liste de BSC exécutables avec les critères de QoS requis: {SExpα=BSC1.1+BSC1.2, SExpβ=BSC2.1+BSC2.2+BSC2.3, SExpγ =BSC3.1}. Le résultat est la liste des BSCs avec la QoS offerte adéquate.

Troisièmement, le FMSC est sollicité pour trouver la combinaison {ID-BSC, @logique} pour chaque BSC selon la position géographique de l'utilisateur pour gagner en terme de temps de réponse durant la phase opérationnelle, parce qu'il existe plusieurs BSC ubiquitaires qui sont déployés dans différentes plates-formes.

Initialement, la recherche est effectuée dans la zone ambiante de l'utilisateur qui est limitée par un rayon R et le résultat est envoyé par la suite à l'AMSC. Si la recherche n'aboutit pas, on incrémente le rayon R jusqu'à ce qu'on trouve le BSC qui réponde au besoin de l'utilisateur.

Ensuite l'AMSC utilise le résultat obtenu pour vérifier les capacités statistiques de chaque BSC de la liste pour joindre le VPSN. Si le résultat est positif, l'AMSC ajoute l'ID-VPSN au profil du BSC dans la base de connaissance «INFOWARE».

Ensuite, l'infoware envoie un évènement au MMSC pour ajouter le BSC sélectionné à la table VPSN. Finalement, la table VPSN est créée et elle contient la chaine logique de tous les BSC et leur ordre d'exécution. Le diagramme de Séquence (Figure 28) représente les différentes interactions entre l'utilisateur, les composants de service de type gestion et l'infoware.

A la fin de ce processus nous avons pré-provisionné nos composants de service par le double attachement (BSC/VPSN et VPSN/BSC) sans réserver effectivement les ressources. Le provisioning se fait au moment de la consommation des services. La consommation des services au cours de la session « user-centric » se fait en fonction des transactions définies par l'utilisateur au moment de l'ouverture de la session, une transaction permet l'exécution d'une sélection d'un ensemble de services exposables. La consommation de service se traduit par la réservation réelle des ressources (provisioning) pendant la phase de l'exploitation et plus précisément au moment de la consommation effective.

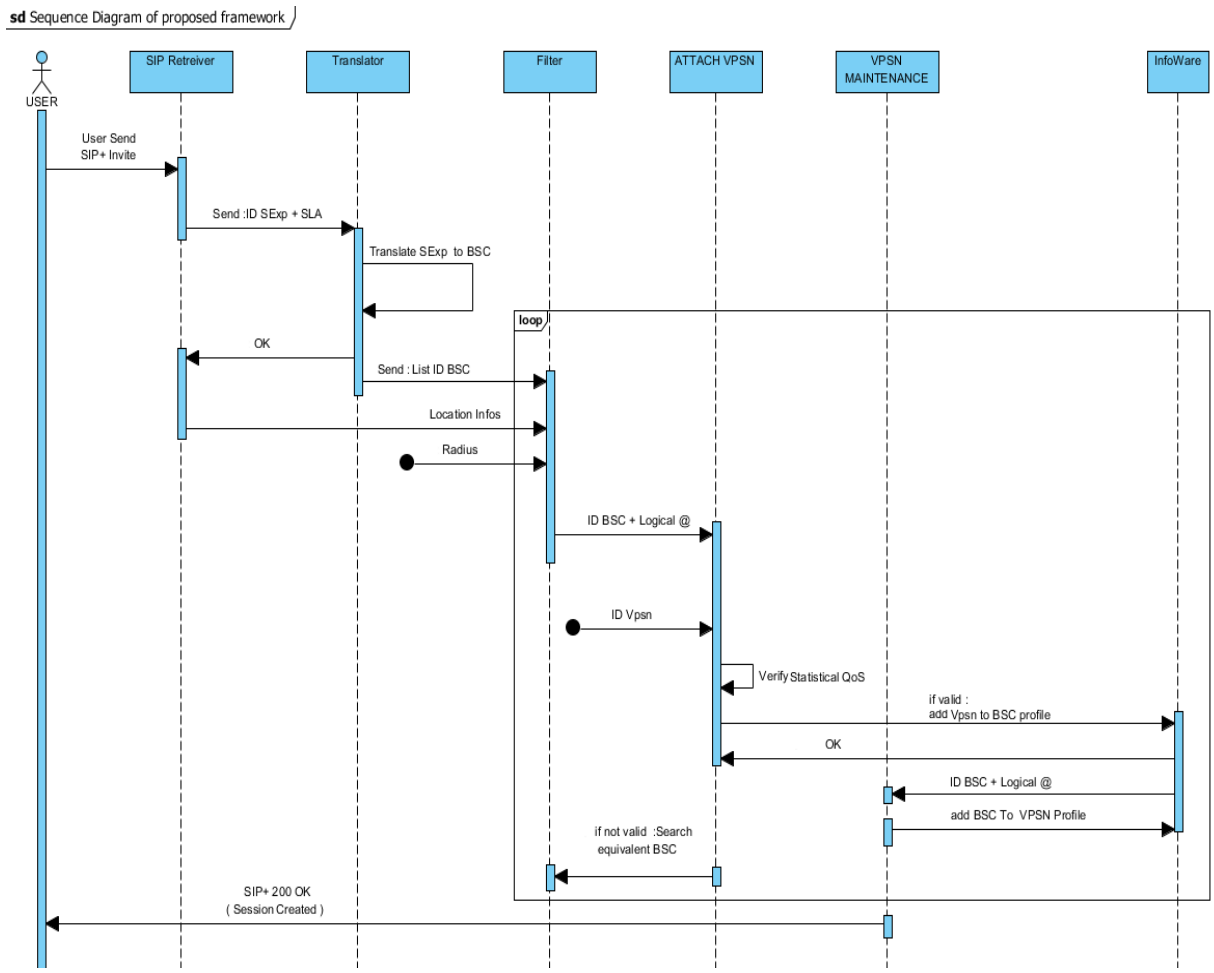

Figure 30: Diagramme de séquence pour la création du VPSN

(b.4) La modification de la session UBIS

Pour offrir à l'utilisateur une continuité de service avec une QoS de bout en bout dans un contexte hétérogène et mobile, il faut lui garantir une session unique et continue pendant la phase d'exploitation.

En fait, le bout en bout suppose que, hormis les ressources réseaux, les ressources services devraient être prises en compte en tant que ressources de premier plan lors de la session de l'utilisateur. Sur la couche réseau, de nombreux protocoles sont utilisés pour déclencher le processus de renégociation de la QoS si une dégradation de la QoS réseau se produit. Par contre ce problème n'est pas encore étudié dans la couche de services. Maintenir, donc, une session de service unique à l'utilisateur nécessite de fortes interactions plus flexibles qui

agissent sur la couche de service pour renégocier la QoS. Un événement de renégociation de la QoS au niveau service peut être déclenché si une dégradation de la QoS d'un composant de service se produit suite à une insuffisante de ressources ou bien elle peut également être invoquée par l'utilisateur qui décide volontairement de changer de services selon ses préférences, comme le passage d'une vidéo de son téléphone PDA à son PC.

a) Le protocole SIP+: Message SIP+ NOTIFY

Avec la convergence du plan de contrôle et du plan de gestion, le protocole SIP+ au niveau service peut transporter en plus des informations de contrôle les informations de gestion pour garantir plus de dynamicité pendant la phase d'exploitation. En effet, cette « QoS signalisation » communiquera en plus de la description des médias, les exigences de QoS au niveau des composants de service et elle enverra la notification de l'état de la QoS surveillée (in/out contrat) par le message SIP+ Notify d'un BSC faisant partie d'un VPSN à sa communauté VSC. Le protocole SIP+ offre une signalisation dynamique au niveau service pour fournir le service demandé et garantir la continuité de session en conformité aux SLA.

La Figure 31 représente la structure d'un message SIP+ NOTIFY utilisé pour gérer la session de service. Les champs utiles dans l'en-tête du message SIP+ NOTIFY qui sont utilisés pour notifier l'état de QoS (IN / OUT contrat) d'un composant de service en cours d'exécution sont les suivants:

- Allow-Events: il permet la notification des événements concernant le contrat de qualité de service (IN/OUT contract).
- Event: Ce champ est utilisé pour envoyer l'état de qualité de service (IN Contract/Out Contract) d'un composant de service.

```
+-----------------------------------------------------+
|Via:                 SIP/2.0/ protocol host:port     |
|From:                User<sip: source@domain>        |
|To:                  User<sip: destination@domain>   |
|Call-ID:             localid@host                    |
|Cseq:                seq#NOTIFY                      |
|Content-length:      length of body                  |
|Content-Type:        Application/XML                 |
|Contact:             <sip:user@pc.example.com        |
|Subscription-State:  Active                          |
|Allow-Events :       contract(IN/OUT contract)       |
|Event:               IN/OUT contract                 |
+-----------------------------------------------------+
+-----------------------------------------------------+
|                                                     |
+-----------------------------------------------------+
+-----------------------------------------------------+
| <?xml version="1.0" encoding="UTF-8"?>              |
| <contract-response version="1.0"  text="IN/OUT" />  |
|                                                     |
+-----------------------------------------------------+
```

Figure 31: Structure du message SIP+ NOTIFY

b) La communauté VSC

Au cours d'une session centrée sur les besoins de l'utilisateur « user centric », on peut avoir une dégradation de la qualité de service suite à un changement de la condition de QoS dans un nœud (par exemple la valeur courante dépasse les valeurs seuils). Le remplacement de ce dernier est géré par la communauté de services «VSC». La VSC regroupe les composants de service ubiquitaires qui sont fonctionnellement et QoS équivalents, ces communautés sont constituées et gérées par l'ensemble des fonctions des services de base (localisation, présence, découverte). Le pilotage de la qualité de service vise à évaluer l'état de la qualité de service de bout en bout au cours d'une session. Pour cela, il faut évaluer l'état de la qualité de service d'un composant de service qui intervient pendant la phase d'exploitation. La politique d'analyse utilisée repose sur un traitement interne de la QoS effectué par l'agent QoS et les notifications (In contrat / out contrat) reçues par la communauté VSC, pour assurer le remplacement d'un élément de service défaillant par un composant équivalent dans le cas d'une dégradation de la QoS.

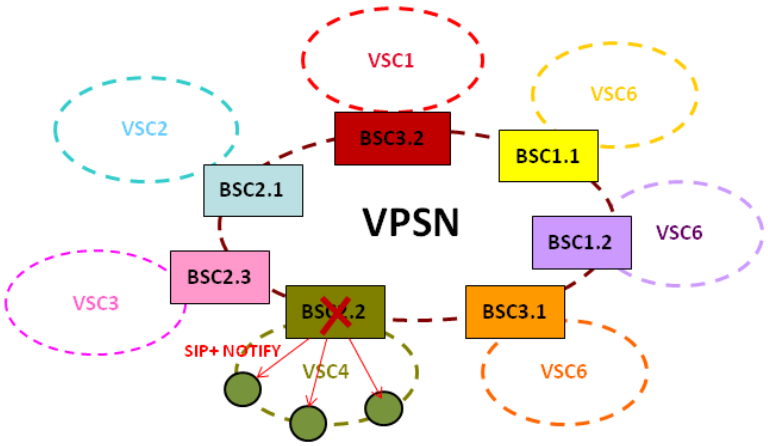

Figure 32: la communauté VSC

Pour cette raison, nous avons besoin d'identifier et de sélectionner l'élément le plus adéquat dans une communauté VSC en prenant en considération la QoS du nœud, du lien et du réseau car les composants de service peuvent être déployés dans des plates-formes hétérogènes, ce qui peut entrainer des temps de réponse variables. Pour faire face à cette hétérogénéité, nous proposons de s'appuyer sur les tables «QoS NLR» pendant la phase de sélection. En effet, chaque nœud de service dispose d'une table QoS qui contient les informations mises à jour de la QoS du nœud, du lien entre deux nœuds et aussi du réseau qui représente le réseau de transport entre ses nœuds. Dans la table VSC les composants de services sont classés selon la QoS du lien entre le BSC activé et ceux qui sont susceptibles de le remplacer.

- La QoS du Nœud représente les caractéristiques fonctionnelles de traitement responsables d'un processus spécifique.
- La QoS du Lien représente la qualité de service du canal de communication virtuel entre les deux nœuds.
- La QoS du Réseau définit la table de routage qui enregistre la qualité de service de chacun des chemins possible dans la couche de transport. La qualité de service est calculée et mise à jour en temps réel.

Pour ce faire, nous proposons une autogestion basée sur un algorithme «VSC QoS NLR selection» afin de trouver un autre composant de service avec une QoS équivalente pour le remplacer. Si cette autogestion de VSC échoue, la signalisation dynamique peut interagir avec le système de l'utilisateur dans la base de données, et réapprovisionner la qualité de service

dans le réseau de services (VPSN) afin de trouver un autre composant de service de remplacement avec une QoS correspondante aux préférences de l'utilisateur.

Au niveau du VSC, la communication se fera en P2P. Dans ce cas, la gestion de la QoS est répartie. En effet, l'intelligence est distribuée sur les composants de service du VSC. En d'autre terme, la requête SIP+ QoS Notify peut être envoyée aux BSCs du VSC en P2P selon deux approches:

- ***1ère approche: Transmission P2P de proche en proche***

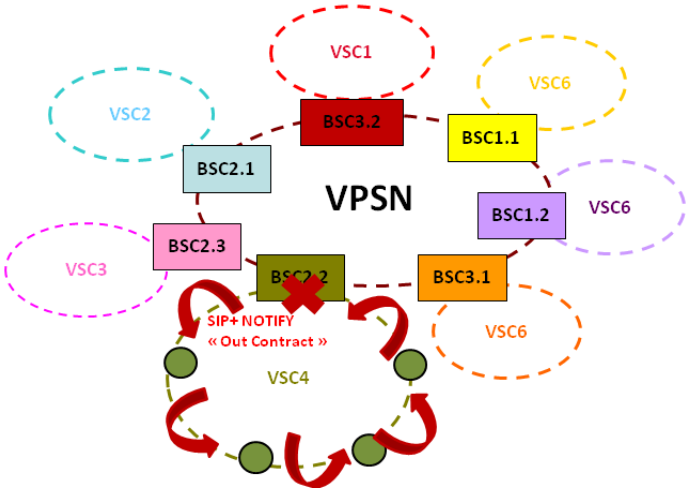

Figure 33:VSC Transmission P2P de proche en proche

La requête SIP+ QoS Notify est envoyée de proche en proche selon le classement QoS NLR des éléments de service dans la table «VSC QOS NLR». La sélection du candidat s'appuie sur l'algorithme «QoS NLR selection» (Figure 34) pour élire le composant qui offre une QoS NLR équivalente ou bien meilleure de celle du composant qui est hors contrat. Cet algorithme s'appuie sur les tables VSC QOS NLR, pour que la requête SIP+ Notify soit envoyée en premier lieu vers un composant de service qui a probablement les ressources nœud, lien et réseau pour satisfaire la demande.

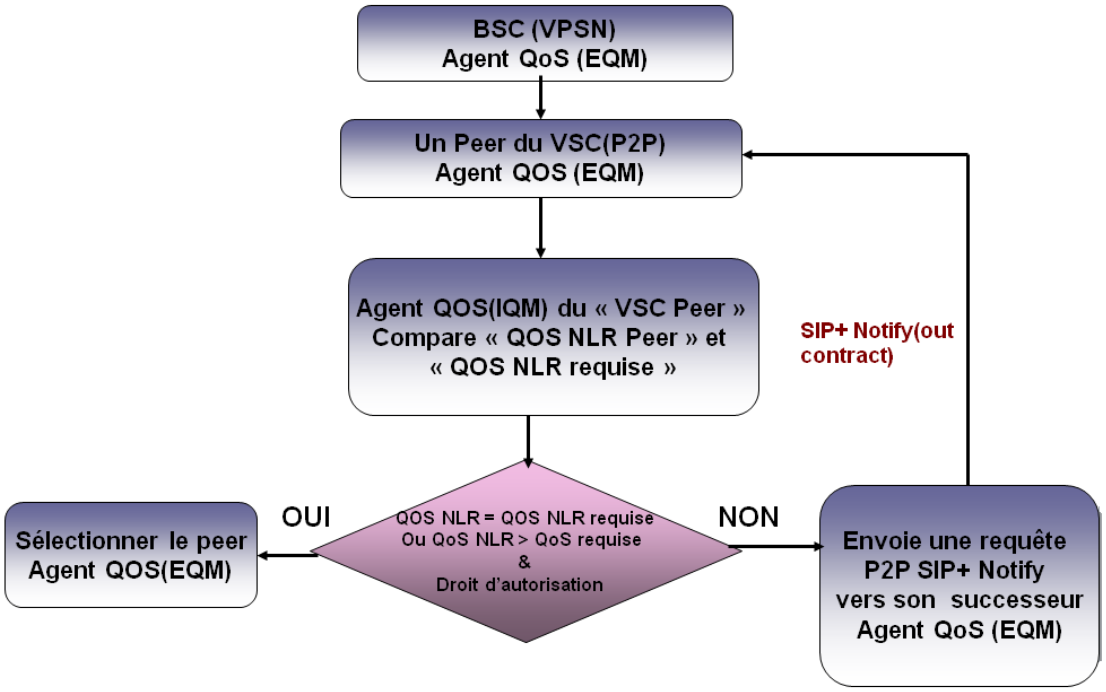

Figure 34: Algorithme «QoS NLR selection» : proche en proche transmission

Lorsque l'agent QOS (IQM) du BSC (VPSN) détecte une dégradation de la QoS, l'EQM assure la notification de la défaillance de ce dernier via le protocole SIP+ Notify (QoS NLR requise) aux BSC de la communauté VSC, et cela, pour assurer son remplacement par un autre BSC offrant une QoS NLR équivalente à celle du composant défaillant. Ce calcul se fait par une communication en P2P SIP+, le P2P SIP+ permet d'étendre le protocole SIP+ au niveau du VSC et le router selon le mécanisme P2P.

Le VSC constitue le réseau des BSCs en P2P, la requête SIP+ Notify est envoyée de proche en proche selon l'ordre recommandé par la table QoS NLR, qui favorise le BSC qui a une QoS équivalente ensuite les BSCs qui ont une «QoS NLR» meilleure.

Le premier BSC de ce réseau traite la requête SIP+ Notify. Si ce dernier a les droits, il envoie une requête P2P «SIP + message SET» pour rejoindre le VPSN et par la suite remplacer le nœud qui est hors contrat, sinon il assure le routage de cette requête à son successeur de la communauté VSC en s'appuyant sur la table QoS NLR.

Les différentes possibilités de traitement:

- Si le BSC a les droits d'autorisation et la QoS NLR BSC_i «VSC» >= QoS NLR Conception du BSC défaillant, ce nœud envoie un message SIP+ MESSAGE SET au VPSN pour assurer le remplacement de ce dernier.

- Si le BSC n'a pas les droits d'autorisation, ce nœud renvoie un message SIP+ Notify (out contrat) à son successeur selon la logique de la table QoS NLR(VSC) jusqu'à la satisfaction de la demande.

  - ***2ème approche: Transmission P2P multicast***

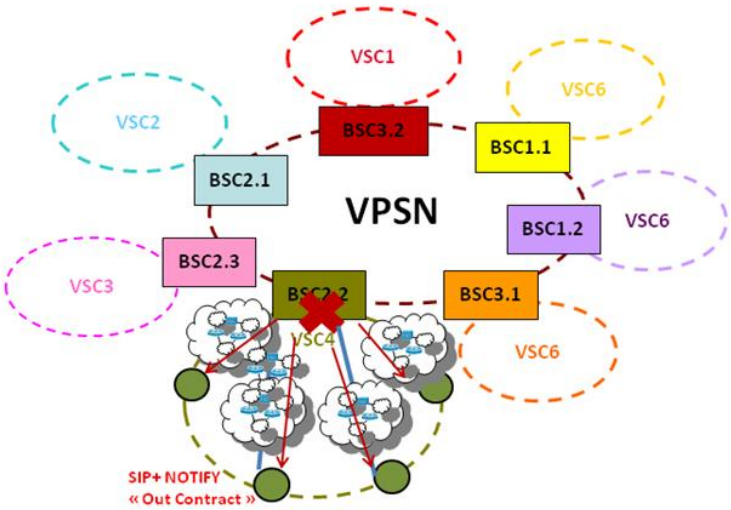

Figure 35: Transmission P2P dans la communauté VSC en multicast

Dans cette approche, le SIP+ QoS Notify est envoyé en multicast à tous les éléments de service de la communauté P2P. Dans ce cas la sélection du candidat se fait selon la table QoS NLR. En effet, dans la table QoS NLR les BSCs sont classés suivant leur QoS NLR par rapport au BSC activé. Le BSC classé au début de la table aura forcément une QoS équivalente ou meilleure à celle du BSC hors contrat. S'il est autorisé, il envoie un message SIP+ message SET pour rejoindre le VPSN.

Dans le cas ou plusieurs BSC dans la table ont une QoS NLR équivalente à celui qui est hors contrat, le BSC qui a un meilleur temps de réponse sera sélectionné pour rejoindre le VPSN.

Le diagramme (Figure 36) représente le déroulement de l'algorithme QOS NLR Selection.

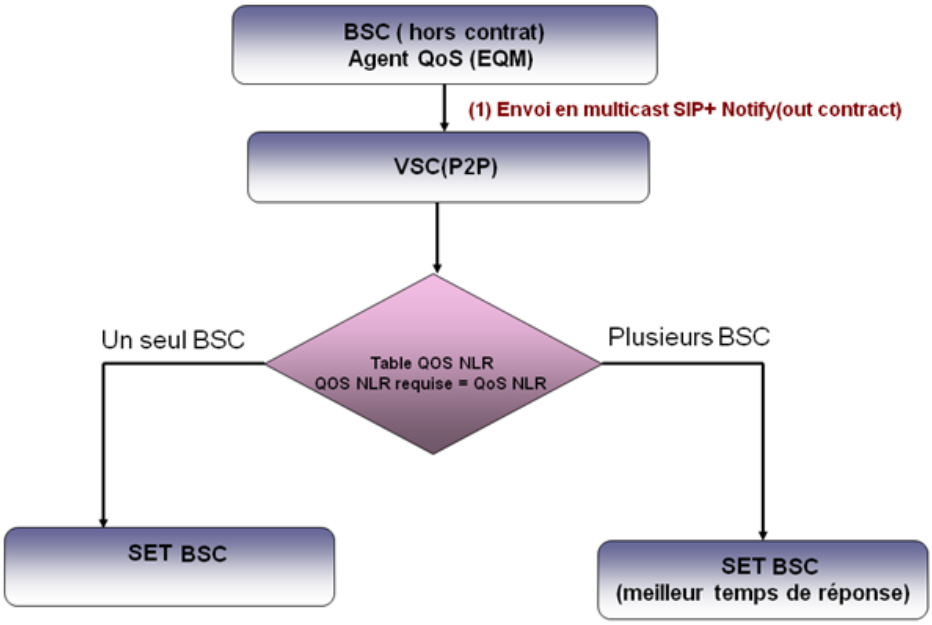

Figure 36: Algorithme «QoS NLR Selection»: transmission P2P en multicast

La Figure 37 représente les différents messages échangés entre le BSC défaillant du VPSN et la communauté du VSC. Au début de la session, la requête Subscribe Event QoS est envoyée en SIP+ P2P par tous les éléments de service de la communauté VSC aux BSCs (VPSN) pour qu'ils souscrivent à l'événement (Out contrat) afin d'être notifiés de son occurrence.

Le BSC (VPSN) répond par un message 200 OK, ce message indique que la demande de souscription a été bien reçue.

Les agents QoS(EQM) des BSCs appartenant à la communauté VSC restent à l'écoute. Si une dégradation de la qualité de service de BSC (VPSN) se produit pendant la phase d'exploitation, l'agent QOS (EQM) de ce dernier envoie un message de notification SIP+ à la communauté VSC. Ce message est routé en P2P et traité selon l'algorithme «QoS NLR Selection» jusqu'à la sélection du peer de remplacement.

Lorsqu'un agent QoS du VSC se reconnaît candidat de remplacement en s'appuyant sur l'algorithme QoS NLR Selection, il envoie un message SIP+ MESSAGE SET pour rejoindre le VPSN.

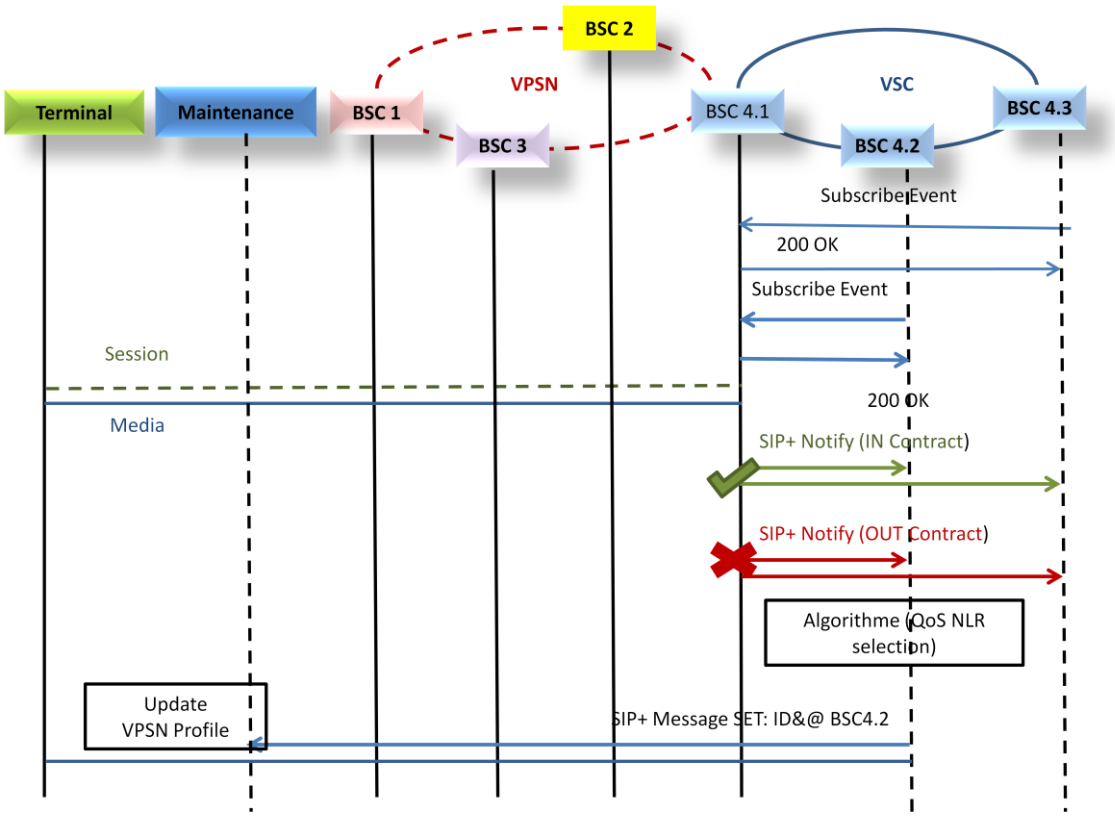

Figure 37: Diagramme d'autogestion du VSC

Nous décrivons un scénario (Figure 38) qui illustre l'utilité de l'automatisation de la gestion de la session de l'utilisateur pendant les différents types de mobilité. Lorsque l'utilisateur (Point A) est à la maison (localisation {LO, LA}), il choisit une composition de services exposables : SExpα, SExp β et SExp γ (**Erreur ! Source du renvoi introuvable.**Figure 23), donc après traitement de cette requête par les composants de gestion du ServiceWare, la logique des composants de service de base selon les préférences de l'utilisateur (pré-provisioning) est BSC1.1 + BSC1.2 + BSC2.1 + BSC2.2 + BSC2.3 + BSC3.1. Durant la phase d'exploitation, la performance du BSC2.2 (Point B) se dégrade, et elle ne peut pas continuer à fournir la QoS requise par le SExpβ. Par conséquent, le BSC2.2 envoie un message SIP+ Notify à tous les membres de sa communauté VSC pour procéder à son changement. Il existe toujours des services ubiquitaires qui ont les capacités fonctionnelles et non-fonctionnelles (QoS) pour remplacer le composant de service défaillant, l'algorithme "QoS NLR Selection" est utilisé pour faire le choix dans la communauté VSC, en se basant sur la localisation de l'utilisateur

pour garantir un temps de réponse équivalent au temps de réponse fourni par le composant dégradé.

Lorsque la communauté VSC sélectionne le composant BSC2.2' (point B') localisé dans une autre plate-forme de services. Le gestionnaire de la VSC invoque le composant de gestion qui est responsable du maintien du VPSN «MMSC» en envoyant un message qui contient les informations sur le BSC2.2' du remplacement (ID-BSC2.2', @logique), ID-VPSN) pour mettre à jour le profil VPSN. Ensuite la connexion est transférée du BSC2.2 au BSC 2.2' (provisioning dynamique).

L'utilisateur (point A) se déplace (mobilité du terminal), et quand il arrive au bureau (point A'), il change son terminal du PDA à un PC (mobilité de l'utilisateur). Dans cette situation, en raison du changement de l'emplacement de l'utilisateur du point A {LO, LA} au point A{LO', LA'} et le changement de son terminal du PDA au PC, les composants de service de base BSC1.1+ BSC1.2 ne peuvent plus répondre à ses exigences. Pour cela, le composant de maintenance de VPSN se charge de les remplacer par deux nouveaux services BSC4.1 + BSC4.2, qui sont adaptés à son nouveau terminal. Par conséquent, nous obtenons un réapprovisionnement dynamique du VPSN après la mobilité: BSC2.1 + BSC2.2' + BSC2.3 + BSC3.1+ BSC4.1+ BSC4.2.

Finalement, la session de service de l'utilisateur est maintenue et le service delivery est assuré. Nous mentionnons que toutes ces opérations sont transparentes à l'utilisateur et elles sont effectuées d'une manière autonomique durant la session.

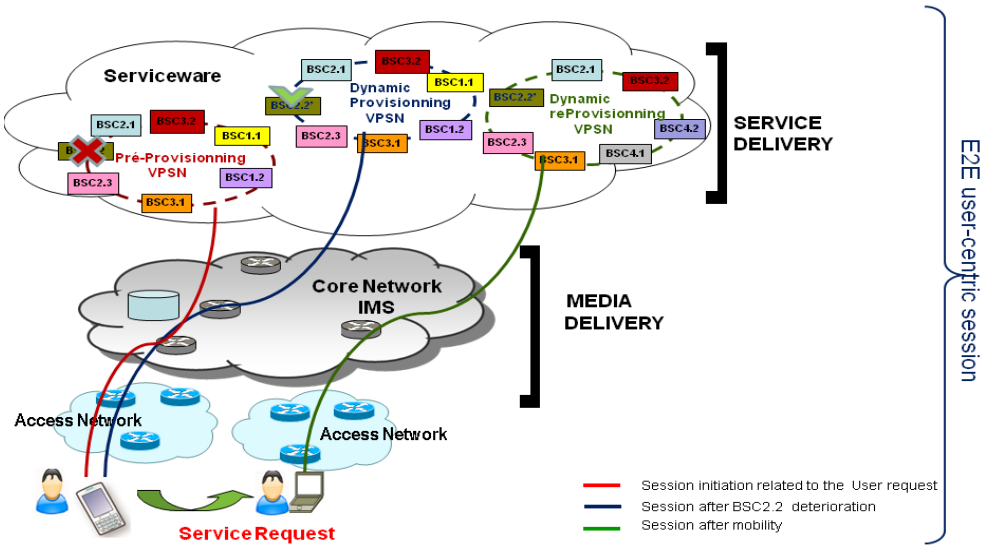

Figure 38: Scénario d'autogestion pendant la mobilité

V.4 Conclusions des propositions

Dans ce chapitre, nous avons consigné nos propositions qui visent à répondre au besoin du pilotage dynamique de la qualité de service de bout en bout d'une session «User-centric» qui tient compte des besoins fonctionnels et non-fonctionnels (QoS) de l'utilisateur et de ses préférences dans son nouveau contexte qui se caractérise par l'hétérogénéité et la mobilité.

Pour faciliter ce pilotage avec un maximum de dynamicité et de flexibilité, nous avons centré notre étude sur 3 axes principaux: la dimension organisationnelle, la dimension fonctionnelle (le composant de service autonomique) et la dimension protocolaire.

Basée sur les travaux existants de notre groupe de travail qui traitent la dimension architecturale représenté par l'architecture UBIS et la dimension informationnelle représentée par le modèle informationnel, nous avons proposé une nouvelle organisation pour faire face au problème de gestion dans les architectures contemporaines, et avoir une gestion distribuée qui favorise le principe de coopération entre les acteurs de l'architecture selon leur rôle pour répondre à un besoin non-fonctionnel (QoS) qui sera piloté à tous niveaux architecturaux.

Dans l'objectif de satisfaire le SLA contacté, Nous avons proposé un composant de service autonomique qui dispose d'un agent de QoS qui contrôle ses ressources en temps réel et qui réagit dynamiquement et de manière autonome dans le cas d'un changement dans le contrat de QoS (Disponibilité, Fiabilité, Délai et Capacité) au cours de la session de l'utilisateur. Le contrôle et la gestion de l'agent QoS s'appliquent durant toutes les phases de la session (pré-provisioning, provisioning et consommation) pour garantir à l'utilisateur une QoS en totale adéquation avec ses attentes et le prix qu'il paye.

Nous avons proposé dans la dimension protocolaire, un protocole de signalisation SIP+ qui a pour rôle l'établissement, la modification et la terminaison d'une session de service unique et continue dans une architecture trans-organisationnelle. Ce protocole sert à une négociation de la QoS des services personnalisés lors de la phase d'initialisation de la session de service et aussi à une renégociation de la QoS lors de la phase d'exploitation, pour remédier aux différents problèmes de dégradation de la QoS qui sont principalement causés par la mobilité.

Enfin, Ce protocole SIP+ permet d'avoir un «Service Delivery» au niveau de la couche service qui se rajoute au «Media Delivery» du niveau transport.

	Existant	**Défi ?**	**Proposition**
Organisation	Pas d'autogestion selon les besoin non-fonctionnels	Comment avoir une organisation Auto-suffisante Automatisée et auto-gérable?	Dimension organisationnelle: Acteurs, rôle, Agent QoS
Service	Gestion centralisée de la QoS Application monolithique	Comment offrir des services personnalisés? Comment décentraliser la gestion de la QoS?	Composant de service autonomique
Session et protocoles	Multimédia service Delivery SIP/SDP négociation de la QoS Media	Comment avoir une session unique et mobile? Service Delivery?	Dimension protocolaire : Protocole SIP+ Initiation / modification /terminaison d'une session mobile de services personnalisés

Chapitre VI Implémentation

Dans cette partie, nous présentons un scénario (§VI.1) qui illustre un cas d'usage de la personnalisation des services et qui montre l'utilité d'avoir une autogestion de la QoS, basée sur un agent de QoS dans un contexte ubiquitaire, au cours de la mobilité de la session. Puis dans (§VI.2), nous détaillons notre plate-forme de développement et de tests UBIS, qui est utilisée pour le développement des services applicatifs ainsi que ceux de gestion que nous avons proposé pour gérer la création et la modification de la session UBIS.

VI.1 Scénario

Dans cette section, nous utilisons un cas d'usage pour démontrer l'utilité de l'ensemble de nos propositions. Le scenario suivant illustre le cas où un utilisateur souhaite personnaliser ses services pour chercher un bien immobilier à acheter dans sa zone géographique.

Il choisit dans son catalogue trois services exposables dont il a besoin : Search Proprety, Google Maps , et banking.

Services exposables	Recherche bien immobilier (Search proprety)			Google Maps		Banque (Banking)		
Services de base	FIND	LOCATION	GET	Display on Map	Calculate Itiniraire	Demand credit	Insurance	Secure Payment
Qualité de service	Haute			Haute		Haute		
Fournisseur	Square Habitat			Google		Crédit agricole		

Le scénario se déroule comme suit (Figure 39):

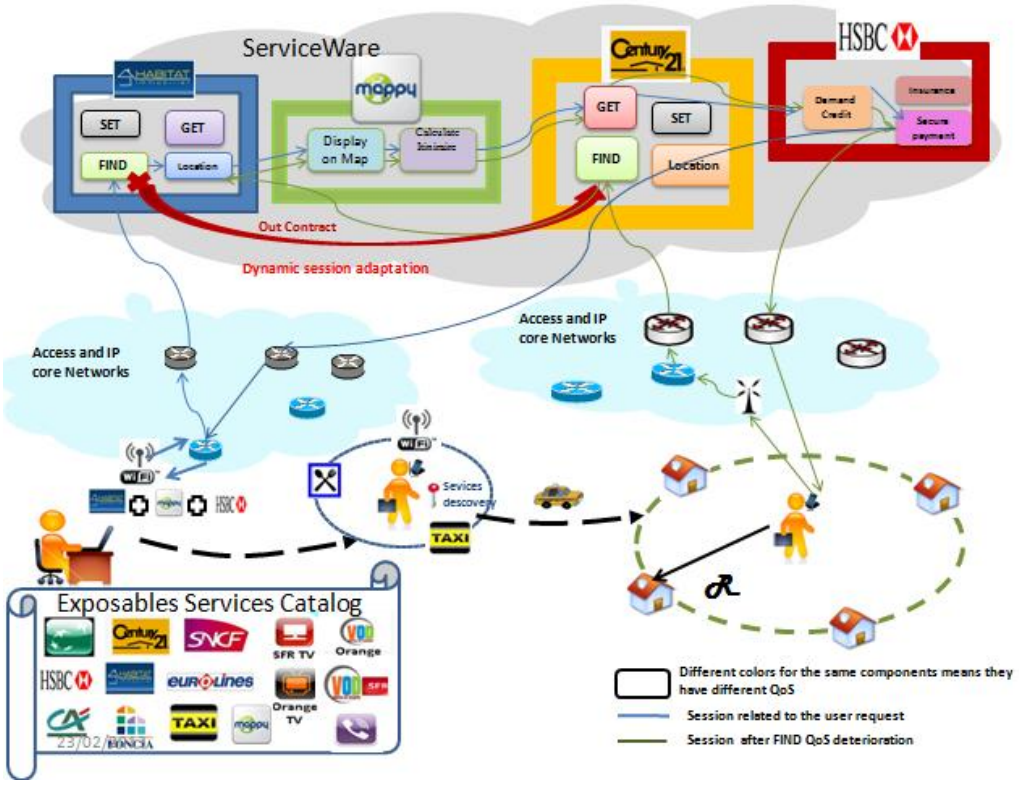

Figure 39 : Scénario de la mobilité de la session des services personnalisés

Lorsque l'utilisateur est en train de se déplacer, il va utiliser le composant FIND pour rechercher des biens immobiliers en fonction de sa position géographique (Longitude, Latitude). Cette recherche est effectuée dans la zone ambiante de l'utilisateur qui est limitée par un rayon R. Ensuite le composant de service «Location» est utilisé pour obtenir les coordonnées GPS pour chaque bien immobilier. Ensuite il utilise Google Maps pour localiser le bien sur une carte, et après il utilise le composant de service «GET» pour voir le profil détaillé de chaque bien sélectionné (la nature, la superficie, le prix, l'adresse). Et finalement, il va utiliser le service «Banking» pour d'une part demander un prêt d'achat et une assurance pour le bien qu'il a choisi, et d'autre part pour effectuer, le paiement sécurisé en ligne.

Au cours de l'exploitation, la performance de " FIND@Habitat" se détériore et ne peut pas fournir la qualité de service requise par l'utilisateur. Par conséquent, "FIND@Habitat" doit être remplacé par un composant de service FIND ubiquitaire. FIND@HABITAT envoie un

évènement de notification «Out contrat» via le message SIP+ Notify à tous les membres de sa communauté VSC pour traiter son changement. Il existe toujours des composants ubiquitaires fournis par d'autres fournisseurs tels que FIND@Century21.

Pour tester la performance de nos propositions, nous avons fait tourner ce scénario dans un premier temps avec des composants de service sans l'agent QoS (BSC), et dans un second temps avec des composants de service autonomiques (ASC).

- 1er cas: nous surchargeons le composant de service FIND, on remarque que des données sont perdues à cause de la dégradation de la QoS et par conséquent la session est interrompue.

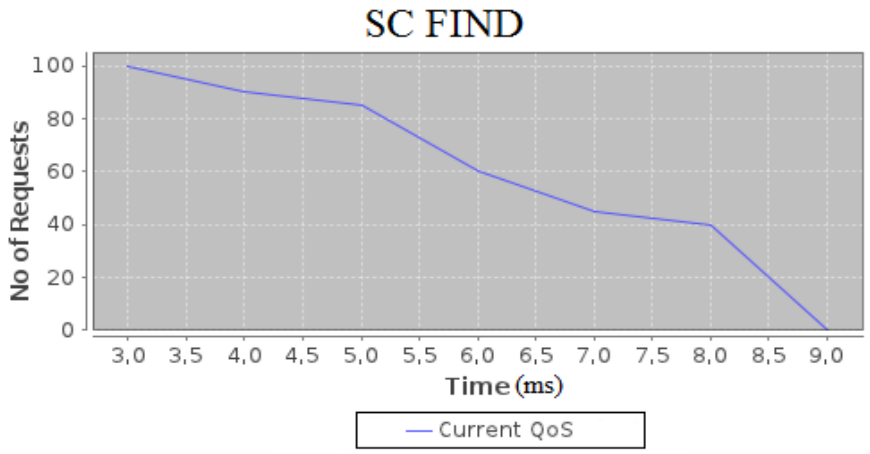

Figure 40 : les performances d'un composant de service basique FIND

- 2ème cas: nous utilisons des composants de service autonomiques ASCs avec les caractéristiques suivantes:

- Find (QoS α): A1= 0, 99, R1= 0, 98, D1=300ms, C1=10.
- Location (QoS β): A2=0, 97, R2=0, 86, D2=500ms, C2=8.
- Get (QoS µ): A4=0, 98, R4=0, 90, D4= 450ms, C4=9.

Nous mentionnons que si un des quatre critères de QoS se dégrade au cours de la session de l'utilisateur, il faut procéder au changement du composant de service parce que tous les critères de QoS sont nécessaires pour maintenir la QoS globale du composant de service.

Durant la session de l'utilisateur, l'agent QoS du composant de service ASC_FIND@Habitat détecte une dégradation de la QoS dans le critère de la Fiabilité (R1min = 0, 8 à T=8ms). Il invoque la communauté VSC pour procéder à son changement par un composant de service équivalent ASC_FIND@Century21.

La performance de la QoS est meilleure grâce au QoS-agent parce qu'il évite une dégradation complète de la QoS d'un composant de service et maintient le «service delivery»

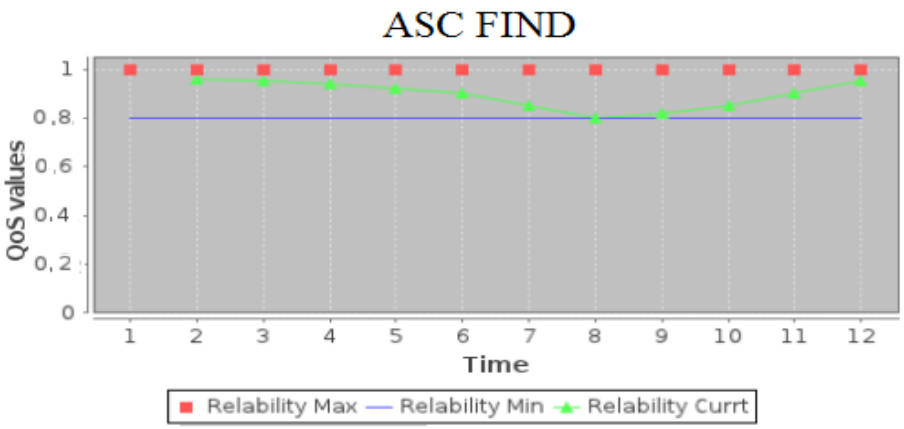

Figure 41: les performances d'un composant de service autonomique FIND

VI.2 Plate-forme de développement et de test UBIS

Pour mettre en place le projet UBIS, nous disposons d'une plate-forme UBIS qui nous permet de démontrer la faisabilité de l'ensemble de nos propositions et les tests des cas d'usages. La plate-forme est développée selon la structure de l'architecture UBIS, elle est structurée en fonctions des quatre niveaux de visibilité: Utilisateur, Terminal, Réseau et Service.

- *Utilisateur:* les abonnements et les préférences de l'utilisateur en termes de personnalisation de service et de mobilité seront stockés dans les bases de connaissances Infosphère et Infoware.
- *Terminal* : l'interface USERWARE fournit les fonctionnalités et les capacités pour permettre à l'utilisateur de personnaliser ses services, et lui offre un accès orienté usage qui est transparent de bout-en-bout aux services demandés.
- *Réseau Accès*: le réseau 3G de la plate-forme R&D de SFR (Société Française de Radiophonie) et le réseau WIFI de Telecom ParisTech sont utilisés pour assurer l'accès aux services.
- *Réseau de transport* : nous nous appuyons sur la solution VIRTUOR qui propose des machines physiques supportant plusieurs entités virtuelles de différents types comme les routeurs IPv4, IPv6, point d'accès virtuel, serveur SIP etc. Cette solution offre la possibilité de mettre en place plusieurs réseaux virtuels, spécifiques aux applications supportées, qui sont vus de la part des utilisateurs comme des réseaux physiques distincts.
- *Plan de contrôle* : nous utilisons le OpenIMS de FOKUS [47] pour mettre en place l'architecture IMS.
- *Les informations* : nous avons une base de connaissance INFOWARE [49] du coté fournisseur et INFOSPHERE [48] du coté utilisateur/terminal.
- *Les services* : nous avons SERVICEWARE, qui représente une architecture de service qui héberge différents types de service : des services applicatifs (Web, ASC) et des services de gestion (MSG).

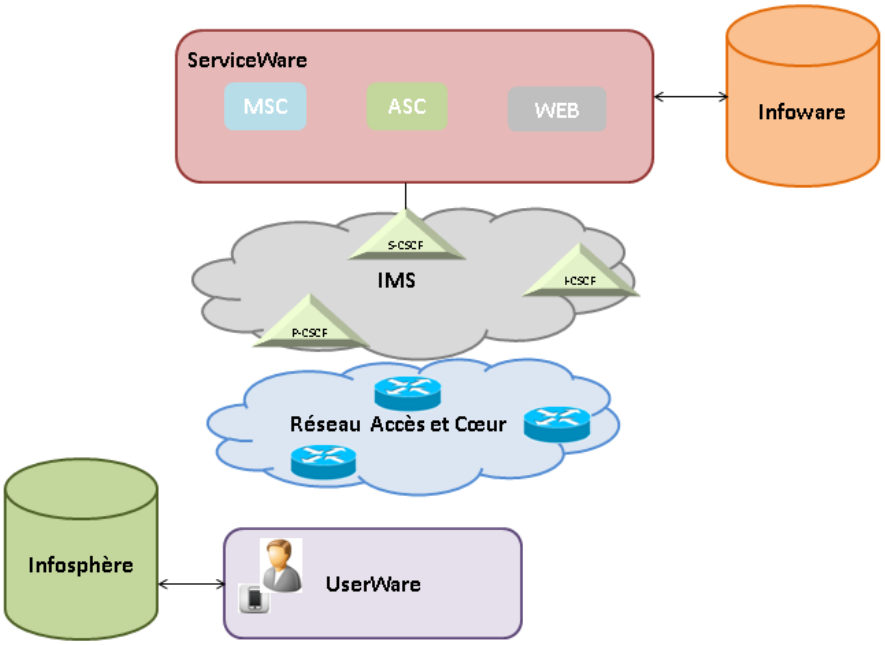

Figure 42: Plate-forme UBIS

VI.2.1 Architecture ServiceWare

Pour mettre en place l'architecture du ServiceWare, nous avons utilisé un service d'application GlassFish V3 /Sailfin pour développer et déployer les services UBIS qui s'appuient sur l'architecture SOA. Nous avons utilisé le langage Java pour développer les composants de service de type gestion et applicatif comme des EJBs indépendants. Il faut mentionner qu'EJB3.0 permet le développement des composants de services autonomes avec un couplage lâche. Ces composants sont déployés dans un serveur Sailfin certifié JEE6, qui supporte différentes APIs comme JMS, JNDI et JDBC et SIP servlet.

Pour les besoins de signalisation de la session et d'usage, les requêtes du protocole SIP + sont créées avec le client SIPP [46] pour créer, modifier et terminer les sessions de services personnalisés, et les requêtes HTTP sont utilisées pour la consommation des services. Nous avons une interface convergée SIP/HTTP «JSR Converged Applications» qui représente un ensemble de Sip Servlet et Http Servlet qui sont gérés par le même conteneur pour assurer l'interfaçage des différents composants EJB ou Web Services.

Le JNDI (Java Naming and Directory Interface) permet la connexion à des annuaires comme LDAP. Il est utilisé pour assurer la gestion des noms des files d'attente rattachée à chaque composant de service. Le JNDI offre les fonctionnalités de nommage des applications développées en JAVA et assure l'association entre les noms et les objets.

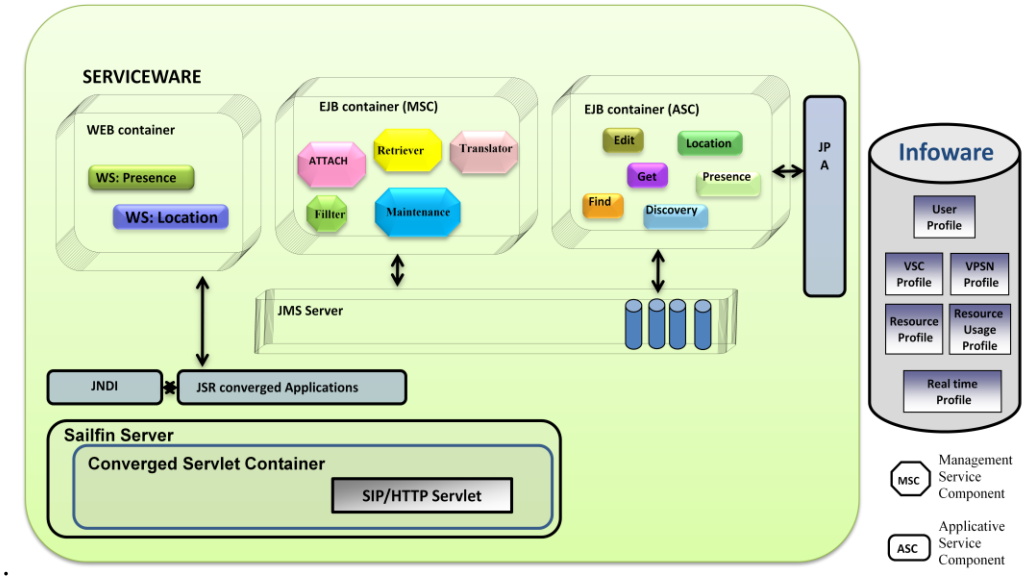

Figure 43: Architecture Serviceware

VI.2.2 La logique de fonctionnement des composants de services de gestion

Comme nous l'avons présenté dans le (§V.3.2), le protocole SIP+ permet le transport des informations fonctionnelles et non-fonctionnelles relatives à la logique des services personnalisés par l'utilisateur. Nous présentons, le modèle fonctionnel et le modèle informationnel, utilisés pour mettre en place les composants de service de types gestion (SIP_Retreiver, Translator, Filter, Attach_VPSN, Maintenance_VPSN) qui sont utilisés pour traiter la demande de l'utilisateur et créer son VPSN.

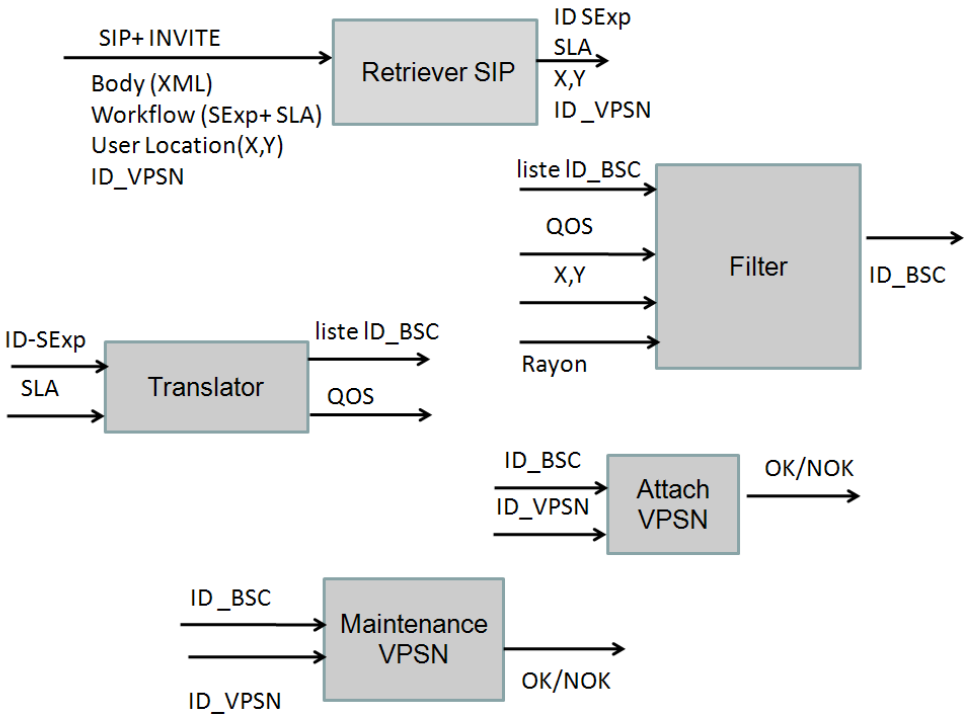

Figure 44 : Modèle fonctionnel des composants de service de gestion

Pour représenter le fonctionnement des différents composants de gestion, nous avons rédigé un pseudo code qui détaille le rôle de chaque composant dans la gestion de la demande de pré-provisioning du VPSN pour l'utilisateur. Ce pseudo code montre également la chaine d'exécution de chaque service suite à la réception d'une requête SIP+ INVITE de demande de création de session par l'utilisateur.

```
 ----------------------------
| ALGORITHM : Vpsn Creation |
| INPUT     : Invite SIP+   |
| OUTPUT    : VPSN Created  |
 ----------------------------

    Input :Sip+ Invite
        I# (-ID-VPSN-) ,(LA,LO)(-User's Location-)
        µ# (-WorkFlow of SExp-)
    BEGIN:
        Step1:
            Sip+ Invite goes to RETREIVER
        RETREIVER:
            Input  : SIP+ Invite Body = {µ#,(LA,LO),I#}
                BEGIN:
                    retrieve information from sip+ Invite Body
                END
            Output :ID-SExp,SLA,(LA,LO),I#
    Step2:
        µ# goes to TRANSLATOR
        TRANSLATOR:
            Input  :ID-SExp , SLA
                BEGIN :
                    FOREACH
                        ID-SExp translate into BSC
                    ENDFOREACH
                END
            Output :List{ID-BSC(Offered-QoS)}
    TRANSLATOR output goes to FILTER

LOOP:BSC-LIST
Step3:
    FILTER:
        Input  :(LA,LO),ID-BSC,Я // User lookup zone radius
            BEGIN:
                    verify Я
                    find ID-BSC
                    send ID-BSC to ATTACH-VPSN
            END
        Output :ID-BSC + Logical-@
    FILTER output goes to ATTACH-VPSN
    ATTACH-VPSN:
    Input : ID-BSC , Logical-@ , I#
        BEGIN:
                verify Ж-QoS // Statistical QoS
                IF !MAX(Ж-QoS)
                    THEN add I# to BSC-Profile
                    ELSE
                        Return to Step3
                    IF FILTER Return == NULL
                        THEN Я++
                        Return to Step3
                    ENDIF
                ENDIF
        END
    Output : OK
    VPSN-MAINTENANCE:
    Input  : ID-BSC , Logical-@ , I#
        BEGIN:
                add ID-BSC + Logical-@ to VPSN-Profile
        END
    Output : OK
ENDLOOP
VPSN-MAINTENANCE Sends SIP+ 200 OK (VPSN Created) to USER
END
```

Figure 45: Pseudo code de la création du VPSN

La base de connaissance Infoware est centralisée et contient tous les profils des différentes ressources: équipements, réseaux services ainsi que le profil de l'utilisateur. Les tables

suivantes sont utilisées pour tester le fonctionnement de nos différents composants de services (usage et gestion) au cours d'une session VPSN.

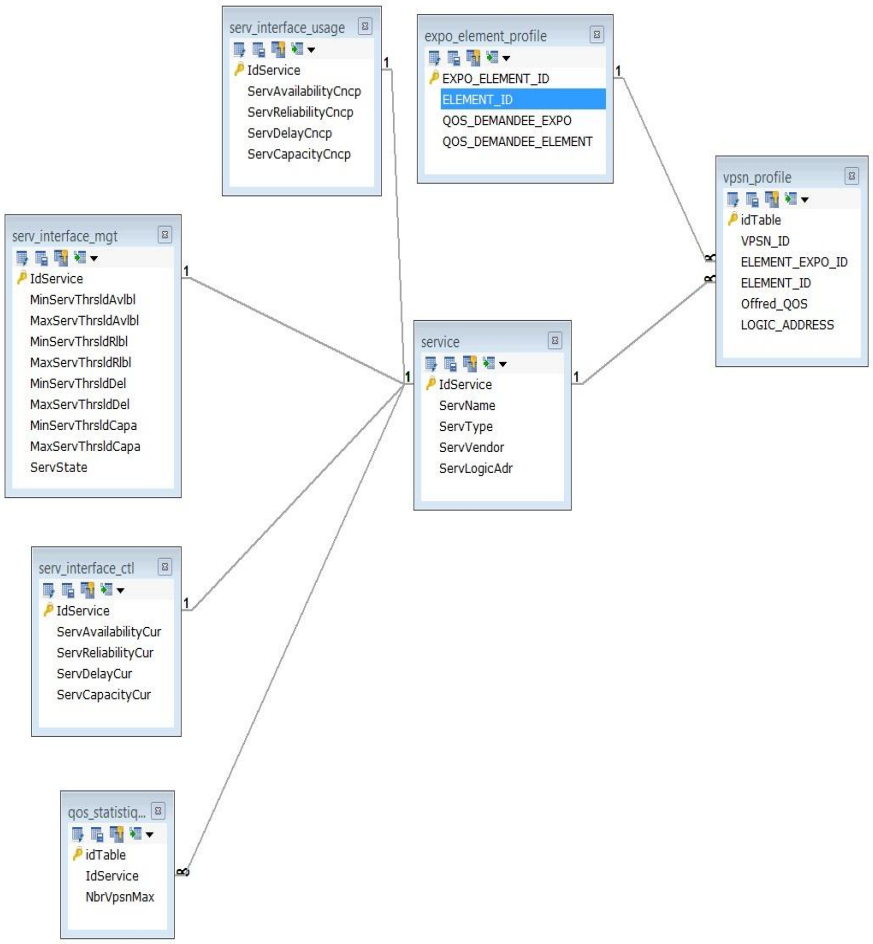

Figure 46: le modèle relationnel des tables du profil des éléments service dans l'Infoware

VI.2.3 Les résultats d'exécution des composants de service de gestion

Dès la réception de l'invite SIP+, le SIP_Retriever procède à la lecture du Contenu XML et extrait les données suivantes

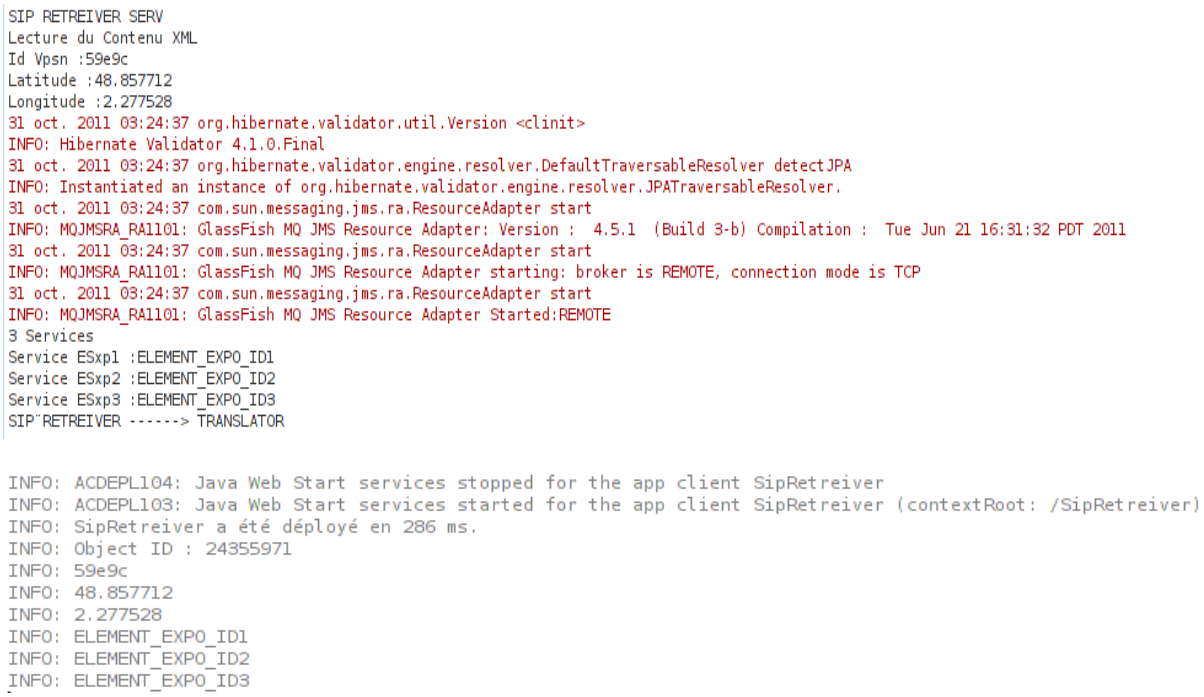

Figure 47: capture d'écran de l'exécution du SIP_Retriever

Le service SIP_Retriever a extrait les données «ID-VPSN», Longitude, Latitude et les IDs de services exposables**.** Ces informations sont envoyées avec des messages objets à la file d'attente JMS et elles sont stockées en FIFO (First In First Out) pour qu'ils soient traités par la suite par le composant Translator.

Chaque service exposable est retiré de la file JMS selon la logique FIFO et traduit par le composant Translator en composants de services élémentaires. Chaque service de base est envoyé conjointement avec sa QoS conceptuelle, l'ID-VPSN et les informations sur la localisation (Lo, La) à la file d'attente du composant de service Filter:

```
------- Translator Serv -------
31 oct. 2011 03:43:12 org.hibernate.validator.util.Version <clinit>
INFO: Hibernate Validator 4.1.0.Final
31 oct. 2011 03:43:13 org.hibernate.validator.engine.resolver.DefaultTraversableResolver detectJPA
INFO: Instantiated an instance of org.hibernate.validator.engine.resolver.JPATraversableResolver.
31 oct. 2011 03:43:13 com.sun.messaging.jms.ra.ResourceAdapter start
INFO: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter: Version :  4.5.1  (Build 3-b) Compilation :  Tue Jun 21 16:31:32 PDT 2011
31 oct. 2011 03:43:13 com.sun.messaging.jms.ra.ResourceAdapter start
INFO: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter starting: broker is REMOTE, connection mode is TCP
31 oct. 2011 03:43:13 com.sun.messaging.jms.ra.ResourceAdapter start
INFO: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter Started:REMOTE
Object Found :::
sipretreiver.MyObject@74187a
*******************************
ELEMENT_EXPO_ID1
ELEMENT_EXPO_ID2
ELEMENT_EXPO_ID3
 Element Complexe 1 :ELEMENT_EXPO_ID1
--Element Basique :ELEMENT_ID1.A / [0.9, 0.9, 2500, 100]
--Element Basique :ELEMENT_ID2.A / [0.85, 0.9, 3000, 60]
 Element Complexe 2 :ELEMENT_EXPO_ID2
--Element Basique :ELEMENT_ID2.A / [0.85, 0.9, 3000, 60]
--Element Basique :ELEMENT_ID3.A / [0.8, 0.8, 3100, 120]
--Element Basique :ELEMENT_ID4.A / [0.9, 0.85, 2500, 80]
 Element Complexe 3 :ELEMENT_EXPO_ID3
--Element Basique :ELEMENT_ID1.B / [0.9, 0.89, 2500, 100]
--Element Basique :ELEMENT_ID5.B / [0.9, 0.87, 2500, 100]
Translator ------> Filter
```

Figure 48 : Exécution du Service Translator

Les messages objets envoyés par Translator sont consommés un par un selon une boucle par le composant Filter

```
Object Found :::
translatorcon.MyExpoObject@10f7ec9
*******************************
Element trouvé dans le rayon :ELEMENT_ID2.A
Qos :[0.85, 0.9, 3000, 60]
Adresse logique :sip:ELEMENT_ID2.A@ubis01.sfr.com

Element trouvé dans le rayon :ELEMENT_ID3.A
Qos :[0.8, 0.8, 3100, 120]
Adresse logique :sip:ELEMENT_ID3.A@ubis01.sfr.com

Element trouvé dans le rayon :ELEMENT_ID4.A
Qos :[0.9, 0.85, 2500, 80]
Adresse logique :sip:ELEMENT_ID3.A@ubis01.sfr.com


FILTER --------> ATTACH-VPSN
```

Figure 49: Exécution du Composant Filter

Le composant Filter reçoit les messages objets stockés dans la JMS Queue et effectue la recherche dans la zone ambiante limitée par le rayon R, pour sélectionner les composants de services qui correspondent au contrat de QoS exigé par l'utilisateur, et il récupère aussi l'adresse logique pour chaque composant de Service.

Finalement, ces informations sont envoyées à une autre file d'attente JMS pour qu'elles soient traitées par les composants de services Attach_VPSN et VPSN_Maintenance pour avoir le double attachement VPSN/BSC et BSC/VPSN)

```
INFO: Instantiated an instance of org.hibernate.validator.engine.resolver.JPATraversableResolver.
INFO: EclipseLink, version: Eclipse Persistence Services - 2.3.0.v20110604-r9504
INFO: file:/home/adnane/Bureau/netbeansprojects/Filter/build/classes/_FilterPU login successful
INFO: Dist :3.714353735606815E-4
INFO: Rayon :10
INFO: Object ID : 20411437
INFO: Vpsn Id :59e9c
INFO: Expo Id :ELEMENT_EXPO_ID2
INFO: BSC ID :ELEMENT_ID2.A
INFO: Logic Address :sip:ELEMENT_ID2.A@ubis01.sfr.com
INFO: Dist :3.7746104614649 62E-4
INFO: Rayon :10
INFO: Object ID : 13287953
INFO: Vpsn Id :59e9c
INFO: Expo Id :ELEMENT_EXPO_ID2
INFO: BSC ID :ELEMENT_ID3.A
INFO: Logic Address :sip:ELEMENT_ID3.A@ubis01.sfr.com
INFO: Dist :3.82786173386934E-4
INFO: Rayon :10
INFO: Object ID : 21062123
INFO: Vpsn Id :59e9c
INFO: Expo Id :ELEMENT_EXPO_ID2
INFO: BSC ID :ELEMENT_ID4.A
INFO: Logic Address :sip:ELEMENT_ID3.A@ubis01.sfr.com
```

Figure 50: Recherche du service dans la zone ambiante

Chapitre VII Conclusion Générale

L'objectif de cette thèse était d'apporter de nouvelles contributions dans ce nouveau contexte «user-centric, mobilité et QoS», pour fournir à l'utilisateur des services personnalisés qui sont en parfaite adéquation avec son usage, ses besoins (fonctionnels et non-fonctionnels (QoS)) et ses préférences tout en respectant le SLA contracté. Nous avons donc centré nos travaux sur les solutions qui vont nous mener au but principal de la thèse qui est le pilotage *dynamique* de la *QoS de bout en bout* pour garantir à l'utilisateur une *session mobile et unique*.

Dans cette conclusion, nous allons synthétiser nos contributions (§VII.1) et donner une idée générale sur nos perspectives (§VII.2).

VII.1 Contributions

Les utilisateurs d'aujourd'hui veulent personnaliser leurs services (vidéo, voix et données) *n'importe où*, *n'importe quand*, et avec *n'importe quel type de terminal* dans un contexte hétérogène et mobile. Pour répondre aux différents verrous et défis que nous avons identifiés précédemment, nous nous sommes intéressés dans cette thèse à trois aspects : la Dimension organisationnelle (§V.1), la Dimension fonctionnelle (composant de service autonomique) (§V.2) et la dimension protocolaire (§V.3), qui sont importants pour répondre aux besoins du nouveau contexte afin d'améliorer considérablement le «*Service Delivery*» au cours de la session de service:

Dans la dimension organisationnelle, nous avons repensé les architectures contemporaines pour avoir une organisation qui pilote d'une façon décentralisée et Top-Down la QoS de bout en bout, et cela pour obtenir le degré *d'automatisation*, *d'autogestion* et *d'autosuffisance* imposé par le nouvel environnement de l'utilisateur, qui demande le maximum de *dynamicité*,

de *flexibilité* et de *transparence* au cours des différents changements qui se produisent lors de la mobilité.

Le composant de service autonomique que nous avons proposé rentre dans le cadre de la dimension fonctionnelle, il joue un rôle très important dans « le Service Delivey » des services personnalisés à l'utilisateur. Grâce à son agent de QOS intégré, ce composant devient autonome et auto-gérable. Il permet de remonter le contrôle et la gestion de la QoS à la couche service de l'architecture pendant toutes les phases opérationnelles (pré-provisioning, provisioning et consommation), et cela pour contrôler les ressources et maitriser le service rendu à l'utilisateur.

Dans la dimension protocolaire, nous nous sommes intéressés au protocole de signalisation SIP+ qui va interagir au niveau applicatif pour répondre au besoin de la négociation et de la renégociation de la QoS à un niveau au-dessus du niveau réseau, et cela dans le but d'étendre la notion d'établissement, de modification et de terminaison de la session au niveau service pour garantir l'unicité et la continuité d'une session de services personnalisés, dans un environnement de l'utilisateur qui est totalement hétérogène et mobile. Cette contribution répond aussi au besoin de communications évènementielles nécessaires pour nous deux précédentes contributions, pour avoir une interaction flexible et en temps réel qui correspond au principe de l'autogestion décentralisée. Pour le traitement des messages de signalisation SIP+ au niveau applicatif, nous avons proposé des composants de services de type gestion qui ont pour rôle de créer la session de services personnalisés (VPSN) en combinant des service exposables (Web, Telco, etc.), et nous avons aussi proposé des mécanismes de gestion basés sur les communautés (VSC) pour adapter d'une manière efficace la session de l'utilisateur en cas de dégradation de la QoS pendant le «Service Delivery».

VII.2 Perspectives

Notre travail de thèse rentre dans le cadre du projet UBIS (User centric : uBiquité et Intégration des Services), pour lequel nous avons développé un démonstrateur qui nous a permis d'étudier la faisabilité de nos différentes contributions, nous avons développé des composants de services autonomique de type applicatif et gestion et nous avons démontré que l'intégration d'un agent de QoS permet une QoS en adéquation aux besoins et un meilleur

«Service Delivery» des services personnalisés. Il faudrait maintenant tester la performance de ces contributions dans un cas d'utilisation réel qui inclut la personnalisation de services multifournisseurs en s'appuyant sur le protocole de signalisation SIP+ dans un environnement hétérogène et mobile.

Il faudrait aussi trouver la meilleure solution pour l'algorithme de Selection utilisé dans les communautés de service virtuelles (VSC), ayant pour rôle de traiter la demande de remplacement d'un service par un autre ubiquitaire, pour maintenir le service avec le même temps de réponse, dans le cas où une dégradation de la QoS se produit au cours de la session de l'utilisateur.

Références Bibliographiques

[1]. J. Kephart and D. Chess, "The Vision of Autonomic Computing" Computer, vol. 36, pp. 41-50, 2003.

[2]. S.Hariri, L.Xue, H.Chen, M.Zhang, S.Pavuluri, S.Rao, "AUTONOMIA: an autonomic computing environment" in IEEE IPCCC, 2003.

[3]. M.Parashar, H.Liu, Z.Li, V.Matossian, C.Schmidt, G.Zhang, and S.Hariri, "AutoMate: Enabling Autonomic Grid Applications," Journal of Networks, Software Tools, and Applications, Special Issue on Autonomic Computing, vol. 9, p. 14, 2006.

[4]. D. Garlan, S. W. Cheng, A. C. Huang, B. Schmerl, and P. Steenkiste, Rainbow: "Architecture-based self-adaptation with reusable infrastructure, " Computer, vol. 37, pp. 46-54, 2004.

[5]. S. Sicard, F. Boyer, and N. D. Palma, "Using Component for Architecture-Based Management," in ICSE, Leipzig, Germany, 2008.

[6]. Ramy.Farha « Autonomic Service architecture for next generation networks », University of Toronto, 2008 -652 pages

[7]. J.Strassner, S.van der Meer, B.Jennings, MP. Dr Leon "An Autonomic Architecture to Manage Ubiquitous Computing Networks and Applications" in proc. Of IEEE, USA June 2009.

[8]. bhakti, M.A.C, Abdullah, A.B" Design of an Autonomic Services Oriented Architecture", international Symposium, June 2010, Med doi : 10.1109/ITSIM.2010.5561601

[9]. Y. Liu, M.Tan, I. Gorton, A. John Clayphan " an Autonomic Middleware Solution for Coordinating Multiple QoS Controls " in proc. of ICSIC Springer – Verlag Berlin, Heidelberg 2008

[10]. F.Zambonelli, N.Bicocchi, G.Cabri, L.Leonardi, M.Puvianil, "On Self-adaptation, Self-expression, and Self-awareness in Autonomic Service Component Ensembles," Michigan, Oct 2011.

[11]. H.liu, M.Parashar, "A component based programming framework for autonomic applications, " Systems, vol 36, May. 2006, PP. 341-352, doi :

10.1109/TSMCC.2006.871577.

[12]. SOA "Portal for SOA and Web Services www.service-architecture.com, 2005.

[13]. E.Badidi, L.Esmahi "A Scalable Framework for Policy-based QoS Management in SOA Environments", Journal of Software, Vol6, 544-553, April 2011, Med doi : 10.4304/jsw.6.4.544-553.

[14]. D. Menascé, H.Ruan, H. Gomaa "QoS Management in Service-oriented Architectures" Journal of performance Evaluation, volume 64 issue 4-8, August, 2007, Med doi : 10.1016/j.peva.2006.10.001

[15]. M.Gillen, A.Paulos, J.Edmondson, P.Varshneya, D.C.Schmidt, L.Bunch,M. Carvalho, A.Martignoni "Dynamic Policy-Driven Quality of Service in Service-Oriented Systems" ISORC, May, Cambridge, USA 2010.

[16]. V. Cardellini, S.Iannucci "Designing a Broker for QoS-driven Runtime Adaptation of SOA Applications" ICWS, Rome, Italy July 2010.

[17]. A.Di Marco, A. Sabetta "Model-based dynamic QoS-driven service composition" in proc .of INRIA, Nov 2010.

[18]. S.Neelavathi and K.Vivekanandan "An Innovative Quality of Service (QOS) based Service Selection for Service Orchestration in SOA" International Journal of Scientific & Engineering Research Volume 2, Issue 4, April 2011.

[19]. Hamid M, Djamel B, Guy B (2009) Session continuity and splitting of multimedia applications using qualitative user preferences J mol Med doi:10.1145/1710035.1710087.

[20]. Chitra B, Khalid Al-B (2007) Towards a User-Centric and quality-aware multimedia service delivery Implementation on IP Multimedia Subsystem. Paper presented at the 7th international conference on next generation mobile applications, Services and technologies, IEEE Computer Society, Washington, DC, USA. 12-14 Sept. 2007.

[21]. José S, Thomas M (2010) Contextualized user-centric multimedia delivery system for next generation networks. In Telecommunication Systems Journal, Springer (in press).

[22]. Maamar Z, AlKhatib G, Mostefaoui S.K (2004) Context-based personalization of web services composition and provisioning. Paper presented at the 30 th Euromicro Conference, University of Zayed, United Arab Emirates, 31 Aug-3 Sept 2004.

[23]. M.Poikselkä, G.Mayer, H.Khartabil, A.Niemi - The IMS: IP Multimedia Concepts and Services, 2nd edition, Wiley, 2006

[24]. ETSI TS 129 214 V8.2.0 (2008-10): "Universal Mobile Telecommunications System (UMTS);Policy and charging control over Rx reference point (3GPP TS 29.214 version 8.2.0 Release 8)"

[25]. ETSI TS 129 212 V8.1.0 (2008-10): "Universal Mobile Telecommunications System (UMTS); Policy and charging control over Gx reference point (3GPP TS 29.212 version 8.1.0 Release 8)"

[26]. TSI ES 282 001 V2.0.0 (2008-03) NGN Functional Architecture

[27]. ETSI ES 282 004 V2.0.0 (2008-02): "Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); NGN Functional Architecture; Network Attachment Sub System (NASS)."

[28]. ETSI ES 282 003 V2.0.0 (2008-05): "Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); Resource and Admission Control Sub-system (RACS); Functional Architecture"

[29]. 3GPP TS 22.228 V9.1.0 (2009-03): "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Service requirements for the Internet Protocol (IP) multimedia core network subsystem"

[30]. 3GPP TS 129 214 V10.2.0 (2011-04):"Technical SpecificationUniversal Mobile Telecommunications System (UMTS); LTE; Policy and charging control over Rx reference point "

[31]. ETSI TS 122 135 V3.1.0 (2000-01): "Technical Specification Universal Mobile Telecommunications System (UMTS); Multicall; Service description"

[32]. ETSI TR 121 905 V10.3.0 (2011-03): "Technical Report Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS);

LTE; Vocabulary for 3GPP Specifications"

[33]. ETSI TR 122 978 V10.0.0 (2011-05): "Technical Report Universal Mobile Telecommunications System (UMTS); LTE; All-IP network (AIPN) feasibility study"

[34]. ETSI TS 123 279 V10.0.0 (2011-03): "Technical Specification Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Combining Circuit Switched (CS) and IP Multimedia Subsystem (IMS) services"

[35]. ETSI TS 123 203 V10.4.0 (2011-06): "Technical Specification Digital cellular telecommunications system (Phase 2+) ;Universal Mobile Telecommunications System (UMTS);LTE ;Policy and charging control architecture"

[36]. ETSI TR 102 805-2 V1.1.1 (2009-11): "Technical Report User Group; End-to-end QoS management at the Network Interfaces; Part 2: Control and management planes solution -QoS continuity"

[37]. TINA : "Telecommunications Networking Information Architecture, Service Architecture, Version: 5.0, 16 june 1997"

[38]. RFC 3261: SIP (Session Initiation Protocol) (2002-06)

[39]. RFC 4566 : SDP (Session Description Protocol) (2006)

[40]. RFC 5631: Session Initiation Protocol (SIP) Session Mobility

[41]. RFC 3725: Best Current Practices for Third Party Call Control (3pcc) in the session initiation protocol (SIP).

[42]. RFC 3515 : The Session Initiation Protocol (SIP) Refer Method

[43]. Z.Benhamed Daho, N.Simoni, M.Chevanne, S.Betgé-Brezetz, "An information model for service and network management integration: from needs towards solutions", IEEE/IFIP NOMS, Korea, 2004.

[44]. Noëmie SIMONI, Xiaofei XIONG, Chunyang YIN Virtual Community for the Dynamic Management of NGN Mobility ICAS'09, Avril, 09, Valencia, Spain.

[45]. S Kessal, N Simoni, "Service Provisioning oriented QoS," Paris, Oct 2011

[46]. SIPP " Portal for SIPP" www.sipp.sourceforge.net

[47]. Open IMS Core, FOKUS, http://www.openimscore.org/

[48]. N.Simoni, C.Yin, G. Du Chene, « An intelligent user centric middleware for NGN: Infosphere and AmbientGrid", In Proc. of COMSWARE, pages 599–606, Bangalore, India, Jan. 2008.

[49]. N.Ornelas, N.Simoni, K.Chen, and A.Boutignon, "VPIN: User-session knowledge base for self-management of ambient networks", UBICOMM'08, Oct. 2008, Spain. doi:10.1109/UBICOMM.2008.71.

Liste des publications

Houda ALAOUI SOULIMANI, Noémie SIMONI, Philippe COUDE, "Modèle organisationnel pour le pilotage dynamique de la qualité de service de bout en bout pour une session user-centric**"**, GRES 2010, Montréal, Canada Octobre 2010.

Houda ALAOUI SOULIMANI, Noémie SIMONI, Philippe COUDE "User-centric and QoS-based service session, conférence IEEE APSCC**"** APSCC 2011, Jéju, Corée du sud, Décembre 2011,

Houda ALAOUI SOULIMANI, Noémie SIMONI, Philippe COUDE "QoS-based Autonomic Service Component for Service Delivery**"**, IARIA CTRQ 2012, Avril 2012

Houda ALAOUI SOULIMANI, Noémie SIMONI «Delivrable ANRT: Architecture UBIS : Définition des interactions et protocoles», 2011

Houda ALAOUI SOULIMANI, Rachad Nassar, Noémie SIMONI «Delivrable ANRT Composants de services : Spécifications de la session et de la couche de Service», 2011