Titre: Title:	Architectures 3GPP et évolution vers IPv6
Auteur: Author:	Maxime De Roucy de Flacourt
Date:	2011
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	De Roucy de Flacourt, M. (2011). Architectures 3GPP et évolution vers IPv6 [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/511/

UNIVERSITÉ DE MONTRÉAL

ARCHITECTURES 3GPP ET ÉVOLUTION VERS IPV6

MAXIME DE ROUCY DE FLACOURT
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
FÉVRIER 2011

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

ARCHITECTURES 3GPP ET ÉVOLUTION VERS IPV6

présenté par: DE ROUCY DE FLACOURT, Maxime
en vue de l'obtention du diplôme de: Maîtrise ès Sciences Appliquées
a été dûment accepté par le jury d'examen constitué de:

Mme. BELLAÏCHE, Martine, Ph.D., présidente
M. PIERRE, Samuel, Ph.D., membre et directeur de recherche
M. QUINTERO, Alejandro, Doct., membre

À mes parents, Rémi et toute ma famille.

REMERCIEMENTS

Je voudrais particulièrement remercier mon directeur de recherche, le professeur Samuel Pierre, pour son support et son encadrement durant mon cursus de maîtrise. Je tiens aussi à remercier Yves Lemieux, chargé de mon encadrement au sein de la société Ericsson, pour son aide et son soutien, ainsi que Suresh Krishnan spécialiste sénior chez Ericsson pour ses réponses claires et précises.

Parmi les membres du Laboratoire de Recherche en Informatique Mobile (LARIM), je tiens particulièrement à remercier Stéphane Ouellette pour ses conseils et ses réponses à mes nombreuses questions. Merci, encore, à tous les membres du LARIM pour leur aide, entre autres Méral Shirazipour, Georges Abou Khalil et Angélo Rossi.

Parmi les stagiaires avec qui j'ai été amené à travailler, je tiens particulièrement à remercier Mehdi Hamdoun dont le travail m'a été très utile.

RÉSUMÉ

Dans les réseaux mobiles de troisième génération, le trafic de voix est séparé du trafic de données. Dans les réseaux de quatrième génération, tout le trafic passe par un unique réseau. Les technologies utilisées dans la 4^{e} génération (4G) sont des évolutions de celles développées pour General Packet Radio Services (GPRS) et n'utilisent que très peu les capacités du protocole Internet Protocol (IP) (présent sur le plan de données).

La norme IP version 6 (IPv6), finalisée en décembre 1998, définit le nouveau standard du protocole IP. Elle simplifie le protocole et augmente ses fonctionnalités, elle facilite notamment le support de la mobilité. Aujourd'hui les principaux protocoles étendant IPv6 ou l'utilisant pour fournir de la mobilité sont Mobile IPv6 (MIPv6) et Proxy Mobile IPv6 (PMIPv6). Malgré cette évolution de IP, le réseau 4G développé par le consortium Third generation partnership project (3GPP) utilise, pour gérer la mobilité des usagers, le protocole GPRS Tunneling Protocol (GTP) développé à la base pour GPRS. Il gère aussi en partie la Quality of Service (QoS) sur le réseau cœur Evolved Packet Core (EPC).

Les techniques de QoS permettent de différentier les trafics transitant sur le réseau et de leur appliquer des restrictions et/ou priorités différentes. Ces techniques sont généralement utilisées sur des réseaux susceptibles d'être saturés, ce qui est le cas dans les réseaux mobiles. Elles sont donc très importantes sur le réseau Evolved Packet System (EPS) d'autant plus que celui-ci transporte du trafic de voix. Le trafic de voix est critique, car la fonction première des réseaux mobiles est de connecter des téléphones cellulaires au Public Switched Telephone Network (PSTN). Le réseau doit aussi permettre un accès à un dorsal IP couramment appelé Internet, en théorie n'importe quel type de trafic peut y circuler : vidéo, web, File Transfer Protocol (FTP), Peer-to-Peer (P2P), etc.

Dans le protocole GTP des tunnels sont créés pour chaque usager, appelé User Equipment (UE), les tunnels d'un même UE ayant chacun une QoS différente. Mais tous les paramètres de QoS ne sont pas appliqués par le protocole GTP, certains comme la priorité de transmission (*forwarding priority*) doivent être gérés par les couches inférieures du réseau. Les techniques de QoS à utiliser ne sont pas définies par 3GPP, en revanche des documents comme le 23.401 [8] mentionnent Differentiated Services (DiffServ) comme exemple.

Le protocole PMIPv6 a l'avantage sur MIPv6 de ne pas modifier les fonctionnalités de la couche IPv6 du nœud mobile. De base il n'utilise qu'un tunnel entre un Mobile Access Gateway (MAG) et un Local Mobility Anchor (LMA) (voir section 2.2.3 pour plus d'explications) pour acheminer le trafic de tous les terminaux attachés au MAG. L'utilisation d'un seul tunnel empêche la différentiation des trafics et donc l'utilisation des méthodes de

QoS classiques. C'est pourquoi Hui *et al.* [70] propose de modifier le protocole PMIPv6 pour permettre la création d'un tunnel par flux, de façon à permettre l'application de règles de QoS différentes. En nous inspirant de ces modifications nous pouvons donc transformer le protocole PMIPv6 de façon à ce qu'il supporte les mêmes fonctionnalités que GTP User data tunneling (GTP-U).

Le protocole IPv6 est actuellement géré par quasiment tous les systèmes d'exploitation et les adresses IP version 4 (IPv4) encore disponibles disparaissent rapidement, nous avons donc considéré que le réseau EPS utilise uniquement le protocole IPv6. Partant de ce principe et ayant remarqué qu'IPv6 peut être utilisé pour gérer la mobilité des usagers, nous avons cherché à maximiser son utilisation. Nous proposons donc deux nouvelles architectures du réseau EPC utilisant une version de PMIPv6 légèrement modifiée pour supporter la gestion de plusieurs tunnels entre un MAG et un LMA, à la manière de GTP.

Étant donné le peu de bande passante dont disposent ces réseaux mobiles comparés aux réseaux d'accès filaires et le caractère critique de certains trafics qu'ils véhiculent, nous avons dû vérifier que nos modifications ne réduisent pas les performances du réseau du point de vue des usagers. Pour ce faire nous avons implémenté dans le simulateur Network Simulator version 2 (NS-2) les architectures EPS décrites dans les documents 23.401 [8] et 23.402 [9] ainsi que celles que nous proposons. Au vu des résultats nous pouvons dire que l'architecture 23.401 [8] offre de meilleures performances que toutes les autres. Cependant, 23.402 [9] a été développée pour répondre à une problématique d'ordre politique, survenant lors de l'ajout d'un réseau d'accès non développé par 3GPP au réseau cœur. Cette architecture, sur laquelle nous nous sommes basés, peut donc être mise en place dans n'importe quel environnement contrairement à la 23.401 [8]. Grâce aux simulations nous pouvons dire que les modifications que nous avons apportées ne modifient pas, voire améliorent légèrement, les performances de l'architecture 23.402 [9].

En conclusion nous conseillons aux opérateurs désirant mettre en place un réseau 4G utilisant une des architectures proposées par 3GPP, d'utiliser 23.401 [8] (étant donné ses meilleures performances) si cela leur est politiquement possible. Sinon d'utiliser l'une des architectures basées sur 23.402 [9] que nous proposons.

ABSTRACT

In third generation mobile network, the voice traffic is separated from the data traffic. In fourth generation all goes through a unique network. Technologies used in 4G network evolved from those developed for GPRS, they don't use IP's capacity.

IPv6 norm, released in 1998, describes the new standard for IP protocol. It simplifies the way IP works and increases its functionality, notably mobility integration were facilitated. Today MIPv6 and PMIPv6 are leaders protocols to include mobility in IPv6. Even with those evolutions, 3GPP's 4G network still uses GTP (firstly developed for GPRS) to manage its users mobility; but GTP also manages some part of the QoS mechanism.

QoS techniques permit differentiating network flows and apply some restrictions and/or priorities to them. Those techniques are usually used when the network can be congested, which is the case for mobile network. So they are important for EPS network, specially since it carries voice traffic. Voice traffic is critical since the one of the first goals of mobile network is to allow cellular phone to connect to the PSTN. The network must also provide facilities to the users to them to access an IP backbone, which means that theoretically all sorts of traffic can be presented, video, web, FTP, P2P, etc.

GTP protocol creates tunnels for each user, called UE, each tunnel of a UE has a different QoS. But all the QoS parameters aren't applied by GTP, some (like the forwarding priority) need to be applied by underlying stack. 3GPP doesn't specify the QoS techniques used by stacks under GTP. However, it mentions DiffServ as an example in the document 23.401 [8].

PMIPv6 protocol has the advantage against MIPv6 that it doesn't need the mobile node to have a modified IPv6 stack. Its basic version uses one tunnel between an LMA and a MAG (see section 2.2.3) to carry all traffics from and to mobiles nodes attached to this MAG. The use of a unique tunnel prevents the use of common QoS techniques like DiffServ since it hardeners the differentiation of traffic flows. That's why Hui *et al.* [70] suggest to modify PMIPv6 in order to create one tunnel by flow. So by applying small modifications to PMIPv6 (inspired by Hui *et al.* [70] suggestions) we should be able to support the same functionalities as GTP-U.

Currently, near all the operating systems have an IPv6 stack and the number of IPv4 addresses still unallocated is decreasing quickly. So we decided to assume that EPS network only uses the IPv6 protocol. As IPv6 can be used to manage some part of QoS and mobility through PMIPv6, we searched to increase its use in EPC. So we suggest two new architectures based on 23.402 [9] for EPC network, which used a modified version of PMIPv6 to support the creation of several tunnels between a MAG and an LMA and to match GTP-U functionalities.

However, due to the lack of bandwidth in such network and the criticism of voice traffic, we had to check that our suggestion doesn't damage the network performances from the users' point of view. To test these performances, we implement architecture 23.401 [8], 23.402 [9] and those that we propose. The results of our simulations show that 23.401 [8] has the best performances. However, this architecture suffers from some political problems and can't be built in every situation. That's why 3GPP released the 23.402 [9] architecture which has the advantage to politically support a larger scale of access networks. For our work, we based our architecture on 23.402 [9]. Simulations show that we don't damage and sometimes we even improve the performances of 23.402 [9].

To conclude, we advise operators wishing to build 4G networks based on 3GPP architectures to use the 23.401 [8] if it's possible. But if, for political reason, they can't use 23.401 [8] to consider ours since their performances are better and they improve the use of IPv6.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ALGORITHMES

LISTE DES SIGLES ET ABRÉVIATIONS

AAA	Authentication, Authorization, and Accounting
ACQ	Active Queue Management
AF	Assured Forwarding
AF	Application Function
AKA	Authentication and Key Agreement
AMBR	Aggregate Maximum Bit Rate
AMPS	Advanced Mobile Phone System
APN	Access Point Name
APN-OI	Access Point Name Operator Identifier
ARP	Allocation and Retention Priority
ARQ	Automatic Repeat reQuest
ASN	Access Service Network
ASN-GW	ASN Gateway
ASP	Application Service Provider
AuC	Authentication Center
AVP	Attribute Value Pairs
BA	Binding Acknowledgement
BBRR	bit-by-bit Round-Robin
BCE	Binding Cache Entry
BG	Boarder Gateway
BMC	Broadcast/Multicast Control
BSC	Base Station Controller
BSS	Base Station Subsystem
BSSGP	BSS GPRS Protocol
BTS	Base Transceiver Station
BS	Base Station
BU	Binding Update
BULE	Binding Update List Entry

CBQ	Class Based Queueing
CDMA	Code Division Multiple Access
CGF	Charging Gateway Functionality
CGI	Cell Global Identifier
CK	Confidentiality-Key
CN	Correspondent Node
CoA	Care of Address
CPU	Central Processing Unit
CQI	Channel-Quality Indicator
CSN	Connectivity Service Network
CSG	Closed Subscriber Group
DF	Default Forwarding
DFT	Discrete Fourier Transform
DFTS-OFDM	DFT-spread OFDM
DiffServ	Differentiated Services
DL	Downlink
DPI	Deep Packet Inspection
DRR	Deficit Round-Robin
DS-CDMA	Direct Sequence CDMA
DSCP	differentiated services codepoint
DWBBRR	Deficit Weighted bit-by-bit Round-Robin
EBI	EPS Bearer ID
ECGI	EUTRAN Cell Global Identifier
ECN	Explicit Congestion Notification
EDGE	Enhanced Data rates for GSM Evolution
EF	Expedited Forwarding
EGPRS	Enhanced GPRS
EIR	Equipment Identity Register
EMM	EPS Mobility Management
eNodeB	Evolved Node B

EPC	Evolved Packet Core
EPS	Evolved Packet System
ESM	EPS Session Management
ETSI	European Telecommunications Standards Institute
EUL	Enhanced Uplink
EUTRAN	evolved UMTS Terrestrial Radio Access Network
FA	Foreign Agent
FDD	Frequency Division Duplexing
FDM	Frequency Division Multiplexing
FDMA	Frequency Division Multiple Access
FIFO	First In First Out
FQ	Fair Queueing
FQ-CSID	Fully-qualified PDN Connection Set Identifier
FTP	File Transfer Protocol
F-TEID	Fully-Qualified Tunnel Endpoint Identifier
GBR	Guaranteed BitRate
GERAN	GSM EDGE RAN
GGSN	Gateway GPRS Support Node
GMLC	Gateway Mobile Location Center
GMM/SM	GPRS Mobility Management & Session Management
GMSC	Gateway MSC
GMSK	Gaussian Minimum Shift Keying
GNU	GNU is Not Unix
GPRS	General Packet Radio Services
GRE	Generic Routing Encapsulation
GSM	Global System for Mobile communications
GSN	GPRS support node
GTP	GPRS Tunneling Protocol
GTP-C	GTP Control
GTP-U	GTP User data tunneling

GTPv0	GTP first stage
GTPv1	GTP second stage, first version for EPS
GTPv2-C	GTP-C version 2
GTPv1-U	GTP-U version 1
GUI	Graphic User Interface
HA	Home Agent
HAA	Home Agent Address
HoA	Home Address
HARQ	Hybrid ARQ
HLR	Home Location Register
HNP	Home Network Prefix
HPLMN	Home Public Land Mobile Network
HSDPA	High-Speed Downlink Packet-Data Access
HS-DSCH	High Speed Downlink Shared channel
HSPA	High Speed Packet Access
HSUPA	High-Speed Uplink Packet-Data Access
HSS	Home Subscriber Server
HTB	Hierarchical Token Bucket
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secured
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
ICMPv6	Internet Control Message Protocol version 6 (ICMP pour IPv6)
IE	Information Element
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IK	Integrity-Key
IMEI	International Mobile Equipment Identity
IMS	IP Multimedia Subsystem

IMSI	International MS Identity
IMT-2000	International Mobile Telecommunications 2000
IntServ	Integrated services
IP	Internet Protocol
IP-CAN	IP Connectivity Access Network
IPDV	IP Packet Delay Variation
IPPM	IP Performance Metrics Working Group
IPSec	Internet Protocol Security
IPv4	IP version 4
IPv6	IP version 6
IR	Incremental Redundancy
ISDN	Integrated Services Digital Network
ITU	International Telecommunication Union
LAN	Local Area Network
LARIM	Laboratoire de Recherche en Informatique Mobile
LBI	Linked EPS Bearer ID
LDPC	Low-Density Parity-Check
LLC	Logical Link Control
LLQ	Low Latency Queueing
LMA	Local Mobility Anchor
LMAA	Local Mobility Anchor Address
LQC	Link Quality Control
LTE	Long Term Evolution
MAC	Medium Access Control
MAG	Mobile Access Gateway
MBR	Maximum BitRate
MCC	Mobile Country Code
ME	Mobile Equipment
MEI	ME Identity
MIMO	Multiple Input Multiple Output

MIP	Mobile IP
MIPv4	Mobile IPv4
MIPv6	Mobile IPv6
MME	Mobility Management Entity
MN	Mobile Node
MNC	Mobile Network Code
MPLS	Multiprotocol Label Switching
MS	Mobile Subscriber
MSC	Mobile Switching Center
MSISDN	MS ISDN Number
MT	Mobile Terminal
NAI	Network Access Identifier
NAP	Network Access Provider
NAS	Non-Access Stratum
NSAPI	Network layer Service Access Point Identifier
NSP	Network Service Provider
NS-2	Network Simulator version 2
NS-3	Network Simulator version 3
OFDM	Orthogonal Frequency-Division Multiplexing
OFDMA	Orthogonal Frequency-Division Multiple Access
OMC	Operations and Maintenance Center
OSI	Open Systems Interconnection
OTcl	Object Tool Command Language
PAA	PDN Address Allocation
PAPR	Peak-Average-Power Ratio
PBA	Proxy Binding Acknowledgement
PBU	Proxy Binding Update
PCC	Policy and Charging Control
PCEF	Policy and Charging Enforcement Function
PCRF	Policy and Charging Rules Function

PDCP	Packet Data Convergence Protocol
PDN	Packet Data Network
PDP	Packet Data Protocol
PDU	Protocol data unit
PFu	Policy Function
PFi	Packet Filter
PFID	Packet Filter Identifier
PGW	Packet Data Network Gateway
PHB	Per-Hop Behavior
PLMN	Public Land Mobile Network
PMIPv6	Proxy Mobile IPv6
PPP	Point to Point Protocol
PQ	Priority Queueing
PRO	Procedure
PS	Packet Switched
PSK	Phase-Shift Keying
PSTN	Public Switched Telephone Network
PTI	Procedure Transaction Id
P2P	Peer-to-Peer
QAM	Quadrature Amplitude Modulation
QCI	QoS Class Identifier
QoS	Quality of Service
QPSK	Quadrature PSK
RADIUS	Remote Authentication Dial-In User Service
RAI	Routing Area Identity
RAN	Radio Access Network
RAT	Radio Access Type
RED	Random Early Detection
RFC	Request for Comments
RLC	Radio Link Control

RNC	Radio Network Controller
RR	Round-Robin
RRC	Radio Resource Control
RTP	Real Time Protocol
SAE	System Architecture Evolution
SAI	Service Area Identifier
SASN	Service Aware Support Node
SC-FDMA	Single-carrier FDMA
SCTP	Stream Control Transmission Protocol
SDU	Service Data Unit
SF	Service Flow
SFI	Service Flow Identifier
SFQ	Stochastic Fairness Queueing
SGSN	Serving GPRS Support Node
SGW	Serving Gateway
SIM	Subscriber Identity Module
SMS	Short Message Service
SNDCP	Sub Network Dependent Convergence Protocol
SOFDMA	Scalable OFDMA
SPI	Security Parameters Index
srTCM	single-rate, three-color marker
SSH	Secure Shell
STN	Session Transfer Number
S1-AP	S1 Application Protocol
TA	Timing Advance
TAI	Tracking Area Identity
TAD	Traffic Aggregate Description
TBF	Token Bucket Filter
Tcl	Tool Command Language
TCP	Transmission Control Protocol

TD-CDMA	Time Division CDMA
TDD	Time Division Duplex
TDMA	Time Division Multiple Access
TEID	Tunnel Endpoint Identifier
TFT	Traffic Flow Template
TID	Tunnel Identifier
ToS	Type of Service
trTCM	two-rate, three-color marker
TSN	Transmission Sequence Numbers
UCI	User CSG Information
UDP	User Datagram Protocol
UE	User Equipment
UE-AMBR	UE Aggregated Maximum BitRate
ULA	Update-Location-Answer
ULI	User Location Information
ULR	Update-Location-Request
UMTS	Universal Mobile Telecommunications System
USIM	Universal subscriber identity module
UTRA-FDD	UMTS Terrestrial Radio Access - Frequency Division Duplexing
UTRAN	UMTS Terrestrial Radio Access Network
UTRA-TDD	UMTS Terrestrial Radio Access - Time Division Duplexing
VoIP	Voice over IP
VLR	Visitors Location Register
VoD	Video on Demand
VPLMN	Visited Public Land Mobile Network
WBBRR	Weighted bit-by-bit Round-Robin
W-CDMA	Wideband Code Division Multiple Access
WFQ	Weighted Fair Queueing
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network

WRED	Weighted Random Early Detection
WRR	Weighted Round-Robin
XMPP	Extensible Messaging and Presence Protocol
3GPP	Third generation partnership project
4G	quatrième génération

LISTE DES ANNEXES

CHAPITRE 1

INTRODUCTION

Les réseaux cellulaires sont classés par « génération » et actuellement ceux de deuxième et troisième générations sont les plus courants. Avec la croissance d'Internet, le besoin d'offrir des services habituellement réservés aux réseaux filaires comme le *streaming* audio ou vidéo ou le *web browsing*, augmente. Les réseaux 3G ont été développés dans le but d'augmenter la capacité de transmission des données et ainsi améliorer le confort des usagers utilisant des services multimédia. Cependant, la capacité des réseaux mobiles est encore limitée par rapport à celle des accès traditionnels filaires. Leurs développement est donc toujours très actif et la 4^{e} génération est actuellement en préparation.

Nous avons étudié les architectures des réseaux 4G proposées par le consortium 3GPP et cherché à les améliorer.

Dans ce chapitre nous allons définir quelques concepts de base, puis nous présenterons notre problématique et nos objectifs de recherche. Nous finirons en détaillant le plan de ce mémoire.

1.1 Définitions et concepts de base

Avec le développement important qu'a connue ces derrières années l'informatique mobile et le marché des téléphones cellulaires, de plus en plus de services sont offerts sur ces plateformes. Les réseaux mobiles ont dû évoluer pour permettre le transport d'autres types de trafics que celui de voix. Aujourd'hui les réseaux 3G sont adaptés à la transmission de voix et de données (e.g. web, FTP, Extensible Messaging and Presence Protocol (XMPP)), mais ceux-ci restent séparés, cela implique une gestion plus complexe des réseaux. Dans la nouvelle génération de réseau mobile, la 4G, il est prévu de considérer tout le trafic, y compris la voix, comme du trafic de données et de le faire passer sur un même réseau à commutation de paquets. Dans la 4G il n'y a donc plus de réseaux à commutation de circuits.

Pour acheminer le trafic, les protocoles IPv4 (défini par Postel [88]) et/ou IPv6 (défini par Deering et Hinden [56]) sont utilisés, cependant ils ne spécifient pas de système de priorisation des types de trafics. IPv4 contient un champ Type of Service (ToS) renommé *Traffic Class* en IPv6, et maintenant divisé en differentiated services codepoint (DSCP) et Explicit Congestion Notification (ECN) respectivement défini par Nichols *et al.* [80] et Ramakrishnan *et al.* [91]. Ces champs indiquent la politique de QoS qui devrait être appliquée aux paquets, mais pas

la manière de l'appliquer, si bien qu'aujourd'hui la majorité des applications et des routeurs ne les traitent pas.

Ce problème n'est pas très important tant que le réseau reste assez rapide pour qu'aucune congestion ne survienne. Mais les réseaux mobiles sont très limités en terme de bande passante par rapport aux réseaux câblés classiques.

Les différents types de trafics n'ont pas les mêmes exigences en matière de bande passante, taux de perte de paquets, délai, IP Packet Delay Variation (IPDV), etc. La vidéo non interactive peut supporter un délai entre le serveur et le récepteur assez important sans que l'utilisateur ne soit gêné ; en revanche, ce trafic est très sensible aux pertes de paquets. Une perte, ne serait-ce que de 0,5 % des paquets peut se traduire par une dégradation importante de la qualité de l'image [76]. Le trafic de voix est sensible au délai, mais peut supporter un taux de perte de paquets plus important (jusqu'à 1 % selon Wallingford [100, chap. 9]). Un transfert de document ne requière pas de fortes contraintes sur le délai, mais une bande passante élevée est préférable.

Les mécanismes qui permettent de traiter différemment les paquets suivant leur contenu sont appelées techniques de QoS. Dans les réseaux mobiles 4G, pour offrir une qualité de service comparable à celle de la téléphonie traditionnelle en plus de permettre l'accès aux réseaux de données (communément appelé Internet), le consortium d'industriels 3GPP propose d'utiliser un système de *bearer*. Les *bearers* sont des tunnels ayant chacun leurs paramètres de QoS [63, 73].

Le document 23.401 [8] propose une architecture utilisant des *bearers* GTP (protocole développé par 3GPP). GTP se divise en deux branches, GTP-U et GTP Control (GTP-C) ; GTP-U est destiné à encapsuler les paquets IP [24], il gère les fonctions associées aux *bearers* (débit maximum, priorité, etc.). GTP-C est destiné à envoyer des données de contrôle relatives aux *bearers* [22] (création, destruction, modification), il est notamment utilisé pour gérer la mobilité des usagers.

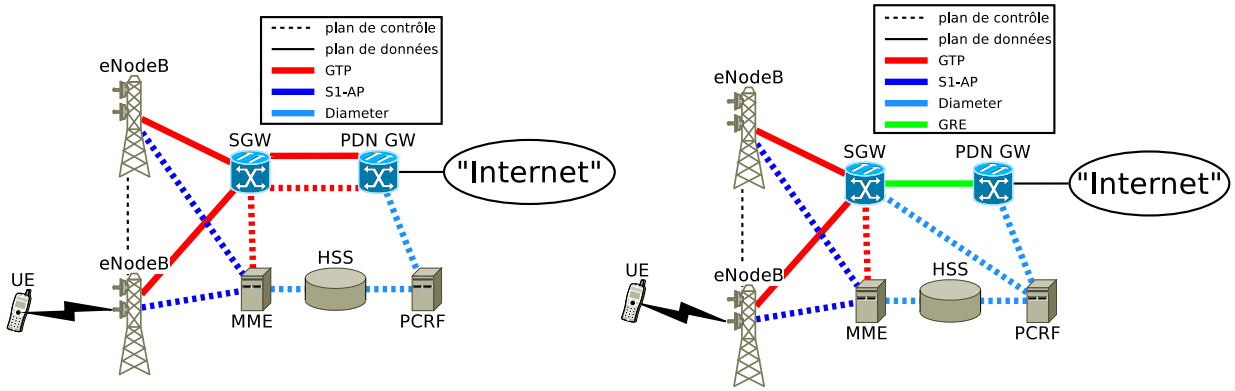
Les réseaux mobiles 4G ont aussi été prévus pour supporter plusieurs types de réseaux d'accès, tels que evolved UMTS Terrestrial Radio Access Network (EUTRAN) ou Worldwide Interoperability for Microwave Access (WiMAX). Il est politiquement difficile de faire accepter à un opérateur de connecter son réseau d'accès WiMAX à un réseau utilisant GTP étant donné que ce protocole appartient au consortium 3GPP (ce n'est pas un standard Internet Engineering Task Force (IETF)), qui comme mentionné plus haut est un consortium d'industriels. Il a donc été décidé de proposer une nouvelle architecture utilisant les protocoles Generic Routing Encapsulation (GRE) [64] et PMIPv6 [65] sur la partie partagée du réseau, cette architecture est décrite dans le document 23.402 [9]. GRE est un protocole d'encapsulation, il ne gère pas la signalisation et le trafic de contrôle comme GTP ; le protocole Diameter

[48, 74] est utilisé lorsqu'il s'agit de transférer ce genre d'informations.

Le protocole PMIPv6 permet de gérer la mobilité des usagers. L'architecture basique d'un réseau PMIPv6 se compose d'unités mobiles, de plusieurs MAG et d'un LMA. Les usagers se déplacent d'un MAG à l'autre sans qu'ils puissent le détecter, tout leur trafic transite via le LMA. Les paquets sont donc encapsulés entre le MAG courant et le LMA, respectivement le Serving Gateway (SGW) et le Packet Data Network Gateway (PGW) dans l'architecture 23.402 [9]. Il n'existe qu'un seul tunnel entre un MAG et un LMA pour tous les usagers connectés au MAG.

Étant donné que l'architecture 23.402 [9] n'utilise plus GTP entre le SGW et le PGW, 3GPP considère qu'il n'y a plus de *bearer* entre eux. Cependant, chaque *bearer* GTP a été remplacé par un tunnel GRE, nous considérons donc les tunnels GRE comme des *bearers* pour plus de simplicité. GRE possède un champ *Key* de quatre octets permettant de remplacer le champ Tunnel Endpoint Identifier (TEID) dont GTP se sert pour différentier les *bearers*.

Les deux architectures proposées dans les documents 23.401 [8] et 23.402 [9] sont schématisées à la figure 1.1.



(a) Architecture proposée dans 23.401 [8]

(b) Architecture proposée dans 23.402 [9]

Figure 1.1 – Architecture proposée par 3GPP

Hui *et al.* [70] proposent de modifier PMIPv6 pour permettre la création de plusieurs tunnels entre le MAG et le LMA. Ils utilisent, comme 3GPP dans l'architecture 23.402 [9], le protocole GRE pour différentier les tunnels. Le but de cette modification est de permettre la différentiation de trafics et donc de faciliter l'application de politiques (e.g. de QoS) différentes. Nous pouvons remarquer une similitude avec le modèle GTP, d'ailleurs Hui *et al.* [70] proposent d'utiliser les Traffic Flow Template (TFT) utilisés par 3GPP dans le protocole GTP (un TFT permet d'identifier une session, une connexion).

1.2 Éléments de la problématique

La première architecture développée par le consortium 3GPP est décrite dans le document 23.401 [8] cependant, du fait de la présence de GTP entre le SGW et le PGW, le consortium fut contraint de développer une nouvelle architecture développée dans le document 23.402 [9]. Ces changements ayant été décidés en majorité sur la base de critères politiques.

Du fait des capacités limitées des réseaux mobiles, la maitrise des ressources allouées à chaque usager et à chaque type de trafic est très importante. Les *smartphones* sont de plus en plus courants et sont maintenant capables de gérer un grand nombre d'applications et de types de trafics. La vidéo est aussi apparue sur les plateformes mobiles et suscite de plus en plus d'intérêt. Ces nouveaux facteurs accentuent le besoin d'utiliser les ressources du réseau de la manière la plus efficace possible de façon à maximiser les performances.

Dans ce contexte nous pouvons nous demander si les changements apportés à l'architecture 23.401 [8] dans 23.402 [9] ont affecté les performances du réseau du point de vue de l'usager.

Un des objectifs du passage à la quatrième génération de réseau mobile est l'utilisation du protocole IP pour tous les types de trafic et l'utilisation d'un réseau unique. Cependant, le protocole IP n'est que peu utilisé dans l'architecture 23.401 [8] qui utilise GTP. Dans l'architecture 23.402 [9], par rapport à 23.401 [8], entre le SGW et le PGW GTP a été remplacé par PMIPv6 et GRE. Le protocole PMIPv6 a toutefois été modifié de façon à ce que le MAG (en l'occurrence le SGW) n'ait aucun lien avec la couche IP du Mobile Node (MN) (ou UE). Le routeur par défaut du MN est le PGW qui est le LMA de l'architecture PMIPv6 ce qui est contraire à ce qui est décrit dans son Request for Comments (RFC) [65, sec. 3]. De plus le protocole GRE n'est pas géré par la couche IP ou PMIPv6 comme on pourrait le croire après la lecture de Muhanna *et al.* [78].

Nous pouvons nous demander s'il est possible d'utiliser les caractéristiques de IPv6 de façon plus avancée et si cela a un impact sur les performances ressenties par l'utilisateur.

Hui *et al.* [70] proposent une extension à PMIPv6 pour permettre l'utilisation de plusieurs tunnels entre un MAG et un LMA. Chaque tunnel encapsulant le trafic d'un seul service pour un terminal, de cette façon il est possible d'appliquer des techniques de QoS au tunnel et ainsi prioriser ce qu'il transporte. Cette évolution permettrait de transférer la gestion des *bearers* au niveau IP, au protocole PMIPv6.

GRE n'est utilisé dans l'architecture 23.402 [9] que pour remplacer le champ TEID permettant de différentier les *bearers* dans GTP. Dans l'en-tête du protocole IPv6 il existe un champ appelé « flow label » de 20 bits dont l'utilisation n'est pas encore définie. Un RFC [41] est d'ailleurs en cours de rédaction pour tenter d'en préciser les utilisations.

Nous pouvons nous demander s'il est possible d'utiliser ce champ en remplacement de

GRE. Et si cela est possible, quels impacts auraient cette modification sur les performances du réseau ressenties par l'usager.

1.3 Objectifs de recherche

L'objectif principal de ce mémoire est d'évaluer et comparer les performances des architectures 23.401 [8] et 23.402 [9] proposées par le consortium 3GPP. Nous allons aussi proposer et évaluer des changements à apporter à ces architectures pour essayer d'utiliser au mieux les caractéristiques du protocole IPv6 et limiter l'utilisation de GTP et GRE. L'évaluation des différentes architectures se fera sur la base des performances du réseau au niveau du terminal.

D'une manière plus spécifique nous visons les objectifs suivants :

- évaluer les performances de l'architecture 23.402 [9] par rapport à 23.401 [8], et ce du point de vue des terminaux, c'est-à-dire que seules les performances constatées sur les paquets circulant sur le plan de données nous intéressent ;
- proposer et évaluer un moyen de confier la gestion des *bearers* entre le SGW et le PGW au protocole PMIPv6 dans l'architecture 23.402 [9] ;
- proposer et évaluer un moyen de s'affranchir de GTP-U au profit de PMIPv6 ;
- créer des simulateurs capables d'évaluer des métriques de performance pour les architectures 23.401 [8], 23.402 [9] et pour les architectures modifiées que nous proposons.

1.4 Plan du mémoire

Suite à l'introduction, ce mémoire se poursuit avec le chapitre 2 dressant un historique non exhaustif des technologies utilisées dans les réseaux mobiles. Au chapitre 3, nous exposons nos propositions de modification des architectures EPC de 3GPP après quoi nous dévoilons et comparons les performances obtenues avec et sans modification. Le chapitre 4 décrit les simulateurs et outils utilisés pour obtenir les résultats. Enfin, la conclusion synthétise le travail de recherche effectué lors de cette maîtrise en faisant ressortir les principales contributions apportées ; nous y abordons aussi les possibles recherches futures susceptibles de découler de ce travail.

CHAPITRE 2

ANALYSE DES RÉSEAUX MOBILES

Avant de pouvoir exposer nos propositions nous pensons qu'il est préférable de décrire les différentes technologies employées par les réseaux mobiles. Nous allons donc tout d'abord les passer en revue ; nous nous attarderons sur les réseaux cellulaires en détaillant leur évolution de la première à la dernière génération (la 4G). Nous introduirons différentes techniques permettant de gérer la mobilité au niveau IP, en premier lieu sur IPv4 ce qui nous permettra ensuite d'expliquer plus facilement celles utilisant IPv6. Pour finir nous expliquerons le fonctionnement de l'architecture de QoS DiffServ.

2.1 Définitions et concepts de base

Les réseaux dits « mobiles » sont des réseaux dont les terminaux sont amenés à se déplacer. Il existe plusieurs types de ces réseaux, les plus connus étant les Wireless Local Area Network (WLAN) (e.g. les réseaux Wi-Fi), et les réseaux cellulaires. Les réseaux cellulaires sont généralement classifiés en plusieurs générations décrivant leur niveau de technicité et leur apparition chronologique.

Wi-Fi : Les réseaux Wi-Fi sont des réseaux WLAN respectant les spécifications de la norme 802.11legacy développée par l'organisme Institute of Electrical and Electronics Engineers (IEEE). Il existe plusieurs amendements à cette norme dont les plus connus sont 802.11a, b, g et n. Chacun de ces amendements définit un sous-type de Wi-Fi. L'acronyme Wi-Fi désigne aussi un ensemble de protocoles utilisés dans ce type de réseau.

Cellulaire : Les réseaux cellulaires courants sont appelés Public Land Mobile Network (PLMN), chacun de ces réseaux est mondialement identifié par un Mobile Country Code (MCC) et un Mobile Network Code (MNC). Ils sont généralement connectés au PSTN pour permettre à leurs usagers de communiquer avec les autres PLMN et d'appeler des numéros fixes. Parfois ces réseaux sont aussi connectés à Internet, notamment dans les dernières générations de réseaux cellulaires (à partir de la 2,5G).

Il existe une multitude d'algorithmes, de normes et de protocoles concernant les réseaux cellulaires. Nous allons décrire les plus connus par ordre d'appartenance aux différentes générations.

2.1.1 Advanced Mobile Phone System

La technologie Advanced Mobile Phone System (AMPS) fait partie de la famille des réseaux cellulaires 1G. À chaque communication est allouée une bande de fréquences (principe Frequency Division Multiple Access (FDMA)) ce qui permet de facturer le client en fonction de la durée d'appel. La voix est transmise de manière analogique sur le canal radio ce qui rend la communication très sensible au bruit, mais aussi susceptible d'être interceptée et espionnée.

2.1.2 Global System for Mobile communications

Les réseaux Global System for Mobile communications (GSM) appartiennent aux réseaux cellulaires 2G, l'acronyme GSM désigne aussi un codec de compression de signal audio utilisé justement dans ces réseaux. Cette norme utilise le principe de la commutation de circuit i.e. un chemin physique est réservé par chaque connexion/communication, c'est la méthode la plus ancienne utilisée pour les communications réseau.

L'architecture du PLMN GSM est schématisée à la figure 2.1, elle est reliée au PSTN et contient plusieurs types de nœuds décrits par Eberspächer *et al.* [61] et Pierre [86] :

Base Transceiver Station est une antenne réseau montée sur une station de base, elle est en liaison physique avec le Mobile Terminal (MT).

Base Station Controller est un contrôleur de Base Transceiver Station (BTS), il gère la bande passante, les ressources radio et contrôle la procédure de *relève* entre deux BTS.

Mobile Switching Center est chargé d'aiguiller les trafics de données et de contrôle aux différents nœuds et fonctions du réseau. Il gère la localisation des MT en coordination avec le Visitors Location Register (VLR), l'aiguillage des appels (commutation de circuit), la sécurité en coordination avec le Authentication Center (AuC) et la mobilité des MT (gestion des *relèves*).

Home Location Register est une base de données contenant toutes les informations relatives aux MT appartenant au PLMN de l'opérateur ; par exemple : la zone (correspondant à un VLR) courante où se trouve le MT. Elle est unique dans le PLMN.

Visitors Location Register est une base de données contenant les informations relatives au MT se trouvant dans la zone qu'il dessert avec son Mobile Switching Center (MSC). Elle contient entre autre la BTS sur laquelle se trouve le MT.

Authentication Center est une base de données contenant les informations de sécurité des MT du PLMN.

Equipment Identity Register est une base de données contenant une liste de tous les International Mobile Equipment Identity (IMEI) des MT enregistrés sur le PLMN.

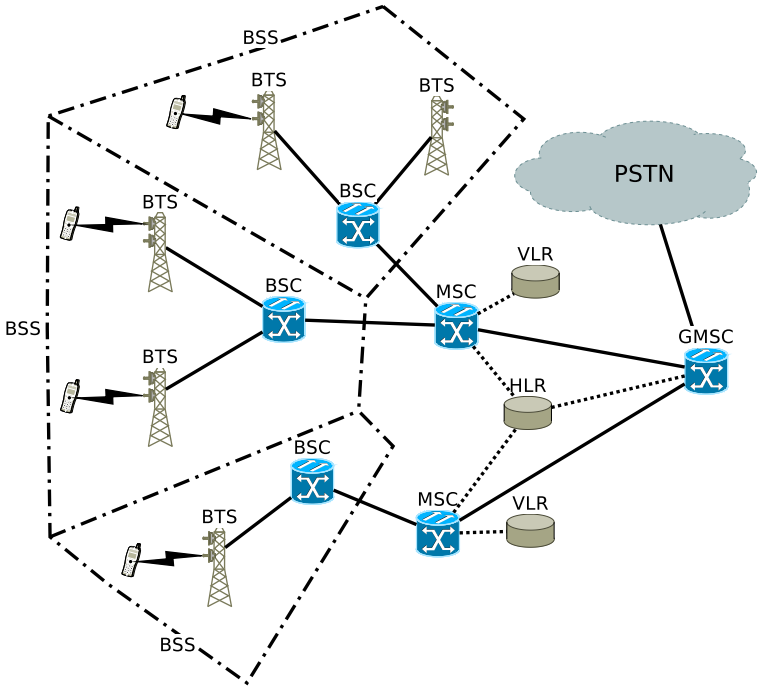


Figure 2.1 – Architecture simplifiée d'un PLMN GSM

Le réseau d'accès GSM utilise à la fois les techniques Time Division Multiple Access (TDMA) et FDMA, il sépare le trafic *downlink* du *uplink* par Frequency Division Duplexing (FDD) comme décrit par Eberspächer *et al.* [61]. La bande de fréquences utilisée initialement est celle des 900 MHz soit 890 à 915 MHz pour le trafic *uplink* et 935 à 960 pour le *dowlink*, cette bande fût ensuite étendue de 10 MHz soit 880 à 915 et 925 à 960. Il existe aujourd'hui une dizaine de types de GSM allant du GSM-380 au GSM-1900, le deuxième nombre correspondant à la bande de fréquences.

Pour partager un même canal fréquentiel entre plusieurs usagers la technique TDMA alloue des *time-slots* à chaque usager sur la fréquence voulue. Une trame TDMA contient 8 *time-slots* et les fréquences de transmission et de réception d'un terminal peuvent changer à chaque *time-slot*. La BTS doit aussi mesurer et informer le MT du temps de propagation des trames appelées Timing Advance (TA) ; cette valeur doit être comprise entre 0 et 63 symboles, ce qui correspond à une durée limitant le rayon des cellules GSM à 35 km.

Dû aux contraintes du terrain, certaines zones ne sont pas couvertes par toutes les fréquences. Pour les terminaux à faible vitesse, les « zones de coupure » peuvent dépasser les 20 ms, ce qui correspond au temps de transmission d'un bloc de données GPRS. Quels que soient le décalage et les bits aléatoires ajoutés, toutes les données seront perdues. Pour remédier à ce problème, le terminal change souvent de fréquence de transmission et de réception.

Un terminal GSM peut être dans deux états différents, le mode dédié est le mode du terminal durant les appels ou lorsqu'il y a échange d'information avec le réseau. Le terminal

entre dans le mode *idle* lorsqu'il n'a plus besoin d'aucune ressource dédiée (aucun échange avec le réseau), dans ce mode il continue cependant à recevoir des notifications des BTS et peut décider d'effectuer une *relève*.

Dans le mode *idle* le réseau sait dans quelle zone se trouve le terminal, mais ignore sur quel BTS il est attaché. En revanche, dans le mode dédié, étant donné qu'il y a échange de données, le réseau est obligé de connaître précisément la position du mobile.

On peut remarquer que malgré sa popularité GSM est une technologie vieillissante amenée à disparaitre dans les prochaines années. En décembre 2009 Karsten Nohl déchiffra et dévoila le code source utilisé par les téléphones GSM ; puis Chris Paget expliqua, lors de la Defcon de juillet 2010, une méthode de fabrication à faible coût (1500 $) d'un espion/intercepteur de communication GSM.

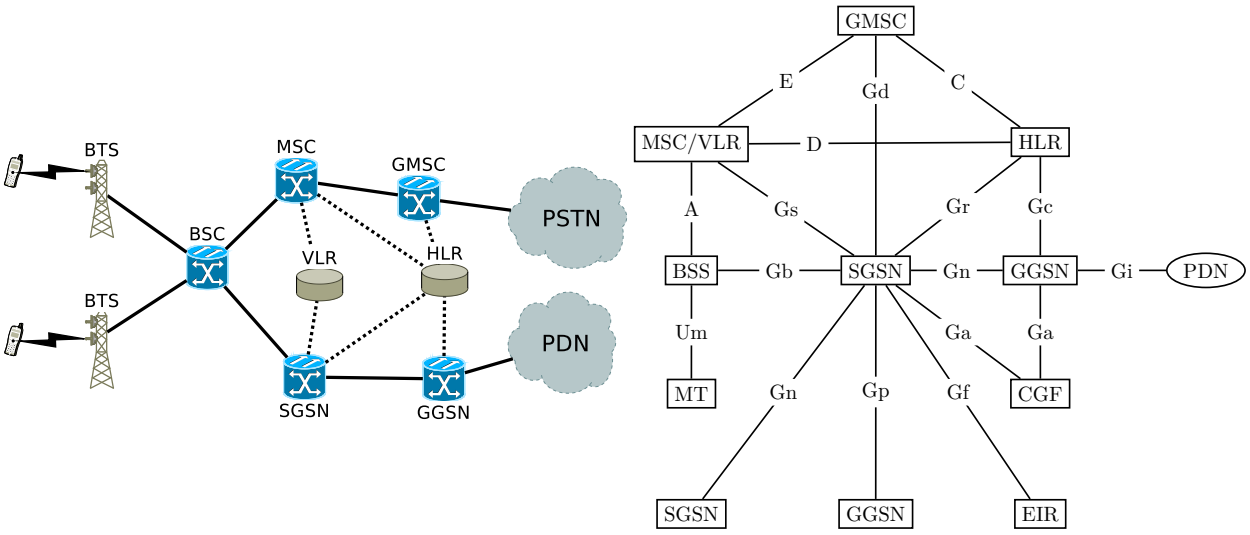
2.1.3 General Packet Radio Services

Le réseau GSM offre la possibilité d'initier ou de recevoir des appels téléphoniques comme avec un téléphone fixe ; il offre aussi la possibilité d'envoyer des Short Message Service (SMS) (originellement prévu pour envoyer des messages de service). Néanmoins, il est assez limité face à la demande croissante de technologies liées au réseau Internet (e.g. mail, web), c'est pourquoi l'European Telecommunications Standards Institute (ETSI) standardisa une extension au réseau appelée GPRS schématisée à la figure 2.2 (cette extension est maintenant standardisée par le consortium 3GPP). Elle est décrite et correspond à la version 97 des spécifications de GSM. Elle permet de supporter la transmission de données par paquet (commutation de paquets), notamment le protocole IP et ainsi la connexion du mobile à Internet.

Les couches de protocoles utilisées sont représentées à la figures 2.3 et 2.4. La couche Radio Link Control (RLC) possède une fonctionnalité Automatic Repeat reQuest (ARQ) de « transmission fiable ».

On peut remarquer que la couche Logical Link Control (LLC) permet d'offrir une sécurité de transmission importante aux couches supérieures, cependant RLC offre déjà des fonctionnalités ARQ et les applications nécessitant des canaux fiables utilisent le protocole de transport Transmission Control Protocol (TCP). Comme il est expliqué par Brand et Aghvami [47] LLC n'est pas nécessaire et a été supprimée des évolutions de GPRS notamment Universal Mobile Telecommunications System (UMTS).

GPRS est couramment considérée comme faisant partie des réseaux cellulaires 2,5G. Sa première version (spécification GSM version 97) permet d'atteindre des débits théoriques de 40 Kbps en *downlink* et 14 Kbps en *uplink* [35]. Les versions ultérieures (version 98 et 99) offrent quant à elles, un débit théorique maximal de 171 Kbps [35] (évalué en pratique à



(a) couplé avec un réseau GSM [4] (b) architecture logique

Figure 2.2 – Architecture des réseaux GPRS

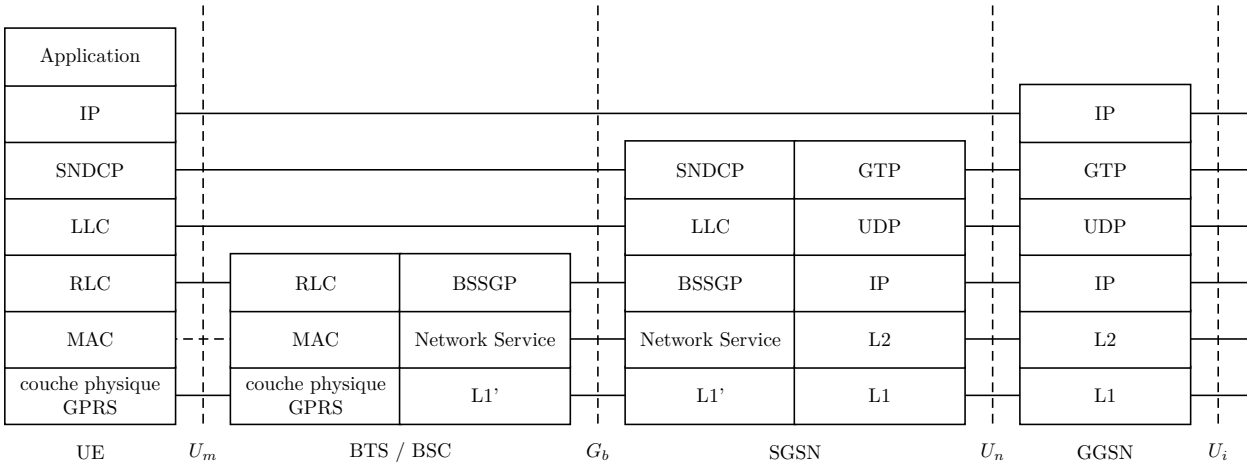


Figure 2.3 – Couches de protocoles du plan de données dans le réseau GPRS (inspirée de Brand et Aghvami [47])

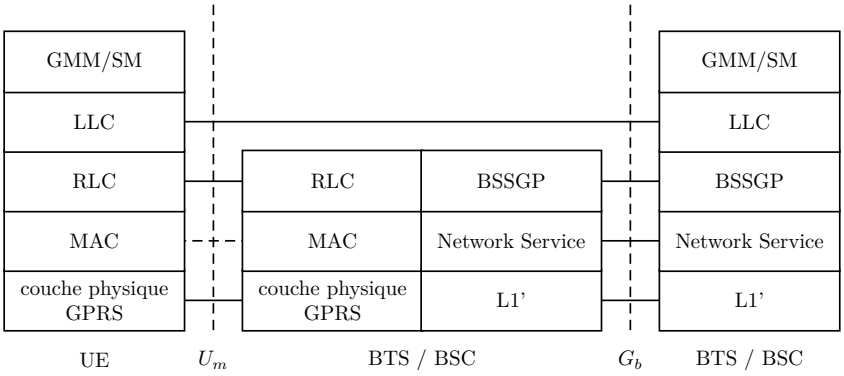


Figure 2.4 – Couches de protocoles du plan de contrôle dans le réseau GPRS (inspirée de Brand et Aghvami [47])

60 Kbps [47]).

Les modifications les plus notables sont (voir figure 2.2b) :

- l'ajout du Serving GPRS Support Node (SGSN) ; l'équivalent « commutation de paquets » du MSC. Il enregistre la position des terminaux et assure des fonctions de sécurité et d'authentification.
- l'ajout du Gateway GPRS Support Node (GGSN) ; l'équivalent « commutation de paquets » du Gateway MSC (GMSC). Il est l'interface entre le réseau mobile GPRS et d'autre Packet Data Network (PDN).
- l'évolution du Home Location Register (HLR) pour supporter les informations nécessaires au réseau GPRS ;
- l'évolution logicielle des Base Station Controller (BSC) [44] pour supporter l'interface Gb ;
- l'évolution matérielle et logicielle des BTS [44] pour supporter les nouveaux canaux physiques.

La technologie GPRS malgré son débit assez faible au regard des connexions Internet actuelles fut déployée dans de nombreux réseaux. En effet un de ses grands avantages est d'utiliser une partie de l'architecture GSM, dont le réseau d'accès, ce qui diminue grandement son coût de déploiement pour les opérateurs disposant déjà de ce réseau.

Pour la gestion de la mobilité, les états que peut prendre le terminal sont gérés indépendamment des états GSM [47]. Il en existe trois :

idle le réseau ne connait absolument pas la position du terminal.

standby le réseau connait la position du terminal en terme de *routing area*. C'est l'équivalent GPRS de l'état *idle* de GSM.

ready le réseau connait exactement la position du terminal.

Le réseau GPRS n'étant pas prévu pour fournir des services temps réel, il n'a pas été jugé intéressant d'implémenter un mécanisme de *soft-handover*. Le *handover* se fait donc par déconnexion/reconnexion et peut entraîner des pertes de paquets.

Les états cités plus haut ne concernent que la gestion de la mobilité, il en existe d'autres types comme par exemple Packet Data Protocol (PDP) *inactivate* et *activate*. Un contexte PDP définit une connexion au réseau, une session d'un usager. Il est représenté par une structure de données présente sur le Mobile Subscriber (MS), le SGSN et le GGSN. Il contient par exemple le International MS Identity (IMSI) du MS et le Network layer Service Access Point Identifier (NSAPI) qui tous deux forment un Tunnel Identifier (TID) (aussi appelé TEID), il peut aussi inclure l'adresse du MS, son type (e.g. IP, Point to Point Protocol (PPP)), et le profil de QoS utilisée [1]. Dans les premières versions de GPRS (version 97 et 98 des spécifications GSM) un seul contexte PDP peut être activé par adresse de MS [1, sec. 9.1][6, sec. 9.1.2.1] ; un MS peut cependant posséder plusieurs contextes PDP non activés. Certains MS peuvent supporter plusieurs connexions simultanées (ils possèdent plusieurs adresses) ce qui leur permet d'avoir plusieurs contextes PDP activés simultanément [89].

Du fait de l'apparition de nombreux nouveaux services relatifs à la commutation de paquets et de la rareté de la bande passante, il a été nécessaire d'introduire une notion de QoS au réseau. Les informations définissant la QoS à utiliser par le MS sont contenues dans les contextes PDP, cependant un seul de ces contextes peut être actif sur un MS à un instant donné. Une différentiation est donc effectuée entre les trafics des usagers, mais pas entre les types de trafics d'un même MS.

Dans les nouvelles versions de GPRS un TFT a été ajouté au contexte PDP ce qui permet de différentier plusieurs contexte PDP ayant la même adresse de MS. Il est alors possible de définir un contexte PDP par service et non plus par connexion au réseau, chaque contexte contenant la politique de QoS à appliquer au service en question. Plusieurs contextes PDP peuvent alors être activés simultanément sur un MS, ce qui permet de faire de la QoS par service et non plus par usager.

GPRS *Core Network*

Le réseau cœur du réseau GPRS est principalement composé du SGSN et du GGSN, ces deux nœuds sont des GPRS support node (GSN).

GGSN Comme décrit plus haut, il est l'équivalent du GMSC du réseau GSM, il peut aussi être vu comme l'équivalent du Home Agent (HA) de l'architecture Mobile IP (MIP). Il assure la mobilité des usagers dans le sens où il connait le SGSN sur lequel se trouve actuellement le MS et aiguille les paquets en conséquence. Il attribue et distribue les

adresses des MS, authentifie les usagers et comptabilise leur trafic. Il applique aussi les paramètres (e.g. de QoS) spécifiés par les contextes PDP actifs.

SGSN Il aiguille les paquets et gère une partie de la mobilité en enregistrant le MSC courant des usagers. Comme le GGSN, il authentifie les usagers et comptabilise leur trafic.

Ces deux nœuds encapsulent et décapsulent les paquets qui transitent entre eux en utilisant le protocole GTP. Ce protocole gère aussi la mobilité des MS. Il utilise lui-même le protocole de transport User Datagram Protocol (UDP). Il existe plusieurs types de GTP (pour plus de détails, voir la section 3.1.1) :

GTP-C transporte certaines données sur le plan de contrôle.

GTP-U encapsule les paquets sur le plan de données.

Des Boarder Gateway (BG) peuvent être installés dans les réseaux cœurs GPRS pour permettre l'interconnexion entre différents PLMN. Entre ces PLMN, le protocole GTP est utilisé [44] ; aussi des accords sont généralement passés entre les opérateurs concernant la traduction des paramètres des contextes PDP, notamment pour les paramètres des QoS.

Le réseau cœur de GPRS défini dans les spécifications 97 de GSM est aussi utilisé (avec quelques améliorations) avec les réseaux d'accès UMTS Terrestrial Radio Access Network (UTRAN) et Enhanced Data rates for GSM Evolution (EDGE).

2.1.4 Enhanced Data rates for GSM Evolution

Les réseaux GSM/GPRS ont eux-mêmes évolué grâce à la technologie EDGE permettant d'augmenter considérablement les débits disponibles, cette évolution est couramment considérée comme appartenant aux réseaux mobiles 2,75G [47, 35]. EDGE aussi appelé Enhanced GPRS (EGPRS), est décrit dans la version 99 des spécifications GSM (reprisent depuis par le consortium 3GPP). C'est une alternative aux couteux réseaux 3G basés sur UMTS qui nécessitent une mise à niveau de toute l'architecture du réseau.

Les technologies GPRS et EDGE peuvent partager les mêmes spectres de fréquence, les opérateurs n'ont donc pas à se procurer de nouvelles licences pour migrer vers ce système comme c'est le cas avec la technologie UMTS (dont le prix des licences à atteint jusqu'à cinq milliards d'euros en France).

EDGE introduit l'utilisation d'un nouveau type de modulation en plus de Gaussian Minimum Shift Keying (GMSK) utilisé dans GPRS, le 8 Phase-Shift Keying (PSK). Grâce à ce type de modulation, un terminal EDGE est capable d'atteindre un débit théorique de 384 Kbps (171 Kbps pour GPRS) [47, 35], ce débit restant inférieur à ce qu'il est possible d'atteindre en UMTS. Dans la nouvelle version EDGE *Evolution*, ce débit est passé à 1,3 Mbps en *downlink* et 653 Kbps en *uplink* ; le délai est aussi passé sous la barre des 100 ms [35].

Cependant, du fait du support de cette nouvelle modulation les téléphones GPRS ne sont pas compatibles avec EDGE et les BTS doivent subir une mise à jour, l'architecture du réseau restant la même.

Un autre apport de EDGE est la création de nouveaux Link Quality Control (LQC) combinant l'adaptation des liens utilisés dans GPRS avec une technique d'Incremental Redundancy (IR), inclut dans la méthode de retransmission Hybrid ARQ (HARQ) de type deux. Le problème du choix du schéma de codage est important, en effet si on choisit un schéma trop sécuritaire, de la bande passante sera gaspillée du fait de la forte redondance. Si on choisit un schéma de codage non sécuritaire on maximise la bande passante, mais si une trame est perdue alors le délai et la bande passante se détériorent.

L'idée de la technique IR de type 2 est de conserver les trames erronées et de ne retransmettre que les parties de cette trame utiles à la reconstruction de la séquence correcte. Ainsi beaucoup de bande passante est sauvegardée.

Si la bande passante était la seule valeur à maximiser, la technique à adopter consisterait à toujours commencer la transmission d'une trame en utilisant le schéma de codage le moins sécuritaire. Si une erreur se produit, le système retransmet les informations erronées ainsi que les données suivantes, mais en utilisant un schéma de codage plus sécuritaire. Le système recommence ces étapes jusqu'à ce que les données soient correctement transmises. Cependant, cette méthode engendre un délai très important étant donné que la transmission commence toujours avec le schéma de codage le moins résistant, les retransmissions peuvent donc être très nombreuses avant qu'une trame soit correctement transmise.

La méthode utilisée par EDGE utilise le principe énoncé au paragraphe précédent, mais n'utilise pas le schéma de codage le plus sensible aux pertes pour commencer. En se basant sur le nombre de retransmission des trames précédentes, soit la qualité actuelle du lien, le réseau choisit le niveau de redondance (le schéma de codage) utilisé pour la première transmission [47].

2.1.5 Universal Mobile Telecommunications System

Le réseau UMTS (représenté figure 2.5) est une évolution du réseau GPRS, il appartient au réseau cellulaire 3G, c'est un standard développé par le consortium 3GPP dans les documents 23.101 [5] et [37]. Il suit les spécifications International Mobile Telecommunications 2000 (IMT-2000) de l'International Telecommunication Union (ITU) dans le sens où il utilise le standard radio Wideband Code Division Multiple Access (W-CDMA). Il existe cependant de nombreux types d'UMTS et tous n'utilisent pas W-CDMA.

Dans ce nouveau réseau, les anciens BSC sont remplacés par des Radio Network Controller (RNC) et les BTS par des NodeB. Une des évolutions notables apportée par les NodeB est

le support de plusieurs cellules radio par une seule entité physique.

UTRAN est le réseau d'accès utilisé, il en existe deux types : UMTS Terrestrial Radio Access - Frequency Division Duplexing (UTRA-FDD) et UMTS Terrestrial Radio Access - Time Division Duplexing (UTRA-TDD). UTRA-FDD utilise le standard W-CDMA se servant de la technique Direct Sequence CDMA (DS-CDMA). Ce type de réseau d'accès fait passer les informations ascendantes sur une fréquence différente de celles descendantes (principe du FDD) et permet l'utilisation d'une même fréquence par plusieurs usagers grâce à l'utilisation de codes de codages différents (principe du Code Division Multiple Access (CDMA)). UTRA-TDD utilise le standard Time Division CDMA (TD-CDMA), qui utilise le principe Time Division Duplex (TDD) pour séparer le trafic ascendant du trafic descendant. Le fait de permettre l'utilisation des techniques FDD et TDD induit un contrôle plus fin des bandes de fréquences à allouer. Par exemple, si l'opérateur ne dispose que de très courtes bandes de fréquences, il préférera surement utiliser la technique TDD qui évite de trop les diviser.

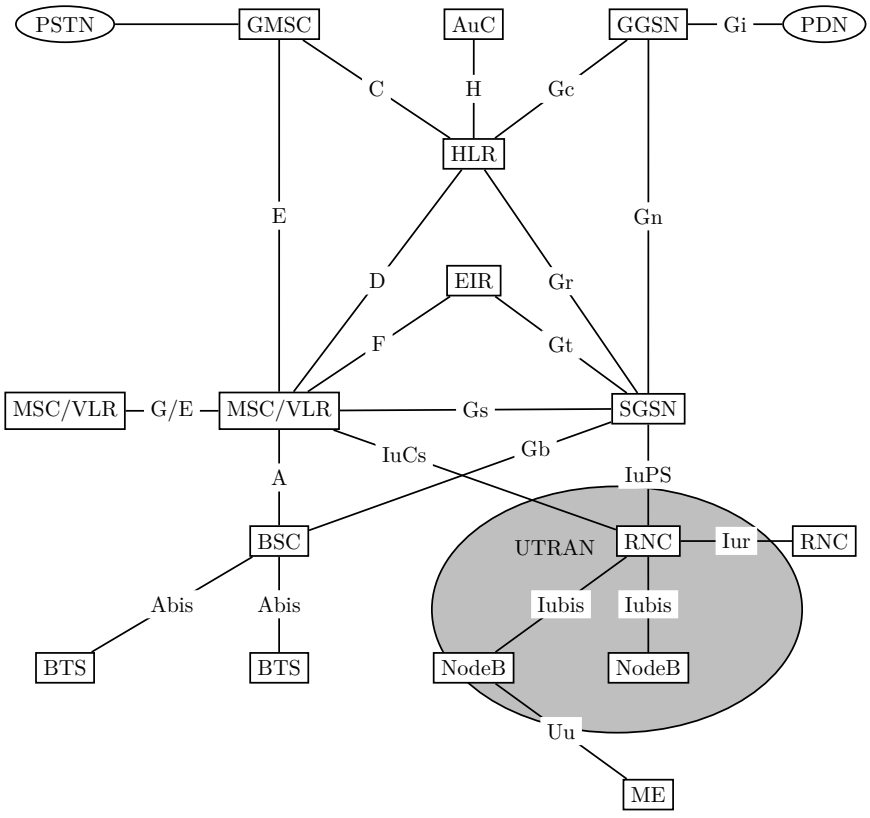


Figure 2.5 – Architecture logique du réseau UMTS (inspirée de 3GPP [37])

Le standard W-CDMA a évolué et une extension appelée High Speed Packet Access (HSPA) est apparue, elle vise surtout à augmenter la capacité du réseau d'accès. Elle se compose de deux modules High-Speed Downlink Packet-Data Access (HSDPA) et High-Speed

Uplink Packet-Data Access (HSUPA) (ou Enhanced Uplink (EUL)) améliorant respectivement les performances en *downlink* et *uplink*. Avec W-CDMA simple, le débit *downlink* peut atteindre environ 400 Kbps, tandis qu'avec HSDPA celui-ci peut atteindre 7,2 Mbps (voir 14 Mbps avec un terminal adapté). HSPA+ est une autre évolution introduisant notamment la technologie Multiple Input Multiple Output (MIMO) et pouvant atteindre 42 Mbps en *downlink* et 11 Mbps en *uplink*. On peut remarquer que HSDPA peut être installé sans que HSUPA soit présent, on peut donc avoir un système « semi-HSPA ».

Le module HSDPA introduit un nouveau canal logique nommé High Speed Downlink Shared channel (HS-DSCH) ayant la particularité d'être partagé entre tous les terminaux d'une même cellule. Ainsi la bande passante est mieux répartie ; celle non utilisée par un terminal n'est pas perdue puisqu'elle peut être utilisée par les autres. Il est préférable d'avoir un canal avec beaucoup de débit partagé entre plusieurs usagers plutôt que plusieurs petits canaux attribués un à un à chaque terminal. Ce partage de canal n'est pas effectué de façon triviale comme avec une file First In First Out (FIFO) par exemple, mais utilise une méthode de *scheduling* trame par trame basée sur la qualité de réception des UE. Pour chaque nouvelle trame, chaque UE envoie un rapport sur la qualité de sa réception (appelé Channel-Quality Indicator (CQI)), le NodeB décide alors lequel d'entre eux recevra cette trame et quelle quantité de données y stocker [54]. Le délai est aussi amélioré grâce à l'utilisation d'une méthode HARQ basée sur les NodeB et non le RNC comme c'est le cas pour EDGE.

Le module HSUPA apporte approximativement les mêmes modifications en *uplink* : *scheduling*, HARQ, etc. Partager le canal *downlink* est aisé étant donné que tous les paquets passent par le NodeB avant d'être envoyés par ondes radio. Le NodeB peut facilement choisir sur quel UE envoyer des données. En revanche, cela est plus délicat en *uplink* étant donné que plusieurs UE peuvent envoyer des informations en même temps sans passer par un point central. La solution utilisée dans HSUPA est de placer le *scheduler* dans le NodeB, cela implique que les UE envoient régulièrement un rapport sur l'état de leurs *buffers* et notifient le NodeB lorsqu'ils veulent envoyer de nouvelles données. Contrairement à HSDPA, où un seul UE à la fois est adressé par le NodeB, plusieurs UE peuvent envoyer des données en même temps, ce qui peut provoquer des interférences. Le *scheduling* effectué par le NodeB permet de s'assurer que les signaux sont toujours décodables, que les interférences ne sont pas trop fortes. Bien que les nouveaux canaux *uplink* se comportent comme des canaux partagés du fait du *scheduling*, ils sont en fait dédiés ; c'est le taux d'interférence au NodeB qui est partagé par les UE [54].

Deux nouveaux types de relève sont introduits [47] :

softer handover désigne le passage d'une cellule à une autre géré par le même NodeB. Le UE est alors connecté par plusieurs canaux parallèles, chacun alloué à une cellule

spécifique. Ce *handover* n'a pas d'impact sur l'architecture UTRAN étant donné qu'il n'y a pas de changement de NodeB.

soft handover désigne le passage d'une cellule à une autre géré par des NodeB différents. Le UE est alors connecté à deux NodeB en même temps ce qui implique la présence du lien Iur (cf. figure 2.5) pour les synchroniser (dans le cas où elles n'appartiennent pas au même RNC).

Les terminaux ne sont plus appelés MT mais UE, ils contiennent désormais un module d'identification Universal subscriber identity module (USIM) évolution du Subscriber Identity Module (SIM) utilisé dans GSM.

Pour augmenter les performances du réseau radio, les couches LLC et Sub Network Dependent Convergence Protocol (SNDCP) sont remplacées par une couche Packet Data Convergence Protocol (PDCP), chargée entre autre de compresser les en-têtes IP.

Comme les réseaux GPRS et EDGE, UMTS dispose de méthodes de QoS (figure 2.6). Cependant, les précédents réseaux n'étaient pas prévus pour supporter des trafics temps réel, ils ne possédaient que des méthodes de gestion de QoS assez basiques [47]. Dans UMTS, la QoS est gérée indépendamment pour chaque service grâce à un système de *bearer*. Un *bearer* est un tunnel par lequel passent les données du ou des services pour lequel il a été construit, chaque *bearer* possédant sa propre QoS.

UMTS dispose de plusieurs couches relatives au *bearer* comme le montre la figure 2.7. Pour l'utilisateur, le service de QoS apparait comme étant de bout en bout. Un des buts du système à sa création était d'être intelligible et simple à configurer, de ce fait seul quatre classes de QoS ont été définies, le tableau 2.1 donne leurs caractéristiques. Chaque classe possède des attributs comme le débit maximum, le débit garanti, la priorité d'allocation et de rétention (voir 23.107 [6] pour plus de détails).

Tableau 2.1 – Caractéristiques des classes de QoS utilisées en UMTS

Classes	conversation	streaming	interactive	best effort
Caractéristiques	peu de gigue peu de délai	peu de gigue	peu de délai peu de d'erreur	peu de perte
exemples d'application	voix	streaming vidéo	web	mail, download

Les trois principaux protocoles utilisés dans le réseau UTRAN sont : Radio Resource Control (RRC), RLC et Medium Access Control (MAC) (voir figure 2.6). Le protocole RRC gère le plan de contrôle du réseau d'accès et notamment les procédures de connexion, les mesures de qualité du lien radio, les *bearers*, les procédures de sécurité et les décisions de

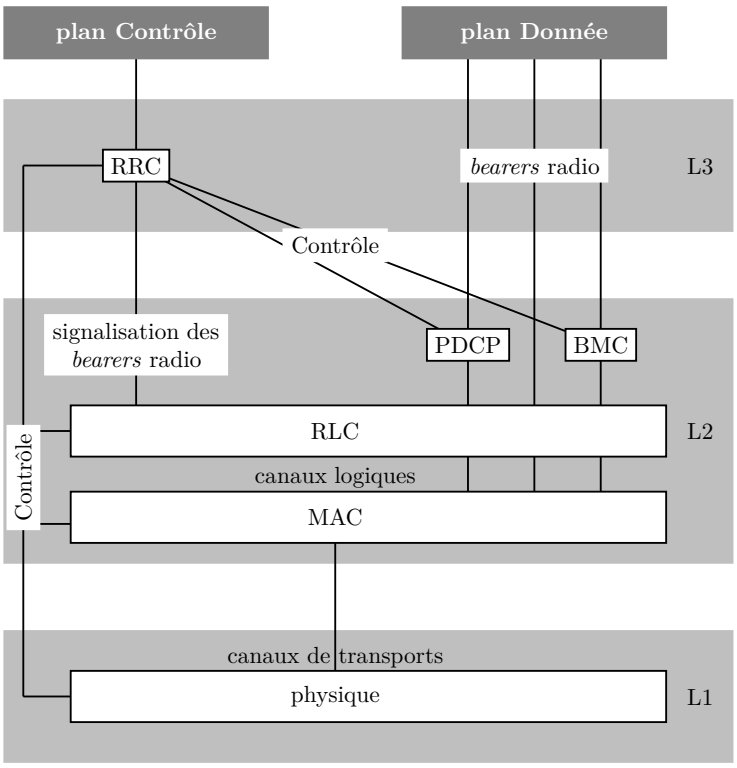


Figure 2.6 – Protocoles radio UTRAN (inspirée de Brand et Aghvami [47])

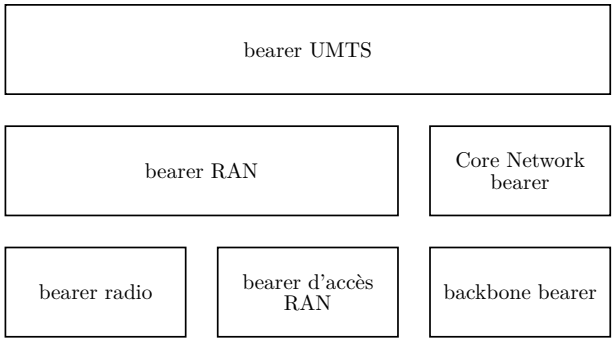


Figure 2.7 – Architecture de *bearer* utilisée dans UMTS (inspirée de 23.107 [6])

handover. Le protocole MAC gère l'envoi des données sur la couche physique en tenant compte des paramètres que lui fourni le protocole RRC, il gère notamment le *scheduling* des données.

Le protocole RLC offre trois types de transfert aux couches supérieures [47] :

transparent Les données sont envoyées sans ajout d'en-tête relative au protocole RLC. Aucune modification n'est effectuée sur les données, il est cependant possible d'utiliser la fonctionnalité de segmentation/ré-assemblage. Les données ne sont pas garanties d'arriver à destination, il est aussi possible que des erreurs soient présentes dans les paquets remontés aux couches supérieures.

unacknoledged Les données ne sont pas garanties d'arriver à destination, mais celles qui arrivent sont garanties d'être exemptes d'erreur.

acknoledge Les données sont garanties d'arriver à destination, sans erreur.

2.1.6 Worldwide Interoperability for Microwave Access

WiMAX est un protocole de communication sans fil donnant son nom au réseau d'accès l'utilisant, il fait partie des standards de l'IMT-2000. Comme le Wi-Fi il a été développé par l'organisme IEEE. Il désigne tous les réseaux respectant la norme 802.16. À l'heure actuelle, par ordre chronologique, nous connaissons 802.16a, 802.16d, e et m. La première version réellement utilisée est la 802.16d, mais cette version ne supporte pas la mobilité. La version suivante possède cette fonctionnalité, c'est pourquoi nous nous y intéressons dans notre étude. Pour supporter la mobilité entre plusieurs Access Service Network (ASN) le réseau se base sur la technologie IP et utilise Mobile IPv4 (MIPv4) (en mode *proxy* ou simple) ou MIPv6 [87, 103].

Avant que la technologie Long Term Evolution (LTE) n'arrive à maturité WiMAX était vu comme le futur des réseaux mobiles. Désormais LTE a pris l'avantage dans le domaine des technologies 4G. En effet, bien qu'à l'heure actuelle les deux technologies aient des performances comparables, LTE a l'avantage d'être un descendant direct des réseaux 3G courants. Certaines parties de l'architecture des anciens réseaux peuvent donc être réutilisées ou mises à jours. WiMAX quant à lui possède une architecture (représentée figure 2.8) et utilise des méthodes différentes de ce qui se fait dans les réseaux 3G. Actuellement la technologie WiMAX (802.16e) permet d'obtenir au maximum 144 Mbps en *downlink* et 35 Mbps en *uplink*. La prochaine version (802.16m) prévoit de supporter jusqu'à 1 Gbps en *downlink*.

Le passage à la version 802.16e a apporté de nombreuses améliorations à la norme, comme :

- la gestion des *soft* et *hard handover* (nécessaire au support de la mobilité) ;
- l'utilisation de méthode HARQ ;
- le support de technologie MIMO ;

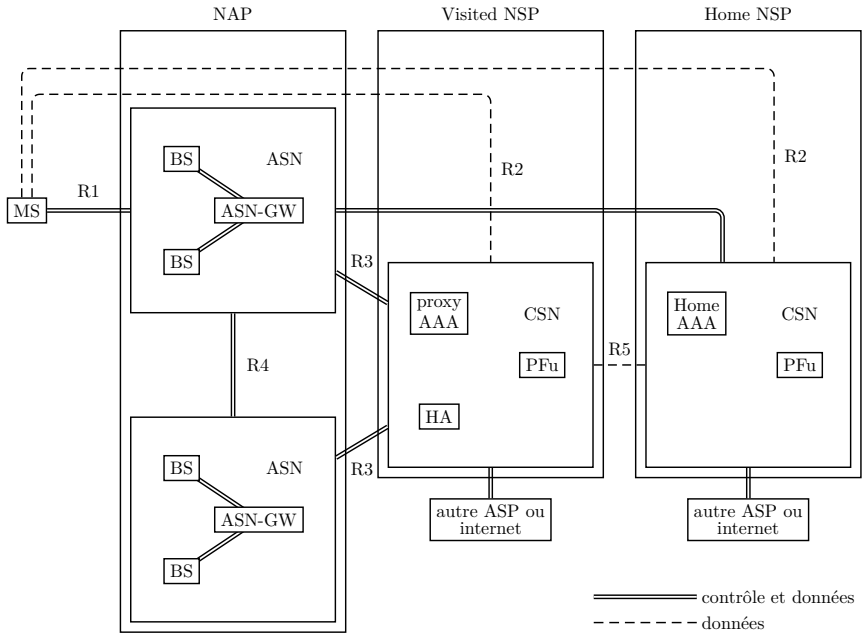


Figure 2.8 – Architecture logique du réseau WiMAX (inspirée de WiMAX Forum® [102])

- un meilleur système de QoS ;
- l'utilisation de la méthode de protection contre les erreurs de transmission Low-Density Parity-Check (LDPC) ;
- l'utilisation de Scalable OFDMA (SOFDMA) (ce qui rend cette version incompatible avec la 802.16d).

On remarque certaines similitudes avec HSPA comme l'utilisation de méthode HARQ ou de MIMO, ils utilisent aussi tous deux les modulations Quadrature PSK (QPSK) 16-Quadrature Amplitude Modulation (QAM) et 64-QAM. En revanche, leurs techniques de transmission radio sont différentes ; HSPA utilise une méthode basée sur CDMA tandis que WiMAX utilise SOFDMA (dérivé de Orthogonal Frequency-Division Multiplexing (OFDM)). OFDM est une technique qui consiste à allouer à chaque utilisateur un grand nombre de sous canaux de fréquence espacés de façon à être orthogonaux. Grâce à cette méthode l'utilisation spectrale du canal est bien meilleure qu'avec une Frequency Division Multiplexing (FDM) classique. Orthogonal Frequency-Division Multiple Access (OFDMA) utilise aussi la division temporelle pour séparer les données des terminaux, comme dans HSPA la modulation des canaux est adaptée dynamiquement en fonction de la qualité du lien observé.

2.1.7 Long Term Evolution

La technologie LTE, inclus dans le réseau EPS représenté à la figure 2.9 est une évolution du réseau d'accès 3G UMTS. Elle a été développée et standardisée par le consortium 3GPP,

sa première version est parue en décembre 2008 [36]. Bien que certains industriels aient déclaré que LTE se trouve dans la catégorie des réseaux 4G, il ne satisfait pas entièrement aux spécifications de l'ETSI ; il est donc plus juste de parler de 3,9G ou presque 4G.

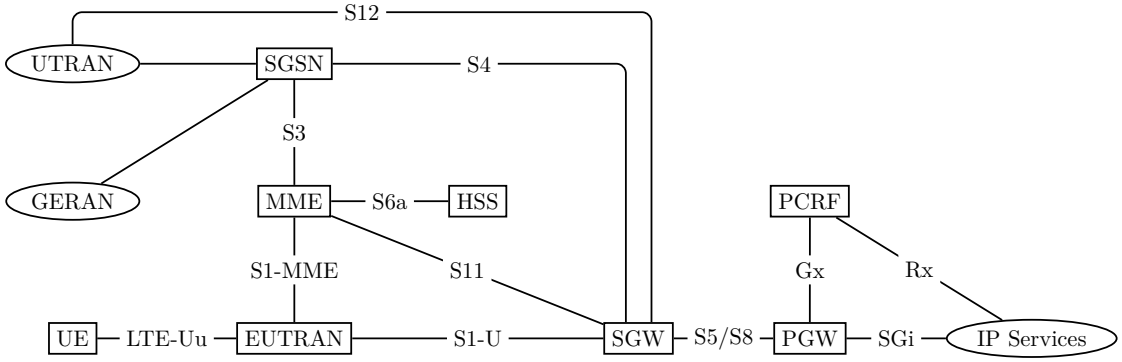


Figure 2.9 – Architecture simplifiée du réseau EPS (inspirée de 23.401 [8])

Le terme LTE désigne la technologie radio utilisée dans le réseau d'accès EUTRAN, celui-ci est prévu pour être attaché au réseau cœur EPC développé par le groupe de travail System Architecture Evolution (SAE). Le réseau au complet (EUTRAN plus EPC) est appelé EPS.

Une nouvelle version de LTE appelée LTE *advance* correspondant à la version 10 des spécifications est en cours de développement [1]. Cette version appartiendra réellement à la catégorie 4G et sera totalement compatible avec la technologie LTE actuelle.

Les performances couramment attribuées au réseau LTE sont de 100 Mbps minimum en *downlink* et 50 Mbps minimum en *uplink* (avec un terminal de catégorie 3) avec un délai d'aller-retour de moins de 10 ms (minimum de 1 ms) [36, 77]. Ces paramètres correspondent au but que s'était fixé le projet à son lancement, soit multiplier par trois ou quatre le débit en *downlink* et par trois le débit *uplink* comparé à HSPA. La taille des cellules a elle aussi augmenté, pouvant aller jusqu'à 100 Km en zone rurale [77] ; bien qu'il soit possible de maximiser le débit en utilisant de hautes fréquences (jusqu'à 2,4 GHz). Chaque cellule pouvant supporter au minium jusqu'à 200 UE.

Le projet LTE, en plus de s'être focalisé sur le passage à la 4G, a cherché à simplifier l'architecture du réseau mobile, aussi bien au niveau du réseau d'accès que du réseau cœur. En effet, le réseau d'accès nommé EUTRAN ne contient plus qu'un seul nœud, le Evolved Node B (eNodeB) et trois interfaces : S1-MME, S1-U et X2 qui est optionnel. S1-U et S1-MME peuvent être combinées en une seule interface S1 reliant le réseau d'accès EUTRAN au réseau cœur EPC. L'interface X2 relie deux eNodeB entre eux et est optionnelle. Il est donc possible de ramener le réseau d'accès à un seul nœud relié au EPC par une seule interface [2].

1. Certains documents sont déjà disponibles, mais ils sont encore susceptibles de changer.
2. Un eNodeB peut être connecté à plusieurs Mobility Management Entity (MME).

Le EPC contient beaucoup de fonctionnalités pouvant être regroupées en plusieurs nœuds physiques. L'architecture complète d'un réseau EPS est représentée à la figure 2.10. Le Service Aware Support Node (SASN) n'est pas un nœud décrit par 3GPP, c'est un produit développé par Ericsson pour prendre en charge une partie des fonctions Policy and Charging Enforcement Function (PCEF) et Assured Forwarding (AF) ; nous l'avons représenté car il est présent dans nos simulateurs (ayant été développé en partenariat avec Ericsson). Le EPC est compatible avec de nombreuses technologies d'accès en plus de EUTRAN. Les anciennes installations des réseaux GSM, EDGE, et UMTS peuvent être rattachées au réseau EPC et la mobilité des usagers entre ces technologies est supportée. Les réseaux WiMAX peuvent aussi s'attacher au EPC. Cependant, WiMAX est un standard de l'IEEE et il est politiquement difficile de faire transiter le trafic issu de ce réseau dans un tunnel utilisant le protocole GTP qui n'est pas un standard de l'IETF et qui est développé par le consortium 3GPP. C'est pourquoi le document 23.402 [9] prévoit de remplacer le protocole GTP par PMIPv6 et GRE entre les nœuds SGW et PGW.

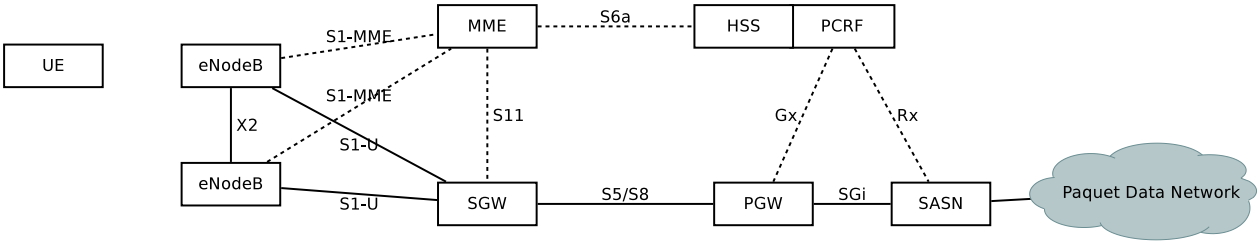


Figure 2.10 – Schéma général du réseau EPS

La partie basique du EPC est composée de cinq nœuds (quatre selon certaines sources) :

PGW Il connecte le réseau EPC avec le réseau extérieur appelé PDN. Il gère la mobilité des UE dans le sens où il agit comme le LMA d'un réseau PMIPv6, il authentifie et autorise les UE à se connecter en leur associant une adresse. Il se charge aussi de leur créer des *bearers* conformes avec leurs contrats et la politique du réseau (dont le Policy and Charging Rules Function (PCRF) a la charge). Étant le point de liaison entre le PLMN et le PDN le PGW fait aussi la correspondance entre les QoS utilisés dans ces réseaux. Il applique les politiques de QoS des *bearers* comme les Maximum BitRate (MBR), les APN-AMBR, les Guaranteed BitRate (GBR), etc. Dans la norme 23.402 [9] il n'existe plus de réel *bearer* entre le SGW et le PGW dans le sens où GTP a été remplacé par PMIPv6 et GRE. Le PGW ne gère donc plus la création de *bearers*, cependant il possède toujours les TFT nécessaires pour filtrer et router les trafics dans les *bearers*. En effet il est toujours chargé d'appliquer les politiques de QoS des *bearers*, chaque tunnel GRE correspondant à un *bearer* (voir figure 3.8). Le PGW est aussi

chargé de comptabiliser le trafic des UE[3] en vue de leurs facturations [73].

SGW Il est en bordure du réseau d'accès, il route les paquets *downlink* vers le eNodeB courant du UE. Il gère la mobilité des UE dans le sens où il peut être vu comme le MAG du système PMIPv6 [73]. Dans la norme 23.401 [8] il gère la correspondance entre les *bearers* GTP SGW-PGW et eNodeB-SGW. Dans la norme 23.402 [9] on considère qu'il contient les TFT des *bearers*, car ceux-ci n'existent plus sur l'interface S5/S8. Cependant, étant donné qu'il a un tunnel GRE par *bearer*, ses TFT ne sont en fait que des correspondances d'étiquettes. Le SGW gère aussi l'interfaçage du EPC avec les réseaux d'accès ; en effet la technologie EUTRAN n'est pas la seule à pouvoir se connecter au EPC. Il peut être lié à plusieurs PGW pour permettre aux UE d'être connectés simultanément à différents PDN. Dans le cas où le UE est en mode *idle* le SGW peut être amené à mettre en cache les paquets *downlink* lui étant destinés, le temps pour le MME de « réveiller » le UE et de construire les *bearers* radio.

MME Il incorpore certaines parties du SGSN n'ayant pas été placées dans le SGW. Il y a un MME par SGW. Il gère toutes les fonctions du plan de contrôle (il n'est d'ailleurs pas sur le plan de données) relatives aux UE (via la couche Non-Access Stratum (NAS) dont il est le nœud terminal) et aux *bearers* [73]. Il authentifie les UE sur le réseau et le réseau auprès d'eux (procédure Authentication and Key Agreement (AKA)), en plus de gérer le cryptage des communications. Il route les demandes de création de *bearer* provenant du PGW ou du UE et gère la négociation de leur QoS associée. Lorsque le UE est en état *idle* et qu'une requête lui est destinée, c'est au MME d'indiquer au SGW où se trouve le UE ; il gère donc la mobilité du UE [73]. Il est aussi responsable du choix du SGW à la connexion des UE ou lors de *handover* entre deux eNodeB nécessitant un changement de réseaux cœur (changement de SGW ou PGW). Le MME est lié au Home Subscriber Server (HSS) pour pouvoir authentifier les UE et enregistrer leurs états et leurs localisations, il est aussi chargé d'allouer aux UE un identifiant temporaire lors de leur connexion. Il gère au niveau du plan de contrôle la mobilité entre les différents réseaux d'accès possibles.

HSS C'est une base de données stockant les informations sur les sessions des UE, comme la QoS à laquelle ils ont souscrit. Le HSS fait aussi partie des nœuds gérant la mobilité du UE dans le sens où il enregistre leur localisation pour le MME ou d'autres nœuds de technologie différente comme le SGSN [81].

PCRF Il gère la QoS et les politiques de facturation sur le EPS. Il se trouve sur le plan de contrôle et ne fait donc que notifier les autres nœuds des décisions qu'il prend. Selon

3. Un UE peut être connecté à plusieurs PGW à la fois.

Olsson *et al.* [81] il ne fait pas partie du EPC, contrairement à Lescuyer et Lucidarme [73] qui le mentionnent dans leur liste.

Dans les réseaux UMTS et ses prédécesseurs, une grande partie de la gestion du réseau d'accès (adaptation des canaux, gestion de la QoS, etc.) s'effectue dans le RNC (3G) ou le BSC (2G). Comme nous l'avons dit, le réseau EUTRAN ne comporte qu'un seul nœud, le eNodeB ce qui le rapproche plus des réseaux WLAN comme Wi-Fi et WiMAX [73]. En plus de simplifier l'architecture, cela permet d'améliorer les capacités du réseau, notamment le délai. Les retransmissions de paquets perdus et les paquets de contrôle n'ont plus à se rendre jusqu'au RNC ou BSC ; ils sont directement traités par le eNodeB [73]. Comme le NodeB de UTRAN, le eNodeB de EUTRAN est capable de gérer plusieurs cellules logiques. Il gère entièrement le lien radio [73] :

- modulation, démodulation : il utilise, comme HSDPA et WiMAX, les modulations QPSK, 16QAM et 64QAM ;
- codage, décodage ;
- contrôle des ressources : création, modification ou destruction de *bearer* (dirigé par le MME), allocation de nouveaux canaux, etc.
- gestion de la mobilité radio : décision de lancer une procédure de *handover* et choix de son type : utilisant l'interface X2, ou S1 ;
- compression et chiffrement des en-têtes IP sur l'interface radio ;
- gestion de la détection et correction d'erreur de transfert au niveau de la couche 2 du modèle Open Systems Interconnection (OSI) ;
- sélection du MME du UE au moment de sa connexion.

L'interface X2 entre deux eNodeB est optionnelle et ne sert que lors des *handovers*, elle permet de diminuer le taux de perte de paquets et le délai. Lors d'un *handover* utilisant l'interface S1 les paquets de l'ancien eNodeB doivent être re-routés vers le nouveau eNodeB via le SGW. Ce passage par l'interface S1 se trouvant dans le chemin de nombreuses autres communications oblige le SGW a créer des tunnels temporaires de redirection [8]. En plus de prendre plus de ressources sur le réseau, cela empêche le SGW de libérer rapidement les ressources allouées entre lui et l'ancien eNodeB. Grâce à l'interface X2, le trafic arrivant à l'ancien eNodeB est re-routé directement vers le nouveau eNodeB. Il n'y a pas de réservation de ressource supplémentaire entre l'ancien eNodeB et le SGW. L'interface X2 servant exclusivement au *handover* les risques de congestion sont très faibles et donc les chances de perdre un paquet sont moins importantes qu'en utilisant S1.

Une grande évolution apportée par le réseau LTE est l'abandon complet de la méthode de commutation de circuits utilisée dans les réseaux GSM pour acheminer la voix. Absolument toutes les données passent par le réseau EPC utilisant la commutation de paquets. Ce choix

d'utiliser la Voice over IP (VoIP) simplifie encore l'architecture du réseau mobile et sa gestion, mais apporte aussi des contraintes. Transférer de la voix sur IP est aujourd'hui très courant, mais la qualité observée peut être très variable suivant la charge que subit le réseau. Les réseaux mobiles servant principalement à offrir un service de téléphonie il serait mal vu que celui-ci ne soit pas de bonne qualité. Le réseau LTE utilise donc des techniques de QoS permettant de différentier les services et notamment la VoIP de manière à prioriser au maximum ses paquets et éviter une dégradation des communications téléphoniques.

Comme dans les réseaux UMTS chaque service peut posséder un *bearer* [62], cependant plusieurs services ou connexion peuvent emprunter le même *bearer*, un *bearer* peut ne pas être alloué à un et un seul service [29]. Dans LTE il existe plusieurs types de *bearer* :

Défaut Chaque UE à sa connexion crée son unique *bearer* par défaut, celui-ci est toujours présent, même quand le UE est en mode *idle*. Tous les trafics provenant ou allant au UE et qui n'appartiennent à aucun autre *bearer* passent par celui-ci.

Dédié Ils sont créés, modifiés ou détruits à la demande du UE ou du réseau. Ils sont généralement réservés à un service en particulier, mais rien n'empêche un opérateur de régler les TFT des UE pour que plusieurs connexions, plusieurs types de trafic ou plusieurs services passent par un même *bearer*. Il existe deux sous-types de *bearer* dédié :

non-GBR Leur débit est limité par les Aggregate Maximum Bit Rate (AMBR) (voir ci-dessous).

GBR Ils possèdent des paramètres de QoS supplémentaires (voir ci-dessous) et leurs bandes passantes n'est pas limitées par les AMBR.

Dans le réseau EPC, la bande passante est divisée en deux ou en trois suivant la juridiction du pays. Aux États-Unis par exemple, il est interdit de couper une communication téléphonique en cours, mais il est obligatoire de pouvoir faire des appels de détresse (même quant le réseau est congestionné). Les opérateurs sont donc obligés de conserver une partie de la bande passante libre pour les appels d'urgence éventuels. Dans d'autres pays, il est autorisé de couper une communication en cas d'urgence, toute la bande passante peut donc être exploitée. Lors d'un appel d'urgence, les services de faibles priorités sont alors susceptibles d'être coupés.

Une partie de la bande passante est réservée aux *bearers* GBR. Si cette portion est totalement utilisée, il est impossible de créer de nouveaux *bearers* GBR, même si aucun *bearer* non-GBR n'est créé.

Chaque *bearer* dédié possède les paramètres de QoS suivants :

Allocation and Retention Priority (ARP) Il désigne la priorité du *bearer* par rapport aux autres, au moment de sa création, ou de la création d'un autre. Par exemple, les

services d'urgence possèdent un *bearer* avec ARP très élevé, ils leur sont donc possible de se créer même quand le réseau est congestionné, voire de couper d'autres *bearers* ayant un ARP plus faible.

QoS Class Identifier (QCI) Il définit toutes les contraintes de QoS (excepté en terme de débit) que doit respecter le *bearer*. Neuf QCI, représentés dans le tableau 2.2, sont standardisés mais les opérateurs ont la possibilité de créer leurs propres QCI. Cependant, il est conseillé d'utiliser en priorité celles prédéfinies, car elles sont valides chez tous les opérateurs et peuvent être traitées sur tous les réseaux sans nécessité de *roaming agrement*.

Les *bearers* dédiés GBR possèdent en plus les paramètres suivants :

GBR Correspond au débit réservé pour le *bearer*, cette bande passante lui est garantie, c'est une limite basse.

MBR Correspond au débit maximum autorisé du *bearer*, elle n'est pas garantie, c'est une limite haute. Actuellement (dans la version 9 des spécifications 3GPP) il est toujours égal au GBR, mais les futures versions prévoient d'autoriser l'utilisation de valeurs différentes.

Les *bearers* non GBR partagent deux paramètres de QoS appartenant au UE [8] :

UE AMBR Ce paramètre n'est utilisé que dans le eNodeB, il permet de limiter la bande passante global d'un UE. La bande passante de tous les *bearers* non GBR (*bearer* par défaut inclus) doit être inférieur à cette valeur. Les *bearers* GBR ne sont pas concernés par cette restriction.

Access Point Name (APN) AMBR Un UE peut être connecté à plusieurs PDN, ou sous-réseaux (subdivision sur un même PDN). Chacun de ces accès se fait sur un APN qui désigne de façon unique un réseau. Grâce à ce paramètre, il est possible de limiter la bande passante d'un UE sur un APN spécifique, il n'est utile qu'à l'entrée de chaque APN et donc sur le PGW. Un UE peut donc posséder plusieurs APN AMBR. Par exemple il peut être connecté sur les APN « Internet » et « réseau de l'entreprise », l'intranet de sa compagnie étant surdimensionné son APN AMBR est très élevé. Sur l'APN « Internet » le UE se verra attribué un APN AMBR correspondant au prix du contrat qu'il a souscrit avec son opérateur.

Les *bearers* peuvent être créés par le réseau ou par le UE. Rendre le UE incapable de demander la création d'un *bearer* (UE QoS *unaware*) offre l'avantage de rendre les terminaux plus simples. Par exemple lorsque le terminal veut lancer un appel téléphonique il communique les informations nécessaires (personne à joindre) au AF pour que celui-ci le mette en

Tableau 2.2 – QCI standardisé du réseau LTE (inspiré de Olsson *et al.* [81])

QCI	Type de *bearer*	Priorité	délai maximum	taux de perte de paquets maximum	Exemple de service
1	GBR	2	100 ms	10^{-2}	VoIP
2		4	150 ms	10^{-3}	vidéo bidirectionnel (conversationnel)
3		3	50 ms	10^{-3}	jeu en temps réel
4		5	300 ms	10^{-6}	vidéo en *streaming* de type VoD, non conversationnel
5	Non GBR	1	100 ms	10^{-6}	signalisation IMS
6		6	300 ms	10^{-6}	*streaming* vidéo basé sur TCP (HTML 5), web
7		7	100 ms	10^{-3}	voix et vidéo en *streaming* temps réel (*live*), jeu interactif
8		8	300 ms	10^{-6}	*streaming* vidéo basé sur TCP (HTML 5), web
9		9	300 ms	10^{-6}	

communication VoIP avec l'interlocuteur. Le AF a donc conscience de la création d'un nouveau service sur le réseau, il peut donc informer le PCRF qui va lancer une procédure de création de *bearer* dédié pour la VoIP.

Une autre possibilité est d'utiliser la technique du Deep Packet Inspection (DPI) sur le plan de données pour analyser en temps réel les types de trafics relatifs à chaque UE. C'est ce que fait le SASN de Ericsson. Par exemple si un nouveau trafic vidéo est détecté, le PCRF est contacté et de la même manière que pour le AF, un nouveau *bearer* est créé. Il est aussi possible de combiner les techniques, rendre le UE QoS *aware* pour certains services et laisser le réseau gérer les autres.

Dans les normes de LTE il n'est pas spécifié comment le réseau doit procéder pour respecter le QCI des *bearers*. Le choix de la technique de QoS utilisée (DiffServ, Integrated services (IntServ) avec ou sans Multiprotocol Label Switching (MPLS), etc.) est laissé au choix de l'opérateur.

Interface Radio

L'interface Radio, comme c'est le cas en UMTS, peut utiliser les techniques de duplexage FDD et TDD ce qui permet de tirer parti de nombreuses plages de fréquence, même très restreintes. Comme WiMAX, elle utilise des techniques de duplexage basées sur OFDM : Single-carrier FDMA (SC-FDMA) aussi appelée DFT-spread OFDM (DFTS-OFDM) en *uplink* et

OFDM en *downlink*. La technique SC-FDMA permet de résoudre un problème lié à OFDM appelé Peak-Average-Power Ratio (PAPR). En *downlink* les eNodeB sont assez puissants pour contrer ce problème et éviter son apparition, en revanche en *uplink* les UE ont des capacités limitées en terme de Central Processing Unit (CPU) et de batterie (voir Lescuyer et Lucidarme [73] pour plus de détails).

LTE supporte MIMO en *uplink* et *downlink* ce qui lui permet de multiplier par quatre l'efficacité spectrale et par dix le nombre d'usagers par cellule, comparé aux premières versions d'UMTS [36].

2.2 Mobilité IP

Dans les réseaux IP, certaines extensions au protocole permettent de gérer directement la mobilité des usagers.

2.2.1 Mobile IPv4

MIPv4 est le plus vieux (2002 [85]) des protocoles de gestion de mobilité que nous allons décrire. Son architecture se compose des nœuds suivants :

MN Il se déplace sur le réseau et doit explicitement supporter le protocole MIPv4.

Correspondent Node (CN) Il est fixe ou mobile et communique avec le MN. Il ne supporte pas forcément MIPv4.

HA C'est le point d'encrage du MN sur le réseau source.

Foreign Agent (FA) C'est un intermédiaire appartenant au réseau visité. Ce nœud est optionnel si le MN est capable de supporter la technique *co-located* Care of Address (CoA).

Initialement le MN découvre son HA grâce au message Internet Control Message Protocol (ICMP) *Agent Advertisement* diffusé par celui-ci, sollicité ou non par un message ICMP *Agent Solicitation* envoyé par le MN. Le MN lance alors une procédure d'enregistrement et d'authentification avec le HA aux termes desquelles il recevra son adresse permanente appelée Home Address (HoA). Le HA crée une entrée pour le MN dans sa table de correspondance entre les HoA et les CoA (une correspondance étant appelée *mobility binding*). Le CoA du MN est son adresse temporaire lors de ses déplacements, pour l'instant (dans le réseau *home*) elle est égale au HoA.

Lorsque le MN se déplace il détecte le changement de réseau, soit de façon autonome, soit en recevant un des messages ICMP *Router Advertisement* diffusé par le FA. Il s'enregistre et s'authentifie alors auprès de lui et reçoit une adresse temporaire, la CoA (généralement

déterminée grâce au *Router Advertisement* reçut). Le MN envoie une requête de mise à jour de *mobility binding* appelée Binding Update (BU). Le HA enregistre la CoA du MN, dès lors tous les paquets arrivants sur le HA à destination de l'adresse HoA sont encapsulés dans un tunnel à destination du CoA [85].

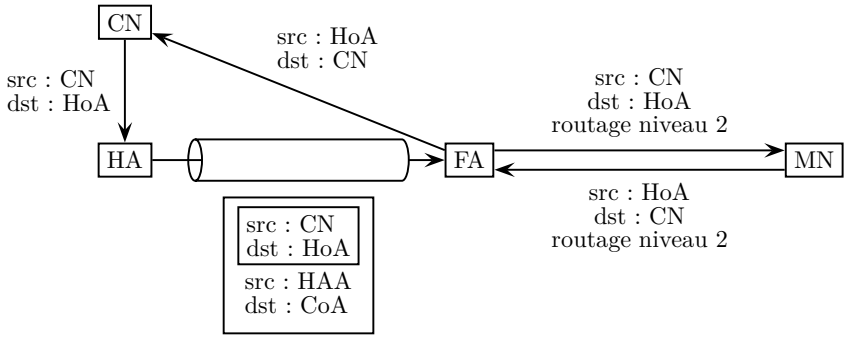
Dans le cas normal, c'est-à-dire lorsque le CoA du MN est l'adresse du FA et que la méthode de routage triangulaire est utilisée, lorsque le tunnel est établi, les données du MN destinées au CN sont acheminées jusqu'au FA par un protocole de routage autre que IP (généralement un protocole de niveau deux tout simplement). Le FA les envoie sur le réseau, ces paquets ayant pour IP source le HoA et pour destination l'adresse du CN, ils sont aiguillés vers le CN. Les données du CN destinées au MN sont routées vers le HA étant donné que pour le CN l'adresse IP du MN est son HoA. Le HA les encapsule dans un tunnel à destination du FA qui les décapsule et les envoie au MN (toujours sans utiliser le protocole IP entre FA et MN). La figure 2.11a schématise ces échanges.

Le fait que le FA envoie directement les données du MN vers le CN peut poser problème. Le HoA n'appartient pas au réseau du FA, il se peut donc qu'un *firewall* rejette les paquets du MN sortant du FA en direction du MN. Pour pallier à ce problème, il est possible d'utiliser la technique de « l'encapsulation retour » (*reverse tunneling*) qui consiste à encapsuler les données provenant du MN sur le FA jusqu'aux HA. Cette technique est schématisée à la figure 2.11b.

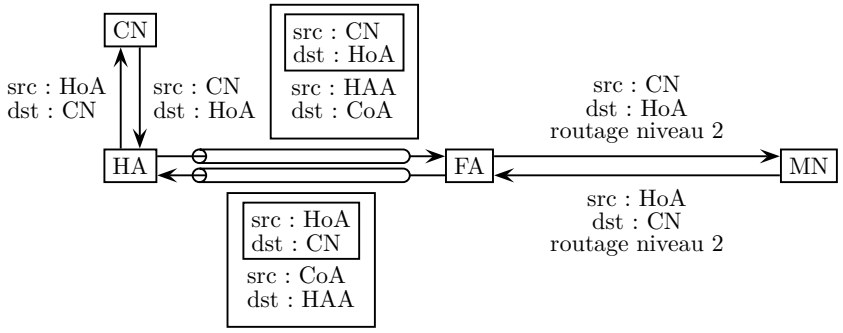
Il est aussi possible pour le MN de se passer du nœud FA en utilisant un *co-located* CoA ; les utilisations de cette technique sont schématisées à la figure 2.12. Cette méthode a cependant le désavantage de nécessiter deux adresses publics IPv4 pour chaque MN, le HoA et le CoA ; les adresses IPv4 se font de plus en plus rares (l'Internet Assigned Numbers Authority (IANA) devrait cesser de distribuer ses adresses IPv4 début 2011 [101, sec. 1]).

2.2.2 Mobile IPv6

Le protocole IPv6 possède lui aussi une extension (ou protocole à part entière) appelée MIPv6 permettant de gérer la mobilité. Ici nous ne nous intéresserons pas à la sécurité du protocole largement détaillée par Johnson *et al.* [72], Patel *et al.* [83] et Devarapalli et Dupont [59], mais plutôt aux quelques différences avec MIPv4. L'architecture MIPv6 est semblable à celle de MIPv4, mais ne comporte pas de FA. La pénurie d'adresses problématiques pour la technique *co-located* CoA de MIPv4 ne touche pas le protocole IPv6 (qui a d'ailleurs été conçu dans le but de pallier à cette pénurie). MIPv6 utilise par défaut le « l'encapsulation bidirectionnelle » (*bidirectional tunneling*) équivalent du *reverse tunneling* de MIPv4 [72]. Cette méthode ne nécessite pas le support de MIPv6 par le CN. Au niveau de l'architecture et de l'échange de données, elle ressemble à la méthode d'encapsulation retour avec *co-located*

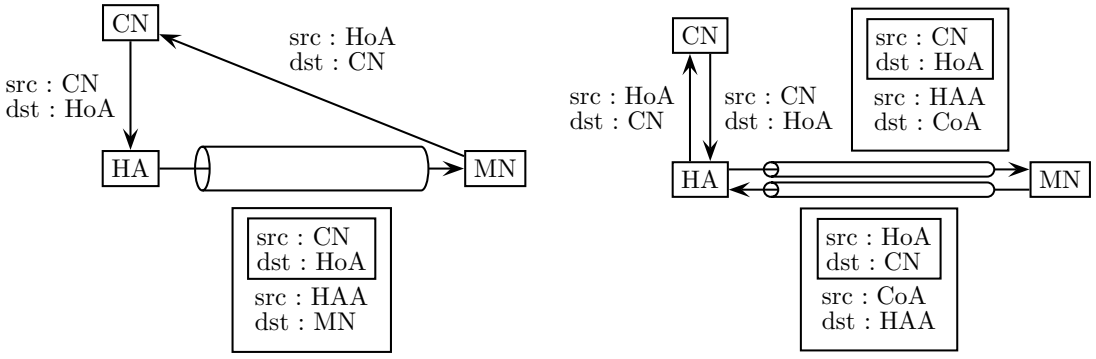


(a) routage triangulaire



(b) encapsulation retour

Figure 2.11 – Communication MIPv4 avec FA CoA



(a) routage triangulaire

(b) encapsulation retour

Figure 2.12 – Communication MIPv4 avec *co-located* CoA

CoA de MIPv4 schématisée à la figure 2.12b.

Si le CN supporte le protocole MIPv6, une méthode d'optimisation du routage (*route optimization*) peut être utilisée. Elle permet d'améliorer la distance parcourue par les paquets de données, mais aussi de limiter les risques de congestion du niveau du HA [72]. Dans ce cas, même si le HA est toujours contacté lors des procédures de sécurité ou pour mettre à jour les *mobility binding*, le trafic de données ne transite plus par lui. Un BU est envoyé par le MN au CN pour l'informer d'un éventuel déplacement et le CN conserve lui-même une table de *mobility binding*. Les données du MN à destination du CN sont envoyées avec l'adresse source CoA, le CN répond en utilisant le CoA comme adresse de destination, mais fournit aussi le HoA dans un nouveau type d'en-tête (*type 2 routing header*). La figure 2.13 schématise cet échange.

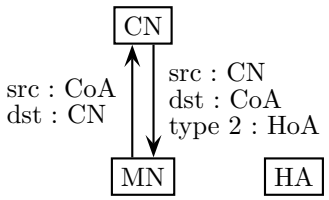


Figure 2.13 – Communication MIPv6 avec optimisation de routage

2.2.3 Proxy Mobile IPv6

Le protocole PMIPv6 utilise les avantages du protocole IPv6 pour apporter la mobilité aux usagers. Contrairement à MIPv6 qui se base sur les terminaux pour gérer leur mobilité [72], PMIPv6 se base entièrement sur le réseau, les usagers doivent simplement posséder une pile IPv6 normale [65]. Il est aussi possible d'utiliser PMIPv6 avec certaines parties du réseau en IPv4, mais nous ne nous intéresserons pas à cette propriété ici.

PMIPv6 reprend une grande partie du protocole MIPv6 et introduit un nouveau nœud appelé MAG gérant la mobilité à la place du MN. Les messages utilisés dans MIPv6 comme le BU, renommé Proxy Binding Update (PBU), et le Binding Acknowledgement (BA), renommé Proxy Binding Acknowledgement (PBA), sont réutilisés avec quelques options en plus. Le LMA de MIPv6 est réutilisé et agrémenté de quelques fonctions ce qui permet de gérer sur un même réseau les MN supportant MIPv6 et ceux qui ne le supportent pas.

Le nouveau nœud MAG a les rôles suivants [65] :

- Il est responsable de la détection de déplacement du MN ;
- Il gère la communication sur le plan de contrôle avec le LMA ;
- Il « fait croire » au MN qu'il est dans son réseau d'origine en lui envoyant des *router advertisement* contenant son/ses Home Network Prefix (HNP) ;

- Il route et encapsule les paquets du MN vers son LMA, et inversement, il décapsule les paquets provenant du LMA et les route vers les MN.

Lorsqu'un MN se connecte au domaine PMIPv6, le MAG le détecte ; il peut par exemple intercepter un message Internet Control Message Protocol version 6 (ICMPv6) *Router Solicitation* envoyé par le MN (mais ce n'est pas optimal). La méthode de détection du MN n'est pas spécifiée par Gundavelli *et al.* [65]. Au lieu d'envoyer tout de suite un message *Router Advertisment* au MN, comme le ferait un routeur IPv6 normal, le MAG informe le LMA du MN de cette connexion. Il interroge donc une base de données locale ou distante contenant le *Policy Profile* du MN ; tous les MAG et les LMA doivent avoir accès à cette base de données ou en avoir une copie locale. Ce *Policy Profile* contient au minimum l'identifiant du MN et son LMA. Il envoie alors à ce LMA un message PBU contenant le *flag* `P` introduit par PMIPv6 (ce *flag* a pour but d'indiquer au LMA que cette connexion utilise le protocole PMIPv6 plutôt que MIPv6), l'identifiant du MN et quelques options utiles comme le *Handoff Indicator* (voir section 4.3.6).

Le LMA met à jour sa table de correspondance appelée *Binding Cache*. Celle-ci contient toutes les informations nécessaires au maintien des « sessions de mobilité » des MN. Les informations spécifiques à PMIPv6 sont [65] :

- un *flag* indiquant que le MN utilise PMIPv6 ;
- l'identifiant du MN ;
- l'identifiant de couche liaison OSI du MN, cet identifiant doit être constant tout au long de la session. Il peut être égal à zéro si la technologie d'accès ne peut pas fournir d'identifiant.
- l'adresse IPv6 *link local* que doivent utiliser les MAG pour communiquer avec le MN. Ce n'est pas l'adresse du MN mais l'adresse source que les MAG doivent adopter pour « faire croire » au MN qu'il est toujours dans le même réseau (cf. paragraphe 2.2.3).
- la liste des HNP du MN, ils peuvent être configurés de façon statique dans le *Policy Profile* ou être générés dynamiquement par le LMA à la connexion de celui-ci ;
- l'interface que le LMA doit utiliser pour correspondre avec le MAG, et donc le tunnel dans lequel les paquets destinés au MN doivent être encapsulés ;
- la technologie d'accès actuellement utilisée par le MN ;
- la date d'envoi du dernier PBU reçu si celui-ci contenait l'option *Timestamp* ou la date de réception s'il ne la contenait pas. Cette valeur permet au LMA de traiter les PBU dans leur ordre chronologique (un « vieux » PBU arrivant sur le LMA est tout simplement rejeté).

Après avoir vérifié l'identité et les droits du MN, accepté le PBU et créé son Binding Cache Entry (BCE) (mise à jour du *Binding Cache* du MN) le LMA envoie un message

PBA au MAG pour lui indiquer qu'il peut continuer la séquence d'attachement. Si le tunnel bidirectionnel entre le LMA et le MAG n'existe pas, alors celui-ci est créé. Ces tunnels peuvent être créés dynamiquement à la connexion des MN ou être configurés de façon statique à la mise en place du domaine PMIPv6. Il ne peut en exister qu'un par couple LMA-MAG, aussi les données de tous les MN connectés à un MAG empruntent le même. Leurs « point d'accès » sont les Proxy-CoA et les Local Mobility Anchor Address (LMAA). Lorsque le MAG reçoit le PBA il crée sa partie du tunnel et met à jour ses tables de routage pour faire en sorte que les données du MN y transitent. Il envoie aussi au MN ses HNP via un message *Routeur Advertisment*. La figure 2.14 schématise un domaine PMIPv6 avec deux LMA et deux MAG.

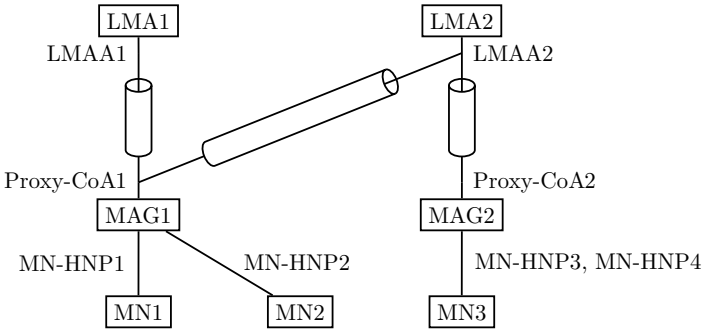


Figure 2.14 – Ensemble des éléments du réseau PMIPv6 (Proxy Mobile IPv6 *Domain*) [65]

Adresse *link local*

L'adresse source du message ICMPv6 *Routeur Advertisment* est une adresse *link-local* fournie par le LMA ou configurée statiquement sur le MAG. Tous les messages du réseau que reçoit le MN doivent avoir cette adresse comme source, de cette façon le MN ne peut détecter les changements de réseaux. Si cette adresse est configurée statiquement alors elle doit être la même pour tous les MAG.

Lorsque le MN reçoit le *Routeur Advertisment* il enregistre ses HNP et configure un HoA à partir de ceux-ci ; comme c'est le cas dans le protocole IPv6 [56]. Le réseau ne connait que les HNP du MN, seul celui-ci connait son HoA.

La procédure d'attachement du MN est schématisée à la figure 2.15. Lorsque le réseau a complété son enregistrement, le MN peut communiquer avec l'extérieur via le MAG qu'il considère comme un routeur IPv6 normal. Le MAG aiguille les données vers le LMA via le tunnel, le LMA décapsule les données pour les envoyer sur le réseau externe. Les paquets provenant de l'extérieur et à destination du MN empruntent le chemin inverse. La figure 2.16 schématise cette communication ; nous pouvons remarquer que celle-ci ressemble en tout point à une communication utilisant le protocole MIPv4 avec encapsulation retour et FA CoA schématisé à la figure 2.11b.

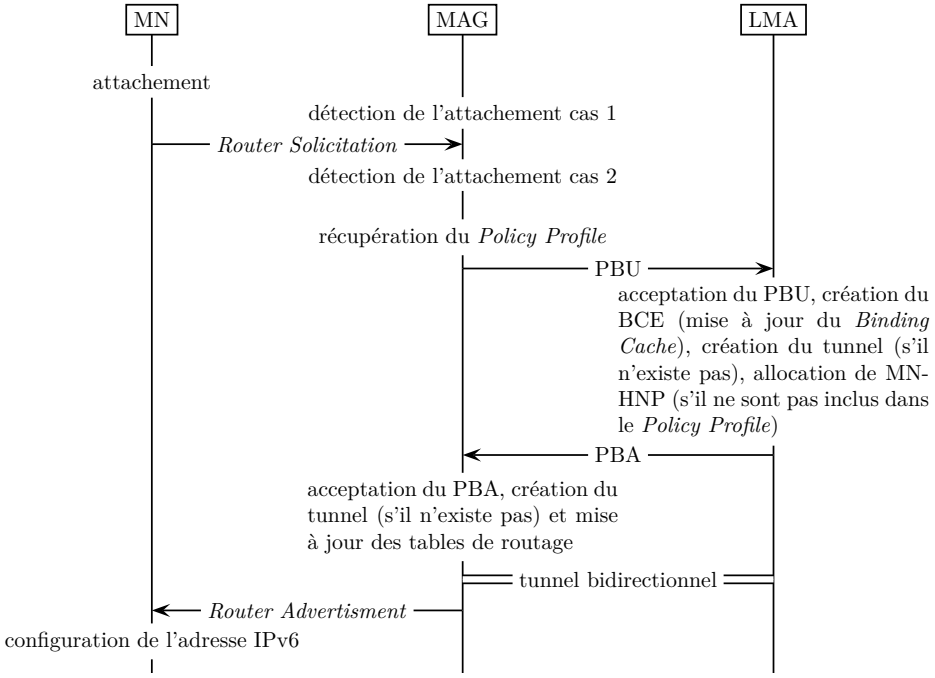


Figure 2.15 – Attache d'un MN à un domaine PMIPv6 (inspiré de Gundavelli *et al.* [65])

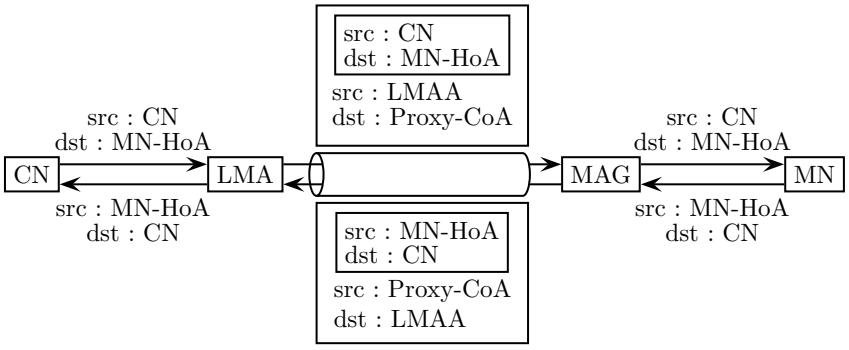


Figure 2.16 – Communication simple dans un réseau PMIPv6 (inspirée de Gundavelli *et al.* [65])

Handover

Lorsqu'un MN se déplace d'un MAG à l'autre un *handover* est effectué dans le domaine PMIPv6. C'est un *hard handover* c'est-à-dire que le MN se détache du réseau pour ensuite s'attacher au nouveau MAG (lors d'un *soft handover* le terminal mobile s'attache au nouveau réseau avant de se détacher de l'ancien).

Le MN se détache de l'ancien MAG qui détecte ce changement et informe le LMA par un PBU possédant un *lifetime* de zero (cf. figure 4.27). À la réception de ce message le LMA met à jour la BCE du MN, répond au MAG par un PBA et lance un compte à rebours. Si le LMA ne reçoit pas d'autre information concernant ce MN avant la fin du compte à rebours, il supprime la BCE et considère le MN comme hors du domaine PMIPv6.

Le MN se lie au nouveau MAG qui lance une procédure d'attache comme décrite précédemment. Si ce MAG est capable de déterminer si le MN provient d'un *handover* ou s'il effectue sa connexion initiale au domaine, il doit en informer le LMA grâce à l'option *Handoff Indicator* du PBU (voir section 4.3.6).

À la réception du nouveau PBU le LMA arrête les comptes à rebours et met à jour le BCE du MN, il enregistre en particulier le nouveau Proxy-CoA. Il renvoie au MAG les HNP, l'adresse *link-local* à utiliser et les autres informations utiles via un message PBA. Mais contrairement au cas où le MN se connecte pour la première fois, il ne génère pas de données dynamiquement, toutes les informations envoyées dans le PBA sont tirées du BCE du MN. De cette façon le MAG peut configurer un environnement en tout point égal à celui du précédent MAG. Le *Router Advertisment* envoyé au MN est donc le même que ceux envoyés par l'ancien MAG, la couche IPv6 du MN ne détecte aucun changement de réseau. La figure 2.17 schématise cette procédure.

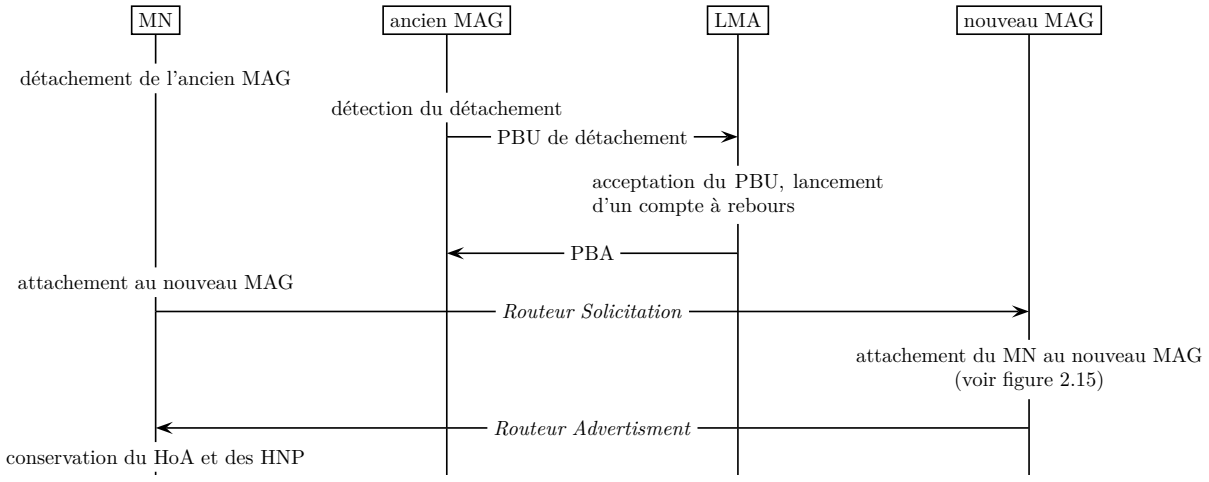


Figure 2.17 – *Handover* dans un domaine PMIPv6 (inspiré de Gundavelli *et al.* [65])

2.3 DiffServ

Dans les spécifications 23.401 [8] et 23.402 [9], les mécanismes de QoS ne sont spécifiés que pour la couche OSI n° 7 : Application. Il est précisé que le trafic doit être séparé dans différents *bearers* ayant chacun leurs propres caractéristiques de QoS. Dans nos simulateurs nous avons implémenté les MBR, GBR et UE Aggregated Maximum BitRate (UE-AMBR) au niveau des applications GTP, la bande passante des *bearers* ne peut donc pas excéder une certaine valeur fixée par l'utilisateur. Cependant, cela ne garantit pas que les paquets passants par les *bearers* GBR ne soient pas rejetés à cause de la charge des *bearers* non-GBR.

Pour supporter les caractéristiques des QCI comme la priorisation, les contraintes de délai, etc. il est conseillé [4] dans 23.401 [8] et 23.402 [9] d'utiliser une architecture DiffServ.

DiffServ (définie par Blake *et al.* [45] et Nichols *et al.* [80]) est une architecture permettant de faire de la QoS, c'est-à-dire de différentier les paquets les uns des autres et de les prioriser/ralentir/rejeter en conséquence. Les paquets IPv4 et IPv6 possèdent des champs DSCP (inclus dans le champ ToS en IPv4 et dans le champ *Traffic Class* en IPv6 comme indiqué par Nichols *et al.* [80]) permettant de spécifier à quelle classe de service ils appartiennent. Typiquement dans une architecture de ce type, les routeurs situés dans la zone DiffServ classifient les paquets en fonction de leurs DSCP pour les mettre dans des files d'attente. Un algorithme vient ensuite enlever les paquets des files d'attente pour les envoyer sur le lien. Les routeurs en bordure de zone DiffServ, en plus de faire ce travail, marquent les paquets.

Il y a donc plusieurs étapes bien distinctes dans cette architecture :

- Le marquage (sur les routeurs de bordures) ;
- La classification ;
- Le *Traffic Conditionning* qui peut remarquer/rejeter le paquet ;
- La mise en file d'attente qui peut rejeter le paquet ;
- Le *Scheduling* qui détermine le prochain paquet à sortir sur le lien.

2.3.1 Le marquage

Le marquage des paquets se fait sur les routeurs en bordure de zone DiffServ, il peut être basé sur les ports, les adresses IP, ou encore utiliser une technique de DPI (comme il est prévu pour le SASN). Le DPI permet de classer un paquet suivant toutes les informations qu'il pourrait contenir, par exemple, l'application destinataire du paquet ou le type de donnée contenu (texte, vidéo, son, etc.), certains outils de DPI permettent même de décrypter les connexions chiffrées (e.g. HyperText Transfer Protocol Secured (HTTPS)). Une fois que le paquet a été analysé, son champ DSCP est changé pour refléter la politique de QoS qui doit

4. Les exemples donnés mentionnent l'utilisation de DiffServ.

lui être appliquée.

DiffServ définit trois principaux types de politiques appelées Per-Hop Behavior (PHB) [43] :

Default Forwarding (DF) est utilisé pour les trafics sans réel importance, il correspond au *Best Effort*. En définitive aucun traitement spécial n'est effectué sur ces paquets.

Assured Forwarding (AF) comme décrit par Heinanen *et al.* [67] ce PHB est utilisé pour garantir que le trafic en deçà d'une certaine limite de bande passante sera acheminé et que les paquets ne seront pas réordonnés. Le trafic au-delà de la limite de bande passante a une forte probabilité d'être acheminé, sans pour autant en être garanti. Il est conseillé pour les trafics de conférence multimédia, de *streaming* et pour les services nécessitants une faible latence.

Expedited Forwarding (EF) comme décrit par Davie *et al.* [55] ce PHB est utilisé pour garantir que le trafic ait un faible délai, un faible IPDV et un faible taux de perte de paquets. Comme le AF, il garantit le passage des paquets en deçà d'une certaine valeur de bande passante, mais garantit aussi que les paquets ne seront pas retardés par le routeur courant au-delà d'une certaine limite ; un IPDV maximum est assuré. Ce PHB est recommandé pour le trafic VoIP.

2.3.2 La classification

Le « classifieur » sépare les paquets en fonction de leurs DSCP et les envoie au *policer* correspondant. Chaque nœud DiffServ en possède un.

2.3.3 Le *Traffic Conditionning* aussi appelé *Policing*

Le *Traffic Conditionning* s'assure que le trafic dont il a la charge ne dépasse pas une certaine bande passante (dans la plupart des cas). Il peut remarquer ou rejeter (*drop*) les paquets non conformes. Les différents types de *Traffic Conditionning* mentionnés par Babiarz *et al.* [43] sont [5] :

- le seau à jetons (*token bucket*) appelé `sr+ts`, représenté à la figure 2.18 ;
- le single-rate, three-color marker (srTCM) décrit par Heinanen et Guerin [68] et représenté à la figure 2.19 ;
- le two-rate, three-color marker (trTCM) décrit par Heinanen et Guerin [69] et représenté à la figure 2.20.

Les actions décrites dans ces schémas peuvent être un re-marquage, une transmission, un rejet du paquet, ou une combinaison de deux de ces actions. Babiarz *et al.* [43] conseille

5. Tous sont disponibles sous NS-2.

d'utiliser un seau à jetons pour le trafic EF et srTCM ou trTCM pour les différents types de trafics AF ; le trafic DF quant à lui ne passe dans aucun *policer*.

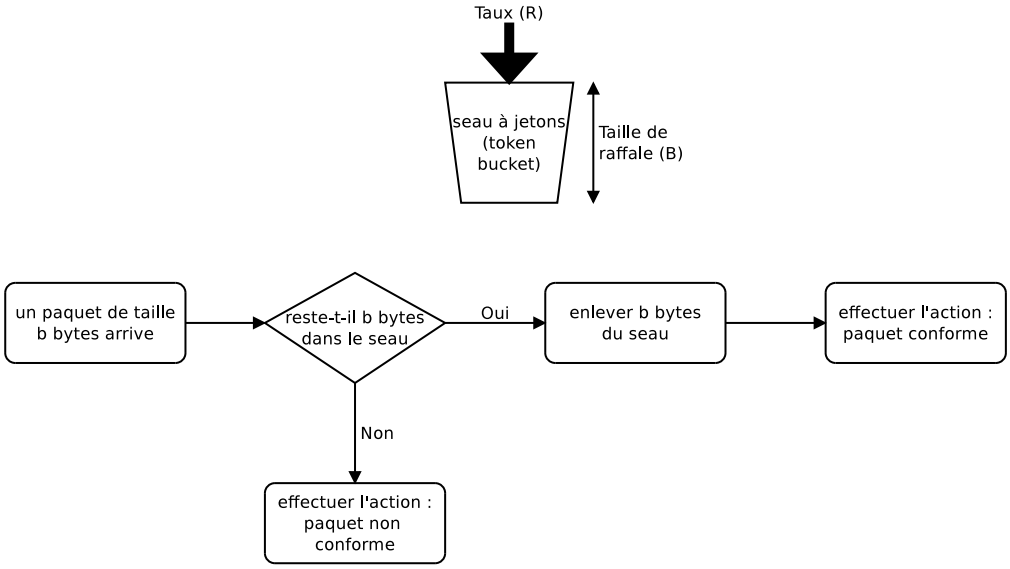


Figure 2.18 – Filtre de Seau à Jetons (inspiré de Menoncin [76])

2.3.4 Active Queue Management

Passé le *Traffic Conditionning* les paquets passent par un algorithme de rejet (il y a un module de rejet par *policer*). Ces modules servent à contrôler la taille des files d'attente dans lesquelles seront mis les paquets. Les principaux modules sont :

- *Drop Tail* dont le fonctionnement est schématisé à la figure 2.21a ;
- Random Early Detection (RED) décrit par Braden *et al.* [46] et dont le fonctionnement est schématisé à la figure 2.21b ;
- Weighted Random Early Detection (WRED) dont le fonctionnement est schématisé à la figure 2.22. C'est une évolution de RED développée par *Cisco* qui permet de distinguer, pour une même file d'attente, les paquets de DSCP différent et d'y associer des paramètres RED distincts (maxp, minth, maxth) [52].

2.3.5 Le *Scheduling*

Le *scheduling* définit la manière dont les files d'attente sont servies. Les types d'algorithmes évoqués par Babiarz *et al.* [43] sont Priority Queueing (PQ) (aussi appelé PRIO) et *Rate Queuing* dont on mentionne Weighted Fair Queueing (WFQ) et Weighted Round-Robin (WRR). L'algorithme PRIO consiste à servir en premier les files configurées avec la priorité la plus forte ; tant qu'il y a des paquets dans cette file aucune des autres n'est servie.

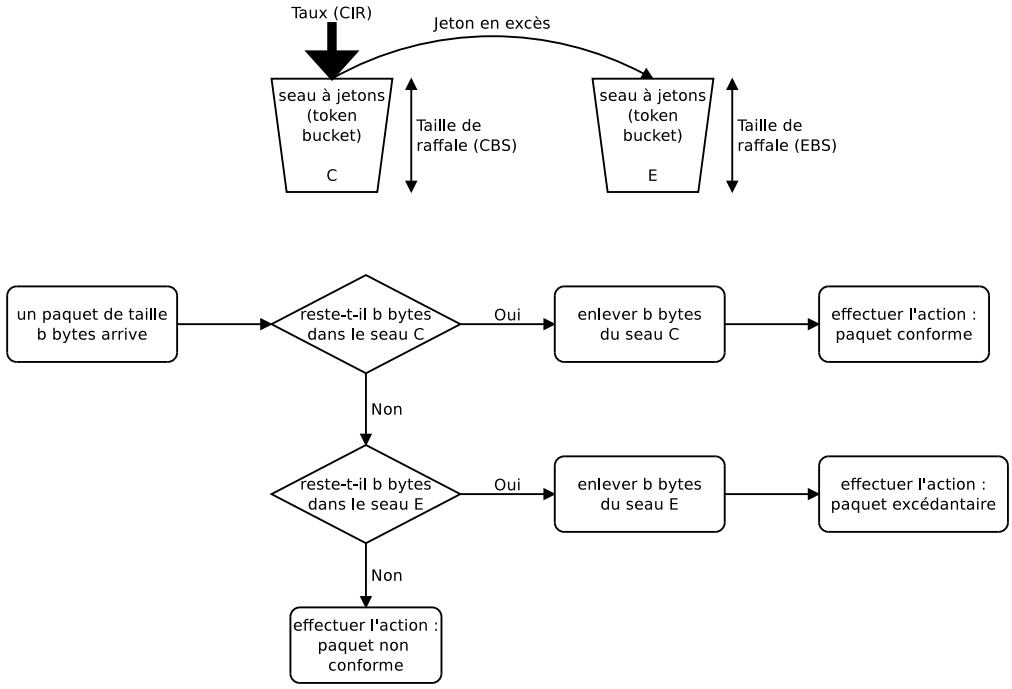


Figure 2.19 – Filtre srTCM (inspiré de Menoncin [76])

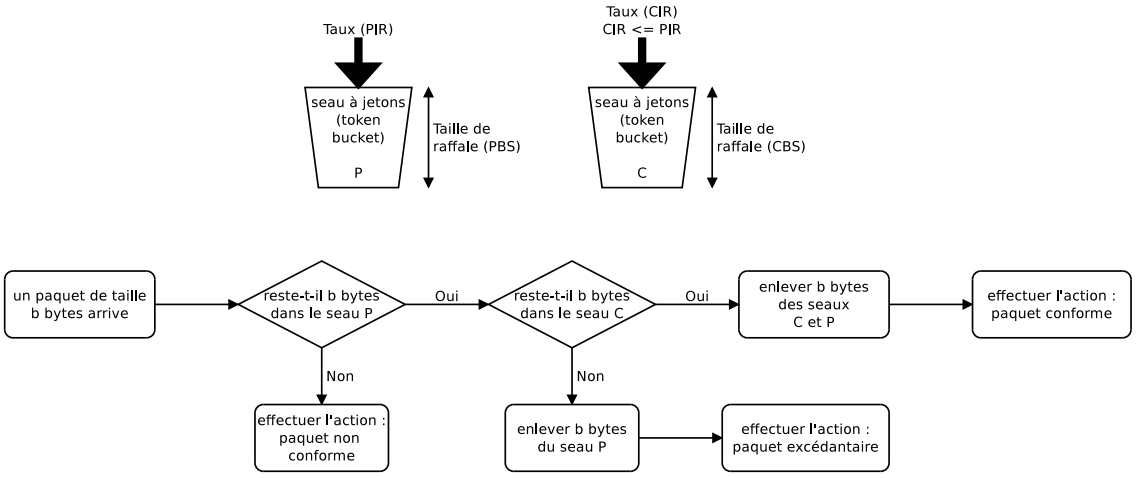
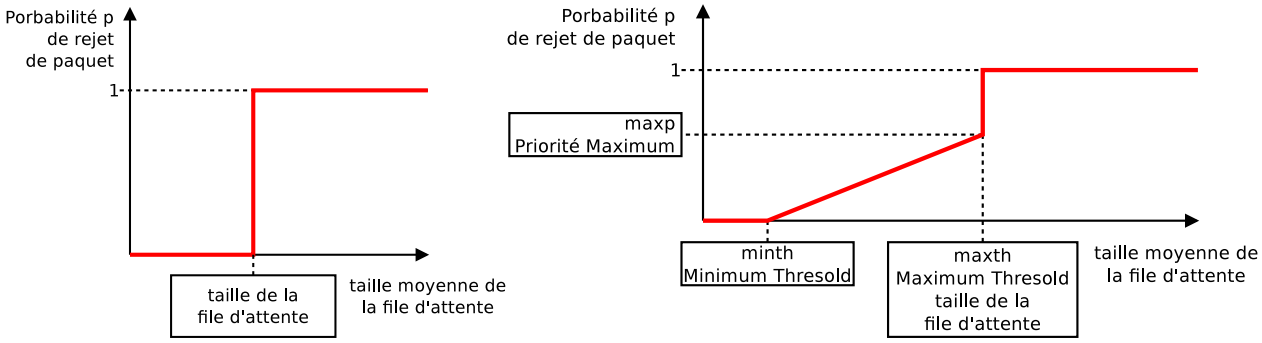


Figure 2.20 – Filtre trTCM (inspiré de Menoncin [76])



(a) *Drop Tail* (b) RED

Figure 2.21 – Méthodes de gestion des files d'attente *Drop Tail* et RED

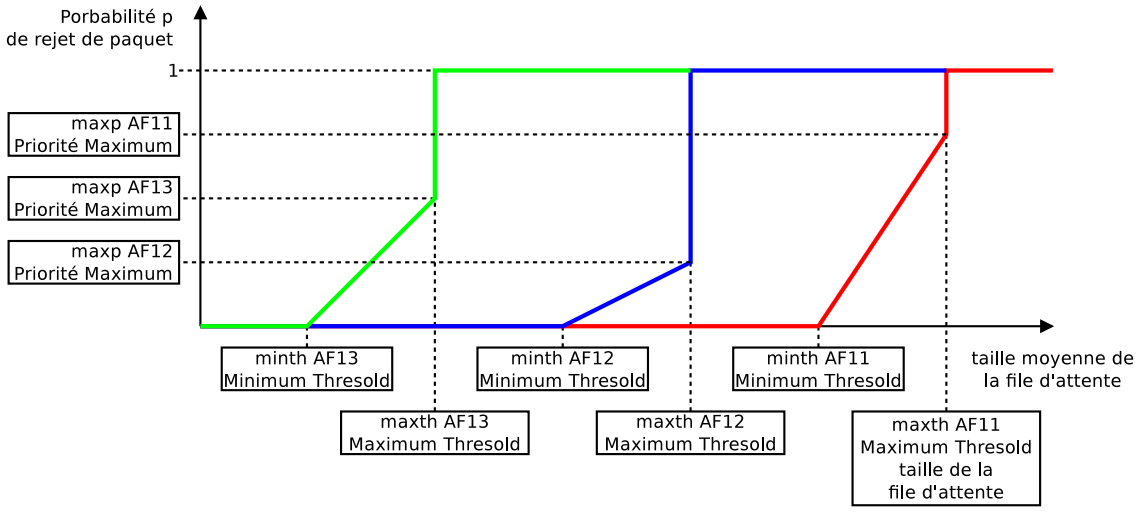


Figure 2.22 – Méthode de gestion de file d'attente WRED

Cet algorithme est utile si l'on souhaite qu'un type de trafic passe avant tous les autres, même si celui-ci risque de s'approprier toute la bande passante. En effet, il existe un risque de « famine » des files de basses priorités. Dans un environnement d'entreprise ou dans le cas de structures plus grandes, ce risque n'est pas envisageable[6]. C'est pourquoi ce type de file est associé à un limiteur de bande passante (*Rate Limiter*) [104].

Pour décrire l'algorithme WFQ comme expliqué par Vega Garcia et Huitema [98] il est plus judicieux d'expliquer le fonctionnement de Round-Robin (RR) et WRR en premier. L'algorithme basique RR consiste à servir successivement les files d'attente les unes après les autres. Par « servir » nous entendons prendre un paquet et l'envoyer sur le lien. Cet algorithme pose problème lorsque deux files ont des paquets de tailles moyennes différentes, la file ayant les paquets les plus grands se verra attribuer une plus grande partie de la bande passante. Pour remédier à cela il existe l'algorithme WRR associant à chaque file un poids que nous appellerons P. Lorsqu'une file est servit elle envoie P paquets sur le lien. De cette façon on peut associer un poids plus important aux files dont on sait que la taille moyenne des paquets est petite.

L'algorithme le plus équitable en terme de partage de bande passante est le bit-by-bit Round-Robin (BBRR), il s'apparente au RR dans le sens où il sert tour à tour chaque file. Cependant, au lieu d'envoyer un paquet à chaque « service » il n'envoie qu'un bit du paquet. Cet algorithme est bien sûr impossible à réaliser étant donné qu'il fragmente chaque paquet en blocs de un bit. Weighted bit-by-bit Round-Robin (WBBRR) est une variante de BBRR où chaque file possède un poids P et chaque service correspond à l'envoi de P bits sur le lien. Il permet de conserver l'équité du BBRR toute en distribuant de façon non égale la bande passante aux différentes files. Il n'est pas réalisable en pratique de part la fragmentation des paquets qu'il implique.

Le but de l'algorithme WFQ décrit par Vega Garcia et Huitema [98] et développé par *Cisco*, est de simuler l'algorithme WBBRR ; il n'envoie pas des bits, mais des paquets. À chaque paquet est associé un nombre, l'envoi se faisant dans un ordre croissant en fonction de cette valeur. Ce nombre représente le numéro du tour (nombre de tours effectués depuis le lancement de l'algorithme) au cours duquel serait servi le paquet si l'algorithme WBBRR avait été utilisé.

Lorsqu'un paquet arrive dans une file, on détermine son nombre associé N_i en fonction du poids de la file α, de la taille du paquet P_i, du nombre N_{i-1} associé au paquet suivant dans la file et R le nombre de tour qu'aurait fait l'algorithme WBBRR (s'il avait été utilisé) au moment où le paquet est arrivé. La formule adaptée à partir de la formule de l'algorithme

6. Excepté dans le cas de trafic relatif à la sécurité, par exemple : une communication téléphonique vers le 911.

Fair Queueing (FQ) de Vega Garcia et Huitema [98], est la suivante :

$$N_i = max\,(N_{i-1}, R) + \alpha P \tag{2.1}$$

Cependant, même si cet algorithme permet de répartir la bande passante de façon équitable entre les différentes files d'attente, il ne permet pas de garantir un délai maximum comme il est demandé pour le PHB EF. Pour cette raison *Cisco* a développé une autre caractéristique de file appelée Low Latency Queueing (LLQ), celle-ci s'apparente à une file prioritaire de l'algorithme PRIO [42], mais comporte une limite de bande passante permettant d'éviter la « famine » des autres files [51, 50, 57, 75]. Les [7] files LLQ s'utilisent dans un environnement WFQ.

L'algorithme Deficit Round-Robin (DRR) décrit par Shreedhar et Varghese [95] permet lui aussi de faire du partage équitable de ressources. Il s'apparente à un RR dans le sens où les files sont servies tour à tour, mais « servir » signifie cette fois l'attribution d'un certain quantum de bits. Les files possèdent une variable égale à la somme des bits servis depuis l'envoi du dernier paquet. Lorsque cette valeur devient supérieure à la taille du premier paquet de cette file alors celui-ci est envoyé et l'algorithme est arrêté jusqu'à ce que le paquet ait été transmis sur le lien. Il est possible d'associé un quantum de un bit aux files de façon à simuler le BBRR, mais cela demande l'exécution d'un nombre important de tours ce qui implique une forte charge CPU. Nous avons appelé Deficit Weighted bit-by-bit Round-Robin (DWBBRR) une variante de DRR dans laquelle toutes les files possèdent un poids P (généralement inférieur à dix) et où le quantum DRR associé à une file est égal à P bits (voir section 4.2.7 pour plus de détails).

2.3.6 Résumé et Notes

Le schéma de la figure 2.23 représente toutes les étapes par lesquelles passe un paquet en sortie d'un routeur DiffServ de bordure. Certains algorithmes n'ont pas été représentés pour simplifier le schéma ou parce qu'ils sont incompatibles entre eux (par exemple les algorithmes de *Scheduling* PRIO et WFQ). Dans un routeur DiffServ central (ou *Core*) la seule différence aurait été la suppression du module de marquage au tout début de la chaîne.

À titre d'information le noyau Linux permet lui aussi de faire de la QoS, mais la gestion des files d'attente est légèrement différente de celle utilisée dans NS-2 et les routeurs *Cisco* [76]. En effet, le schéma le plus approprié pour décrire le module de QoS du noyau Linux serait celui d'un arbre dans lequel les feuilles seraient les files d'attente avec leurs modules

7. On utilise généralement qu'une file LLQ, mais il est tout de même possible d'en créer plusieurs.

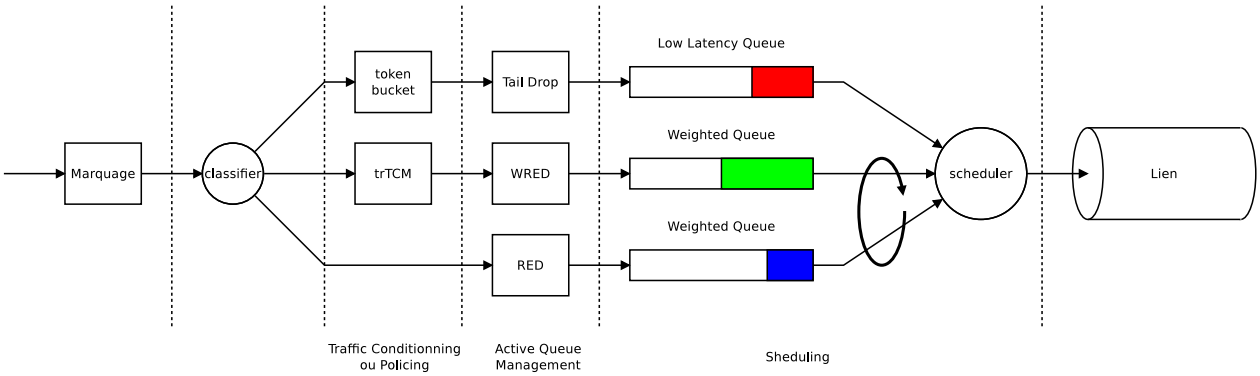


Figure 2.23 – Chemin d'un paquet en sortie d'un routeur DiffServ *Edge* (inspiré de Menoncin [76])

Active Queue Management (ACQ), et chaque fourche représenterait un module de *Traffic Conditionning* ainsi qu'un *Classifier*.

Actuellement le noyau Linux dispose des ACQ suivants :

- b-FIFO et p-FIFO qui sont des files FIFO simples (le b pour Byte et le p pour paquet ne servant qu'à logger de façon différente la charge de la file) ;
- pfifo_fast (file par défaut des interfaces) qui contient trois sous-files servies par l'algorithme PRIO ; cette file classifie elle-même les paquets dans ses sous-files de manière à respecter les spécifications de Almquist [40] ;
- Token Bucket Filter (TBF) est un seau à jetons simple ;
- Stochastic Fairness Queueing (SFQ) permet de répartir équitablement la bande passante entre les différentes connections, les différents flux qu'elle contient.

Quelques modules de *Scheduling* (qui peuvent être plusieurs sur une même interface) :

- PRIO ;
- DRR ;
- Class Based Queueing (CBQ) dont les fonctionnalités sont trop nombreuses pour être décrites ici ;
- Hierarchical Token Bucket (HTB) qui possède moins de fonctionnalité que CBQ, mais qui est beaucoup plus facile à configurer.

Pour avoir plus de détails sur ces différents types de files et algorithmes, nous conseillons la lecture de inetdoc LINUX [71]. Un exemple de configuration se trouve à l'annexe I.

CHAPITRE 3

PROPOSITION D'ÉVOLUTION VERS UNE UTILISATION PLUS AVANCÉE D'IPv6

Comme nous l'avons expliqué précédemment, contrairement aux 3G, le trafic de voix partage le même chemin que les autres dans les réseaux mobiles 4G. Cependant, malgré l'utilisation du protocole IP, le réseau est toujours géré en grande partie par GTP, qui fut développé initialement pour GPRS (2.5G). Dans le cadre de cette maîtrise, nous avons cherché à utiliser au maximum les possibilités offertes par la couche IP et ainsi réduire l'usage d'autres protocoles comme GTP ou GRE. Nous avons considéré que tous les terminaux utilisent IP version 6.

Les réseaux mobiles nécessitent une attention particulière en ce qui concerne les performances et donc les techniques de QoS à employer. Ils sont limités en ressources et l'apparition de nouvelles formes de médias sur les téléphones cellulaires (e.g. la vidéo) les soumet à des contraintes importantes. Le phénomène s'est amplifié dans les réseaux 4G du fait du passage de la voix sur IP, utilisé par les autres types de trafic. Les changements proposés seront testés sur la base des performances du réseau au niveau des terminaux.

Dans ce chapitre nous allons détailler le fonctionnement des architectures 23.401 [8] et 23.402 [9] proposées par 3GPP. Puis nous exposerons les suggestions de Hui *et al.* [70] pour modifier le protocole PMIPv6 ; ce qui nous permettra d'introduire les architectures 403 et 404 que nous proposons. Nous finirons en décrivant les différentes métriques de performances que nous avons utilisées pour évaluer nos modifications.

3.1 L'architecture 23.401

Dans l'architecture 23.401 [8] la QoS est gérée et appliquée grâce à des tunnels appelés *bearers* (voir section 2.1.7) ; ceux-ci étant gérés par le protocole GTP. Ce système mis en place par 3GPP apporte une certaine sécurité étant donné que les *bearers* GBR sont gérés de façon différente des non GBR. Par exemple, la bande passante des liens est divisée, une partie pour les GBR et le reste pour les non GBR. De plus la bande passante des *bearers* GBR, comme son nom l'indique, est réservée.

L'architecture 23.401 [8] est représentée à la figure 1.1a et les couches de protocoles utilisées sur le chemin de données à la figure 3.1. La figure 3.2 donne une vue schématique

des différents types de *bearers* GTP et du paramètre AMBR[1]. Dans chaque *bearer* passe un ou plusieurs services, les non GBR sont limités par les UE-AMBR et APN-AMBR, ce qui n'est pas le cas des GBR.

Les diagrammes de séquences des procédures auxquelles nous nous sommes intéressés, sont représentés dans l'annexe A.

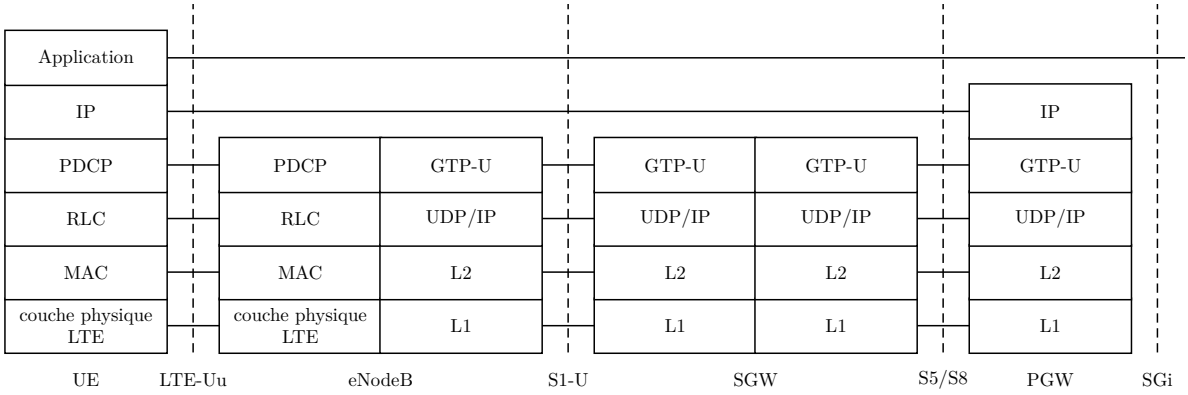


Figure 3.1 – Couches de protocoles du plan de données dans un EPS (inspiré de 23.401 [8])

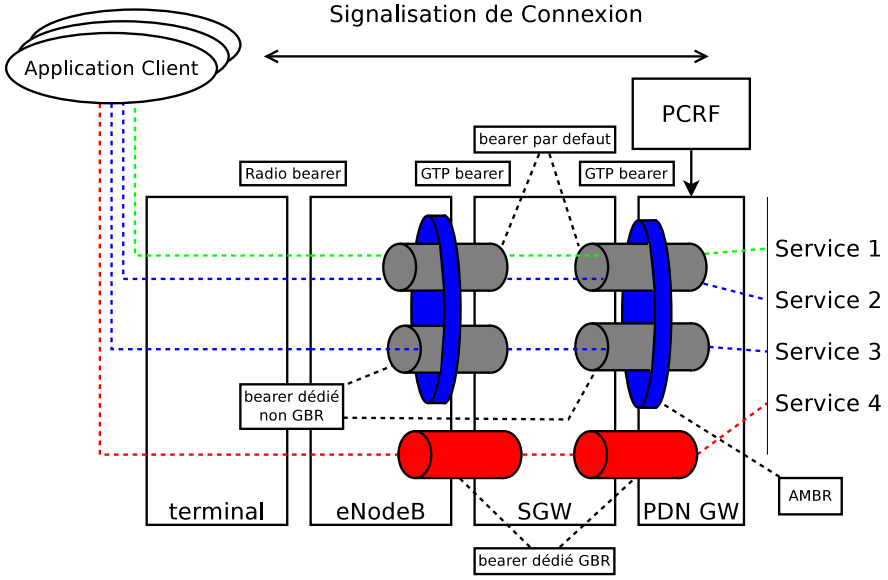


Figure 3.2 – Schéma représentatif des *bearers* GTP dans l'architecture 23.401

3.1.1 GTP

Comme mentionné dans la section 1.1 le protocole GTP se divise en deux domaines d'application. Il agit dans le plan de données pour encapsuler les paquets des usagers ; et

1. Le AMBR n'est pas un tunnel comme on pourrait le croire à la vue de ce schéma.

dans le plan de contrôle pour gérer la mobilité et les *bearers*. Il intervient au dessus du protocole UDP comme représenté à la figure 3.3.

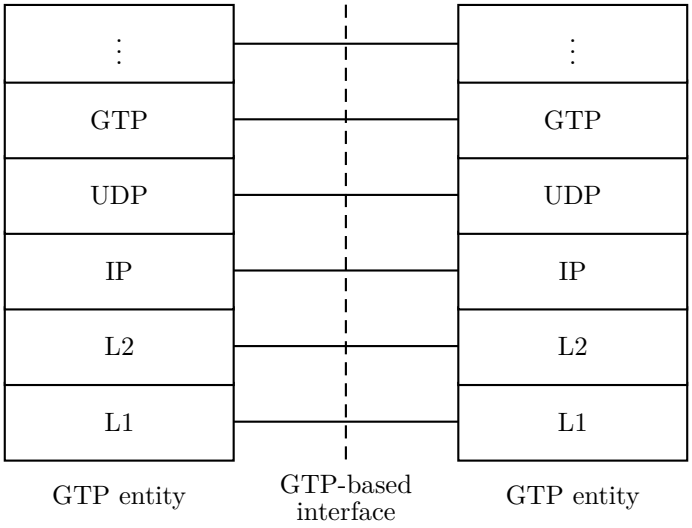


Figure 3.3 – Couches de protocoles utilisées sous GTP

GTP fut initialement développé pour les réseaux GPRS. Le consortium 3GPP l'améliora ensuite pour le rendre compatible avec UMTS ; cette version appelée GTP second stage, first version for EPS (GTPv1) [2] est décrite dans le document 29.060 [16]. 3GPP continua d'apporter des améliorations au protocole pour permettre son utilisation dans le réseau EPS et à partir de la version 8 de ses spécifications, séparât clairement le protocole GTP en GTP-U version 1 (GTPv1-U) décrit dans 29.281 [24], et GTP-C version 2 (GTPv2-C) décrit dans 29.279 [23]. GTP-U est utilisé sur les interfaces S1-MME et S5/S8, le GTP-C est utilisé sur S11, S5/S8 et X2 si celle-ci existe (voir figure 2.10).

GTP User data tunneling

Sur le plan de données l'en-tête GTP est de taille variable, il comporte les *flags* PN, S et E qui indiquent la présence de champs optionnels. Le *flag* S est fixé à zéro par les nœuds eNodeB, SGW et PGW. Le *flag* PN est activé lors de relèves inter-technologiques (*vertical handover*). Le *flag* E est activé lorsque le paquet contient une autre extension, nous ne l'utilisons pas dans le cas d'une simple encapsulation de paquet. Il existe aussi les *flags* P et T, nous avons choisi de toujours fixer P à zéro (il permet d'activer le *PiggyBacking*). Le *flag* T permet de différentier le trafic de données du trafic de contrôle. En résumé, sur le plan de données l'en-tête du protocole GTP fait 8 octets (voir schéma 3.4).

2. La première version étant désignée par GTP first stage (GTPv0).

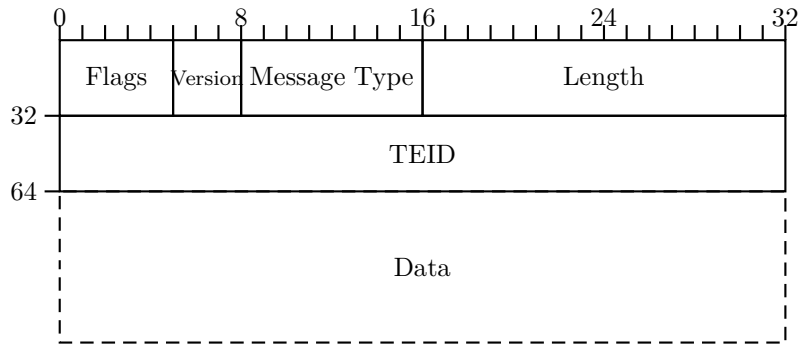


Figure 3.4 – En-tête GTP-U utilisé dans le plan de données (inspiré du document 29.281 [24])

Le champ TEID représenté dans le schéma 3.4 identifie le *bearer* à emprunter sur le « chemin » GTP ; un « chemin » correspondant à deux IP et deux ports UDP.

GTP-C

L'en-tête du protocole GTP-C contient une partie commune à tous les messages de contrôles, schématisée à la figure 3.5 ; cet en-tête fait 12 octets.

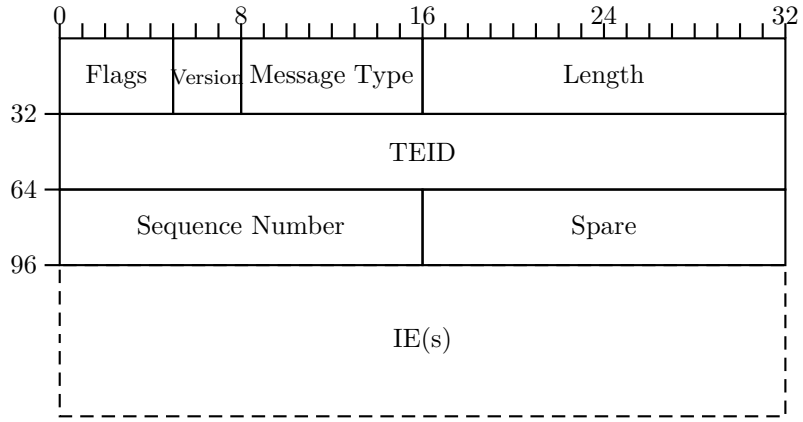


Figure 3.5 – En-tête GTP-C utilisé dans le plan de contrôle (inspiré du document 29.274 [22])

Les Information Element (IE) sont des informations utiles au message, par exemple dans un message de contrôle « Create Session Request », un IE « Radio Access Type (RAT) Type » et un ou plusieurs IE Fully-Qualified Tunnel Endpoint Identifier (F-TEID) seront inclus. Le format général d'un IE est représenté à la figure 3.6. Les différents messages de contrôle du protocole GTP-C et les IE sont trop nombreux pour être tous décrits ici, on peut cependant les trouver dans les documents 29.060 [16] et 29.274 [22].

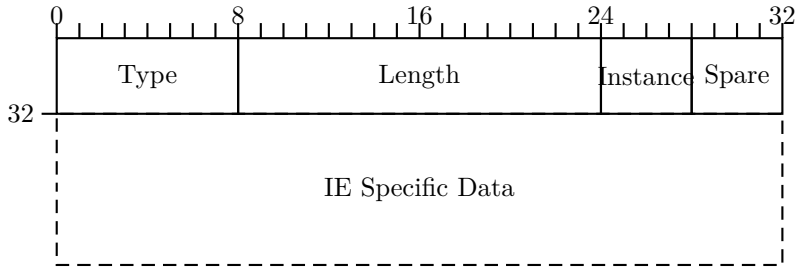


Figure 3.6 – En-tête classique d'un IE dans le protocole GTP-C (inspiré de 29.274 [22])

3.2 L'architecture 23.402

L'architecture 23.402 [9] est une évolution de la 23.401 [8], le principe des *bearers* reste le même mais l'interface S5/S8 a changé. Elle est représentée à la figure 1.1b, les couches de protocoles utilisées sur le chemin de données sont exposées à la figure 3.7 et un schéma représentatif des *bearers* est disponible à la figure 3.8. Les diagrammes de séquences des procédures de cette architecture auxquelles nous nous sommes intéressés, sont représentés dans l'annexe B.

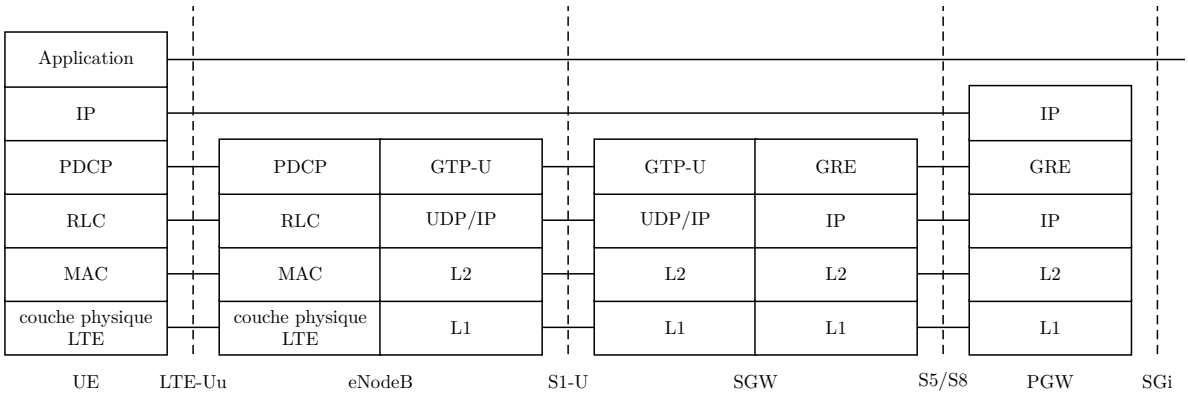


Figure 3.7 – Couches de protocoles du plan de données dans un EPS suivant l'architecture 23.402 [9]

Le protocole GRE permet de remplacer GTP-U pour encapsuler les paquets et différentier les *bearers*. Ces tunnels GRE sont eux même encapsulés dans un tunnel PMIPv6 entre le SGW faisant office de MAG et le PGW faisant office de LMA. L'architecture PMIPv6 a déjà été présentée à la section 2.2.3.

Cependant nous pouvons remarquer à la figure 3.7 que le protocole PMIPv6 a été modifié. Dans le RFC de PMIPv6 [65] il est décrit que le MAG doit être le routeur par défaut du MN (sous entendu pour la couche IP). Or la couche IP du UE est en relation avec celle du PGW et non celle du SGW, le routeur par défaut du UE est donc le PGW qui est le LMA dans l'architecture PMIPv6.

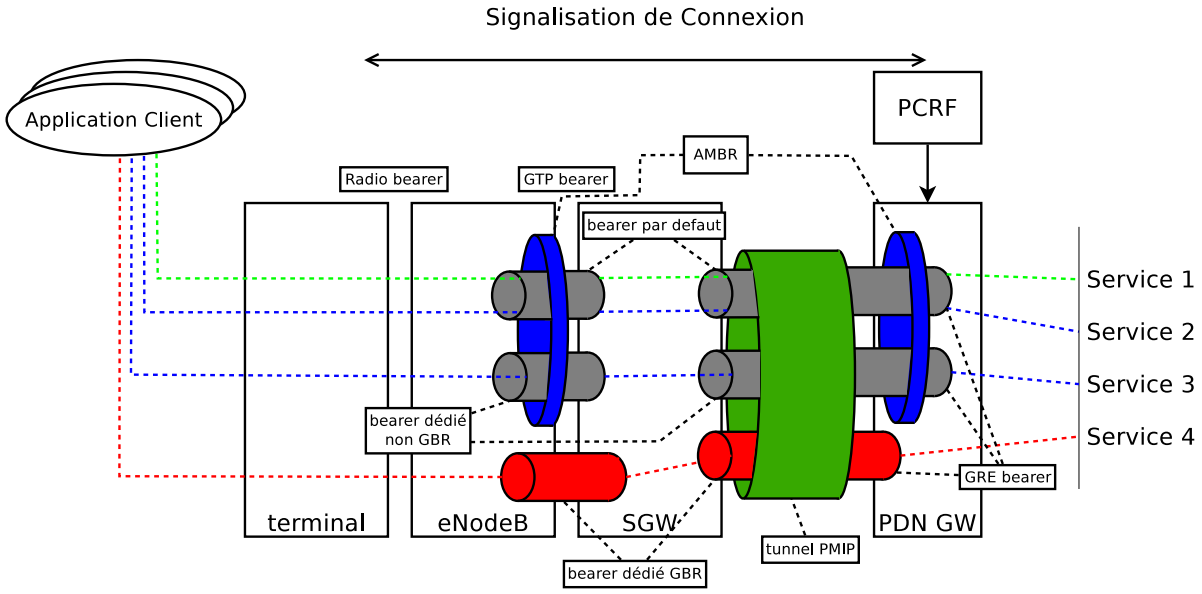


Figure 3.8 – Schéma représentatif des *bearers* GTP et GRE dans l'architecture 23.402

Comme décrit à la section 3.1.1, le protocole GTP se divise en deux branches, le GTP-U et le GTP-C. Le GTP-U qui encapsule les paquets de données a été remplacé par GRE entre le SGW et le PGW. Cependant GRE ne permet pas de transmettre des informations de contrôle. Les informations qui transitaient via GTP-C sur le lien S5/S8 ne peuvent plus emprunter le même chemin. Un nouveau lien a donc été créé entre le PCRF et le SGW, sur lequel est utilisé le protocole Diameter.

3.2.1 Diameter

Le protocole Diameter, décrit par Calhoun *et al.* [48], est utilisé dans le EPS pour transporter des informations de contrôle sur les liens S6a, Gx et Rx. Il est le successeur du protocole Remote Authentication Dial-In User Service (RADIUS), d'où il tire son nom (voir Olsson *et al.* [81]). Il a l'avantage d'être très facilement modifiable et évolutif. En effet, un en-tête Diameter contient une partie basique obligatoire, pouvant contenir plusieurs Attribute Value Pairs (AVP) que l'organisme utilisateur du protocole peut définir comme il veut du moment qu'il ait reçu un identifiant de l'IANA. Plusieurs messages, AVP, sont aussi définis de façon native dans Diameter. Nous avons représenté l'en-tête de base ainsi que le schéma d'un AVP dans les figures 3.9 et 3.10.

Un ensemble d'AVP définit une application, il existe par exemple l'application NAS définie par l'IETF, celles utilisées sur les liens Gx, S6a et Rx (cf. figure 2.10) développées par le consortium 3GPP, etc. Un schéma de la structure de Diameter est représenté à la figure 3.11. Les différents messages et AVP Diameter utilisés sur l'interface S6a sont décrits dans

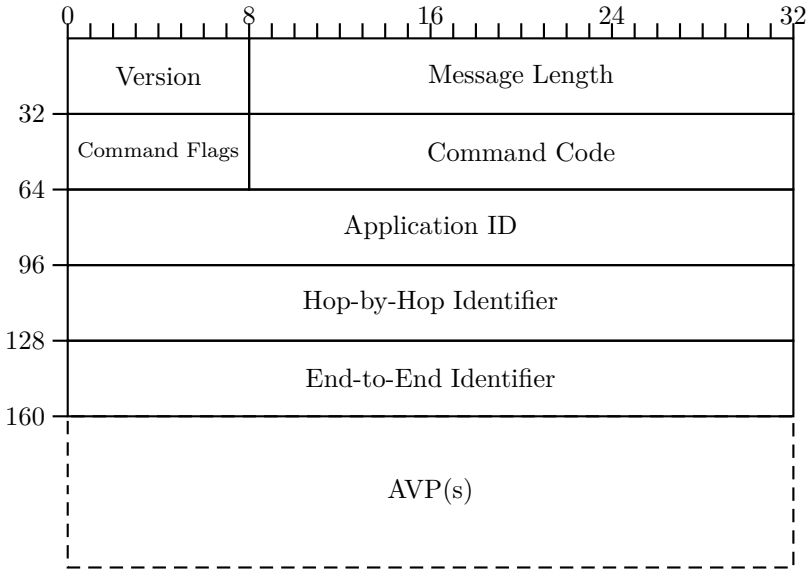


Figure 3.9 – En-tête du protocole Diameter (inspiré de Calhoun *et al.* [48])

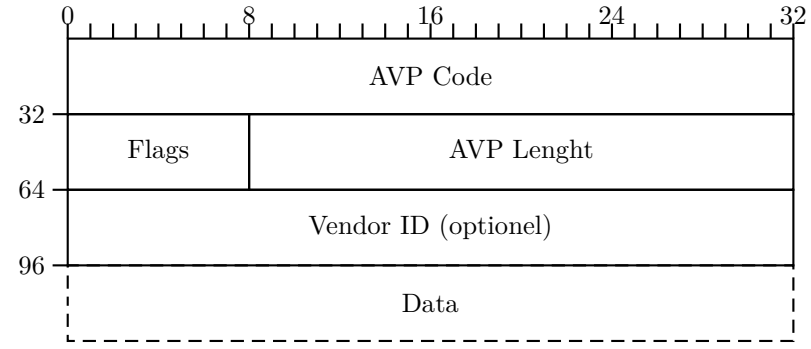


Figure 3.10 – En-tête classique d'un AVP dans le protocole Diameter (inspiré de Calhoun *et al.* [48])

le document 29.272 [21], ceux de l'interface Gx sont dans le document 29.212 [18] et ceux de Rx sont dans 29.214 [19].

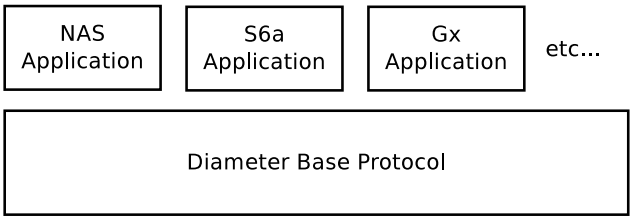


Figure 3.11 – Structure classique du protocole Diameter (inspiré de Olsson *et al.* [81])

Diameter se place au dessus de TCP ou Stream Control Transmission Protocol (SCTP) comme représenté à la figure 3.12. La taille de ces messages est décrite section 4.3.2.

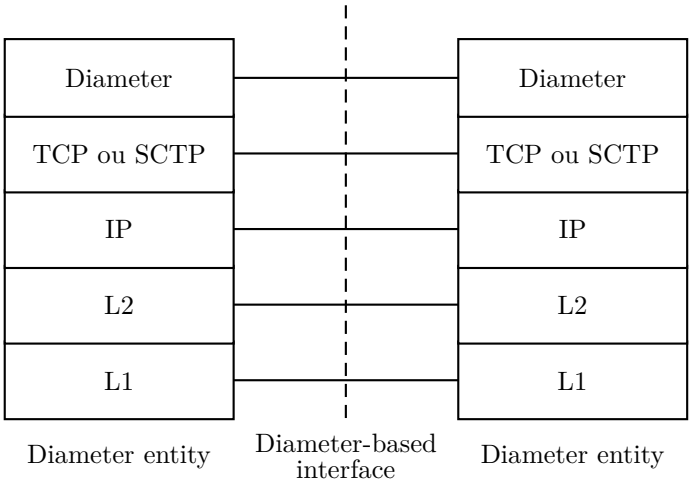


Figure 3.12 – Couches de protocoles utilisées sous Diameter

3.3 Proxy Mobile IPv6 multi-tunnel

Hui *et al.* [70] propose une évolution du protocole PMIPv6 pour permettre le déploiement de plusieurs tunnels entre un MAG et un LMA. Dans cette architecture, chaque service possède son propre tunnel, ce qui permet de différentier les trafics entre le MAG et le LMA et ainsi de faciliter la mise en place de techniques de QoS comme DiffServ. La figure 3.13 donne un exemple d'utilisation de PMIPv6 avec plusieurs tunnels par MN.

Pour déployer les tunnels, ils introduisent deux nouvelles options PMIPv6 : « Service Flow Identifier » et « Service Flow Description » représentées aux figures 3.14 et 3.15. Ils utilisent le protocole GRE pour différentier les tunnels, notamment grâce au champ *key* du BCE qu'ils proposent. Pour diffuser ces clefs, ils utilisent l'option *GRE Key* de PMIPv6 présentée à la figure 3.16.

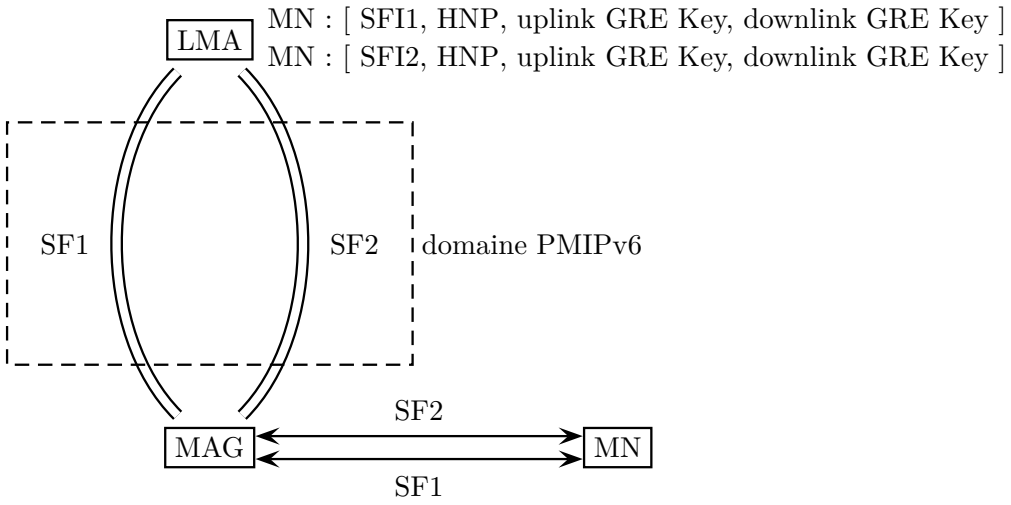


Figure 3.13 – Exemple d'utilisation de PMIPv6 avec plusieurs tunnels (inspiré de Hui *et al.* [70])

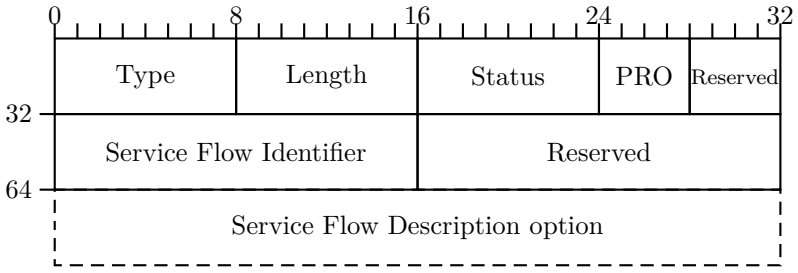


Figure 3.14 – Option *Service Flow Identifier* de PMIPv6 [70, sec 3.1]

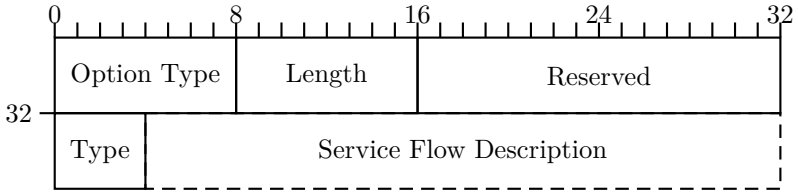


Figure 3.15 – Option *Service Flow Description* de PMIPv6 [70, sec 3.2]

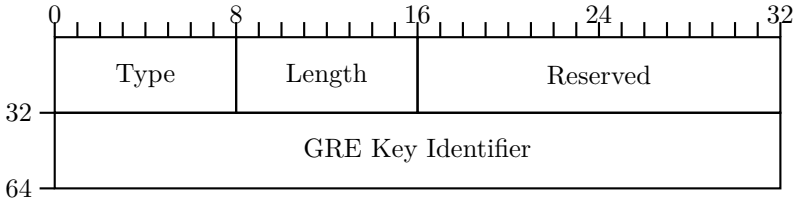


Figure 3.16 – Option GRE *Key* de PMIPv6 [78, sec 6.1]

Le MAG analyse le trafic et déclenche une procédure de création de tunnel lorsqu'un nouveau flux, qu'une nouvelle connexion, est détecté comme représenté à la figure 3.17.

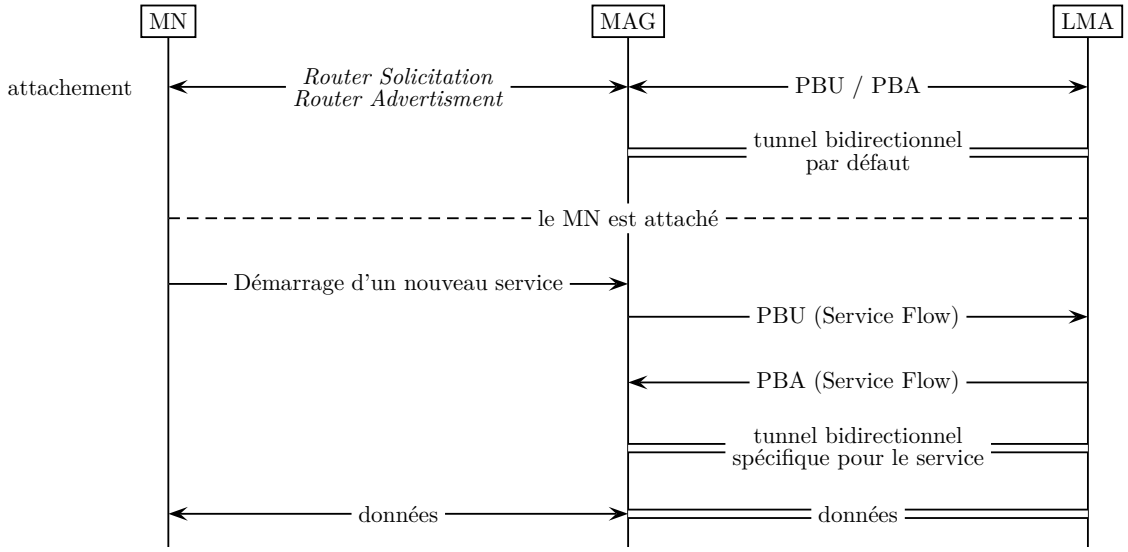


Figure 3.17 – Création d'un tunnel spécifique à un service dans l'architecture PMIPv6 (inspiré de Hui *et al.* [70, sec 2])

3.4 L'architecture 403

Dans le cadre de cette maîtrise nous avons essayé de maximiser l'utilisation du protocole IPv6, ce qui se traduit indirectement par une minimisation de l'utilisation de GTP et GRE. Nous nous sommes tout d'abord basés sur l'architecture 23.402 [9] et l'avons modifiée de façon à transférer la gestion des *bearers* à PMIPv6. Nous nous sommes donc servi du *draft* de Hui *et al.* [70] que nous avons modifié de façon à mimer le comportement des méthodes actuelles de gestion des *bearers*.

Dans ce *draft* chaque service possède son propre tunnel. Cependant un *bearer* (pour les architectures 23.401 [8], 23.402 [9]) peut être partagé entre plusieurs services, par exemple le *bearer* par défaut d'un UE est emprunté par tous les services qui n'ont pas de priorité particulière. Les *bearers* dédiés peuvent aussi faire transiter plusieurs trafics, et donc plusieurs services, requérant la même priorité.

Dans un premier temps, nous avons donc créé de nouvelles options PMIPv6 permettant de déployer des tunnels pour plusieurs services. Le protocole GTP utilise des TFT pour décrire le trafic passant par un *bearer* [4, sec 15.3]. Un TFT est un ensemble de Packet Filter (PFi) identifiés par des Packet Filter Identifier (PFID) (unique dans un même TFT) et ayant chacun une priorité appelée précédence. Les valeurs de précédence des PFi définissent l'ordre dans

lequel ils doivent être évalués. Un PFi est une combinaison des caractéristiques suivantes :

- adresse du CN et masque de sous réseau ;
- *Next Header* (étant donné notre choix de ne considérer que l'IPv6) ;
- intervalle de port, coté UE ;
- intervalle de port, coté CN ;
- Security Parameters Index (SPI) pour Internet Protocol Security (IPSec) ;
- *Traffic Class* IPv6 et masque (voir figure 3.18) ;
- *Flow Label* IPv6 (voir figure 3.18).

Les PFi permettent d'isoler les trafics des différents services du UE.

Le protocole GTP-C possède un IE dédié au transport de TFT [10, sec 10.5.6.12]. Il contient notamment :

- un « TFT operation code » permettant de décrire l'action à appliquer à la réception de ce IE. Il peut indiquer par exemple la création d'un nouveau TFT, le remplacement, la modification ou la suppression d'un ancien TFT, etc.
- le nombre de PFi qu'il inclut ;
- une liste de PFi contenant uniquement des PFID si l'action décrite plus haut correspond à une suppression de PFi d'un TFT existant. Sinon chaque élément de cette liste est composé de :
  - un PFID ;
  - la précédence du PFi ;
  - la taille de la description ;
  - la description du PFi (combinaison de caractéristiques décrites plus haut).

Nous avons donc reproduit ce schéma pour l'architecture PMIPv6 à quelques différences près. Nous avons tout d'abord créé une option Traffic Flow Template schématisée à la figure 3.19. Cette option contient un champ Procedure (PRO) correspondant au « operation code » GTP et un champ « Status » tiré de l'option *Service Flow Identifier* représenté à la figure 3.14. Nous n'y avons pas directement inclus les PFi comme dans GTP car cela nous permet de spécifier autant de PFi que nécessaires pour une seule option TFT. En effet pour GTP comme pour PMIPv6 la taille d'une option est décrite sur un octet, ce qui la limite à un maximum de 255 octets. Or la description d'un PFi peut prendre jusqu'à soixante octets (en IPv6), ce qui limite donc un IE TFT à quatre PFi dans le pire des cas. La solution adoptée par GTP dans le cas où l'option TFT est trop courte, est de répartir tous les PFi d'un TFT dans plusieurs IE TFT. Un TFT n'est donc pas forcément décrit par un seul IE. Nous avons préféré créer deux options différentes, une pour la description du TFT (ou plutôt de l'action à appliquer à un TFT) et une pour chaque PFi. L'option TFT est donc forcément suivie d'une ou plusieurs options PFi sur lesquelles elles s'appliquent. L'option PFi représentée à la figure

3.20 permet de décrire les caractéristiques d'un PFi, elle correspond à l'option *Service Flow Description* à laquelle nous avons ajouté le PFID de GTP (ou le Service Flow Identifier (SFI) de l'option Service Flow Identifier décrit par Hui *et al.* [70]) et sa valeur de précédence. Seul le champ PFID est obligatoire, en effet lors d'une suppression de PFi d'un TFT la description complète du PFi n'est pas nécessaire, seul son identifiant suffit.

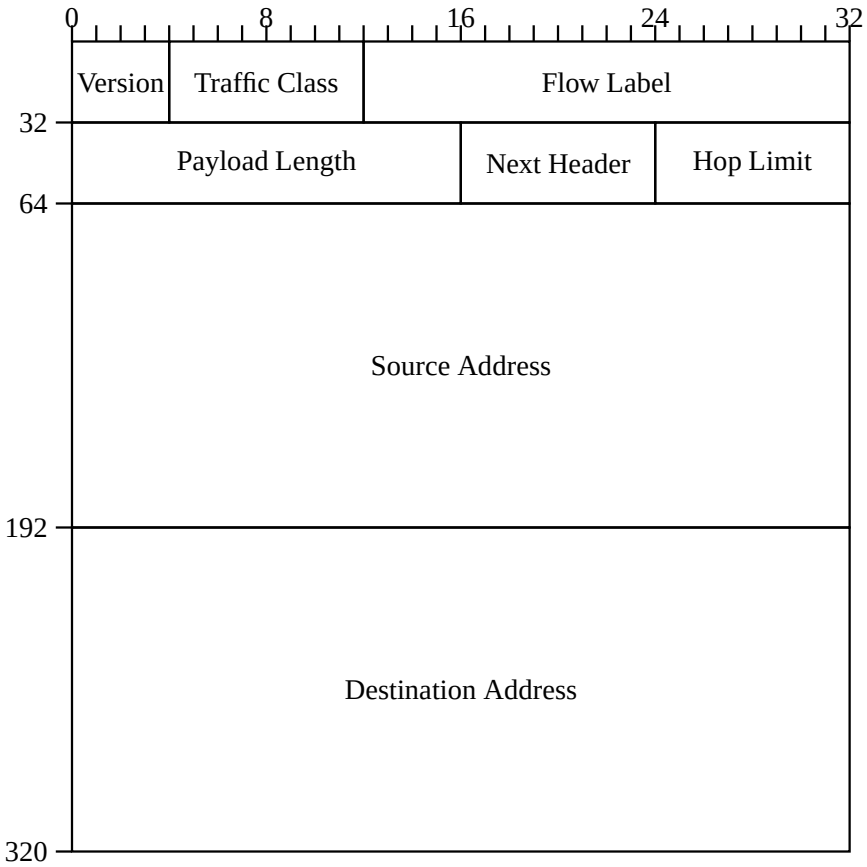


Figure 3.18 – En-tête IPv6 (inspiré de [56, sec 3])

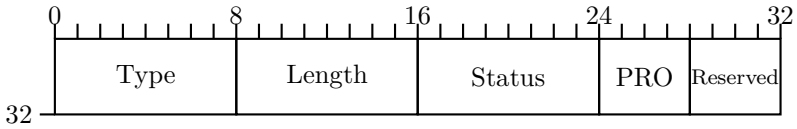


Figure 3.19 – Option Traffic Flow Template proposée pour PMIPv6

Hui *et al.* [70] utilise l'option GRE *Key* représentée à la figure 3.16, introduite par Muhanna *et al.* [78]. Cependant celle-ci a été prévue pour permettre l'identification de destination, par exemple dans le cas où le LMA est connecté à plusieurs réseaux ayant les mêmes plages d'adresses IP (e.g. deux réseaux Local Area Network (LAN) utilisant la plage 10.0.0.0/24). GRE peut aussi encapsuler un très grand nombre de protocoles ; mais cette

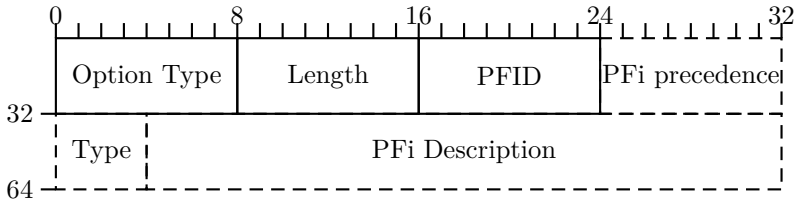


Figure 3.20 – Option Packet Filter proposée pour PMIPv6

fonctionnalité ne nous est pas utile étant donné que seul IPv6 transite dans les *bearers*, tel que présenté aux figures 3.1 et 3.7. D'après Gundavelli *et al.* [65, sec. 6.10.2] la méthode d'encapsulation préconisée pour PMIPv6 est *IPv6-In-IPv6* (décrit par Conta et Deering [53]) c'est à dire que le Protocol data unit (PDU) IP du paquet encapsulé devient le Service Data Unit (SDU) IP du paquet final. L'utilisation de GRE n'a rien d'obligatoire, nous avons donc utilisé une encapsulation *IPv6-In-IPv6* ; il fallait donc un nouveau moyen de distinguer les *bearers*.

L'en-tête IPv6 représenté à la figure 3.18 possède un champ *Flow Label* n'ayant pas encore d'utilité concrète, la seule contrainte est qu'il doit arriver au destinataire avec la valeur fixée par le nœud source (il peut être changé sur le trajet du moment qu'il soit remis à l'initial à la fin). Il pourrait être intéressant de l'utiliser pour distinguer les différents tunnels PMIPv6 entre un MAG et un LMA. Ce champ fait vingt bits, ce qui permet de distinguer environ un million de tunnels. Les entreprises développant le EPS considèrent qu'une ville comme Montréal (approximativement 1 700 000 habitants) nécessite approximativement 30 SGW. Chaque couple *bearer*/UE possède un QCI unique, et il existe neuf QCI prédéfinis par le consortium 3GPP. Si l'on suppose que tous les habitants de Montréal sont connectés au réseau LTE de façon équitablement répartie, et qu'ils utilisent chacun neuf *bearers* (un par QCI) cela fait environ 510 000 *bearers* par SGW. Limiter le nombre de *bearer* par SGW à un million n'est donc pas aberrant.

Nous avons donc décidé d'utiliser le champ *Flow Label* d'IPv6 pour différentier les tunnels entre un MAG et un LMA. C'est à dire que chaque tunnel possède un identifiant de 20 bits unique pour le MAG, celui-ci est inclus dans le champ *Flow Label* de l'en-tête des paquets IPv6 encapsulant. Pour le déployer entre le MAG et le LMA nous avons créé une nouvelle option *Flow Label* pour le protocole PMIPv6. Cette option est représentée à la figure 3.21. Étant donné que les *mobility header* doivent avoir une taille multiple de huit octets nous avons préféré ajouter un *padding* de 12 bits pour que cette option fasse huit octets.

Si la limitation du nombre de *bearers* à un million par SGW devient problématique, l'utilisation de GRE reste possible. Nous avons décidé de bâtir notre modèle en utilisant le *Flow Label* d'IPv6 comme identifiant des *bearers* cependant cet identifiant peut être facilement

remplacé par une clef GRE. Le modèle utilisera alors l'option GRE *Key* du *mobility header* PMIPv6 plutôt que l'option *Flow Label*.

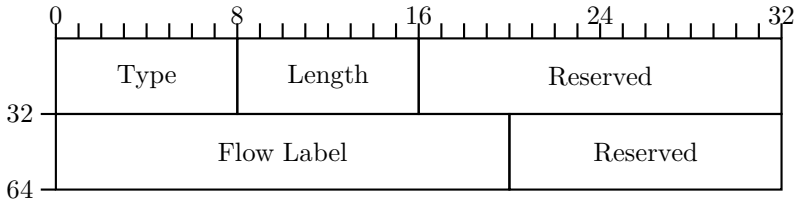


Figure 3.21 – Option *Flow Label* de PMIPv6

Pour supporter plusieurs tunnels entre un LMA et un MAG Hui *et al.* [70] étendent les Binding Update List Entry (BULE) et BCE avec un PFID et une description de PFi. Étant donné les transformations apportées, nous avons ajouté aux BULE du MAG et aux BCE du LMA :

- un *Flow label* ;
- une liste de PFi dont chacune des entrées contient :
  - un PFID ;
  - une valeur de « précédence » ;
  - un identifiant décrivant le type de description du PFi, trois de ces types sont prédéfinis dans le document 23.060 [4, sec 15.3.2.0]. Hui *et al.* [70] utilisent les mêmes types de base que ceux définis par 3GPP.
  - une description du PFi (combinaison de caractéristiques).

Sur le MAG le *Flow Label* identifie de façon unique le tunnel, sur le LMA en revanche le couple HNP/*Flow Label* est l'identifiant.

Nous avons défini les modifications à apporter au protocole PMIPv6 pour permettre un déploiement de tunnel mimant les *bearers* présents dans les architectures 23.401 [8] et 23.402 [9]. Nous allons maintenant préciser les modifications à apporter à ces architectures pour utiliser ces nouvelles fonctionnalités.

3.4.1 Séquence

Nous nous sommes basés sur l'architecture 23.402 [9] car elle inclut déjà une portion du réseau cœur utilisant une version de PMIPv6 remaniée. Dans un premier temps nous n'avons modifié que l'interface S5/S8 (SGW-PGW) qui utilise déjà PMIPv6, et avons appelé cette nouvelle architecture 403. Nous avons transformé le SGW en un routeur IP pour qu'il apparaisse comme un vrai MAG. Les créations/modifications/destructions de *bearers* sont maintenant gérées au niveau PMIPv6. Nous avons supprimé GRE car il ne nous est désormais

plus utile. Les couches de protocoles du plan de données de l'architecture 403 sont représentées à la figure 3.22.

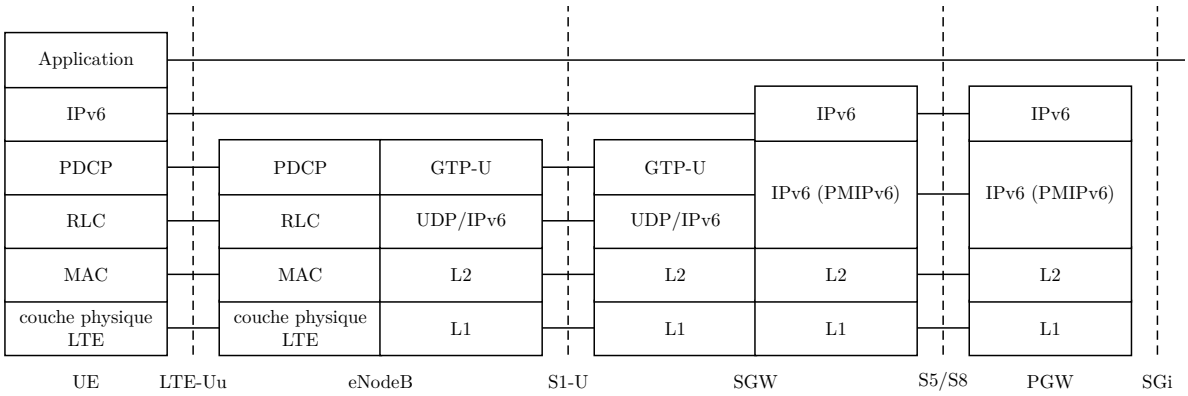


Figure 3.22 – Couches de protocoles du plan de données dans un EPS suivant l'architecture 403 proposée

Il nous a aussi fallu modifier les diagrammes de séquences des différentes procédures de l'architecture 23.402 [9] impliquant les *bearers* (voir Annexe B). Ces nouveaux diagrammes de séquences sont représentés dans cette section et dans l'annexe C. Dans les procédures de complétion de *bearer* (ajout d'un service), représentées aux figures B.1 à B.4, nous avons ajouté un échange PBU/PBA au niveau PMIPv6 au moment où le tunnel GRE est modifié dans l'architecture 23.402 [9]. Les diagrammes de séquences de ces nouvelles procédures sont représentés aux figures 3.23 et C.1 à C.3. La bande passante des liens, la réservation et la limitation du débit des trafics sont encore gérées par le SGW, le PGW et le eNodeB et ne peuvent pas être administrées directement par PMIPv6. Nous avons donc gardé les autres messages de contrôle de ces procédures pour déployer les informations comme les politiques de facturations, les limitations à appliquer aux trafics, les réservations de bande passante, etc.

Le LMA/PGW attend de recevoir les messages PBU et *Policy and Charging Rules Provision* respectivement du MAG/SGW et du PCRF avant d'envoyer le PBA et l'accusé de réception. De cette façon, nous nous assurons qu'un tunnel n'est créé que lorsque toutes les informations le concernant sont arrivées sur le PGW.

La procédure de connexion représentée à la figure B.5 n'a pas été modifiée en apparence, c'est à dire que les mêmes messages sont envoyés. Cependant, nous avons ajouté au PBU de connexion une option *Flow Label* pour spécifier l'identifiant du *bearer* par défaut.

Les procédures de création de *bearer* représentées aux figures B.6 et B.7 ont été modifiées de la même manière que celles de complétions (voir figures C.4 et C.5). Les procédures de destruction et de réduction de *bearer*, représentées aux figures B.8 à B.13, ont été modifiées

de façon à inclure un échange de PBU/PBA au moment où le tunnel GRE est modifié dans l'architecture 23.402 [9] (voir figures C.6 et C.11). La réduction ou destruction effective du *bearer* se fait au début de la procédure car elle n'a pas lieu d'être rejetée contrairement à une création ou une complétion. Le PGW attend toujours de recevoir les informations du SGW et du PCRF pour répondre.

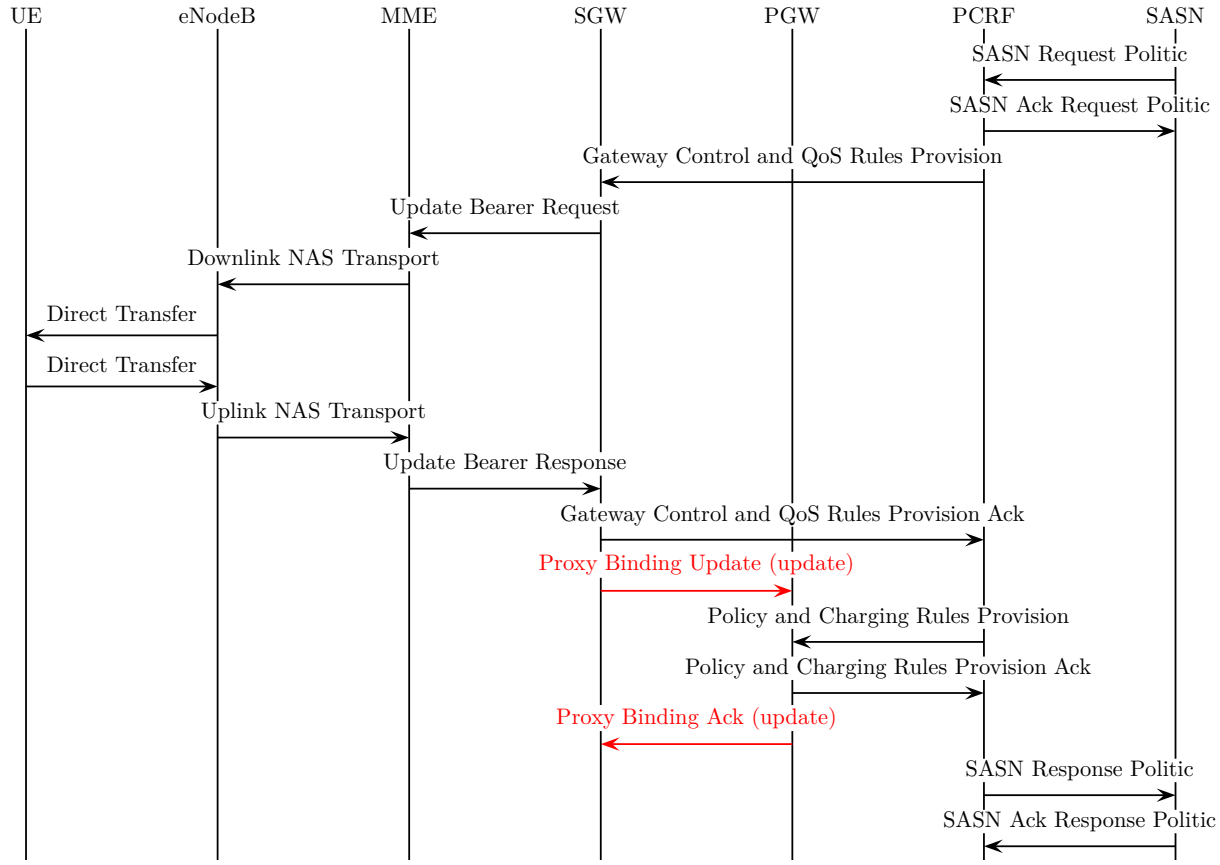


Figure 3.23 – Complétion de *bearer* dédié non GBR avec UE QoS *unaware*

Les procédures de relève sans changement de SGW avec ou sans interface X2 (liant deux eNodeB) ne changent pas. En effet aucun modification n'y est effectuée entre le SGW et le PGW. En revanche, si le eNodeB de destination ne possède pas assez de bande passante restante pour accueillir tous les *bearers* du UE, il y a destruction des *bearers* de plus faible ARP. Dans ce cas nous avons inclus une destruction de tunnel PMIPv6 comme lors d'une destruction de *bearer* normal (cf. figure B.14).

Dans le cas de relève avec changement de SGW représentée aux figures B.15 et B.16 nous avons modifié les PBU/PBA utilisés pour créer la connexion entre le nouveau SGW et le PGW et pour détruire l'ancienne (voir figures C.13 et C.14). Dans notre architecture le PBU de connexion est complété d'options TFT, PFi et *Flow Label* nécessaires à la création de tous

les anciens tunnels qui n'ont pas été détruits par la relève. Il y a donc une option TFT et une option *Flow Label* par tunnel, chaque TFT étant suivi d'une liste de PFi (incluant les champs optionnels) décrivant tous les services passant par le tunnel correspondant.

Pour détruire la connexion entre l'ancien SGW et le PGW nous avons ajouté un échange de PBU/PBA de déconnexion. Nous n'avons pas complété ces messages d'option particulière et avons utilisé ceux décrits par Gundavelli *et al.* [65]. En effet le but de ces messages est d'informer le LMA/PGW que le UE n'est plus rattaché à l'ancien SGW, il n'y a pas de notion de *bearer*.

Comme pour les architectures 23.401 [8] et 23.402 [9] nous avons créé un simulateur mimant le comportement de 403. Cependant nous nous sommes demandés si les changements apportés pouvaient être étendus jusqu'au eNodeB. Nous avons donc imaginé une nouvelle architecture appelé 404.

3.5 L'architecture 404

Dans cette architecture (évolution de la 403) nous avons étendu le domaine PMIPv6 aux eNodeB. Pour correspondre au fonctionnement de GTP nous avons installé un LMA sur le SGW (en plus du MAG déjà présent) et un MAG sur chaque eNodeB. Les eNodeB deviennent donc des routeurs IPv6. Les couches de protocoles sur le chemin des paquets de données sont représentées à la figure 3.24. Nous avons utilisé les mêmes options PMIPv6 que celles utilisées dans l'architecture 403.

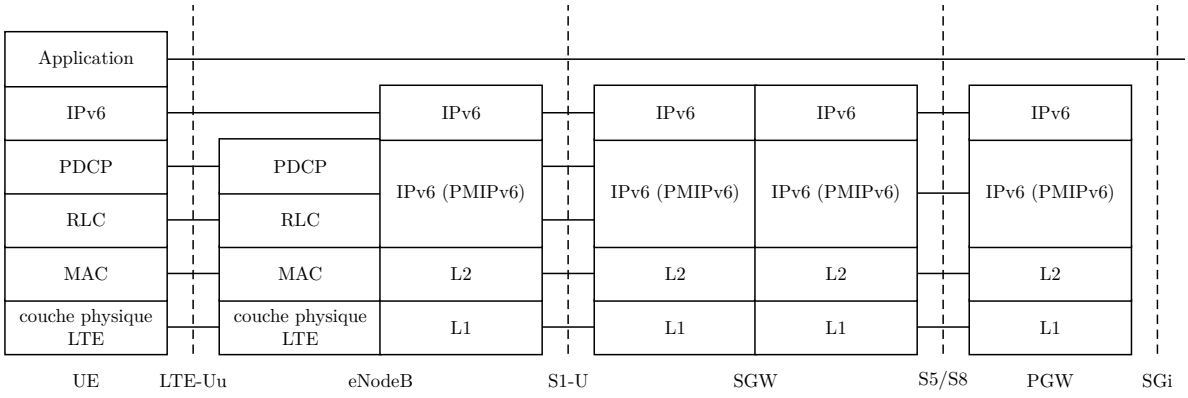


Figure 3.24 – Couches de protocoles du plan de données dans un EPS suivant l'architecture 404 proposée

Pour les procédures de complétion[3] de *bearer*, nous avons ajouté un échange de PBU/ PBA entre le eNodeB et le SGW pour modifier le tunnel au niveau PMIPv6. Les nouveaux

3. Ajout d'un service à un *bearer* déjà existant.

diagrammes de séquences sont représentés aux figures 3.25 et D.1 à D.3. Comme pour l'architecture 403 nous avons fait en sorte que la modification du tunnel PMIPv6 se fasse au même moment que celle du *bearer* GTP dans l'architecture 23.402 [9]. Le SGW attend de recevoir le PBU et le *Update Bearer Response* avant de continuer la procédure, de cette façon nous nous assurons que le *bearer* est modifié si et seulement si toutes les informations le concernant sont disponibles sur le SGW (MBR, type, TFT, ARP, etc.).

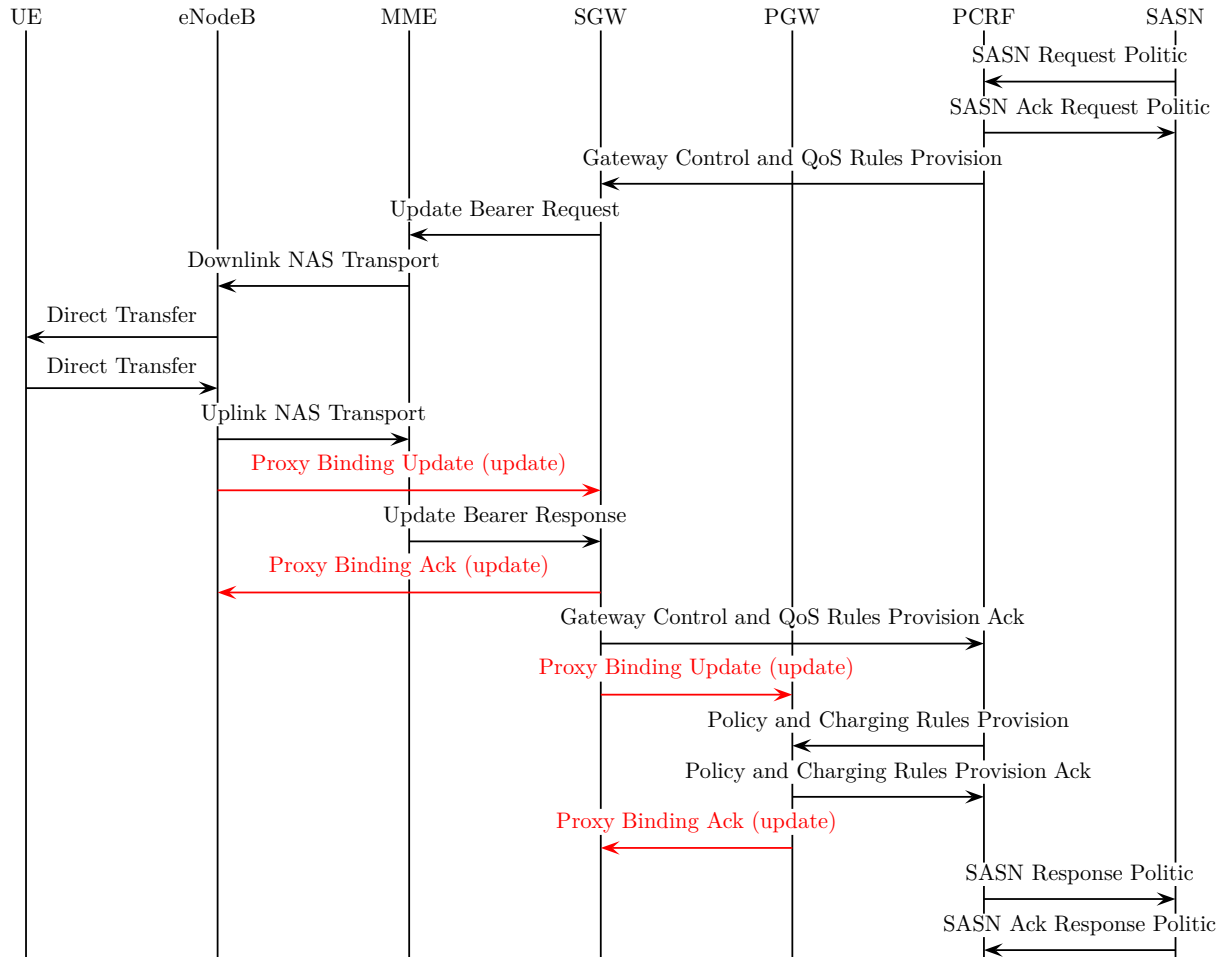


Figure 3.25 – Complétion de *bearer* dédié non GBR avec UE QoS *unaware*

De la même manière les diagrammes de séquences des procédures de création de *bearer* dédié pour les UE QoS *aware* et *unaware* ont été modifiés et sont représentés aux figures D.5 et D.6. Le SGW attend la réception du PBU et du message *Create Bearer Response* avant de continuer. Pour la création et la complétion de *bearer*, l'action (création ou modification effective) est effectuée en fin de procédure car elle est susceptible d'être rejetée du fait de la réservation de bande passante.

Pour la connexion du UE, nous avons ajouté un échange PBU/PBA à l'activation finale du *bearer* (voir figure D.4). Tout comme précédemment, le SGW attend la réception du PBU

et du message *Create Bearer Response* avant de continuer. Les procédures de destruction de *bearer* représentées aux figures 3.26 et D.7, ont été modifiées pour supprimer le tunnel PMIPv6 au moment où le eNodeB envoie l'accusé de réception signifiant la destruction du *bearer* radio. La suppression ne se fait pas à la réception de la requête mais de la réponse, en effet dans le document 23.401 [8] il est indiqué que le SGW détruit ses *bearers* à la réception du *Delete Bearer Response* (dans le cas où l'on utilise seulement GTP). Cependant, le *bearer* entre le SGW et le PGW est supprimé au début de la procédure car dans le 23.402 [9] il est indiqué que le PGW peut être notifié de la destruction du *bearer* en même temps que le SGW. Nous avons donc choisi de paralléliser la notification du SGW et du PGW, ce qui implique la suppression du *bearer* sur le PGW en début de procédure. Pour suivre cette logique nous avons choisi de détruire le tunnel SGW-PGW avant que le SGW ne reçoive le message *Delete Bearer Response* du MME.

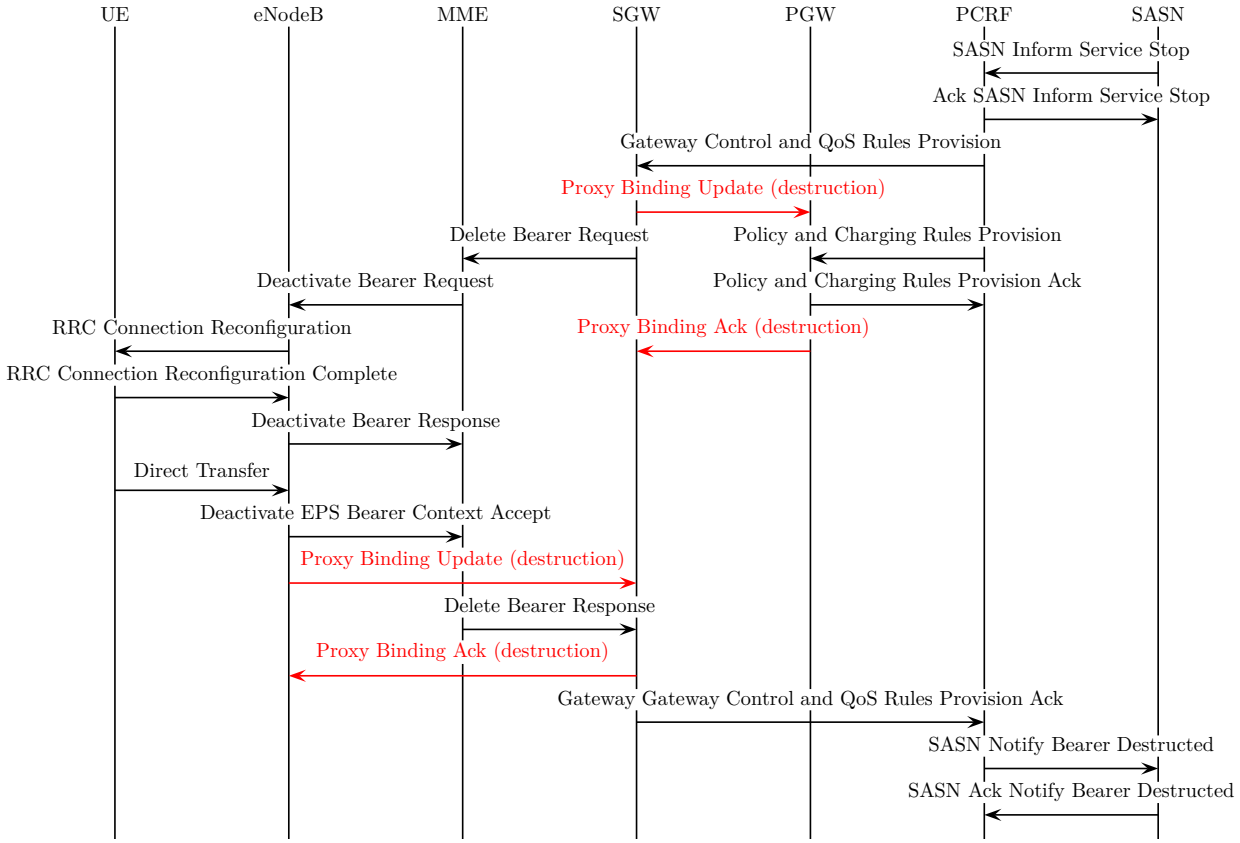


Figure 3.26 – Destruction de *bearer* dédié avec UE QoS *unaware*

Les procédures de réduction[4] de *bearer*, représentées aux figures D.8 à D.11, ont été modifiées de la même façon que celle de destruction. La différence est que les PBU/PBA ne

4. Suppression d'un service, PFi d'un *bearer*.

contiennent pas une demande de destruction [5] mais de modification de TFT. Ils contiennent donc des options TFT avec un champ PRO fixé à *Delete packet filters from existing TFT*[6] et des options PFi désignant les filtres à supprimer.

Concernant la procédure de relève sans interface X2 et sans changement de SGW représentée à la figure 3.27, plusieurs changements ont été apportés. Dans cette figure, les messages en rouge, incluant flèche et texte, sont ceux ajoutés par rapport à la procédure de l'architecture 23.402 [9] représentée à la figure B.14. Les messages dont seule la flèche est en rouge sont ceux qui ont été déplacés.

Nous avons désactivé la possibilité de faire une redirection indirecte car PMIPv6 ne le supporte pas. Le eNodeB source n'aurait pas été en mesure de délivrer les paquets arrivés après la resynchronisation du UE ce qui aurait pu engendrer des pertes de paquets. Nous avons donc décidé de changer très tôt l'adresse de destination des *bearers* du UE sur le SGW, avant même la resynchronisation du UE avec le eNodeB de destination. Les paquets arrivant à cet eNodeB sont mis en cache comme c'est le cas dans les architectures précédentes ; une fois que le UE est connecté au nouveau eNodeB, le cache est vidé et le paquet envoyé au UE.

Pour déplacer le changement d'adresse de destination des *bearers* nous avons décalé l'échange de *Modify Bearer Request/Response* entre le MME et le SGW au moment où le MME reçoit le *Handover Request Ack* du eNodeB de destination. Dans le cas où le *Handover Request Ack* est négatif le SGW n'est pas notifié et la relève n'est pas faite.

Au moment où le MME reçoit le *Handover Request Ack* celui-ci prend connaissance des *bearers* à conserver et à rejeter ; le eNodeB de destination a connaissance de la relève et a préparé un cache pour recevoir les paquets du UE. Il est donc possible de changer le chemin des paquets *downlink* du UE sur le SGW à ce moment. En même temps une procédure PMIPv6 de création des nouveaux tunnels entre le eNodeB destination et le SGW, est lancée par le eNodeB. Le SGW attend d'avoir reçu le *Modify Bearer Request* et le PBU pour changer les adresses des *bearers*, de cette façon nous nous assurons que le SGW possède toutes les informations concernant les *bearers*. Nous avons aussi choisi de déplacer la procédure de destruction de *bearer* avec le message *Modify Bearer Request* car dans la documentation de l'architecture 23.401 [8], son lancement accompagne ce message.

Lorsque le UE se désynchronise du eNodeB source, cet eNodeB envoie un PBU de déconnexion au SGW de façon à entériner l'information de déplacement du UE du précédent PBU envoyé par le eNodeB destination. Une fois que le UE est synchronisé avec le eNodeB destination, comme pour les architectures 23.401 [8] et 23.402 [9], un compte à rebours est lancé à la fin duquel toutes les ressources associées au UE sont libérées sur le eNodeB source.

5. Une demande de destruction de *bearer* ne contient aucune option TFT, l'option *Flow Label* est suffisante.
6. *Delete packet filters from existing TFT* = 5 , si l'on prend la norme du document 24.008 [10].

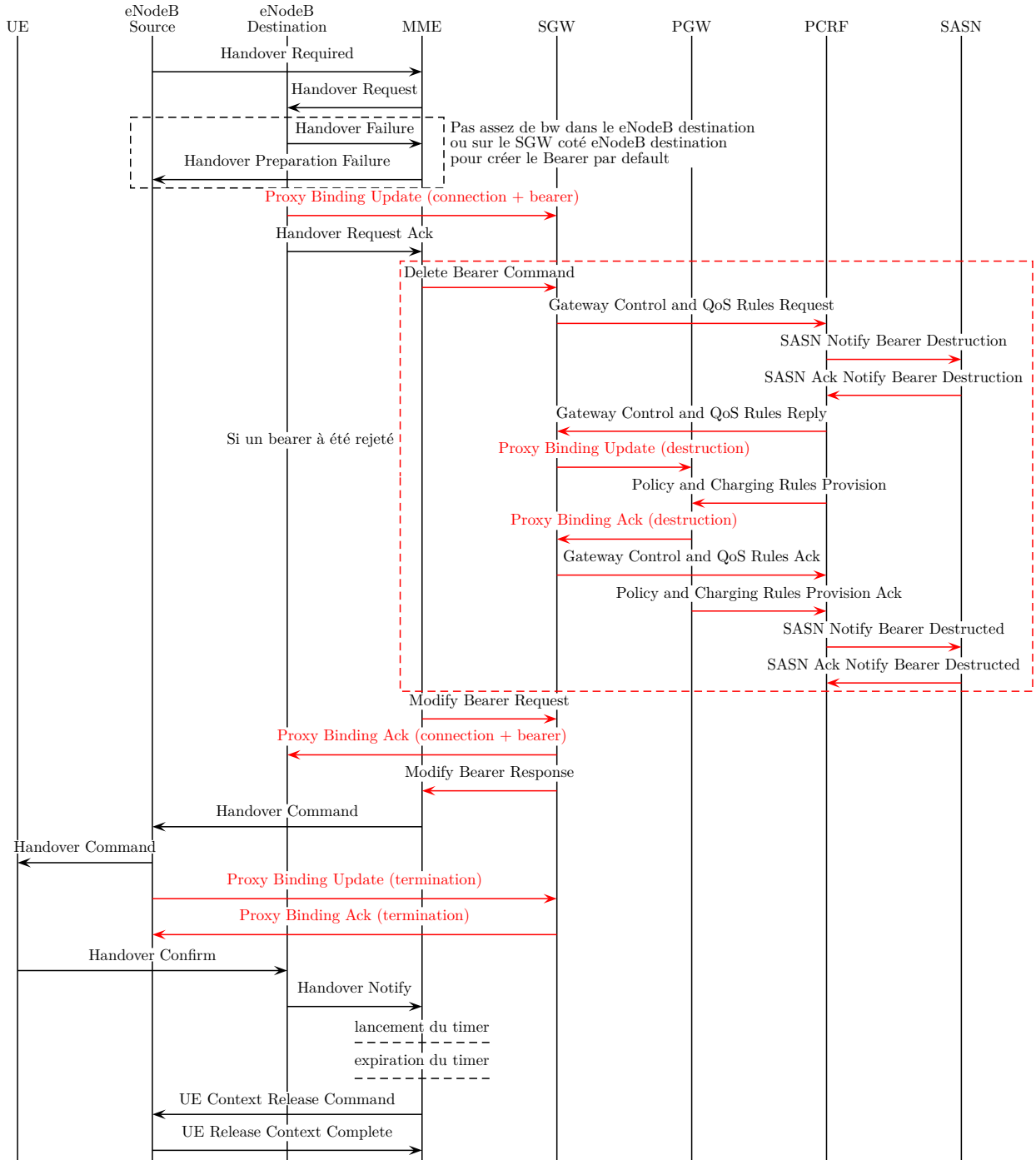


Figure 3.27 – S1 Handover

Les *bearers* qui n'ont pas pu être portés sur le nouvel eNodeB sont supprimés grâce à la même procédure que celle utilisée dans l'architecture 403.

Pour la procédure de relève sans interface X2 avec changement de SGW représentée à la figure 3.28 nous avons ajouté un échange de PBU/PBA entre le SGW et le PGW lors du détachement du UE de l'ancien eNodeB et donc de l'ancien SGW. Le message *Modify Bearer Request* est aussi suivi d'une procédure de connexion/création de *bearers*, entre le SGW et le PGW comme c'est le cas lors de ce type de relève dans l'architecture 23.402 [9].

En ce qui concerne la procédure de relève avec interface X2 sans changement de SGW représentée à la figure D.12 la présence de l'interface X2 nous autorise à changer le *Proxy* CoA [7] du UE sur le SGW après que celui-ci ait changé de eNodeB. En effet l'interface X2 permet de rediriger les paquets directement du eNodeB source vers le eNodeB destination, où ils seront mis en cache en attendant que le UE se resynchronise. Nous avons donc préféré conserver l'ordre des messages de l'architecture 23.402 [9], ce qui veut dire que les paquets ne sont redirigés vers le eNodeB destination qu'au moment où le UE se détache du eNodeB source.

Du fait de l'ajout de PMIPv6 entre le eNodeB et le SGW, nous avons ajouté des échanges de messages PBU/PBA. Comme pour la procédure de relève précédente nous y avons ajouté un échange accompagnant le message *Modify Bearer Request* pour changer le *Proxy* CoA des *bearers* du UE sur le SGW. Un autre, lancé par le eNodeB source en toute fin de procédure vient entériner ce changement.

La procédure de relève avec interface X2 et changement de SGW est représentée à la figure D.13. Par rapport à celle sans changement de SGW, nous avons ajouté un échange de PBU/PBA de déconnexion entre le SGW source et le PGW au moment où le SGW reçoit le PBU de déconnexion du eNodeB source. De cette façon, les tunnels entre ce eNodeB, le SGW et le PGW sont supprimés l'un après l'autre. Les messages PBU et PBA entre le SGW destination et le PGW ont été modifiés pour contenir les informations nécessaires à la création des *bearers*, c'est à dire une option TFT par *bearer* et une liste de PFi. Nous n'avons pas rattaché l'envoi du PBU de déconnexion de l'ancien SGW à la réception du message *Delete Session Request* comme c'est le cas dans l'architecture 403. Nous avons préféré l'attacher à la réception du PBU de déconnexion du eNodeB source pour rester au niveau PMIPv6. Ce que nous ne pouvions pas faire dans l'architecture 403 étant donné que le lien S1-U était géré par GTP.

Nous avons implémenté les architectures décrites précédemment dans le simulateur NS-2 pour pouvoir les comparer et les avons opposées en fonction des performances du réseau au

7. Adresse de destination des *bearers*.

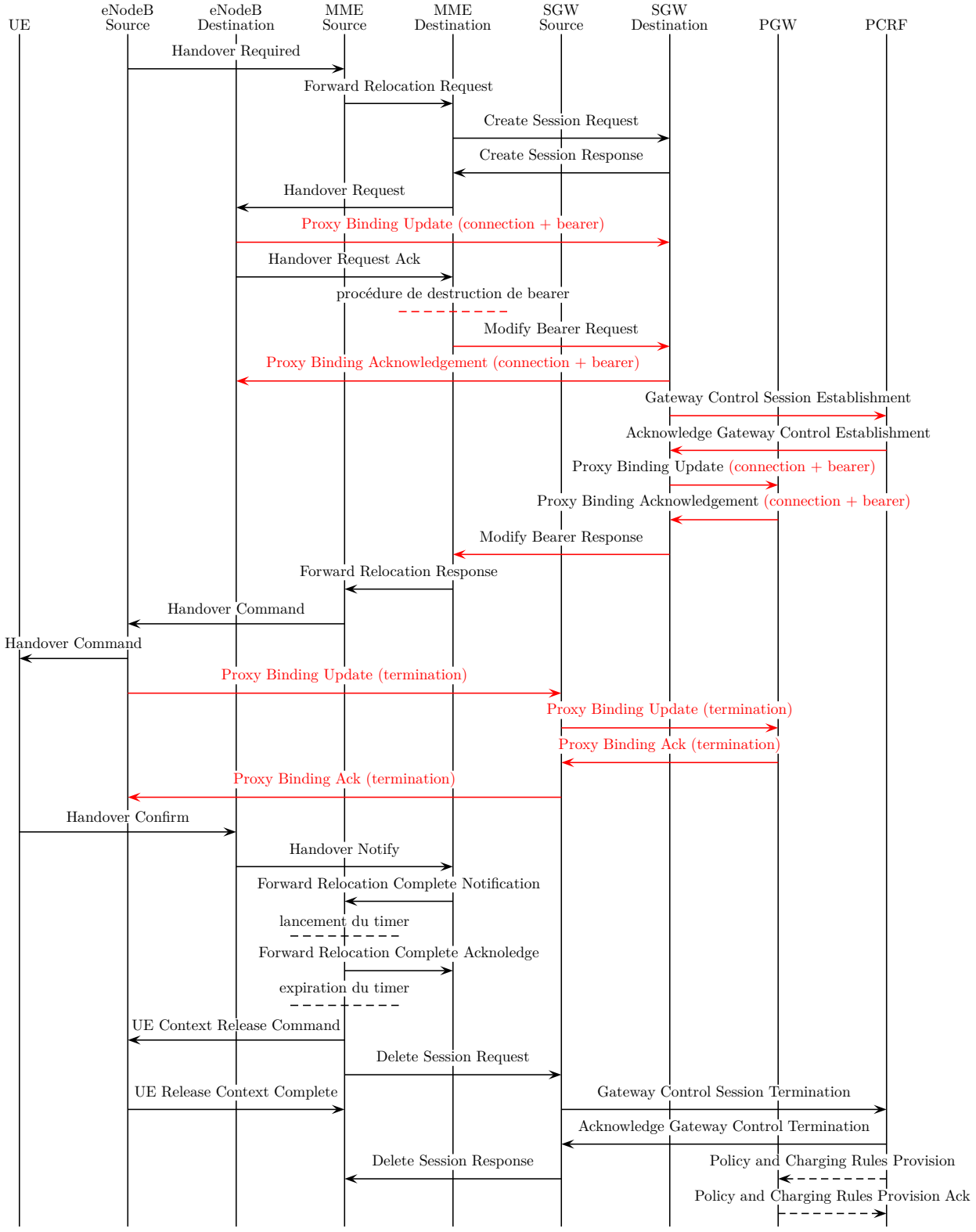


Figure 3.28 – S1 Handover avec changement de SGW

niveau du UE. Pour ce faire nous avons recherché quelles seraient les meilleures métriques à utiliser.

3.6 Performance du réseau

Pour comparer efficacement les performances des réseaux simulés nous avons besoin de plusieurs métriques. Paxson *et al.* [84] explique les bases d'une bonne métrique réseau, ainsi que quelques caractéristiques à respecter. Par exemple, pour une unité utilisant les bits, les préfixes sont en base dix et non en base deux, e.g. 1 kbits = 1000 bits et non 1024 bits comme cela aurait pu être le cas dans le domaine des unités de stockages.

Shalunov et Swany [94] ont proposés, par le biais du *working group* IP Performance Metrics Working Group (IPPM), plusieurs métriques destinées à l'utilisateur final, ces métriques se veulent simples à comprendre et facilement implémentables. Parmi ces métriques on retrouve :

- le délai médian, défini plus particulièrement par Almes *et al.* [38] ;
- le taux de perte de paquets, aussi défini par Almes *et al.* [39] ;
- l'étalement du délai (*delai spread*), tiré d'un calcul sur le délai médian, couramment apparenté à la gigue (*jitter*).

Dans les premières versions de notre logiciel, nous avons utilisé ces métriques mais beaucoup de personnes du LARIM et d'Ericsson se sont étonnés de certaines définitions, entre autre pour la gigue.

Certains problèmes se posaient avec les métriques proposées par Shalunov et Swany [94]. En effet ces métriques n'ont pas été conçues pour s'appliquer à des types de trafics réels comme nous en avons dans notre simulateur. Beaucoup nécessitent d'avoir une source de paquets à débit constant, les paquets ayant tous la même taille, ce qui pose problème, e.g. pour la vidéo.

Lorsque nous avons décidé de ré-implémenter le système de traces de NS-2 comme décrit à la section 4.2.6, nous avons décidé de conserver les types de métriques décrites précédemment, cependant nous en avons choisi au calcul plus simple :

- le délai moyen sur un temps donné, sans considération de la taille des paquets.

$$D(t,T) = \sum_{i=i_0}^{i_n} d(i) \tag{3.1}$$

Où $D(t,T)$ est le délai moyen au temps t sur la période T ; $d(i)$ est le délai du paquet i ; i_0 est le premier paquet passé après le temps $t-T$ et i_n est le dernier paquet passé avant le temps t.

- le taux de perte de paquets sur un intervalle de temps donné (voir section 3.6.1).

$$P(t,T) = \frac{n_p(t,T)}{n(t,T)} \tag{3.2}$$

  Où $P(t,T)$ est le taux de perte de paquets au temps t sur la période T ; $n_p(t,T)$ le nombre de paquets passés entre t et $t-T$ et perdus ; $n(t,T)$ le nombre de paquets passés entre t et $t-T$.
- la gigue, comme défini par Schulzrinne *et al.* [93] ; nous ne calculons cette métrique que pour les trafics temps réel (voir section 3.6.2).
- le débit (*throughput*).

$$Throughput(t,T) = \frac{\sum_{i=i_0}^{i_n} s(i)}{T} \tag{3.3}$$

  Où $Throughput(t,T)$ est le débit au temps t sur la période T et $s(i)$ est la taille totale du paquet i.
- le débit au niveau application (*goodput*) ; seulement pour les applications utilisant le protocole TCP, comme les protocoles HyperText Transfer Protocol (HTTP) ou FTP dans notre simulation. Le débit au niveau application au temps t est égal à la somme des bits arrivés à la couche application du récepteur entre les temps $t-T$ et t, divisée par T.

3.6.1 Le taux de perte de paquets

Nous utilisons le taux de perte de paquets en suivant la définition de Almes *et al.* [39, sec 2.4], c'est à dire que nous ne prenons pas de limite de délai maximum pour considérer un paquet comme perdu. Un paquet est considéré comme perdu si et seulement s'il est rejeté quelque part dans le réseau.

Dans nos simulations nous avons calculé ce taux qu'entre le SASN et les UE. Nous ne voulions pas que des paquets perdus à l'extérieur du réseau soient comptabilisés.

3.6.2 La gigue

La gigue est une métrique assez spéciale dans le sens où l'on trouve énormément de définitions différentes dans les documents de l'IETF. Pour Demichelis et Chimento [58], la gigue est appelée IPDV et correspond à la différence entre les *one-way-delay* (définie par Almes *et al.* [38]) des paquets sélectionnés. Elle est définie par Shalunov et Swany [94] comme étant la différence entre le troisième et premier quartile des *one-way-delay* des paquets sélectionnés.

Nous avons considéré que la gigue n'était une métrique pertinente que dans le cas de

trafics temps réel. Nous avons décidé d'utiliser la gigue définie par Schulzrinne *et al.* [93, sec 6.4.1] qui est utilisée par le protocole Real Time Protocol (RTP), spécialement adapté aux trafics temps réel. Pour un paquet i on définit une valeur de gigue $J(i)$ égale à :

$$J(i) = J(i-1) + \frac{(|D(i-1,i)| - J(i-1))}{16} \tag{3.4}$$

où $D(i-1,i)$ est le délai, comme défini par Almes *et al.* [38], entre le paquet i et le paquet $i-1$ précédent. Soit :

$$D(i,j) = (R_j - S_j) - (R_i - S_i) \tag{3.5}$$

où R_i et R_j sont les temps auxquels sont arrivés les paquets i et j, et S_i et S_j les temps à l'envoi de ces mêmes paquets.

CHAPITRE 4

ÉVALUATION ET ANALYSE DES PERFORMANCES DES DIFFÉRENTES ARCHITECTURES

Maintenant que nous avons détaillé les architectures proposées et déjà existantes, nous allons analyser leurs performances respectives. Pour ce faire nous devons aussi expliquer le fonctionnement de nos simulateurs et des scénarios utilisés.

Dans ce chapitre nous allons donc commencer par détailler notre premier scénario, peu chargé, et nous analyserons les résultats obtenus sur les quatre architectures 23.401 [8], 23.402 [9], 403 et 404. Nous ferons ensuite de même pour notre deuxième scénario dans lequel la charge du réseau est plus importante. Dans les deux dernières sections nous détaillerons l'implémentation de nos simulateurs et les techniques utilisées pour imiter les principaux protocoles utilisés.

4.1 Résultats

4.1.1 Scénario simple

Pour pouvoir comparer les performances des différentes architectures EPS, nous les avons implémentées dans le simulateur NS-2. Nous avons utilisé deux scénarios, le premier est très basique, il contient un seul UE et deux eNodeB [1]. À $t = 1$s le UE démarre une communication de voix utilisant le codec GSM (environ 15 Kbps), une seconde plus tard il lance la visualisation d'un film en *streaming*. Nous avons pris la trace du film « Mr Bean » encodé en H.264, le codec peut utiliser une compression *Low*, *Medium* ou *High* suivant les performances du réseau (taux de perte de paquets, gigue et délai).

La seconde suivante le UE démarre son navigateur web et lance une requête HTTP. La durée de vie d'une page web a été fixée à une minute, sa taille à 150 Ko et l'intervalle entre les requêtes à 30 secondes. Nous avons choisi des valeurs fixes de façon à ce que le scénario ne change pas d'un simulateur, d'une architecture, à l'autre. Trente secondes après s'être connecté le UE effectue une relève sur le second eNodeB et vingt secondes plus tard, lance un téléchargement FTP. Chaque connexion dure cent secondes. Nous avons créé ce scénario en essayant de réunir les principaux types de trafics susceptibles d'être rencontrés sur le EPS.

La voix utilise un *bearer* GBR marqué avec la classe DiffServ EF, la vidéo et le web utilisent des *bearers* non GBR respectivement marqués AF4 et AF1, le trafic FTP passe par

1. Nos simulateurs sont limités de façon statique à un MME, un SGW, un PGW, un SASN et un PCRF.

le *bearer* par défaut marqué avec DF. Le UE est QoS *aware* ce qui permet de s'assurer que les services assignés à un *bearer* rejeté ne soient pas lancés. Si le UE était QoS *unaware*, les trafics relatifs aux services dont les *bearers* auraient été rejetés passeraient par le *bearer* par défaut, ce qui fausserait les résultats.

Nous avons utilisé des liens d'un gigabit par seconde et d'une longueur d'environ 300 mètres (1 μs de délai) pour les interfaces S1-U et S1-MME. Tous les autres liens du réseau cœur utilisent des interfaces de 10 Gbps. Concernant les distances nous avons considéré que le SASN était dans la même armoire que le PGW, leur lien n'est donc que de quelques mètres (délai de 10 ns) ; à terme le SASN doit être intégré au PGW. Le MME et le SGW sont proches (de l'ordre du kilomètre) tout comme le PGW et le PCRF. Mais étant donné la rareté des PGW nous avons préféré mettre une grande distance, de l'ordre de la centaine de kilomètre, entre le PGW et le SGW. Les liens extérieurs au EPS ont été surdimensionnés (1 Tbps entre le SASN et le nœud *core*) pour ne pas représenter un goulot d'étranglement. Un schéma du réseau simulé est représenté à la figure 4.1.

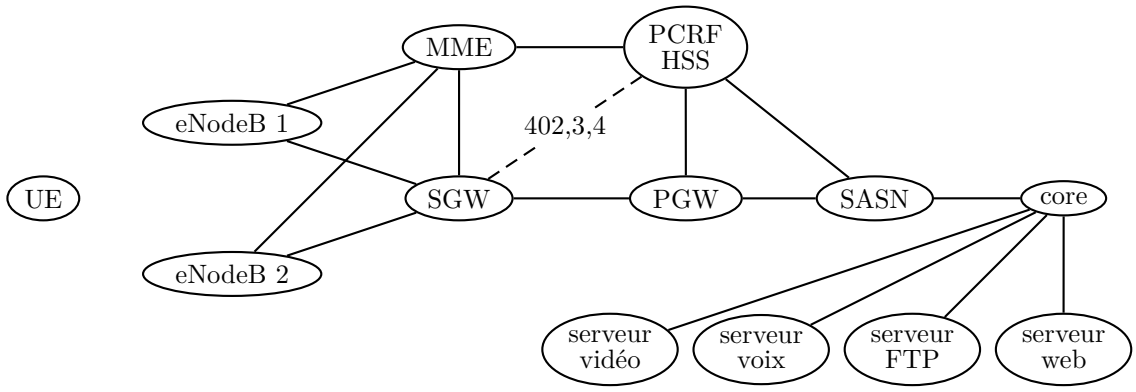


Figure 4.1 – Architecture du réseau simulé

Les résultats de ce scénario simple sont présentés à l'annexe E. On remarque à la figure E.1 qu'il n'y a aucun changement en ce qui concerne le débit. Les débits des trafics de voix, web et FTP n'ont pas été représentés, car ils sont, eux aussi, exactement égaux d'une architecture à l'autre. Le délai et donc la gigue, ont été calculés entre le SASN et le UE de façon à ne pas être influencés par la portion du réseau extérieur à l'architecture EPS. Pour la même raison le taux de perte de paquets est calculé entre le SASN et le UE.

Le délai moyen du trafic vidéo représenté à la figure E.2 ne change pas d'une architecture à l'autre, il en va de même pour les trafics web et FTP. En revanche on remarque que le trafic de voix (figure 4.2) a un délai très légèrement plus élevé (5 μs) au démarrage de la connexion sur l'architecture 23.401 [8] que sur les autres, puis plus bas (de 20 μs en moyenne) après la relève. En fait il existe aussi un écart de délai entre la 23.401 [8] et les autres pour les trafics

vidéo, web et FTP mais celui-ci est imperceptible par rapport à l'amplitude des courbes, il est de l'ordre de la microseconde.

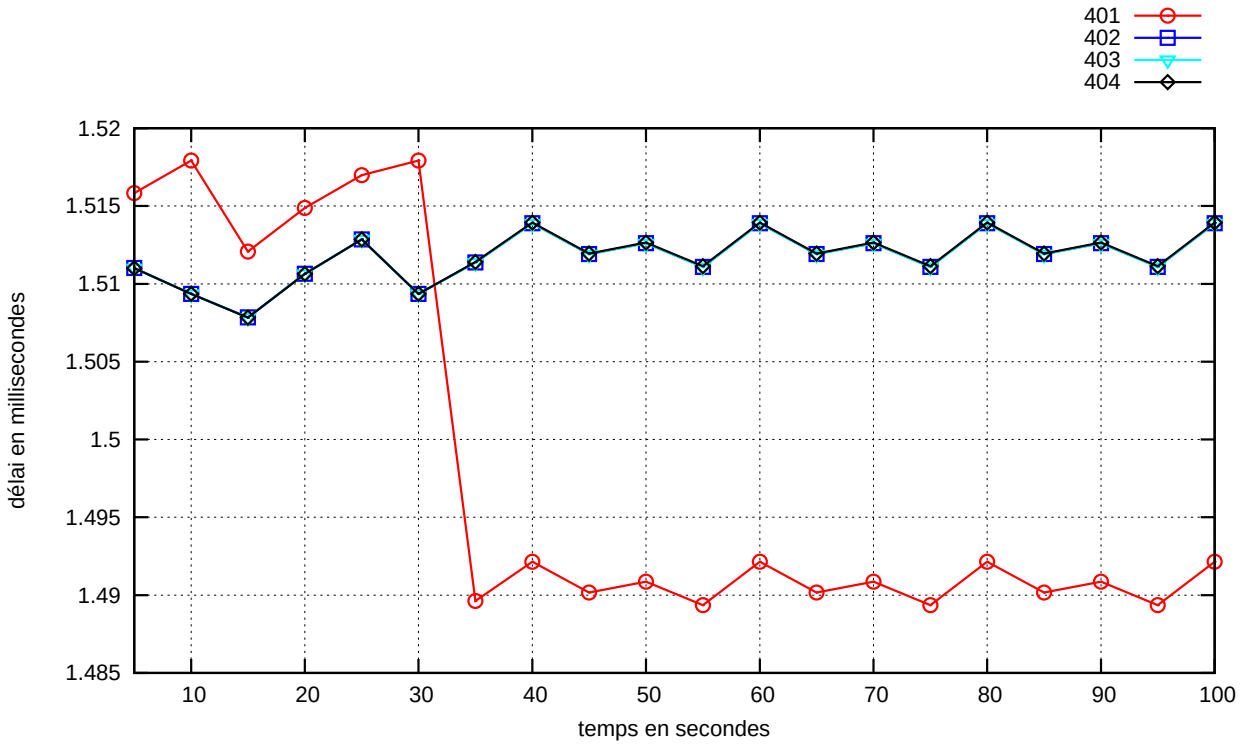


Figure 4.2 – Délai moyen du trafic de voix entre le SASN et le UE

Cette différence n'est pas due au changement d'encapsulation entre les architectures 23.401 [8] et 23.402 [9]. En effet, d'après la figure E.3 représentant le délai entre le SASN et le eNodeB pour le trafic de voix, nous remarquons que c'est la courbe de la 404 qui sort du groupe. En fait le décalage observé à la figure 4.2 est dû aux différences de temps de connexion et de création d'un *bearer* dédié. Ce temps est très similaire voire égal entre les 23.402 [9], 403 et 404 du fait de la parallélisation des messages PMIPv6 avec l'ancienne signalisation. En revanche la création de l'interface Gxc (SGW-PCRF) a ajouté quelques millisecondes à ces procédures. Les transferts de données ne démarrent donc pas en même temps dans l'architecture 23.401 [8] que dans les autres. Le *scheduling* des *bearers* radio *downlink* se faisant toutes les millisecondes sur les eNodeB, un changement du temps de démarrage de la connexion peut entrainer une variation du délai, celle-ci étant inférieur à une milliseconde. D'ailleurs on remarque que la connexion de voix démarre à $t = 0.131783$ dans l'architecture 23.401 [8] et $t = 0.131761$ dans toutes les autres, ce qui nous conforte dans notre hypothèse. Si on retarde de vingt microsecondes la connexion du UE sur la 23.402 [9] le délai du trafic de voix passe en dessous de 1,5 ms avant la relève. En décalant la relève de cent microsecondes sur la 23.401 [8] et de 250 microsecondes sur la 23.402 [9] le délai de 23.401 [8] se maintient au dessus et

celui de 23.402 [9] au dessous de 1,5 ms (voir figure 4.3). Une amélioration ou détérioration du délai inférieur à une milliseconde lorsque le réseau ne comporte que peu de UE ne peut donc pas être attribué avec certitude au changement d'architecture.

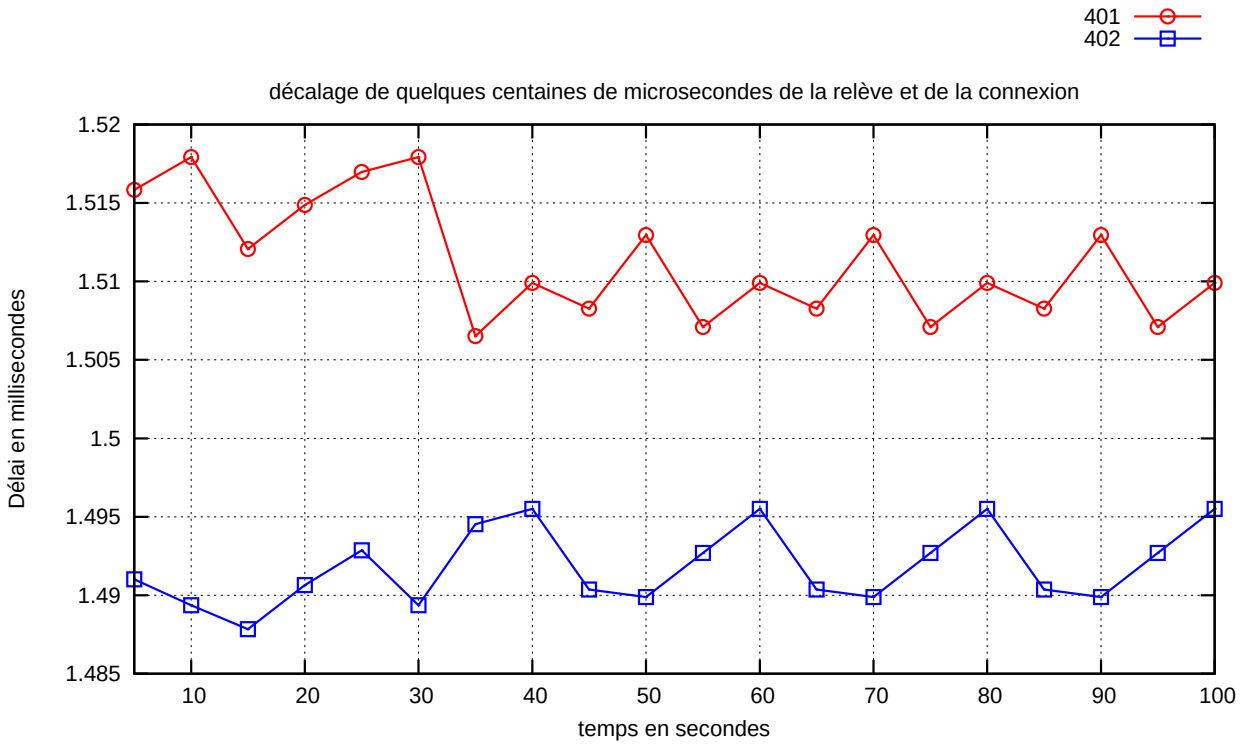


Figure 4.3 – Délai moyen du trafic de voix, la connexion et la relève ayant été décalées

Certains auront peut-être remarqué que nous avons coupé [2] les graphiques du délai et de la gigue du trafic de voix. Dans nos simulateurs, pour pouvoir détecter la fin d'une communication UDP ou RTP, les applications utilisant ces protocoles envoient un paquet spécial de petite taille (20 octets) servant de « marqueur de fin ». Cette technique fonctionne bien pour un trafic vidéo mais le trafic de voix est très stable (son débit est fixe) et utilise des paquets de petites tailles. Le délai de notre paquet de fin vient donc perturber le délai moyen de la connexion de voix, de plus cette moyenne est calculée toutes les cinq secondes et seulement dix paquets (de voix) sont envoyés après la centième seconde. Nous n'avons pas voulu représenter cette augmentation brutale de délai car elle ne traduit en rien une quelconque dégradation de la qualité de la communication. Elle traduit simplement une astuce d'implémentation pour contourner les restrictions intrinsèques à NS-2.

La gigue des trafics vidéo, web, FTP et voix (représentée figure E.4) est identique d'une architecture à l'autre. Il en va de même pour le débit et le *goodput* (voir figure E.5). Le taux

2. Normalement ces graphiques vont jusqu'à $t = 105$s

de perte de paquets est nul (voir figures E.6 et E.7) ce qui n'est pas surprenant étant donné que ce scénario ne comporte qu'un UE.

Au tableau 4.1, la seule procédure dont la durée change de plus d'une milliseconde est la procédure de destruction de *bearer* pour les UE QoS *aware*. Cette perte de performance n'est pas relative à nos modifications étant donné qu'elle s'observe entre les architectures 23.401 [8] et 23.402 [9]. De plus le temps de destruction d'un *bearer* n'est pas une variable critique, le temps de création d'un *bearer* ou le temps de relève sont beaucoup plus importants.

Tableau 4.1 – Comparaison des durées des principales procédures entre les architectures

Architecture	type	durée (en ms) d'une connexion	création de *bearer*	destruction de *bearer*	relève
401	aware	8.101	3.111	3	500
402	aware	8.101	3.102	5	500
403	aware	8.101	3.102	5	500
404	aware	8.101	3.102	5	500
401	unaware	8.101	2.084	2	500
402	unaware	8.101	2.084	2	500
403	unaware	8.101	2.084	2	500
404	unaware	8.101	2.084	2	500

En conclusion, des résultats de ce premier scénario nous pouvons dire que nos modifications n'apportent pas de changement significatif des performances par rapport à l'architecture 23.402 [9], dans le cas d'un réseau faiblement chargé. Les performances des 23.401 [8] et 23.402 [9] sont tout à fait similaires. On note toutefois que la procédure de destruction prend deux millisecondes de plus dans la 23.402 [9]. Pour tester ces architectures avec une charge plus importante, nous avons développé un nouveau scénario.

4.1.2 Scénario chargé

Dans ce nouveau scénario nous avons repris la même configuration que précédemment, les liens ont les mêmes caractéristiques, en revanche le réseau d'accès contient dix eNodeB sur lesquels mille UE se connectent (voir figure 4.4). Les UE sont répartis aléatoirement [3] et équitablement entre tous les eNodeB. Par « équitablement » nous entendons que si le scénario

3. La « graine » de la séquence aléatoire est la même d'une architecture à l'autre.

contient n UE et m eNodeB numérotés de 1 à m alors le eNodeB i possède x_i UE avec :

$$x_i = \left\lfloor \frac{n}{m} \right\rfloor + y_i \tag{4.1}$$

$$\text{où } y_i = \begin{cases} 1 & \text{si } i \leq n - \left\lfloor \frac{n}{m} \right\rfloor m \\ 0 & \text{sinon} \end{cases}$$

Les milles UE se connectent progressivement au rythme d'un par seconde. Ils se comportent tous comme celui du premier scénario ; ils lancent successivement une communication VoIP, un trafic vidéo, un trafic web et un téléchargement FTP. Ils font une relève trente secondes après s'être connecté.

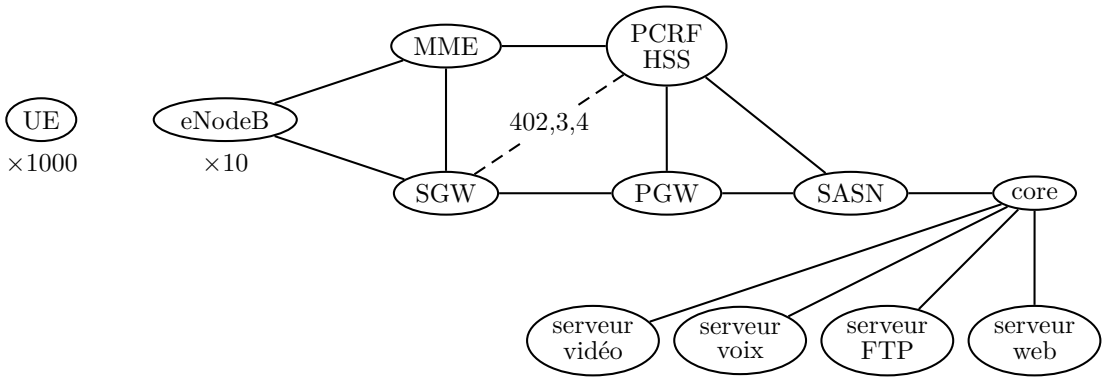


Figure 4.4 – Architecture du réseau simulé (version chargée)

Lors des simulations, les architectures 23.402 [9], 403 et 404 se sont comportées exactement de la même façon en ce qui concerne la gestion des *bearers*. Tous les UE n'ayant pas réussi à se connecter dans l'architecture 23.402 [9] ont aussi été rejeté dans la 403 et la 404 et vice versa. Il en va de même pour les *bearers* rejetés à leur création ou lors d'une relève. En revanche quelques divergences sont apparues entre la 23.401 [8] et les autres à partir de la 850$^{\text{e}}$ seconde.

Le UE numéro 965 (devant se connecter à la 965$^{\text{e}}$ seconde) a été accepté dans l'architecture 23.401 [8] et rejeté dans les autres. Les *bearers* réservés au trafic de voix des UE 847 et 862 ont été rejetés lors des procédures de relèves (environ aux secondes 880 et 890) dans les 23.402 [9], 403 et 404 et acceptés dans la 23.401 [8]. En revanche les *bearers* des trafics vidéos des UE 965 et 902 et le *bearer* du trafic de voix du UE 965 ont été rejetés (lors de leur création) dans la 23.401 [8] et acceptés dans les autres. Ces différences sont dues au paramétrage du scénario plus qu'aux performances des architectures elles même, en effet les *bearers* réservent exactement les mêmes bandes passantes dans toutes les architectures. Le temps d'exécution des procédures couplé aux instants de lancement de celles-ci font que la

configuration du réseau peut être différente d'une architecture à l'autre à un instant donné. Par exemple un UE qui se connecte peut arriver sur un eNodeB supportant déjà dix ou onze UE suivant l'architecture. Beaucoup de facteurs entrent en compte, comme la rapidité d'exécution des procédures, le rejet ou l'acceptation des précédents UE et de leurs *bearers*, l'instant de connexion, de relève, etc.

Étant donné ces divergences, une différence de performance observée à partir de la 850^{e} seconde non incluse (880 pour être exact) ne peut pas être imputée avec certitude aux modifications apportées étant donné que la charge dans le réseau n'est plus la même.

Tel que représenté à la figure F.1, le délai du trafic vidéo est identique dans les différentes architectures, avec une légère différence entre la 23.401 [8] et les autres à partir de la 900^{e} seconde incluse. Il en va de même pour les trafics de voix (représentés à la figure F.2), web et FTP.

Pour avoir plus de détails sur ces résultats nous avons observé le délai au niveau du eNodeB, les modifications portant sur la portion de réseau entre le eNodeB et le PGW. Pour la vidéo (voir figure F.3), le web et le FTP les courbes se superposent (excepté un artefact à $t = 1000$). Pour le trafic de voix, du fait qu'il passe par la file DiffServ de plus haute priorité, que ses *bearers* sont GBR et que ses paquets sont de petites tailles, le délai ne varie que très peu (de l'ordre de la microseconde) au cours de la simulation. L'amplitude des courbes est donc moins grande ce qui nous permet de distinguer les très faibles différences de délai entre les architectures. Tel que représenté à la figure F.4, les courbes des 23.402 [9], 403 et 404 sont presque parallèles, ce qui peut paraitre normal étant donné que les architectures 403 et 404 sont basées sur la 23.402 [9] et reprennent en grande partie ses diagrammes de séquences. Le délai de la 404 est le plus bas sur 80% du temps que dure la simulation. Cela est dû au fait que l'*overhead* des paquets y est moins grand (les paquets sont directement encapsulés dans IPv6). On peut aussi remarquer qu'en quatre points à partir de la 750^{e} seconde, le délai de la 23.401 [8] est le plus bas (de l'ordre d'une microseconde). Bien qu'il y ait un léger écart entre ces courbes, celui-ci reste de l'ordre de la microseconde, ce qui est négligeable. Les différences observées sur les courbes des figures F.1 et F.2 sont donc dues au réseau d'accès (identique sur toutes les architectures).

Concernant le délai nous pouvons donc dire que les performances des différentes architectures sont identiques.

Les débits sont similaires d'une simulation à l'autre (voir figure F.6). Il en va de même pour le *goodput* comme le montre les figures F.7 et F.8. Concernant la gigue (figures F.5 et 4.5) on remarque que, comme pour le délai, les performances sont identiques entre les architectures 23.402 [9], 403 et 404. En revanche, celles de la 23.401 [8] sont nettement meilleures ; à partir de la 650^{e} seconde la gigue du trafic vidéo sur la 23.401 [8] est en moyenne inférieure de

3,5 ms par rapport aux autres, avec une différence maximum de 6,5 ms. Pour le trafic de voix, la gigue est en moyenne inférieure de 1,7 ms avec une différence maximum de 5 ms. Ces divergences surviennent avant la 850^{e} seconde il n'y a donc aucun doute sur la supériorité de l'architecture 23.401 [8]. En revanche, la gigue des 403 et 404 ne changent pas par rapport à la 23.402 [9], nos modifications n'ont donc pas eu d'impact visible.

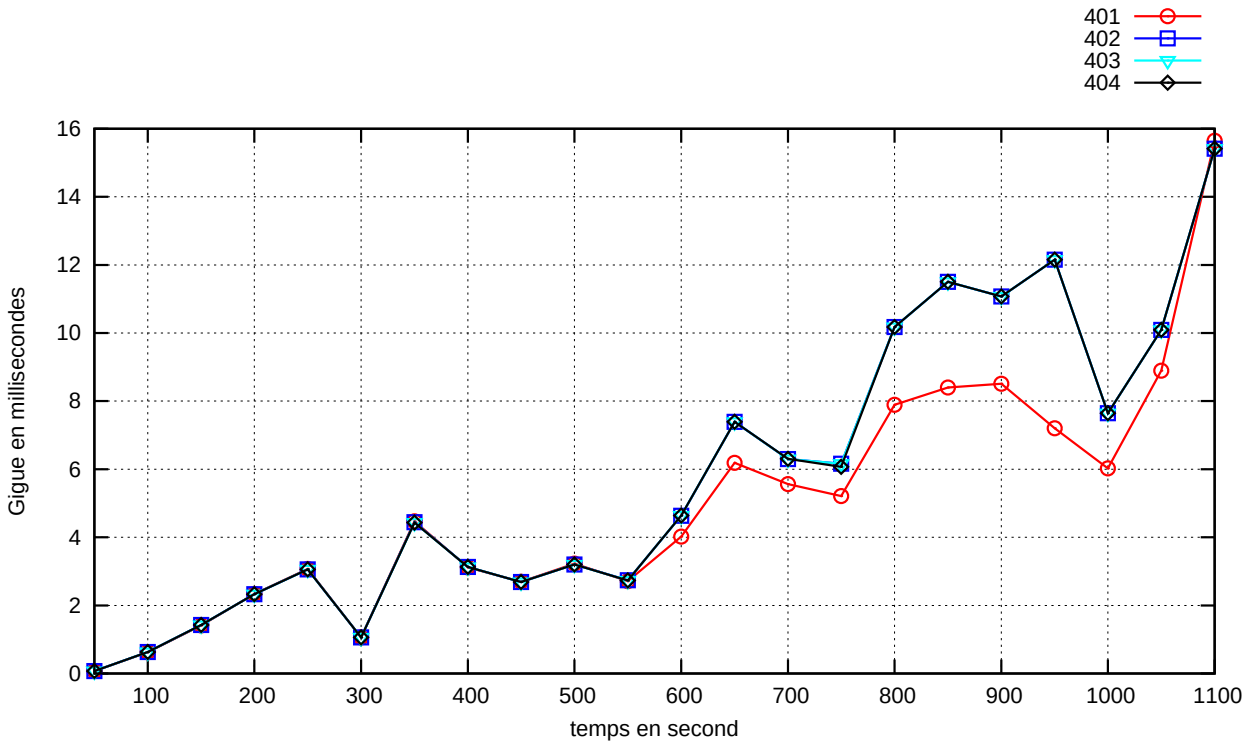


Figure 4.5 – Gigue du trafic de voix entre le SASN et les UE

En ce qui concerne le taux de perte de paquets, celui-ci est nul pour les trafics vidéo, web et FTP comme le montrent les figures F.9 à F.11. En revanche celui du trafic de voix représenté à la figure 4.6 est non nul et atteint un maximum de 1 % au début du scénario (150^{e} seconde) pour les 23.402 [9], 403 et 404. Selon Wallingford [100, chap. 9], un réseau de VoIP ne doit pas souffrir d'un taux de perte de paquets supérieur à 1 %, nous sommes donc à la limite de l'acceptable. Une augmentation des caches des *bearers* GBR associés au trafic de voix serait à envisager sur ces architectures. En revanche le taux de perte de paquets ne dépasse jamais les 0,25 % dans la 23.401 [8], même s'il est supérieur à celui des autres entre la 500^{e} et la 850^{e} seconde.

Si l'on compare les architectures 23.402 [9], 403 et 404 nous observons que les 403 et 404 perdent légèrement moins de paquets lors des pics de pertes ; mais cela reste négligeable (environ 0,02 % de différence). Nous pouvons donc dire que nos modifications n'affectent pas les performances du réseau en ce qui concerne le taux de perte de paquets et que les

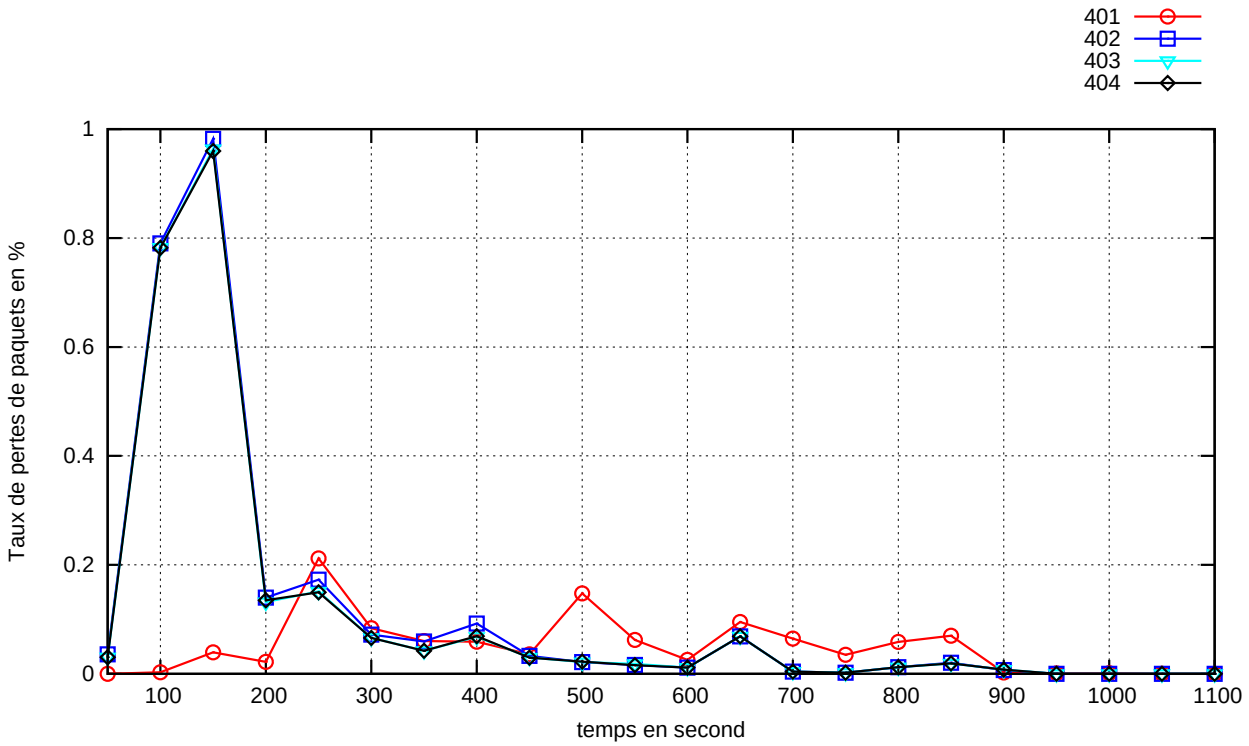


Figure 4.6 – Taux de perte de paquets global du trafic de voix entre le SASN et les UE

performances de l'architecture 23.401 [8] sont meilleures que celles des autres.

4.1.3 Synthèse

Ces résultats nous indiquent que les modifications que nous avons apportées à l'architecture 23.402 [9] dans 403 et 404 n'ont pas ou très peu affecté les performances du réseau au niveau du UE. Les faibles différences entre les 23.402 [9], 403 et 404 étant des améliorations en faveur de la 403 et la 404. Il semble que l'architecture 23.401 [8] ait de meilleures performances que les autres en ce qui concerne la gigue et le taux de perte de paquets ; celle-ci est donc préférable pour les opérateurs. Cependant, si pour des raisons politiques l'opérateur doit abandonner le protocole GTP sur l'interface S5/S8, par exemple pour intégrer un réseau d'accès WiMAX, nous lui conseillons d'utiliser l'architecture 403 voire 404. Celles-ci utilisant de façon plus poussée les avantages d'IPv6 que 23.402 [9].

4.2 Implémentation

Pour évaluer l'impact de nos propositions nous avons développé quatre simulateurs basés sur NS-2 représentant chacun une architecture.

4.2.1 Network Simulator version 2

NS-2 est un simulateur de réseau *open source*, il supporte un nombre considérable de protocoles, mais possède aussi de nombreuses restrictions que nous avons dû contourner. Aujourd'hui une nouvelle version majeure du simulateur est disponible, Network Simulator version 3 (NS-3) ; elle pallie un grand nombre de défauts des précédentes. Malheureusement lorsque NS-3 est arrivé à maturité nous étions déjà trop avancé dans l'implémentation de nos simulateurs pour envisager une migration. De plus son architecture est très différente de celle des versions précédentes.

NS-2 se base sur les langages de programmations C++ et Object Tool Command Language (OTcl). Le noyau est en C++ et les scénarios [4] sont en OTcl de cette façon une modification du scénario ne nécessite pas une re-compilation du logiciel. Chaque classe OTcl doit avoir une classe correspondante en C++, et chaque instanciation d'un objet OTcl entraine la création d'un objet C++. L'idée est bonne, mais la liaison entre le C++ et le Tool Command Language (Tcl) entraine une plus grande complexité du code C++ [5].

Dans NS-2 il n'y a pas de notion de couche de protocole, il existe des objets de type `Application` qui sont chargés de générer les trafics et des `Agent` qui simulent toutes les couches inférieures. L'utilisation d'`Application` n'est pas obligatoire, par exemple le trafic FTP se résume à l'envoi d'un maximum de paquets TCP jusqu'à la réception d'une commande « stop », un `Agent` suffit. La figure 4.7 représente l'architecture C++ d'un nœud simple.

Dans les sections suivantes nous allons décrire quelques-unes des classes et méthodes développées dans le cadre de cette maîtrise.

4.2.2 Plan de contrôle

Dans la réalité le plan de contrôle est composé de plusieurs protocoles couplés entre eux. Pour simplifier notre code et du fait de l'absence de couches distinctes dans NS-2 nous n'avons pas implémenté ces éléments de façon séparée. Nous avons créé une application par UE, par eNodeB et par nœud du EPC. Ces application sont chargées de simuler le comportement de tous les protocoles au-dessus de TCP ou UDP, utilisés sur ces nœud. Chacune possède sa propre classe dérivant au minimum de `ApplicationEPS`. Celles gérant des *bearers* dérivent de la classe `ApplicationBearer`. La figure 4.8 représente le diagramme de classe de ces applications et des principales classes qu'elles utilisent.

Chaque application possède au moins un `Agent` lui permettant d'envoyer des données sur le réseau. Dans le EPC celles de contrôle utilisent les protocoles de transport UDP, TCP

4. La définition des nœuds, de la topologie du réseau, des trafics, etc.

5. Dans NS-3 il est possible d'écrire le code et les scénario en C++, une liaison avec Python est possible, mais reste optionnelle

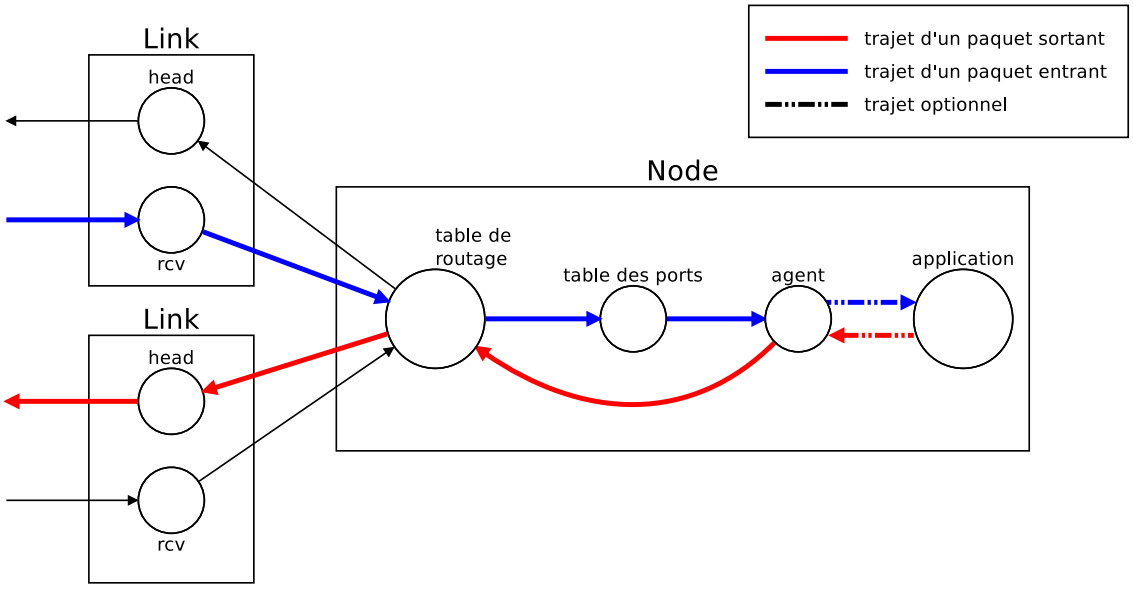


Figure 4.7 – Représentation des différentes composantes d'un nœud dans le simulateur NS-2 (inspiré de The VINT Project [97, sec 5.1])

et/ou SCTP. Le protocole SCTP, que nous simulons grâce au code du TCP, est décrit à la section 4.3.3. Les `Agent` UDP et TCP ne peuvent normalement pas envoyer de données[6] (UDP le peut, mais avec de nombreuses restrictions) mais simplement un certain nombre d'octets, nous avons donc dû modifier leurs implémentations.

Nous avons fait évoluer UDP pour qu'il puisse envoyer des données[6] plus grande que la taille maximum d'un paquet, simulant ainsi la segmentation au niveau de la couche IP. Comme nous l'avons dit un `Agent` prend en charge toutes les couches de protocoles situées sous l'application. Nous avons aussi créé une nouvelle classe appelée `TcpAppAgent` dérivée de l'agent TCP standard permettant d'utiliser ce protocole pour envoyer des objets C++. Nous l'utilisons sur les liens S1-MME, S6a, Gx et Rx.

Chaque application possède une méthode `process_data` dans laquelle aboutissent toutes les données[6] reçues par ses agents. Dans cette méthode nous faisons un tri des données en fonction de leur type (correspondant au nom du message dans la réalité) et nous appliquons les changements qui lui sont relatifs.
Par exemple, pour une donnée de type `PACKET_ENCAPSULE_DATA`, le paquet qu'il contient est extrait, on en détermine la destination, on le ré-encapsule dans une nouvelle donnée du même type et on l'envoie à la prochaine application sur le chemin menant à cette destination. Une donnée de type `DELETE_BEARER_REQUEST_DATA` déclenchera une procédure de suppression des informations et des objets simulant le comportement du *bearer* concerné sur le nœud courant.

6. Par donnée j'entends ici : objet C++.

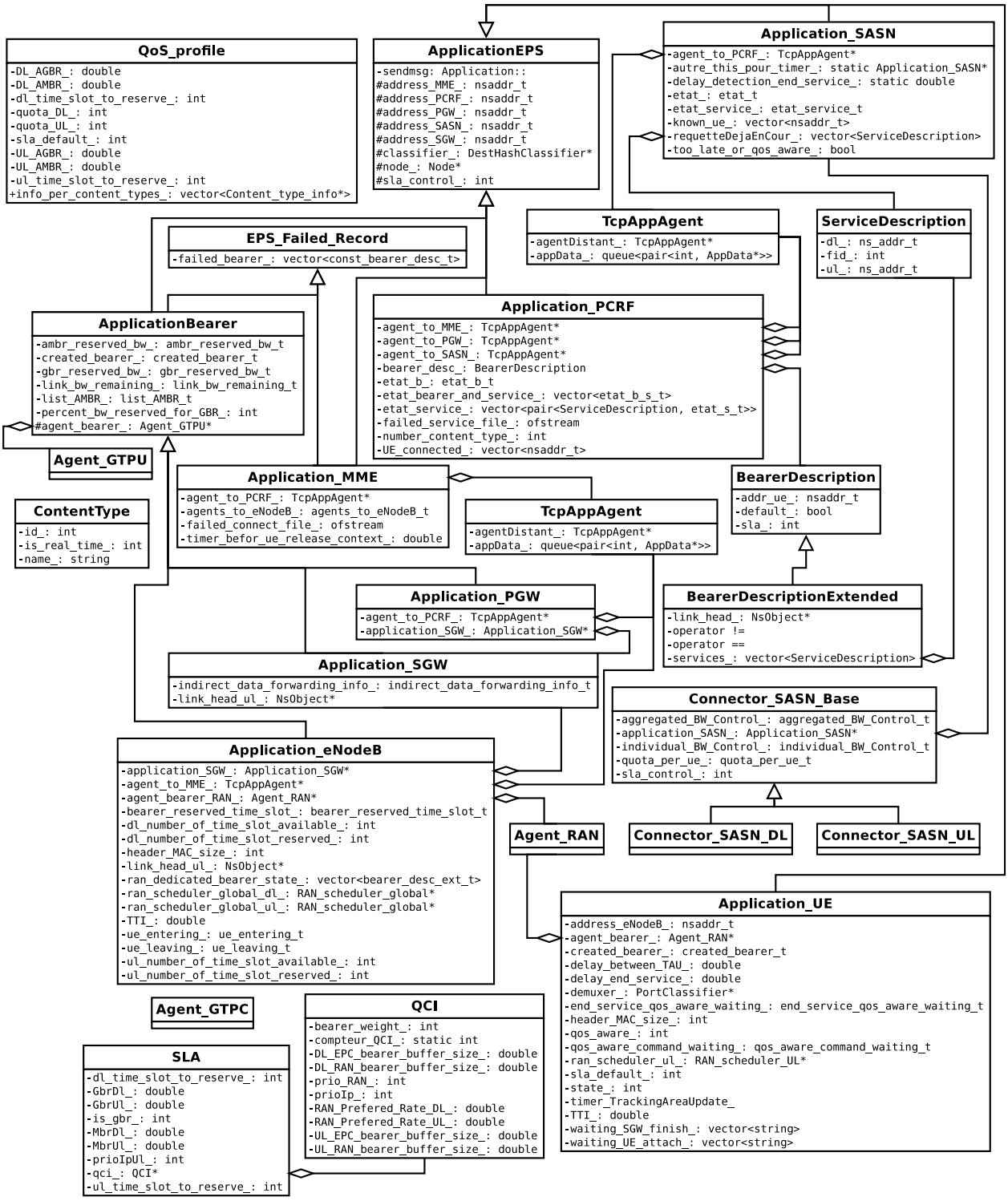


Figure 4.8 – Diagramme des principales classes de notre simulateur (celles que nous avons implémentées)

Il existe plusieurs classes permettant de représenter des données ; les principales étant :

BearerManagementData représente des données de contrôle, elle est utilisée dans toutes les séquences de création/modification/destruction de *bearer*.

UELocationData représente des données de localisation du UE (contrôle), elle est utilisée principalement dans les séquences de relève.

PacketEncapsuleData représente un paquet encapsulé, elle est utilisée sur le plan de données.

Ces classes dérivent toutes de `AppDataExtended` ajoutant quelques fonctionnalités à la classe `AppData` de NS-2. Par exemple à la réception d'une donnée, l'application n'a pas accès au paquet l'ayant transporté, il n'y a donc pas de moyen de connaître sa source. Nous avons donc ajouté une adresse source dans `AppDataExtended` pour essayer de pallier à ce problème. La figure 4.9 représente le diagramme de classe de ces types de données.

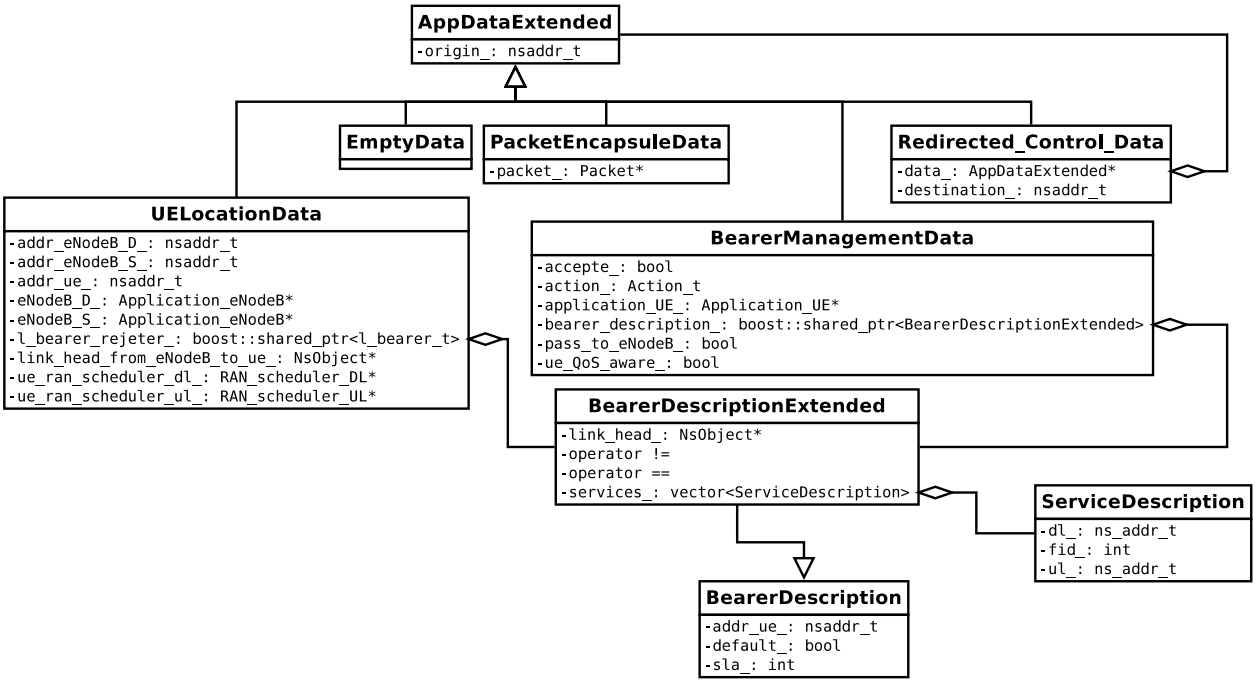


Figure 4.9 – Diagramme des classes représentant les données échangées par les nœuds

4.2.3 Service Aware Support Node

Le SASN est un appareil développé par Ericsson permettant d'effectuer une partie du travail du PCEF et du Application Function (AF). Les paquets qui transitent par le SASN ne sont pas encapsulés et ne lui sont donc pas destinés. Comme le PGW en *downlink* et le UE en *uplink*, il doit capturer ses paquets afin de pourvoir les traiter. Dans le schéma

représenté à la figure 4.7 nous pouvons remarquer que les paquets ne remontent au niveau des agents que si leur IP de destination est celle du nœud courant. Nous avons donc dû modifier l'architecture du SASN, du PGW et des UE pour permettre la remontée de tous les paquets. Nous décrivons ici le SASN mais le principe est le même sur le PGW et le UE. La figure 4.10 donne une vue des objets présents dans le SASN, les principales classes utilisées étant décrites à la figure 4.8.

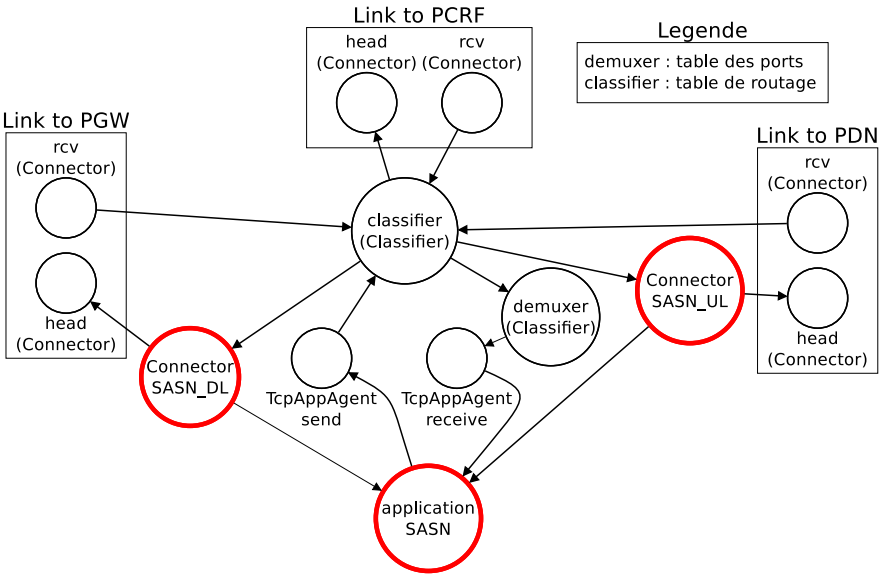


Figure 4.10 – Diagramme d'objet du SASN dans notre simulateur

Pour pouvoir capter les paquets *downlink* et *uplink* dans le SASN nous avons modifié la classe `Classifier`. De base, les `Classifier`, représentant les tables de routage des nœuds, sont statiques ou dynamiques. Nous n'utilisons pas de protocole de routage particulier, nous devons donc utiliser des `Classifier` statiques. Ceux-ci se configurent soit manuellement soit automatiquement (en utilisant l'algorithme de Dijkstra [60]), mais il n'existe pas de méthode « semi-automatique ». Nous avons donc ajouté quelques fonctions pour permettre la modification des routes d'un classifier statique automatique.

Désormais les tables de routage se calculent automatiquement grâce à l'algorithme de Dijkstra [60], puis les modifications apportées manuellement par l'utilisateur viennent changer les routes. Nous avons par exemple redirigé les sorties du `Classifier` pointant vers les liens SASN-PGW et SASN-extérieur respectivement vers des objets de type `Connector_SASN_DL` et `Connector_SASN_UL`. Nous avons aussi fait en sorte que les paquets de données ne passent ni par le PCRF ni par le MME.

Les `Connector` du SASN marquent les paquets et appliquent des limiteurs de bande passante en fonction du type de trafic et du UE. Ils détectent aussi les nouvelles connexions et avertissent le PCRF via l'application `ApplicationSASN` et les `TcpAppAgent`.

4.2.4 *Bearers*

Pour représenter les *bearers* au niveau application sur les nœuds du plan de données nous avons créé les classes `GBR_Bearer` et `Non_GBR_Bearer` schématisées à la figure 4.11. De façon basique elles mettent en cache les paquets en attendant leur envoi sur le lien. La classe `GBR_Bearer` envoie elle même les paquets et s'assure que son débit de sortie n'excède pas le MBR du *bearer*. Pour cela elle utilise une file spéciale appelée `MBR_queue` qui programme l'envoi du paquet suivant en fonction du MBR et de la taille du dernier paquet sorti.

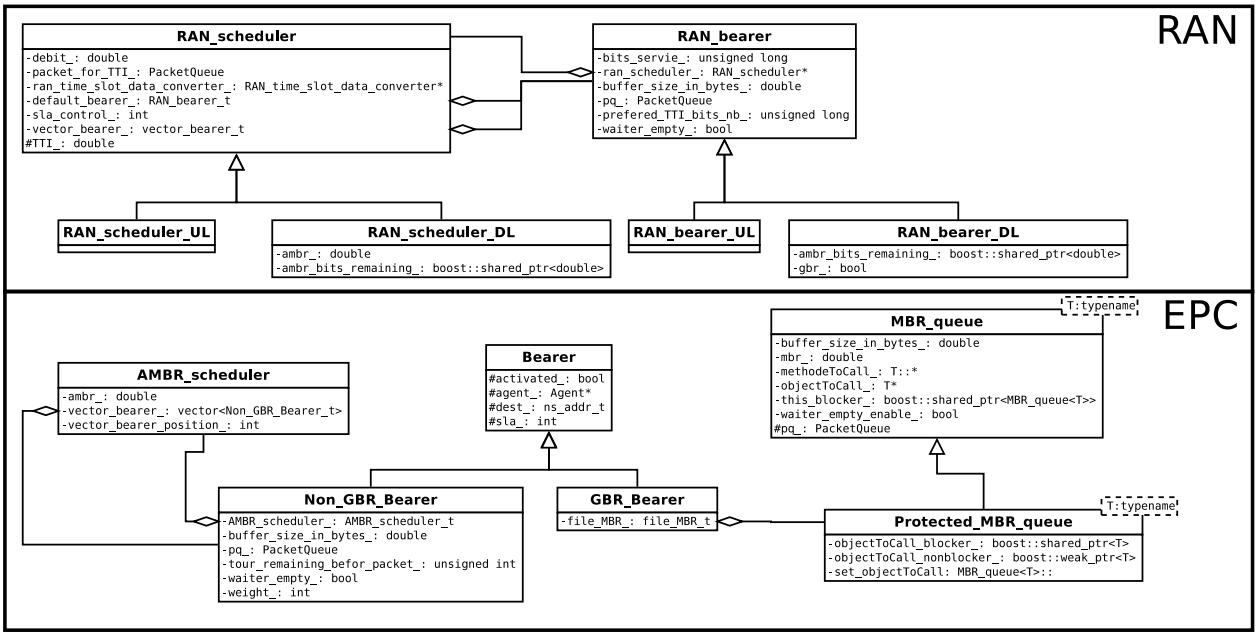


Figure 4.11 – Diagramme des classes représentant les *bearers* de notre simulateur

En vérité les objets de type `GBR_Bearer` utilisent des files `Protected_MBR_queue` étendant les fonctionnalités des `MBR_queue`. Une file `Protected_MBR_queue` possède la capacité, si elle est correctement utilisée, d'empêcher la destruction de l'objet auquel elle est rattachée (en l'occurrence un objet de type `GBR_Bearer`) tant qu'elle contient des paquets. Cette fonctionnalité nous est utile pour empêcher les pertes de paquets lors d'une destruction de *bearer* dû à un *handover*. On peut détruire un *bearer* sur une application, celle-ci n'a alors plus aucun lien avec ce *bearer*, sans que celui-ci soit effectivement détruit. Comme l'application ne « connait » plus le *bearer* celui-ci ne reçoit plus de nouveau paquet, mais continue de se vider étant donné qu'il décide lui même quand envoyer ses paquets. Lorsqu'il a envoyé tous ses paquets il s'auto-détruit.

Les objets de type `GBR_Bearer` décident eux-mêmes du moment où envoyer leurs paquets étant donné qu'ils possèdent leur propre MBR. Ce n'est pas le cas des objets `Non_GBR_Bearer` qui partagent un même AMBR, nous avons donc dû créer une classe `AMBR_scheduler` effec-

tuant à la manière de DiffServ un DWBBRR sur les *bearers* d'un même UE. Ce fonctionnement est schématisé à la figure 4.12. Les *bearers* non GBR n'ont pas de MBR ils n'ont donc pas besoin de files spéciales comme `MBR_queue`, ils utilisent donc la classe `PacketQueue` de NS-2.

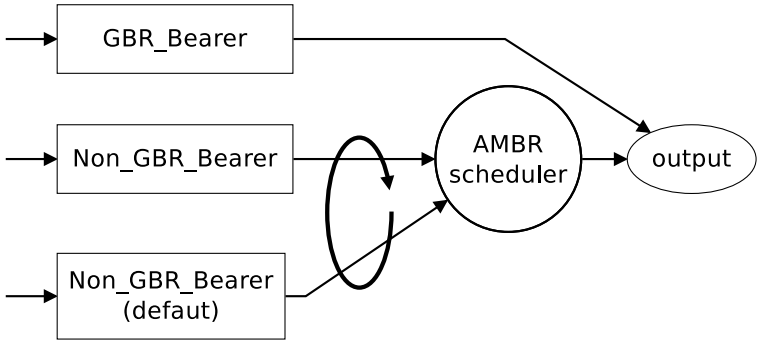


Figure 4.12 – Diagramme d'objet représentant le *scheduling* des *bearers*

4.2.5 Réseau radio

Sur le réseau radio les *bearers* sont différents de ceux du EPC, il n'y a plus de GBR ou non GBR. Le eNodeB gère le *scheduling* des *bearers* en *downlink* et le *scheduling* des UE[7] en *uplink*, les UE gèrent le *scheduling* de leurs propre *bearers* en *uplink*. Ces méthode de *scheduling* ne sont pas spécifiées par 3GPP ; cependant Dahlman *et al.* [54, sec. 15.2.2] décrivent de façon générale un algorithme de services des *bearers* sur les UE.

Pour notre part nous avons choisi d'appliquer l'algorithme décrit par Dahlman *et al.* [54, sec. 15.2.2]. Chaque *bearer* Radio Access Network (RAN) possède une priorité ainsi qu'un débit préféré. Le *scheduler* sert d'abord le *bearer* ayant la plus haute priorité jusqu'à ce que celui-ci ait atteint son débit préféré ou qu'il n'ait plus de paquet à transmettre, ensuite il sert de la même façon le *bearer* suivant par ordre de priorité. Lorsque tous les *bearers* ont atteint leur débit préféré le *scheduler* sert le *bearer* de plus haute priorité jusqu'à ce qu'il n'ait plus de paquet et continue avec les *bearers* suivants par ordre de priorité.

Le *scheduling* des UE en *downlink* et *uplink* se fait de façon à optimiser l'utilisation des ressources du eNodeB tout en assurant qu'aucun des UE ne soit privilégié. Ces ressources sont réparties équitablement entre tous les UE, celles non utilisées par un UE sont redistribuées équitablement entre tous les autres.

La figure 4.13 présente les différentes classes utilisées pour représenter le réseau RAN. Nous avons créé la classe `RAN_scheduler_global` pour gérer le *scheduling* des UE, chaque

7. Le eNodeB ne gère pas le *scheduling* des *bearers* des UE en *uplink*.

eNodeB possède deux objets de ce type, un pour le *scheduling uplink* et un autre pour le *downlink*. La procédure utilisée par les `RAN_scheduler_global` pour servir les UE est décrite à l'algorithme 1. Chaque lien UE-eNodeB possède un `RAN_scheduler_UL` et un `RAN_scheduler_DL` gérant le *scheduling* des *bearers*.

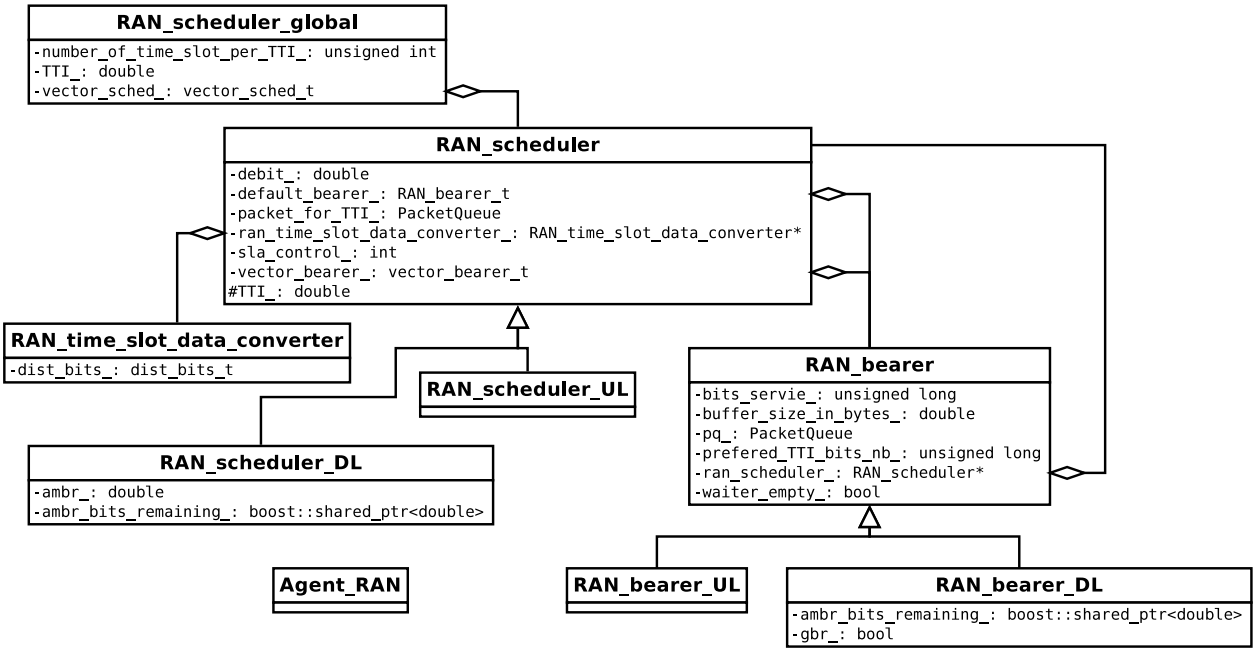


Figure 4.13 – Diagramme des classes utilisées dans la simulation du lien radio

Les objets de type `RAN_scheduler_UL` et `RAN_scheduler_DL` possèdent respectivement des `RAN_bearer_UL` et `RAN_bearer_DL`. La procédure utilisée par les `RAN_scheduler` pour servir les *bearers* est décrite de façon basique grâce à l'algorithme 2. La raison pour laquelle nous avons deux types de `RAN_scheduler` et `RAN_bearer` est que le eNodeB doit appliquer un AMBR aux *bearers downlink*. Les classes `RAN_scheduler_DL` et `RAN_bearer_DL` possèdent donc des attributs supplémentaires pour s'assurer que le trafic issu des *bearers* non GBR du EPC n'excède pas le AMBR autorisé [8].

4.2.6 Traces

De base, NS-2 fournit quelques modules de traces permettant de générer les résultats des simulations. L'un des modules permet de générer une trace exploitable par le logiciel « nam » utilisé pour visualiser les paquets sur le réseau. L'autre module facilement exploitable permet d'enregistrer des informations sur les paquets passant dans une de ces traces. Ces informations regroupent, entre autre :

8. Nous aurions dû implémenter cette fonctionnalité dans l'application du eNodeB mais cela aurait été plus compliqué sans changer le comportement global du nœud.

Algorithme 1 *Scheduler* utilisé sur le RAN pour distribuer la bande passante entre les UE

$nb\_time\_slot\_per\_UE = \frac{\text{nombre de time slote total}}{\text{nombre de UE}}$
$surplus\_total = 0$
Pour tout UE **faire**
  servir le UE avec $nb\_time\_slot\_per\_UE$ time slot
  $surplus$ = nombre de time slot non consommé par le UE
  Si $surplus > 0$ **alors**
    $surplus\_total = surplus\_total + surplus$
  Sinon Si UE veut plus de time slot **alors**
    marquer le UE comme « non rassasié »
  Fin Si
Fin Pour
Tant que reste des UE « non rassasiés » **et** $surplus\_total \neq 0$ **faire**
  $nb\_time\_slot\_per\_UE = \frac{surplus\_total}{\text{nombre de UE}}$
  $surplus\_total = 0$
  Pour tout UE « non rassasié » **faire**
    servir le UE avec $nb\_time\_slot\_per\_UE$ time slot
    $surplus$ = nombre de time slot non consommé par le UE
    Si $surplus >= 0$ **alors**
      $surplus\_total = surplus\_total + surplus$
      marquer le UE comme « rassasié »
    Fin Si
  Fin Pour
Fin Tant que

Algorithme 2 *Scheduler* utilisé sur le RAN pour servir les *bearers*

le `RAN_scheduler_global` fournit nb_time_slot time slot
$nb\_bits\_restant$ = équivalant de nb_time_slot time slot en bits
/*tous les bearers sont limités à leur débit préféré*/
Pour tout bearers par ordre de priorité **faire**
  servir le bearer avec $nb\_bits\_restant$
  $nb\_bits\_restant$ = nombre de time slot non consommé par le UE
Fin Pour
Si au moins un bearer non vide **et** $nb\_bits\_restant > 0$ **alors**
  activer le dépassement des débits préférés sur le bearer
  Pour tout bearers par ordre de priorité **faire**
    servir le bearer avec $nb\_bits\_restant$
    $nb\_bits\_restant$ = nombre de time slot non consommé par le UE
  Fin Pour
Fin Si
retourner $nb\_bits\_restant$ converti en time slot

- l'heure de passage ;
- la trace ayant enregistré l'information ;
- la taille du paquet ;
- la taille totale de ses en-têtes ;
- le type de trafic ;
- la source et la destination du paquet.

En plaçant des traces dans les liens comme représenté à la figure 4.14 il est possible, grâce au fichier généré, de retracer le parcours de chaque paquet et calculer les métriques recherchées. Cependant, cette méthode n'est pas très efficace selon nous, car elle requière un long travail d'analyse de fichiers de traces pouvant avoir des tailles supérieures au giga.

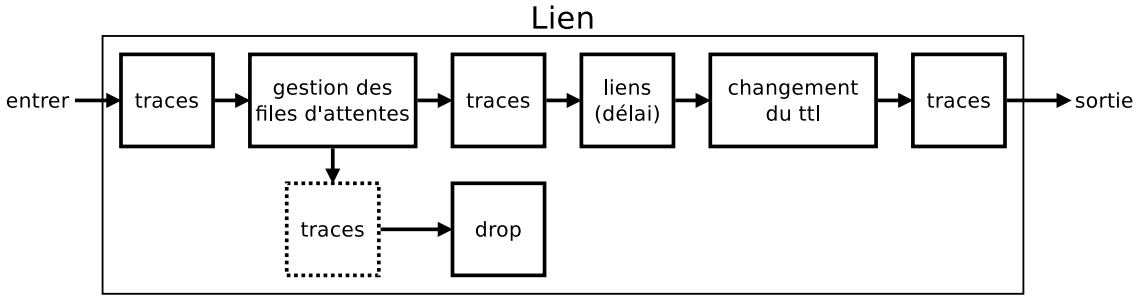


Figure 4.14 – Représentation des différentes composantes d'un lien dans le simulateur NS-2 (inspiré de The VINT Project [97, chap 6])

Dans un premier temps nous avons essayé d'utiliser ces fichiers de traces brutes avec un programme écrit en Ruby. Celui-ci permettait de calculer le délai, le débit, le *goodputs* ou les gigues d'un trafic en particulier ou de tout le réseau. Il était fonctionnel mais très lent étant donné la quantité de données à mettre en relation.

Lorsqu'il a fallu lancer des simulations de plusieurs centaines de UE nous avons décidé de créer un nouveau système de traces permettant de calculer directement quelques métriques de performances (les mêmes que celles de l'ancien logiciel Ruby). Nous nous sommes inspirés de l'architecture des traces déjà présentes pour créer les classes représentées à la figure 4.15 et notamment `TracePerso`. Chaque paquet passant par un objet `TracePerso` est enregistré dans un fichier à la manière de l'ancien module. Les enregistrements ont été épuré et la classe utilise la nouvelle interface C++ `ofstream` au lieu d'une interface Tcl ce qui rend le code plus simple.

La classe `TracePerso_Send` doit être utilisée à la sortie du nœud source. Elle marque les paquets avec le temps courant et une liste de `Trace_Computer_Loss` pour permettre un calcul du délai et du taux de perte de paquets, plus simple. La classe `TracePerso_Recv` doit être utilisée à la sortie du lien juste avant la destination. Elle calcule le délai des paquets grâce

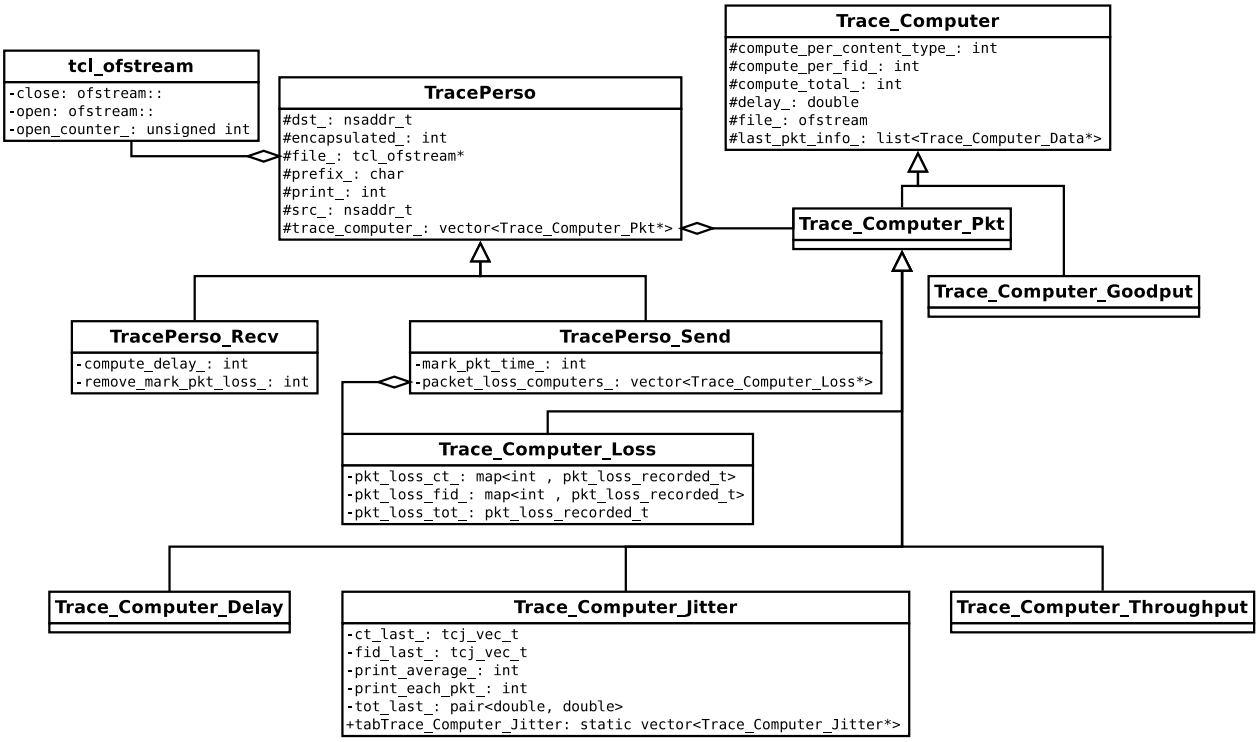


Figure 4.15 – Diagramme des classes générant les traces

au marquage laissé par `TracePerso_Send` et leurs retire les listes de `Trace_Computer_Loss` à notifier en cas de destruction.

Chaque objet de type `TracePerso` possède une liste de `Trace_Computer` qu'il notifie lors du passage d'un paquet. Les `Trace_Computer` sont chargés de calculer durant la simulation les métriques implémentées. Parmi les `Trace_Computer`, `Trace_Computer_Loss` et `Trace_Computer_Goodput` sont un peu spéciales. `Trace_Computer_Loss` comme son nom l'indique, permet de calculer le taux de perte de paquets. La technique utilisée habituellement dans NS-2 consiste à placer une trace à chaque endroit où un paquet peut être rejeté, ce qui offre l'avantage de donner l'emplacement du rejet. Mais cela devient problématique lorsque le nombre de ces endroits est grand ou qu'ils n'ont pas été prévu pour accueillir une trace. Nous avons donc opté pour une autre approche, les `TracePerso_Send` possèdent une liste de `Trace_Computer_Loss`, chaque paquet est marqué avec cette liste. Les `TracePerso_Recv` retirent cette marque. Lorsqu'un paquet n'est plus utilisé, par exemple parce qu'il est arrivé à destination ou qu'il a été rejeté, les règles de NS-2 font que la méthode static `Packet::free` est appelée sur celui-ci. Nous avons donc modifié cette méthode ; si le paquet est marqué avec une liste de `Trace_Computer_Loss` tous les éléments de cette liste sont notifiés. Donc si un paquet passant par un `TracePerso_Send` est détruit avant d'atteindre le `TracePerso_Recv` les `Trace_Computer_Loss` du `TracePerso_Send` en seront informés et le paquet sera considéré

comme perdu.

Les `Trace_Computer_Goodput` s'apparentent aux `Trace_Computer_Throughput` mais se placent au niveau application, c'est le seul type de `Trace_Computer` qui n'est pas rattaché à une `TracePerso`. Elle s'attache à un agent TCP (que nous avons modifié) et est notifié à chaque remontée d'information de l'agent vers l'application.

4.2.7 Differentiated Services

Dans les spécifications de 3GPP la méthode à utiliser pour appliquer la QoS requise par les QCI n'est pas définie. Il est tout de même fait mention de DiffServ comme exemple. Le fonctionnement de DiffServ est détaillé à la section 2.3. NS-2 possède certains modules permettant de simuler son comportement mais ne dispose pas de *scheduler* gérant les files WFQ, DRR, LLQ ou DRR. Nous avons bien trouvé un de ces *schedulers* sur Internet mais son fonctionnement ne nous a pas paru correct. Nous avons donc décidé de créer notre propre *scheduler* comprenant cinq files gérées par l'algorithme DWBBRR, et deux files PRIO. Parmi les files gérées par DWBBRR quatre sont pour les classes AF et une pour le DF. Une des files PRIO est utilisée pour le trafic EF et l'autre pour le trafic de contrôle. Nous avons choisi l'algorithme DWBBRR car il simule le BBRR au même titre que WFQ, tout en étant plus simple à implémenter. Contrairement à ce qu'on pourrait penser, dans NS-2 les modules permettant de simuler DiffServ se trouvent dans les liens, comme représenté à la figure 4.14.

Dans la suite de cette section nous allons détailler le fonctionnement de l'algorithme de *scheduling* que nous avons implémenté. Ce *scheduling* intervient lorsqu'un paquet peut être envoyé sur le lien, c'est à dire lorsque l'interface a fini d'envoyer le dernier paquet ou lorsqu'un nouveau arrive dans le module DiffServ et qu'aucun paquet n'est entrain d'être transmis. Lorsque notre *scheduler* est appelé il vérifie d'abord si la file PRIO réservée au trafic de contrôle du réseau est vide. Si ce n'est pas le cas, il la sert (envoi le premier paquet) sans autre considération. Si la première file PRIO est vide alors il réitère l'opération sur la seconde (utilisée pour le trafic EF). Si les deux sont vides alors il peut lancer le *scheduler* DWBBRR sur les autres.

Nous n'avons pas implémenté l'algorithme DWBBRR en utilisant le pseudo code de Shreedhar et Varghese [95] car cela aurait été trop gourmand en CPU. DWBBRR dans sa version basique, consiste à servir tour à tour chaque file avec n bits, n correspondant au poids de la file. L'algorithme s'arrête lorsque qu'une d'entre elles a reçu assez de bits pour envoyer son paquet de tête (nombre de bits reçus est supérieur ou égal à la taille du paquet). Celles qui sont vides ne sont pas servies et ne conservent pas leurs bits (reste de l'envoi d'un paquet). L'algorithme conserve en mémoire la dernière file qu'il a servi et repart au niveau de

la suivante lorsqu'il est relancé ; il ne repart pas de la première file à chaque fois qu'il envoie un paquet.

Nous avons implémenté notre algorithme de façon à ce qu'il se comporte exactement comme la version basique de DWBBRR décrite ci-dessus tout en limitant la charge CPU. Nous limitons l'algorithme à trois opérations par file : un tour de lecture, un tour d'écriture et un tour (non complet) d'écriture. Tout d'abord nous parcourons toute la liste pour déterminer pour chaque file le nombre de services nécessaires à l'envoi du paquet de tête. Par la même occasion nous déterminons le nombre de tours minimum n à effectuer pour envoyer un paquet. Nous servons chaque file avec $(n-1) \times P_i$ bits, P_i étant le poids de la file courante ; de cette façon nous simulons l'exécution de $n-1$ tours. Le dernier tour est effectué de façon basique, c'est à dire que l'algorithme sert (alloue P_i bits) tour à tour les files. La première servie, étant celle suivant la file ayant envoyé un paquet à la dernière exécution. Il s'arrête lorsqu'une d'entre elles possède assez de bits pour envoyer un paquet.

Le pseudo code de la procédure de sélection d'un paquet dans les files gérées par le *scheduler* DWBBRR est représenté à l'algorithme 3. Dans ce pseudo code chaque file contient une liste de paquets en attente et le nombre de bits qui lui ont déjà été attribués.

- La fonction *nextpkt* retourne le paquet de tête [9] de la liste des paquets d'une file.
- La fonction *size* retourne la taille d'un paquet en bits.
- La fonction *alreadyserved* retourne le nombre de bits qui ont déjà été attribués à une file.
- La fonction *weight* donne le poids d'une file.
- *lastQ* désigne la file par laquelle est sorti le dernier paquet.

4.3 Protocoles

Dans notre réseau les applications s'échangent des messages de contrôle pour connecter les UE, créer/modifier/détruire des *bearers* ou pour faire des relèves. Ces procédures sont représentées dans les annexes A à D. La plupart des messages utilisés proviennent de GTP-C, Diameter, S1 Application Protocol (S1-AP), NAS ou PMIPv6. Les protocoles de transport sont principalement UDP, TCP, et SCTP. Dans cette section nous allons décrire comment nous avons simulé quelques uns des principaux protocoles du réseau.

4.3.1 GTP

Pour simuler le protocole GTP nous nous sommes basés sur le code d'UDP en adaptant la taille des en-têtes et des paquets. Dans les applications nous avons ensuite défini la taille

9. Le paquet de tête d'une file est celui qui doit partir en premier.

Algorithme 3 Sélection du paquet sortant par notre algorithme DWBBRR

Si toutes les files sont vides **alors**
  on arrête là
Fin Si
/*on calcule le nombre minimum de tours à effectuer par un WBBRR basique pour servir un paquet*/
$min\_nombre\_tour = \infty$
Pour tout file non vide **faire**
  $nombre\_tour = \dfrac{\text{size}\,(\text{nextpkt}\,(file)) - \text{alreadyserved}\,(file)}{\text{weight}\,(file)}$
  Si $nombre\_tour < min\_nombre\_tour$ **alors**
    $min\_nombre\_tour = nombre\_tour$
  Fin Si
Fin Pour
/*on simule l'exécution de $min\_nombre\_tour - 1$ tours*/
Si $min\_nombre\_tour > 1$ **alors**
  Pour tout file non vide **faire**
    $\text{alreadyserved}\,(file) += (min\_nombre\_tour - 1) \times \text{weight}\,(file)$
  Fin Pour
Fin Si
/*on effectue le dernier tour*/
$file = lastQ$
Boucle infinie
  $file$ devient sa file non vide suivante
  $\text{alreadyserved}\,(file) += \text{weight}\,(file)$/*on sert la file*/
  Si $\text{size}\,(\text{nextpkt}\,(file)) \leq \text{alreadyserved}\,(file)$ **alors**
    supprimer $\text{nextpkt}\,(file)$ de $file$ et le sauvegarder dans $paquet\_tmp$
    Si $file$ est non vide **alors**
      $\text{alreadyserved}\,(file) -= \text{size}\,(\text{nextpkt}\,(file))$
    Sinon
      $\text{alreadyserved}\,(file) = 0$
    Fin Si
    $lastQ = file$
    retourner $paquet\_tmp$
  Fin Si
Fin Boucle infinie

des messages à envoyer sur le réseau en fonction des informations contenues dans ceux-ci. La plupart des messages GTP sont décrits dans le document 29.274 [22].

Les paquets GTP-U possèdent l'en-tête représenté à la figure 3.4 dans lequel est encapsulé le paquet de données. Cet en-tête fait 16 octets nous avons donc ajouté cette taille à l'en-tête UDP pour simuler GTP-U.

Les paquets GTP-C possèdent l'en-tête de 12 octets représentés à la figure 3.5, nous avons donc ajouté cette taille à celle de l'en-tête UDP pour simuler ce protocole. Les messages GTP-C (GTP-C PDU) encapsulent des blocs d'informations appelés IE. Chaque IE possède un en-tête représenté à la figure 3.6 ainsi que des informations spécifiques. Pour chaque message GTP-C simulé nous avons cherché les IE utilisés ou susceptibles de l'être pour déterminer sa taille. Les tailles des IE utilisés sont représentées dans les tableaux G.1 et G.2. Dans l'annexe G nous détaillons les tailles des messages du protocole GTP-C (ces tailles incluent les 12 octets de l'en-tête).

4.3.2 Diameter

Diameter se base sur les protocoles de transport TCP ou SCTP. Dans notre simulateur, bien qu'il existe une implémentation de SCTP nous ne l'avons pas utilisé car elle ne permet pas la transmission de données et la modifier aurait été trop complexe (cf. section 4.3.3). Nous avons donc simulé Diameter grâce à des agents TCP envoyant des messages de taille pré-calculée.

Un message Diameter se compose d'un en-tête standard et de nombreux sous messages, appelés AVP, eux-mêmes composés d'un en-tête et de données (pouvant être composées d'autres AVP). La taille d'un AVP (données + en-tête) est toujours un multiple de 4 octets.

Dans l'annexe H nous définissons la taille de chacun des messages Diameter que nous avons utilisé dans nos simulations. Celles-ci ont été calculées à partir du tableau H.1 généré grâce à Rigney *et al.* [92], Calhoun *et al.* [48], Loughney [74], Calhoun *et al.* [49] et Hakala *et al.* [66] et des documents 23.003 [2], 23.011 [3], 24.008 [10], 29.002 [15], 29.060 [16], 29.061 [17], 29.212 [18], 29.214 [19], 29.229 [20], 29.272 [21], 29.274 [22], 29.329 [25], 32.298 [26], 32.299 [27] et 32.422 [28].

4.3.3 SCTP

SCTP, défini par Stewart [96], est un protocole de transport assez récent (2000) par rapport à TCP (1981) et UDP (1980). Il a été développé dans le but de pallier aux imperfections de ces deux protocoles, il inclut donc plus de fonctionnalités et est plus tolérant aux pannes. Il est utilisé notamment sur les interfaces S6a et S1-MME, respectivement pour le transport des messages Diameter et S1-AP.

Pour Olsson *et al.* [81] SCTP comme TCP offre une transmission sûre, les données sont assurées d'arriver au niveau application de destination sans erreur. Ce sont des protocoles orientés connexion mais SCTP utilise une méthode de création de session en quatre étapes, contrairement à TCP qui utilise une méthode en trois étapes. SCTP utilise aussi une fonction d'adaptation de la bande passante semblable à celle du TCP. En revanche, il incorpore une méthode de gestion de message, ce que TCP [10] ne possède pas contrairement à UDP, c'est à dire que le protocole lui même sépare les messages à envoyer à l'application. En TCP, une session peut être vue comme un canal sur lequel on peut faire passer un flot de bits, c'est à l'application d'implémenter ses propres marqueurs de début et de fin de message.

SCTP permet aussi de faire du *multi-streaming* et du *multi-homing*, mais ces fonctionnalités ne nous intéressent pas pour nos simulateurs. Un résumé des différences et similarités importantes à nos yeux est représenté au tableau 4.2.

Tableau 4.2 – Comparaison entre SCTP, TCP et UDP (inspiré de Olsson *et al.* [81])

	SCTP	TCP	UDP
Orienté connection	Oui	Oui	Non
Transport sure	Oui	Oui	Non
Gestion des messages	Oui	Non	Oui
Respect de l'ordre des messages	Oui	Oui	Non
Non respect de l'ordre des messages	Oui	Non	Oui
Contrôle de flux et de congestion	Oui	Oui	Non
Multi-streaming	Oui	Non	Non
Multi-homing	Oui	Non	Non

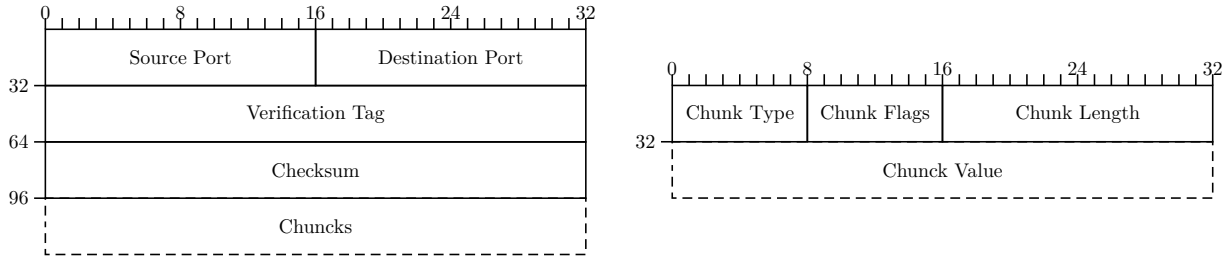
Nous avons choisi d'utiliser le code de TCP pour simuler SCTP. En effet, nous ne nous servons ni du multi-streaming, ni du multi-homing. Les seules différences importantes pour nous sont la taille des paquets et le nombre d'étapes à l'initialisation des connexions. Dans NS-2 il est facile de modifier la taille des en-têtes d'un protocole, le problème de la taille des paquets ne se pose donc pas. Concernant le problème du nombre d'étapes lors des connexions nous ne faisons qu'une seule connexion au démarrage du simulateur, par laquelle passe toutes nos données. Nous avons donc une période de mise en place du réseau, qui n'influence pas le comportement des connexions par la suite. Nos résultats ne dépendent donc pas de cette phase, dès lors le fait d'avoir des initialisations en quatre ou trois étapes importe peu.

Tout comme pour Diameter ou GTP, l'en-tête SCTP se compose d'une partie commune à tous les paquets et de *chunks* (voir figure 4.16), ces *chunks* pouvant transporter des données ou des informations de contrôles. Les deux en-têtes de *chunk* importants pour nous, c'est

10. Nos modifications sur le protocole TCP pour permettre la transmission de données font que celui-ci sépare lui-même les messages sur la destination.

à dire celui de transport de données et celui d'acquittement sont représentés à la figure 4.17 ; l'en-tête complet d'un paquet SCTP pour le passage de données fait 28 octets, les acquittements faisant quant à eux 32 octets.

En fait, les acquittements sont de tailles variables mais inclure cette caractéristique dans NS-2 aurait été trop compliquée, nous avons donc choisi de simuler des acquittements SCTP comportant un *Gap Ack Block*[11] et aucun *Duplicate* Transmission Sequence Numbers (TSN)[12]. Nous n'avons pas mis de TSN car nous n'utilisons qu'un seul chemin pour les paquets.



(a) En-tête commun du protocole SCTP

(b) En-tête de *chunk* du protocole SCTP

Figure 4.16 – En-tête du protocole SCTP (inspiré de Olsson *et al.* [81])

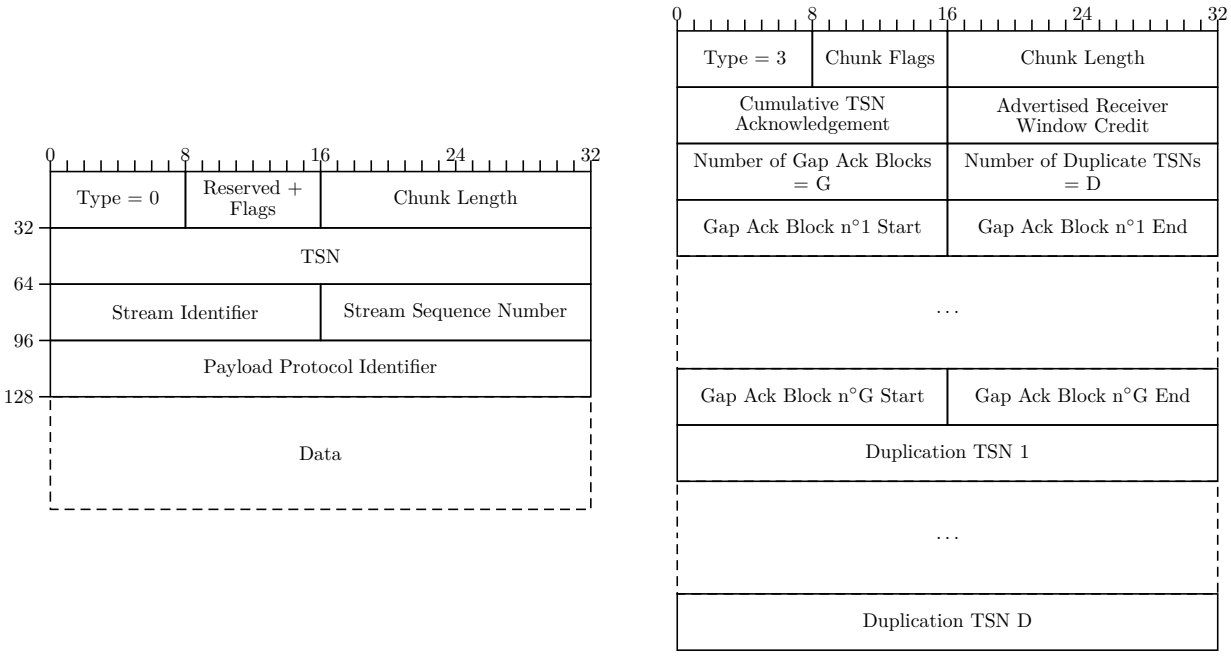
4.3.4 NAS

Le protocole NAS, défini dans le document 24.301 [11], se situe sur la couche NAS entre le UE et le MME, il est chargé de la gestion de la mobilité et des sessions dans cette partie du réseau. Il peut se diviser en deux sous parties, le EPS Session Management (ESM) et le EPS Mobility Management (EMM) ; le ESM gérant les *bearers* et le EMM la mobilité. Ce protocole s'insère au dessus de S1-AP entre le eNodeB et le MME, et RRC entre le UE et le eNodeB.

Pour nos simulations seule la taille des paquets qui transitent est importante, nous nous sommes donc intéressés aux en-têtes du protocole NAS. Un en-tête commun à tous les messages NAS est représenté à la figure 4.18, mais sa taille est variable (1 à 3 octets).

Le plus simple est de différentier les en-têtes NAS en deux catégories : celles utilisées par ESM et celles utilisées par EMM. En effet, comme l'indiquent les figures 4.19 et 4.20, on peut considérer que les paquets ESM ont des en-têtes de 3 octets tandis que les paquets EMM ont des en-têtes de 2 octets exceptés les « Service Request » avec 1 octet. Outre ces en-têtes communs, dans chaque paquet NAS sont incorporés différents IE dont les tailles sont définies dans le document 24.301 [11].

11. Un *Gap Ack Block* traduit l'acquittement d'un paquet arrivé après la perte d'un paquet.
12. Un *Duplicate* TSN informe la source de la duplication d'un paquet.



(a) passage de données (b) acquittement

Figure 4.17 – En-tête des *chunks* de passage de données et d'acquittement du protocole SCTP (inspiré de Olsson *et al.* [81])

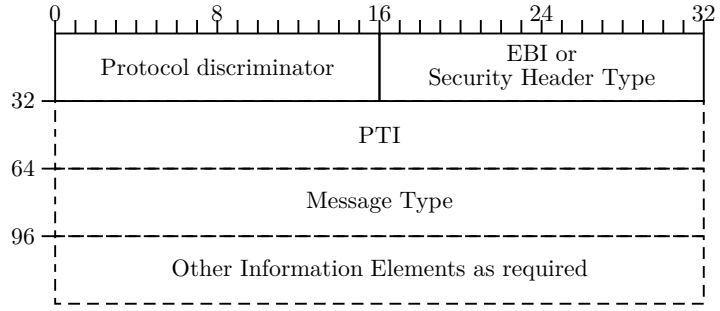
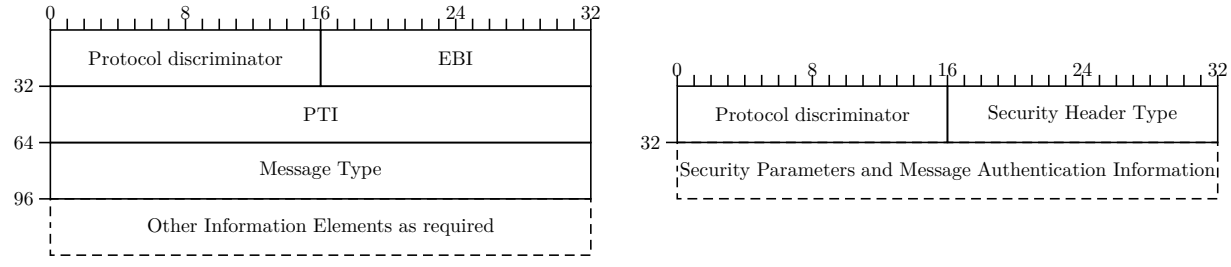


Figure 4.18 – En-tête commun à tous les paquets NAS (inspiré de 24.301 [11])



(a) En-tête normal (b) En-tête spécifique au « Service Request »

Figure 4.19 – En-têtes relatifs à ESM dans le protocole NAS (inspiré de Olsson *et al.* [81])

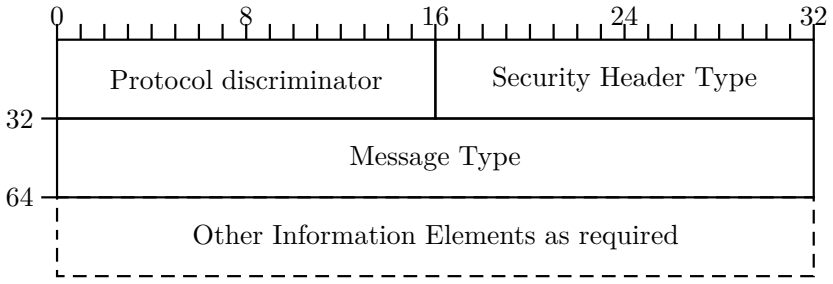


Figure 4.20 – En-tête relatif à EMM dans le protocole NAS (inspiré de Olsson *et al.* [81])

4.3.5 S1-AP

S1-AP n'est utilisé que sur l'interface S1-MME (cf. figure 2.10), il permet de transporter les messages du protocole NAS (dialogue entre le UE et le MME) et de faire communiquer le eNodeB avec le MME. Malheureusement pour nous, bien que le document 36.413 [34] définisse les IE du protocole, leur utilité et toutes les procédures, il ne définit pas le format des messages. Nous n'avons donc pas pu calculer la taille de ses messages.

4.3.6 Proxy Mobile IPv6

Tous les paquets de contrôle PMIPv6 contiennent un en-tête IPv6 classique représenté à la figure 3.18 et un *mobility header* schématisé à la figure 4.21. Un *mobility header* doit avoir une taille multiple de huit octets [72, sec 6.1.1]. Le tableau 4.3 liste les options que nous avons utilisées. Par la suite nous détaillons et calculons la taille des différents messages de contrôle que nous avons utilisés.

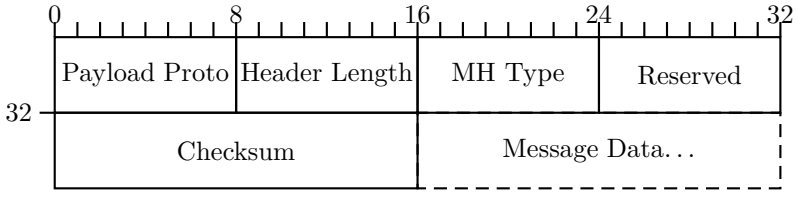


Figure 4.21 – Format des données contenues dans un paquet de contrôle PMIPv6 (*Mobility Header* [72, sec 6.1.1])

Proxy Binding Update (connexion d'un UE)

Le PBU est défini par Gundavelli *et al.* [65, sec 6.9.1.5], son en-tête est représenté à la figure 4.22, il contient :

- une option *Mobile Node Identifier* représentée à la figure 4.25 ;

Tableau 4.3 – Taille des options du protocole PMIPv6

options	taille en octets	notes
Mobile Node Identifier	3 + Identifier	nous avons pris comme identifier le IMSI (12 octets [22])
Prefix Information	32	
Handoff Indicator	4	
Access Technology Type	4	
Service Flow Identifier	8 + Service Flow Description	
Service Flow Description	variable	
GRE Key	8	
Flow Label	8	

- au moins une option *Home Network Prefix* de type *Prefix Information* représentée à la figure 4.26 ;
- une option *Handoff Indicator* représentée à la figure 4.24 ;
- une option *Access Technology Type* représentée à la figure 4.23 ;
- une option *Timestamp*, optionnelle ;
- une option *Mobile Node Link-layer Identifier*, optionnelle ;
- une option *Link-local Address*, optionnelle.

La figure 4.27 schématise les données transportées par les paquets IPv6 que nous avons utilisées, leur taille total est de 72 octets.

$$\left\lceil \frac{544}{8^2} \right\rceil 8 = 72 \tag{4.2}$$

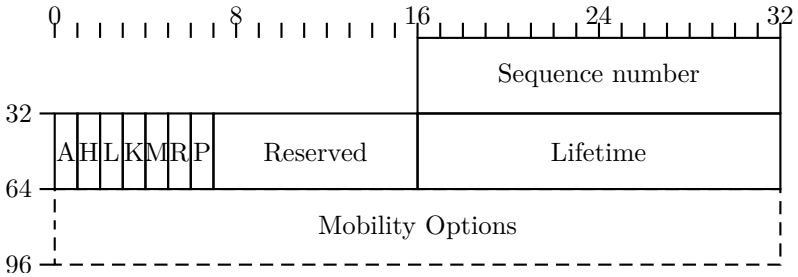


Figure 4.22 – En-tête d'un message Proxy Binding Update (inspiré de Gundavelli *et al.* [65, sec 8.1])

Dans notre modèle (dans les architectures 403 et 404), nous devons ajouter l'option *Flow Label* que nous avons définie et schématisée à la figure 3.21, celle-ci est inspirée de l'option

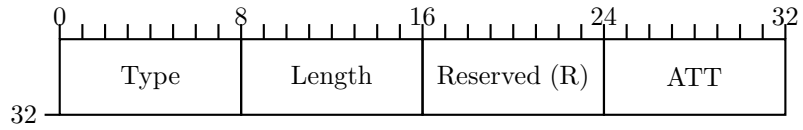


Figure 4.23 – Option *Access Technology Type* de PMIPv6 [65, sec 8.5]

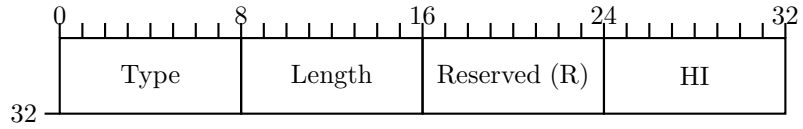


Figure 4.24 – Option *Handoff Indicator* de PMIPv6 [65, sec 8.4]

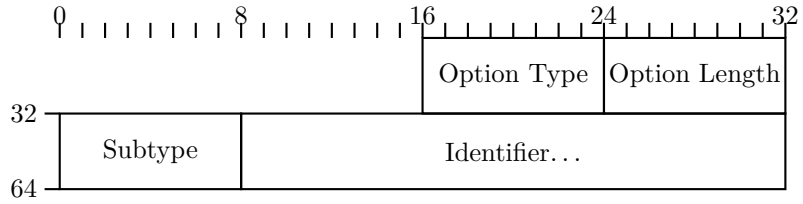


Figure 4.25 – Option *Mobile Node Identifier* de PMIPv6 [82, sec 3]

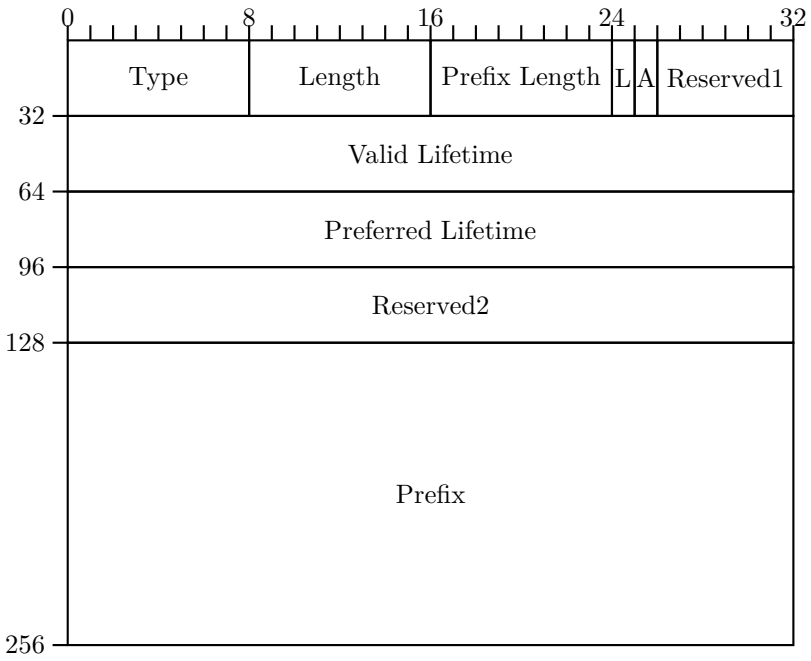


Figure 4.26 – Option *Prefix Information* de PMIPv6 [79, sec 4.6.2]

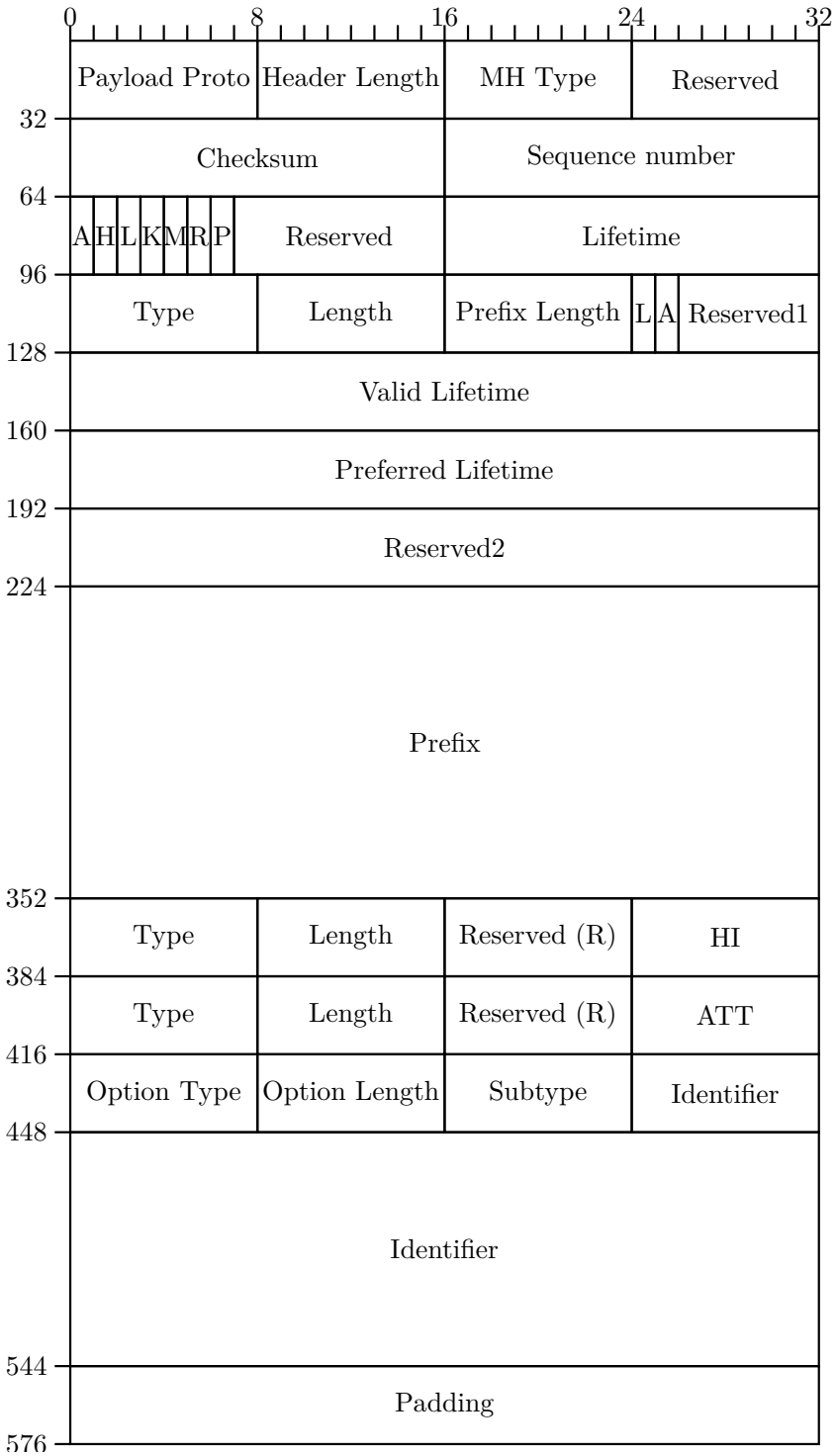


Figure 4.27 – Proxy Binding Update utilisé (inspiré de Gundavelli *et al.* [65, sec 8.1])

GRE *key* définie par Muhanna *et al.* [78]. La taille totale du PBU s'élève donc à 80 octets :

$$\left\lceil \frac{544 + 64}{8^2} \right\rceil 8 = 80 \tag{4.3}$$

Proxy Binding Acknowledgement (connexion d'un UE)

Le PBA est définie dans Gundavelli *et al.* [65, sec 5.3.6] il contient :

- une option *Mobile Node Identifier* représentée à la figure 4.25 ;
- au moins une option *Home Network Prefix* de type *Prefix Information* représentée à la figure 4.26 ;
- une option *Handoff Indicator* représentée à la figure 4.24 ;
- une option *Access Technology Type* représentée à la figure 4.23 ;
- une option *Timestamp*, optionnelle ;
- une option *Mobile Node Link-layer Identifier*, optionnelle ;
- une option *Link-local Address*, optionnelle.

La figure 4.28 schématise l'en-tête des données (sans le *Mobility Header*) du paquet IPv6 que nous avons utilisé. Les options étant les mêmes que pour le PBU et l'en-tête faisant la même taille, nous pouvons dire que dans les architectures 23.401 [8] et 23.402 [9] la taille totale d'un PBA (excluant l'en-tête IPv6) est de 72 octets.

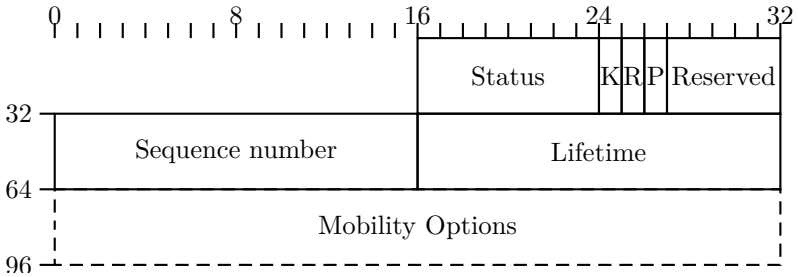


Figure 4.28 – En-tête d'un message Proxy Binding Acknowledgement (inspiré de Gundavelli *et al.* [65, sec 8.2])

Dans notre modèle la taille totale du PBA s'élève donc à 80 octets.

Proxy Binding Update pour la déconnexion

Le protocole PMIPv6 utilise le même paquet que pour la procédure de connexion d'un MN mais certains *flags* changent. Pour nous, cela n'a pas d'importance, la taille finale du paquet reste de 72 octets.

Proxy Binding Update pour la création de *bearer* dédié

Le PBU utilisé pour une création de *bearer* est composé des mêmes options que celles d'un PBU de connexion plus :

- une option *Flow Label* de 8 octets, représentée à la figure 3.21 ;
- une option Traffic Flow Template de 4 octets, représentée à la figure 3.19 ;
- au moins une option Packet Filter de 65 octets au maximum, représentée à la figure 3.20.

Nous pouvons calculer que le PBU utilisé pour une création de *bearer* fait en octets :

$$\left\lceil \frac{544 + 8 \times 8 + 4 \times 8 + 65 \times 8 \times \text{nombre de PFi}}{8^2} \right\rceil 8 \tag{4.4}$$

$$\text{soit } \left\lceil \frac{640 + 520 \times \text{nombre de PFi}}{64} \right\rceil 8 \tag{4.5}$$

Proxy Binding Acknowledgement pour la création de *bearer*

Le PBA utilisé lors de la création, modification ou destruction de *bearer* contient les mêmes informations que le PBU à partir duquel il a été généré. Ce PBA fait donc la même taille que le PBU.

PBU/PBA pour la modification de *bearer* dédié

Ces deux messages sont identiques à ceux utilisés lors de la création de *bearer* dédié. Le nombre de PFi est alors le nombre de PFi modifiés et non tous ceux du *bearer*. Si cette modification est une suppression de service (soit une suppression de PFi), l'option PFi ne fait plus que 3 octets. La taille du message est donc ramenée à :

$$\left\lceil \frac{544 + 8 \times 8 + 4 \times 8 + 3 \times 8 \times \text{nombre de PFi supprimés}}{8^2} \right\rceil 8 \tag{4.6}$$

$$\text{soit } \left\lceil \frac{640 + 24 \times \text{nombre de PFi supprimés}}{64} \right\rceil 8 \tag{4.7}$$

PBU/PBA pour la destruction de *bearer* dédié

Le PBU utilisé pour une destruction de *bearer* est composé des même options que celles d'un PBU de connexion plus une option *Flow Label* de 8 octets représentée à la figure 3.21. Nous pouvons calculer que le PBU utilisé pour une destruction de *bearer* fait en octets :

$$\left\lceil \frac{544 + 8 \times 8}{8^2} \right\rceil 8 = 80 \tag{4.8}$$

4.3.7 Liens Radio

Les protocoles RLC, RRC, MAC et PDCP respectivement définis dans les documents 25.322 [13] et 36.322 [31], 25.331 [14] et 36.331 [33], 25.321 [12] et 36.321 [30], et 36.323 [32] sont utilisés sur le lien radio UE-eNodeB. Notre simulation du réseau radio est assez basic nous ne nous sommes donc pas attardés sur ces protocoles et leur fonctionnement.

PDCP a la capacité de compresser les en-têtes IPv4 et IPv6, cependant cette fonctionnalité n'est pas obligatoirement utilisée, nous ne l'avons donc pas implémentée. Trois en-têtes PDCP sont définis, l'un de zéro octet (qui n'est donc pas, à proprement parler, un en-tête), l'un de un octet et un dernier de trois octets [90]. Nous avons choisi d'utiliser des en-têtes de trois octets pour simuler la présence de PDCP.

Plusieurs modes d'utilisation de RLC sont définis, pour le transfert de données [90] :

- transparent ;
- sans accusé de réception ;
- avec accusé de réception.

Nous avons choisi de simuler la présence d'en-tête RLC relatif au mode avec accusé de réception car il est légèrement plus grand que les autres, il fait deux octets. Nous pouvons remarquer que depuis la septième version des spécifications 3GPP la taille des PDU RLC est variable de manière à éviter les segmentations inutiles.

La couche RRC semble ne pas définir d'en-tête spécifique nous n'avons donc pas simulé son fonctionnement.

Le protocole MAC possède un en-tête de taille variable mais nous avons remarqué l'utilisation de quelques *flags* et d'un champ UE-Id de 32 bits au maximum. Nous avons donc choisi une taille de 5 octets pour l'en-tête du protocole MAC.

CHAPITRE 5

CONCLUSION

Le présent mémoire porte sur la création d'architectures de réseau mobile 4G basé sur les architectures développées par le consortium 3GPP, de manière à maximiser l'utilisation des capacités du protocole IPv6. Nous avons aussi comparé les performances des architectures proposées avec celles développées par 3GPP. Dans la première section de ce chapitre de conclusion nous effectuons une synthèse de nos travaux. Dans la deuxième section nous décrivons les limitations de nos propositions et des méthodes utilisées. Enfin, nous terminons ce chapitre par une description des évolutions futures qui pourraient s'appliquer à nos travaux.

5.1 Synthèse des travaux

Dans le cadre de cette maitrise nous avons proposé des modifications à appliquer au réseau mobile développé par 3GPP et décrit dans le document 23.402 [9]. Ces modifications ayant pour but de maximiser l'utilisation des capacités du protocole IPv6 et donc minimiser l'utilisation du protocole GTP. Dans l'architecture 403 proposée, l'utilisation de GTP est conservée entre les eNodeB et les SGW tandis que le PGW et le SGW sont transformés en vrai LMA et MAG ; la gestion des *bearers* est aussi déléguée à PMIPv6. Dans l'architecture 404 proposée, l'utilisation de PMIPv6 est étendue entre les eNodeB et les SGW.

Les performances des réseaux mobiles sont critiques étant donné qu'ils ne disposent pas d'autant de bande passante que les autres types de réseaux, et que les utilisateurs subissent des relèves. De plus dans les architectures 4G tous les trafics transitent sur un même réseau, ce qui accroit les risques de dégrader les performances des trafics temps réel comme la voix et la vidéo.

Pour vérifier que nos propositions ne dégradent pas les performances de l'architecture 23.402 [9], et pour évaluer les celles de l'architecture 23.401 [8] par rapport aux autres, nous avons décidé de les implémenter dans le simulateur NS-2.

Les résultats obtenus nous montrent que les architectures que nous proposons ne modifient pas, voire améliorent très légèrement les performances de la 23.402 [9]. En revanche la 23.401 [8] reste de loin la plus performante, notamment en ce qui concerne la gigue des trafics temps réel et le taux de perte de paquets.

Nous supposons que l'architecture 23.401 [8] bénéficie de performances supérieures car elle ne possède pas de lien SGW-PCRF. En effet, l'apparition de ce lien dans les architectures

23.402 [9], 403 et 404 permet de paralléliser les diagrammes de séquences des procédures de réductions et destructions de *bearer*. Cela a pour effet de les accélérer, aussi les paquets qui passaient par ces *bearers* sont plus vite redirigés vers les *bearers* par défaut où ils sont susceptibles d'être perdus plus facilement. L'ajout du lien SGW-PCRF pourrait donc expliquer l'augmentation du taux de perte de paquets dans les architectures 23.402 [9], 403 et 404. De la même façon l'accélération des procédures de ces architectures pourrait justifier l'augmentation de la gigue. En effet, les flux changeant plus vite de *bearer*, le nombre de flux dans un *bearer* variant plus vite, les délais de leurs paquets peuvent changer rapidement, impliquant une gigue plus élevé.

Nous conseillons donc aux opérateurs désirant déployer un réseau 4G basé sur les architectures proposées par le consortium 3GPP d'utiliser celle décrite dans le document 23.401 [8]. Cependant, s'ils veulent intégrer des réseaux d'accès non développés par 3GPP, e.g. WiMAX, ils pourraient ne pas pouvoir utiliser cette architecture. Nous leur conseillons alors de choisir l'une de celles que nous proposons.

5.2 Limitations de la solution proposée

Les modifications que nous proposons d'apporter à l'architecture 23.402 [9] souffre de quelques problèmes et limitations. Premièrement les performances sont moins bonnes que celles de la 23.401 [8], ce qui pourrait rebuter les opérateurs à les adopter. De plus nous transformons les SGW (et les eNodeB dans la 404) en routeurs IP ce qui rend ces nœuds sans doute plus complexes et donc plus chers.

Une autre critique que nous pourrions nous faire et que nous avons dû modifier le protocole PMIPv6 pour pouvoir intégrer la gestion des *bearers* au niveau IP. Tant que ces modifications n'ont pas été standardisées par l'IETF nous retombons dans le même problème politique qui a amené 3GPP à créer l'architecture 23.402 [9]. C'est à dire qu'il est difficile de faire accepter aux opérateurs gérants des réseaux d'accès non basés sur des technologies 3GPP, que leurs trafics transitent sur le EPC en utilisant un protocole géré par un consortium d'industriels et non pas standardisé par l'IETF (ou simplement non standardisé par l'IETF dans notre cas).

Concernant l'implémentation des architectures, le simulateur NS-2 possède de nombreux défauts, e.g. la non gestion des couches de protocoles, l'impossibilité de faire transiter de vraies informations sur le réseau, etc. Nos simulations sont donc moins précises que celles que nous aurions obtenues avec NS-3 ou en utilisant un banc d'essai (*testbed*).

5.3 Améliorations futures

Notre projet n'est pas fixe, d'ailleurs le simulateur implémenté pour l'architecture 23.401 [8] est actuellement entrain d'être complété d'un Graphic User Interface (GUI) en vue d'être distribué avec les SASN vendues par Ericsson. De notre point de vue il serait utile d'améliorer ces simulateurs. Comme décrit précédemment NS-2 est assez limité, nous pourrions donc envisager une ré-implémentation sur NS-3 ou sur un *testbed*.

Wakikawa et Gundavelli [99] décrivent une méthode permettant d'utiliser des MN ne possédant qu'une pile IPv4 dans un réseau PMIPv6, voire de faire transiter les données entre le MAG et le LMA sur un réseau IPv4. Dans notre étude nous avons pris comme hypothèse que tous le réseaux étaient en IPv6, cependant la transition d'IPv4 à IPv6 sera sans doute très longue. Il serait donc intéressant d'étudier les modifications proposées par Wakikawa et Gundavelli [99] et de les appliquer à nos architectures, de manière à autoriser l'utilisation d'IPv4 sur les UE.

Nous pourrions aussi mener plus loin nos recherches de façon à mettre en lumière les causes de ces différences de performances entre l'architecture 23.401 [8] et les autres.

RÉFÉRENCES

[1] 03.60 (2002). *General Packet Radio Service (GPRS) ; Service description ; Stage 2*, 7.9.0 édition.

[2] 23.003 (2010). *Numbering, addressing and identification*, 9.2.0 édition.

[3] 23.011 (2009). *Technical realization of Supplementary Services*, 9.0.0 édition.

[4] 23.060 (2010). *General Packet Radio Service (GPRS) ; Service description ; Stage 2*, 9.4.0 édition.

[5] 23.101 (2009). *General Universal Mobile Telecommunications System (UMTS) architecture*, 9.0.0 édition.

[6] 23.107 (2009). *Quality of Service (QoS) concept and architecture*, 9.0.0 édition.

[7] 23.203 (2010). *Policy and charging control architecture*, 9.4.0 édition.

[8] 23.401 (2010). *General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access*, 9.4.0 édition.

[9] 23.402 (2010). *Architecture enhancements for non-3GPP accesses*, 9.4.0 édition.

[10] 24.008 (2010). *Mobile radio interface Layer 3 specification ; Core network protocols ; Stage 3*, 9.2.0 édition.

[11] 24.301 (2010). *Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS) ; Stage 3*, 9.2.0 édition.

[12] 25.321 (2010). *Medium Access Control (MAC) protocol specification*, 9.2.0 édition.

[13] 25.322 (2010). *Radio Link Control (RLC) protocol specification*, 9.1.0 édition.

[14] 25.331 (2010). *Radio Resource Control (RRC) ; Protocol specification*, 10.1.0 édition.

[15] 29.002 (2010). *Mobile Application Part (MAP) specification*, 9.1.0 édition.

[16] 29.060 (2010). *General Packet Radio Service (GPRS) ; GPRS Tunnelling Protocol (GTP) across the Gn and Gp interface*, 9.2.0 édition.

[17] 29.061 (2010). *Interworking between the Public Land Mobile Network (PLMN) supporting packet based services and Packet Data Networks (PDN)*, 9.2.0 édition.

[18] 29.212 (2010). *Policy and charging control over Gx reference point*, 9.2.0 édition.

[19] 29.214 (2010). *Policy and charging control over Rx reference point*, 9.3.0 édition.

[20] 29.229 (2010). *Cx and Dx interfaces based on the Diameter protocol ; Protocol details*, 9.1.0 édition.

[21] 29.272 (2010). *Evolved Packet System (EPS) ; Mobility Management Entity (MME) and Serving GPRS Support Node (SGSN) related interfaces based on Diameter protocol*, 9.2.0 édition.

[22] 29.274 (2010). *3GPP Evolved Packet System (EPS) ; Evolved General Packet Radio Service (GPRS) Tunnelling Protocol for Control plane (GTPv2-C) ; Stage 3*, 9.2.0 édition.

[23] 29.279 (2009). *Mobile IPv4 (MIPv4) based mobility protocols ; Stage 3*, 9.0.0 édition.

[24] 29.281 (2010). *General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U)*, 9.2.0 édition.

[25] 29.329 (2010). *Sh interface based on the Diameter protocol ; Protocol details*, 9.1.0 édition.

[26] 32.298 (2010). *Telecommunication management ; Charging management ; Charging Data Record (CDR) parameter description*, 9.3.0 édition.

[27] 32.299 (2010). *Telecommunication management ; Charging management ; Diameter charging applications*, 9.3.0 édition.

[28] 32.422 (2010). *Telecommunication management ; Subscriber and equipment trace ; Trace control and configuration management*, 9.0.1 édition.

[29] 36.300 (2010). *Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN) ; Overall description ; Stage 2*, 9.3.0 édition.

[30] 36.321 (2010). *Evolved Universal Terrestrial Radio Access (E-UTRA) ; Medium Access Control (MAC) protocol specification*, 9.2.0 édition.

[31] 36.322 (2010). *Evolved Universal Terrestrial Radio Access (E-UTRA) ; Radio Link Control (RLC) protocol specification*, 9.1.0 édition.

[32] 36.323 (2010). *Evolved Universal Terrestrial Radio Access (E-UTRA) ; Packet Data Convergence Protocol (PDCP) specification*, 9.0.0 édition.

[33] 36.331 (2010). *Evolved Universal Terrestrial Radio Access (E-UTRA) ; Radio Resource Control (RRC) ; Protocol specification*, 9.2.0 édition.

[34] 36.413 (2010). *Evolved Universal Terrestrial Radio Access Network (E-UTRAN) ; S1 Application Protocol (S1AP)*, 9.2.2 édition.

[35] 3GPP (2010). General packet radio service / enhanced data rates for global evolution. `http://www.3gpp.org/article/gprs-edge` accédé le 1 Septembre 2010.

[36] 3GPP (2010). LTE. `http://www.3gpp.org/LTE` accédé le 3 Septembre 2010.

[37] 3GPP (2010). *Overview of 3GPP Release 1999*. 3GPP, version 0.1.1 édition. `http://www.3gpp.org/ftp/Information/WORK_PLAN/Description_Releases/Rel-99_description_20100214.zip` accédé le 1 Septembre 2010.

[38] ALMES, G., KALIDINDI, S. et ZEKAUSKAS, M. (1999). A One-way Delay Metric for IPPM. RFC 2679, Internet Engineering Task Force.

[39] ALMES, G., KALIDINDI, S. et ZEKAUSKAS, M. (1999). A One-way Packet Loss Metric for IPPM. RFC 2680, Internet Engineering Task Force.

[40] ALMQUIST, P. (1992). Type of Service in the Internet Protocol Suite. RFC 1349, Internet Engineering Task Force.

[41] AMANTE, S., CARPENTER, B. et JIANG, S. (2010). Update to the IPv6 flow label specification. draft 4, IETF. `http://tools.ietf.org/html/draft-carpenter-6man-flow-update-04` accédé le 12 octobre 2010.

[42] ANOUARD RACHDI, M. (2007). *Optimisation des ressources de réseaux hétérogènes avec coeur de réseau MPLS*. Thèse de doctorat, EDSYS.

[43] BABIARZ, J., CHAN, K. et BAKER, F. (2006). Configuration Guidelines for DiffServ Service Classes. RFC 4594, Internet Engineering Task Force.

[44] BATES, R. J. (2002). *GPRS : general packet radio service*. McGraw-Hill Professional, illustrée édition.

[45] BLAKE, S., BLACK, D., CARLSON, M., DAVIES, E., WANG, Z. et WEISS, W. (1998). An Architecture for Differentiated Service. RFC 2475, Internet Engineering Task Force.

[46] BRADEN, B., CLARK, D., CROWCROFT, J., DAVIE, B., DEERING, S., ESTRIN, D., FLOYD, S., JACOBSON, V., MINSHALL, G., PARTRIDGE, C., PETERSON, L., RAMAKRISHNAN, K., SHENKER, S., WROCLAWSKI, J. et ZHANG, L. (1998). Recommendations on Queue Management and Congestion Avoidance in the Internet. RFC 2309, Internet Engineering Task Force.

[47] BRAND, A. et AGHVAMI, H. (2002). *Multiple Access Protocols for Mobile Communications GPRS, UMTS and Beyond*. Antony Rowe.

[48] CALHOUN, P., LOUGHNEY, J., GUTTMAN, E., ZORN, G. et ARKKO, J. (2003). Diameter Base Protocol. RFC 3588, Internet Engineering Task Force.

[49] CALHOUN, P., ZORN, G., SPENCE, D. et MITTON, D. (2005). Diameter Network Access Server Application. RFC 4005, Internet Engineering Task Force.

[50] CISCO (1999). *Congestion Management Overview*. Cisco. `http://www.cisco.com/en/US/docs/ios/12_2/qos/configuration/guide/qcfconmg.html` accédé le 5 Avril 2010.

[51] CISCO (1999). *Low Latency Queueing*. Cisco. `http://www.cisco.com/en/US/docs/ios/12_0t/12_0t7/feature/guide/pqcbwfq.html` accédé le 5 Avril 2010.

[52] CISCO (1999). *Weighted Random Early Detection (WRED)*. Cisco. `http://www.cisco.com/en/US/docs/ios/11_2/feature/guide/wred_gs.html` accédé le 4 Avril 2010.

[53] CONTA, A. et DEERING, S. (1998). Generic Packet Tunneling in IPv6 Specification. RFC 2473, Internet Engineering Task Force.

[54] DAHLMAN, E., PARKVALL, S., SKÖLD, J. et BEMING, P. (2008). *3G Evolution : HSPA and LTE for Mobile Broadband*. Elsevier, seconde édition.

[55] DAVIE, B., CHARNY, A., BENNET, J., BENSON, K., BOUDEC, J. L., COURTNEY, W., DAVARI, S., FIROIU, V. et STILIADIS, D. (2002). An Expedited Forwarding PHB (Per-Hop Behavior). RFC 3246, Internet Engineering Task Force.

[56] DEERING, S. et HINDEN, R. (1998). Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, Internet Engineering Task Force.

[57] DEKERIS, B., ADOMKUS, T. et BUDNIKAS, A. (2006). Assurance of video conference services with combination of weighted fair queueing scheduling discipline and low latency queue. *traffic*, 2, 4.

[58] DEMICHELIS, C. et CHIMENTO, P. (2002). IP Packet Delay Variation Metric for IP Performance Metrics (IPPM). RFC 3393, Internet Engineering Task Force.

[59] DEVARAPALLI, V. et DUPONT, F. (2007). Mobile IPv6 Operation with IKEv2 and the Revised IPsec Architecture. RFC 4877, Internet Engineering Task Force.

[60] DIJKSTRA, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1, 269–271. `http://www-m3.ma.tum.de/twiki/pub/MN0506/WebHome/dijkstra.pdf` accédé le 25 Octobre 2010.

[61] EBERSPÄCHER, J., BETTSTETTER, C., VÖGEL, H.-J. et HARTMANN, C. (2009). *GSM architecture, protocols and services*. John Wiley and Sons, troisième édition.

[62] EKSTROM, H. (2009). QoS control in the 3GPP evolved packet system. *IEEE Communications Magazine*, 47, 76–83.

[63] EKSTRÖM, H. (2009). QoS control in the 3GPP evolved packet system. *IEEE Communications Magazine*, 47, 76–83.

[64] FARINACCI, D., LI, T., HANKS, S., MEYER, D. et TRAINA, P. (2000). Generic Routing Encapsulation (GRE). RFC 2784, Internet Engineering Task Force.

[65] GUNDAVELLI, S., LEUNG, K., DEVARAPALLI, V., CHOWDHURY, K. et PATIL, B. (2008). Proxy Mobile IPv6. RFC 5213, Internet Engineering Task Force.

[66] HAKALA, H., MATTILA, L., KOSKINEN, J.-P., STURA, M. et LOUGHNEY, J. (2005). Diameter Credit-Control Application. RFC 4006, Internet Engineering Task Force.

[67] HEINANEN, J., BAKER, F., WEISS, W. et WROCLAWSKI, J. (1999). Assured Forwarding PHB Group. RFC 2597, Internet Engineering Task Force.

[68] HEINANEN, J. et GUERIN, R. (1999). A Single Rate Three Color Marker. RFC 2697, Internet Engineering Task Force.

[69] HEINANEN, J. et GUERIN, R. (1999). A Two Rate Three Color Marker. RFC 2698, Internet Engineering Task Force.

[70] HUI, M., CHEN, G. et DENG, H. (2010). Service flow identifier in proxy mobile IPv6. draft 3, IETF. `http://tools.ietf.org/html/draft-hui-netext-service-flow-identifier-03` accédé le 20 Septembre 2010.

[71] INETDOC LINUX (2009). *Gestionnaires de mise en file d'attente pour l'administration de la bande passante*. Linux France. `http://www.linux-france.org/prj/inetdoc/guides/Advanced-routing-Howto/lartc.qdisc.html` accédé le 11 Avril 2010.

[72] JOHNSON, D., PERKINS, C. et ARKKO, J. (2004). Mobility Support in IPv6. RFC 3775, Internet Engineering Task Force.

[73] LESCUYER, P. et LUCIDARME, T. (2008). *Evolved Packet System (EPS) : The LTE and SAE Evolution of 3G UMTS*. Wiley.

[74] LOUGHNEY, J. (2003). Diameter Command Codes for Third Generation Partnership Project (3GPP) Release 5. RFC 3589, Internet Engineering Task Force.

[75] MABE, K. J. (2004). LLQ in integrated services networks. *Southern African Telecommunication Networks and Applications Conference*. 0–0.

[76] MENONCIN, S. (2008). The need for QoS.

[77] MISHRA, A. R. (2010). *Cellular Technologies for Emerging Markets : 2G, 3G and Beyond*. John Wiley and Sons, première édition.

[78] MUHANNA, A., KHALIL, M., GUNDAVELLI, S. et LEUNG, K. (2010). Generic Routing Encapsulation (GRE) Key Option for Proxy Mobile IPv6. RFC 5845, Internet Engineering Task Force.

[79] NARTEN, T., NORDMARK, E., SIMPSON, W. et SOLIMAN, H. (2007). Neighbor Discovery for IP version 6 (IPv6). RFC 4861, Internet Engineering Task Force.

[80] NICHOLS, K., BLAKE, S., BAKER, F. et BLACK, D. (1998). Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. RFC 2474, Internet Engineering Task Force.

[81] OLSSON, M., SULTANA, S., ROMMERM, S., FRID, L. et MULLIGAN, C. (2009). *SAE and the Evolved Packet Core : Driving the mobile broadband revolution*. Elsevier.

[82] PATEL, A., LEUNG, K., KHALIL, M., AKHTAR, H. et CHOWDHURY, K. (2005). Mobile Node Identifier Option for Mobile IPv6 (MIPv6). RFC 4283, Internet Engineering Task Force.

[83] PATEL, A., LEUNG, K., KHALIL, M., AKHTAR, H. et CHOWDHURY, K. (2006). Authentication Protocol for Mobile IPv6. RFC 4285, Internet Engineering Task Force.

[84] PAXSON, V., ALMES, G., MAHDAVI, J. et MATHIS, M. (1998). Framework for IP Performance Metrics. RFC 2330, Internet Engineering Task Force.

[85] PERKINS, C. (2002). IP Mobility Support for IPv4. RFC 3344, Internet Engineering Task Force.

[86] PIERRE, S. (2003). *Réseaux et systèmes informatiques mobiles : Fondements, architectures et applications*. Presses inter Polytechnique.

[87] PINOLA, J. et PENTIKOUSIS, K. (2008). Mobile wimax. *The Internet Protocol Journal*, 11, 19–35. `http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_11-2/ipj_11-2.pdf` accédé le 10 Novembre 2010.

[88] POSTEL, J. (1981). Internet Protocol. RFC 0791, Internet Engineering Task Force.

[89] PRIGGOURIS, G., HADJIEFTHYMIADES, S. et MERAKOS, L. (2000). IP QoS frameworks in GPRS. `http://alexandra.di.uoa.gr/Dienst/UI/2.0/Describe/ncstrl.uoa_cs/TR00-0001` accédé le 2 Septembre 2010.

[90] PROTOCOLS.COM (2010). *UMTS Family Protocol Information*. Protocols.com. `http://www.protocols.com/pbook/UMTSFamily.htm` accédé le 28 Octobre 2010.

[91] RAMAKRISHNAN, K., FLOYD, S. et BLACK, D. (2001). The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168, Internet Engineering Task Force.

[92] RIGNEY, C., WILLENS, S., RUBENS, A. et SIMPSON, W. (2000). Remote Authentication Dial In User Service (RADIUS). RFC 2865, Internet Engineering Task Force.

[93] SCHULZRINNE, H., CASNER, S., FREDERICK, R. et JACOBSON, V. (2003). RTP : A Transport Protocol for Real-Time Applications. RFC 3550, Internet Engineering Task Force.

[94] SHALUNOV, S. et SWANY, M. (2010). Reporting IP performance metrics to users. draft 5, IETF. `http://tools.ietf.org/html/draft-ietf-ippm-reporting-05` accédé le 4 août 2010.

[95] SHREEDHAR, M. et VARGHESE, G. (1996). Efficient fair queuing using deficit round-robin. *IEEE/ACM Transactions on Networking*, 4, 375–385.

[96] STEWART, R. (2007). Stream Control Transmission Protocol. RFC 4960, Internet Engineering Task Force.

[97] The VINT Project (2009). *The ns Manual (formerly ns Notes and Documentation)*. The VINT Project, seconde édition. `http://sourceforge.net/projects/nsnam/files/ns-2/2.34/` accédé le 21 Octobre 2010.

[98] VEGA GARCIA, A. et HUITEMA, C. (1996). *Mecanismes de Controle pour la Transmission de l'Audio sur l'Internet*. Thèse de doctorat, University of Nice-Sophia Antipolis.

[99] WAKIKAWA, R. et GUNDAVELLI, S. (2010). IPv4 Support for Proxy Mobile IPv6. RFC 5844, Internet Engineering Task Force.

[100] WALLINGFORD, T. (2005). *Switching to VoIP*. O'Reilly, première édition.

[101] WEIL, J., KUARSINGH, V. et DONLEY, C. (2010). IANA reserved IPv4 prefix for IPv6 transition. draft 1, IETF. `http://tools.ietf.org/html/draft-weil-opsawg-provider-address-space-01` accédé le 8 septembre 2010.

[102] WiMAX Forum® (2009). *Network Architecture : Architecture Tenets, Reference Model and Reference Points*. WiMAX Forum®, release 1.0 version 4 - stage 2 édition. `http://www.wimaxforum.org/resources/documents/technical/T32` accédé le 1 Septembre 2010.

[103] WiMAX Forum® (2009). *Network Architecture : Detailed Protocols and Procedures*. WiMAX Forum®, release 1.0 version 4 - stage 3 édition. `http://www.wimaxforum.org/sites/wimaxforum.org/files/technical_document/2009/07/WMF-T33-001-R010v04_Network-Stage3-Base.pdf` accédé le 10 Novembre 2010.

[104] ZHANG, H. et FERRARI, D. (1993). Rate-controlled static-priority queueing. *IEEE INFOCOM*. 227–236.

ANNEXE A

Diagrammes de séquences de l'architecture 23.401

Les figures suivantes représentent les diagrammes de séquences des principales fonctions de gestion de bearer dans l'architecture 23.401 [8]. Les informations utilisées pour obtenir ces diagrammes sont tirées des documents 23.401 [8], 23.203 [7] et 36.300 [29].

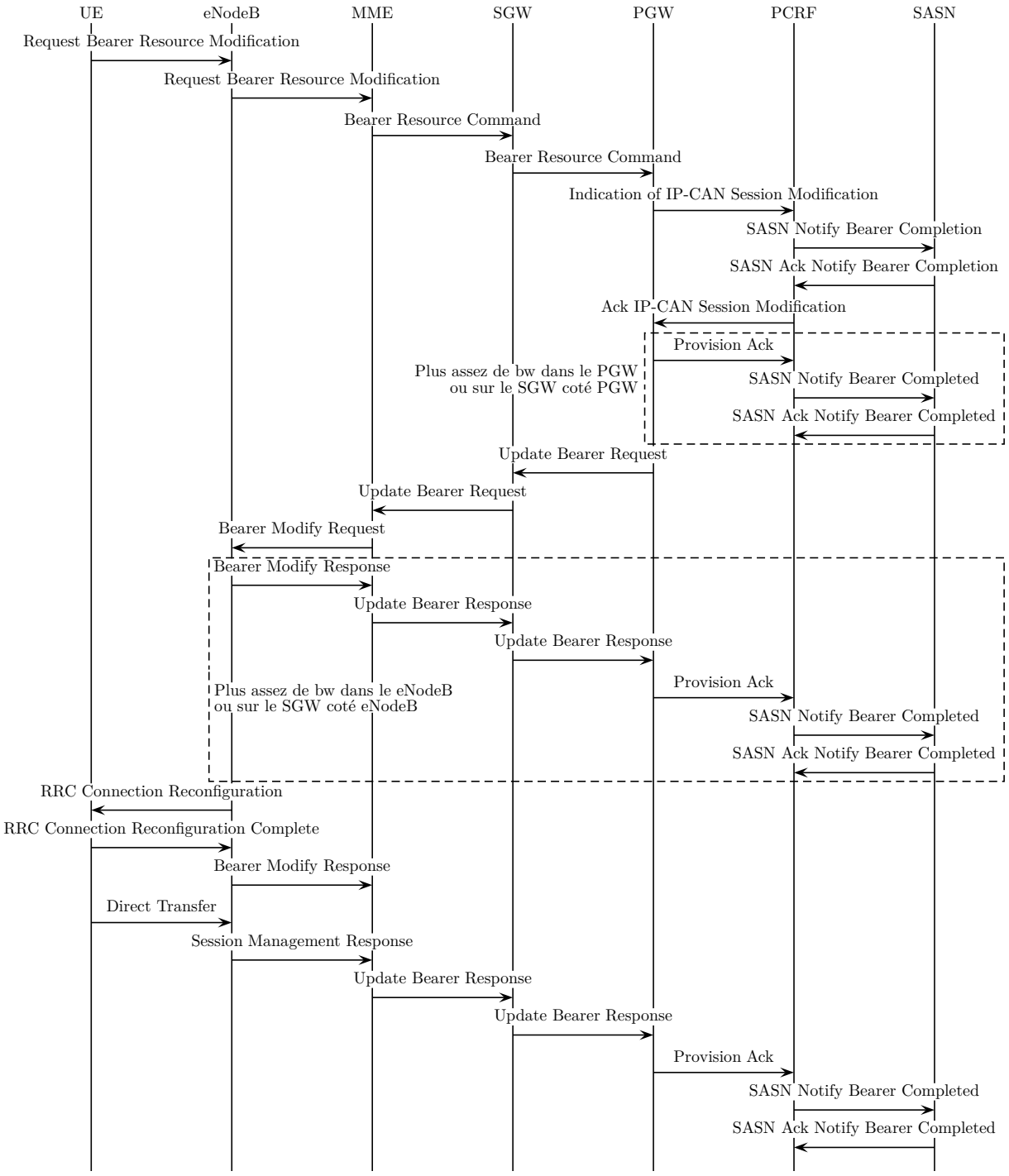


Figure A.1 – Complétion de *bearer* dédié GBR avec UE QoS *aware*

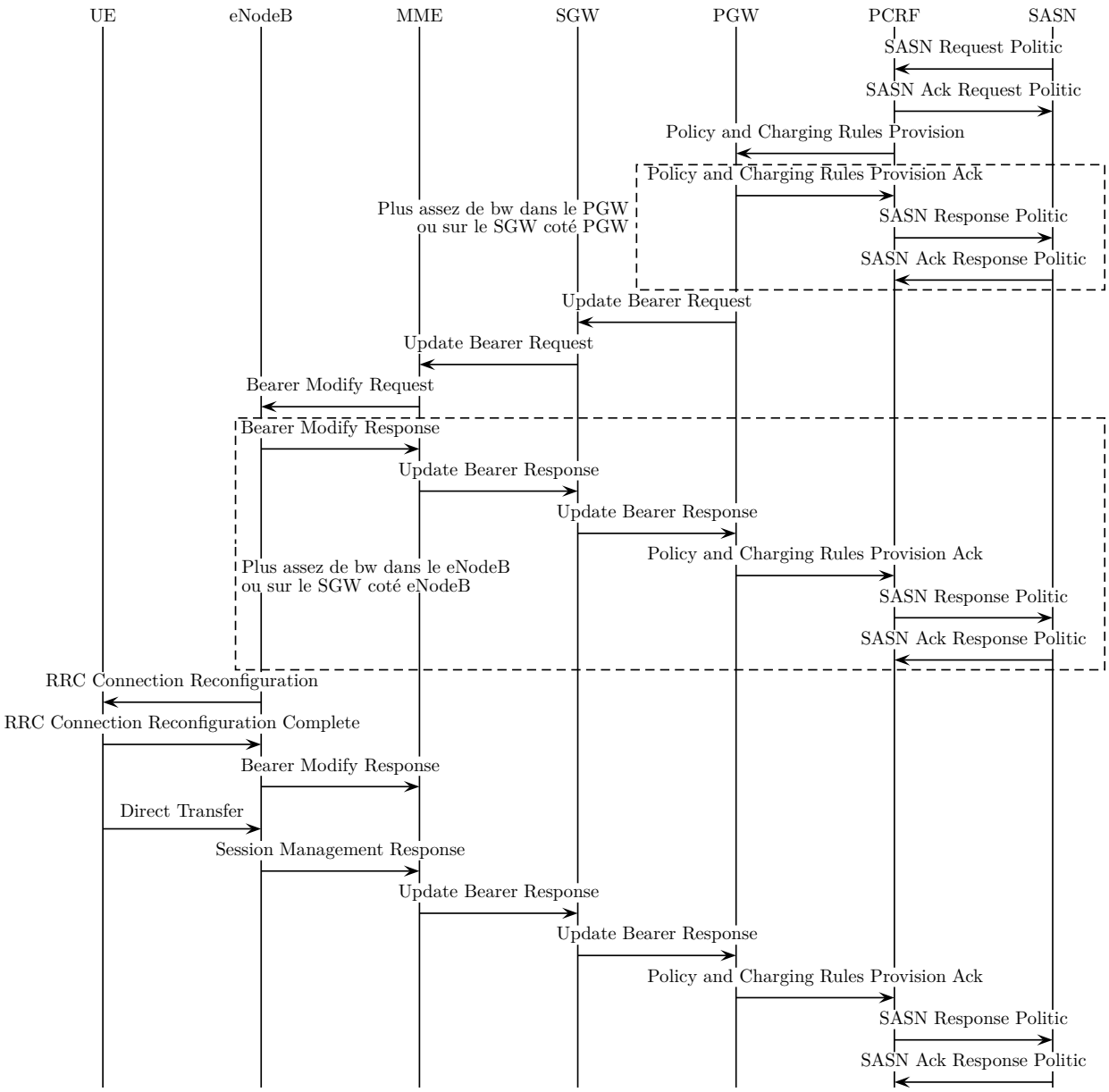


Figure A.2 – Complétion de *bearer* dédié GBR avec UE QoS *unaware*

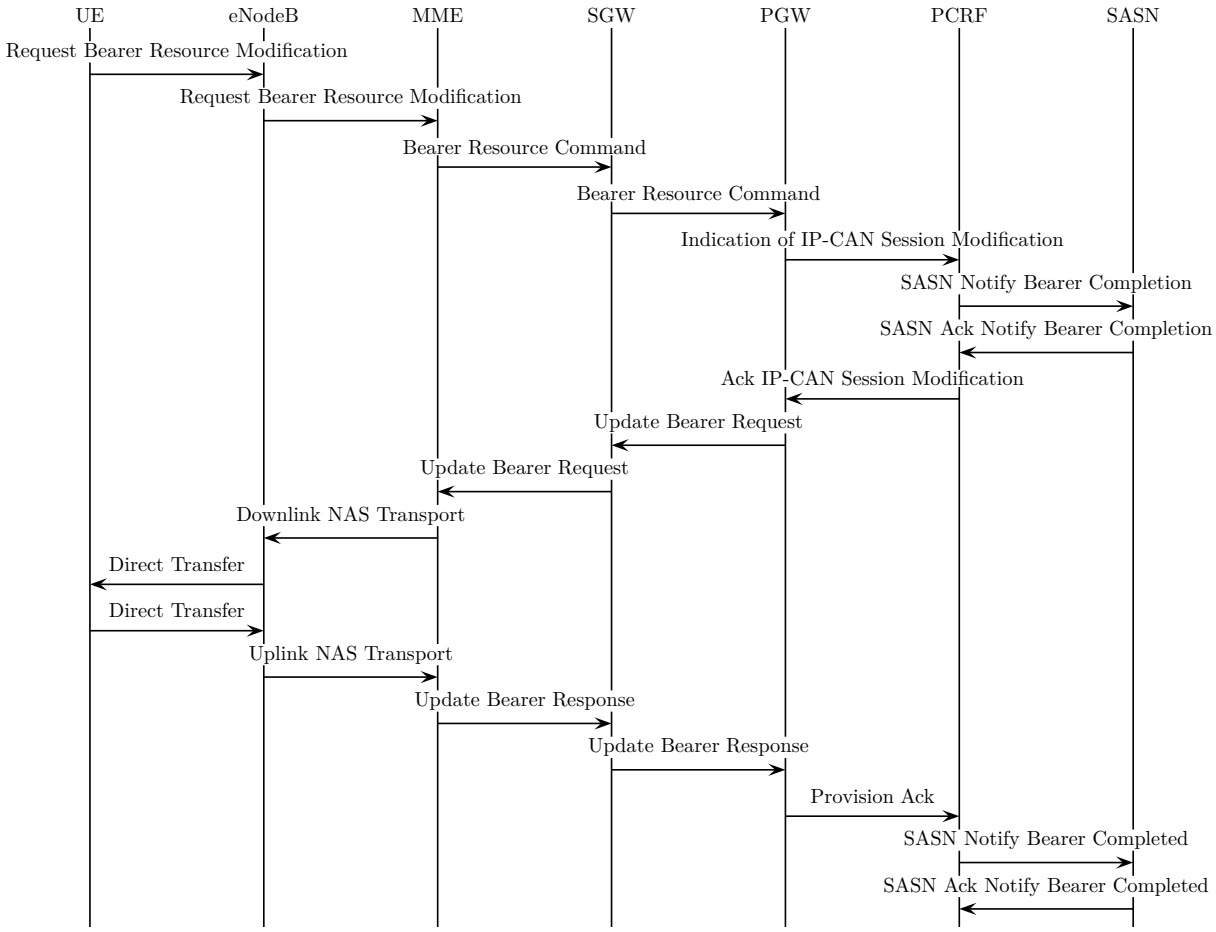


Figure A.3 – Complétion de *bearer* dédié non GBR avec UE QoS *aware*

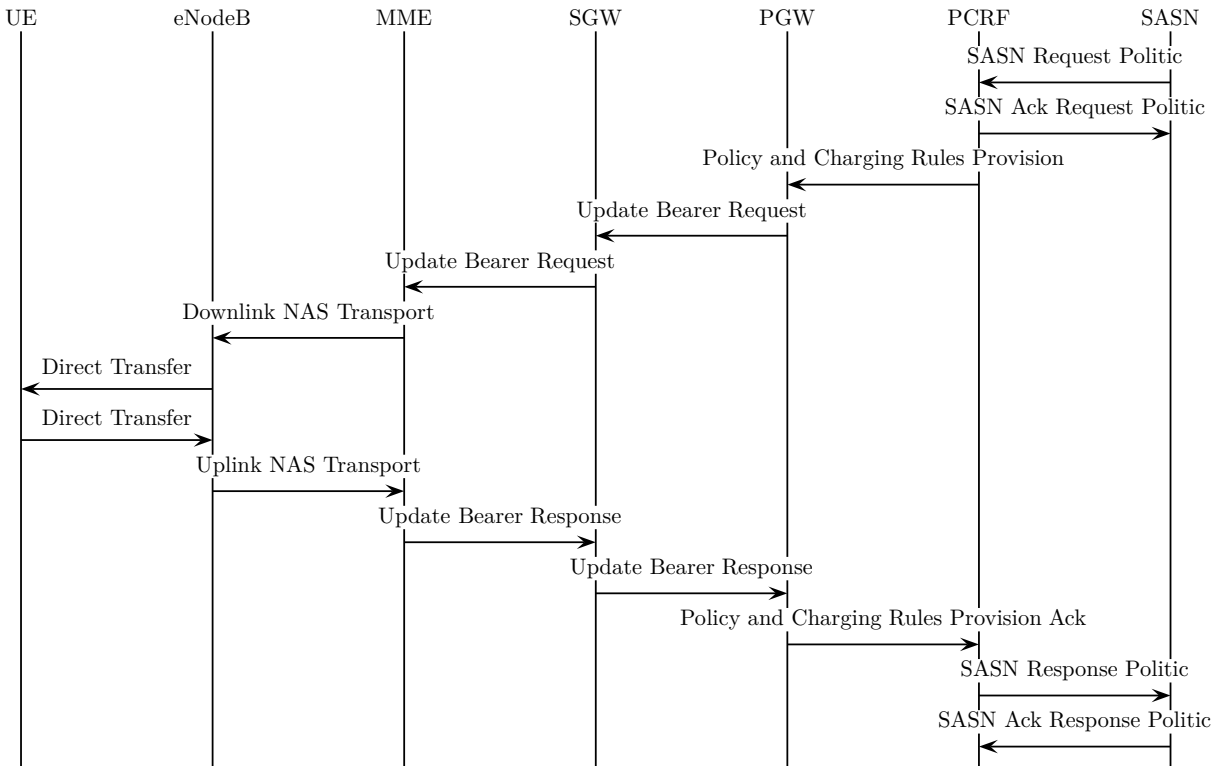


Figure A.4 – Complétion de *bearer* dédié non GBR avec UE QoS *unaware*

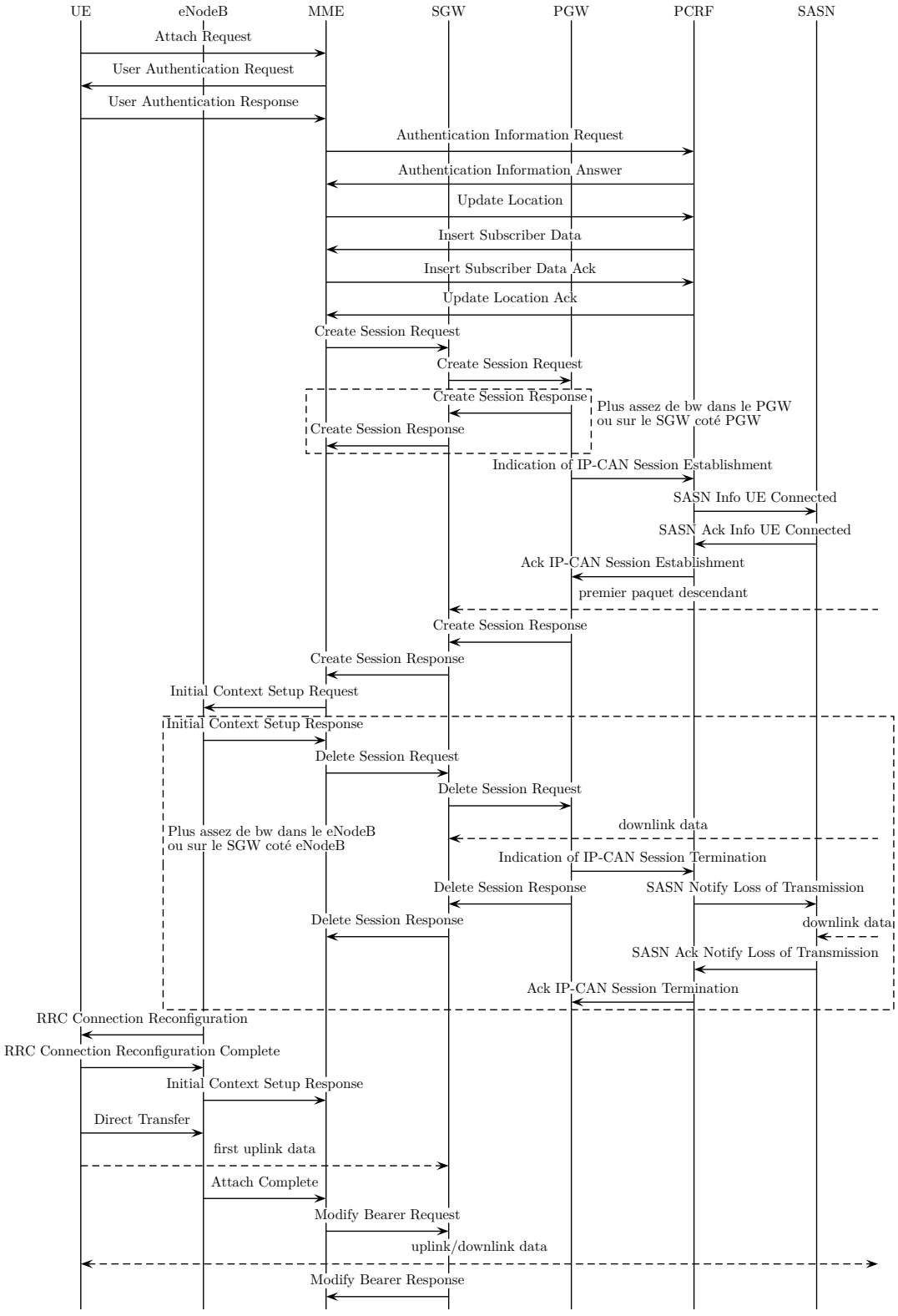


Figure A.5 – Connexion d'un UE

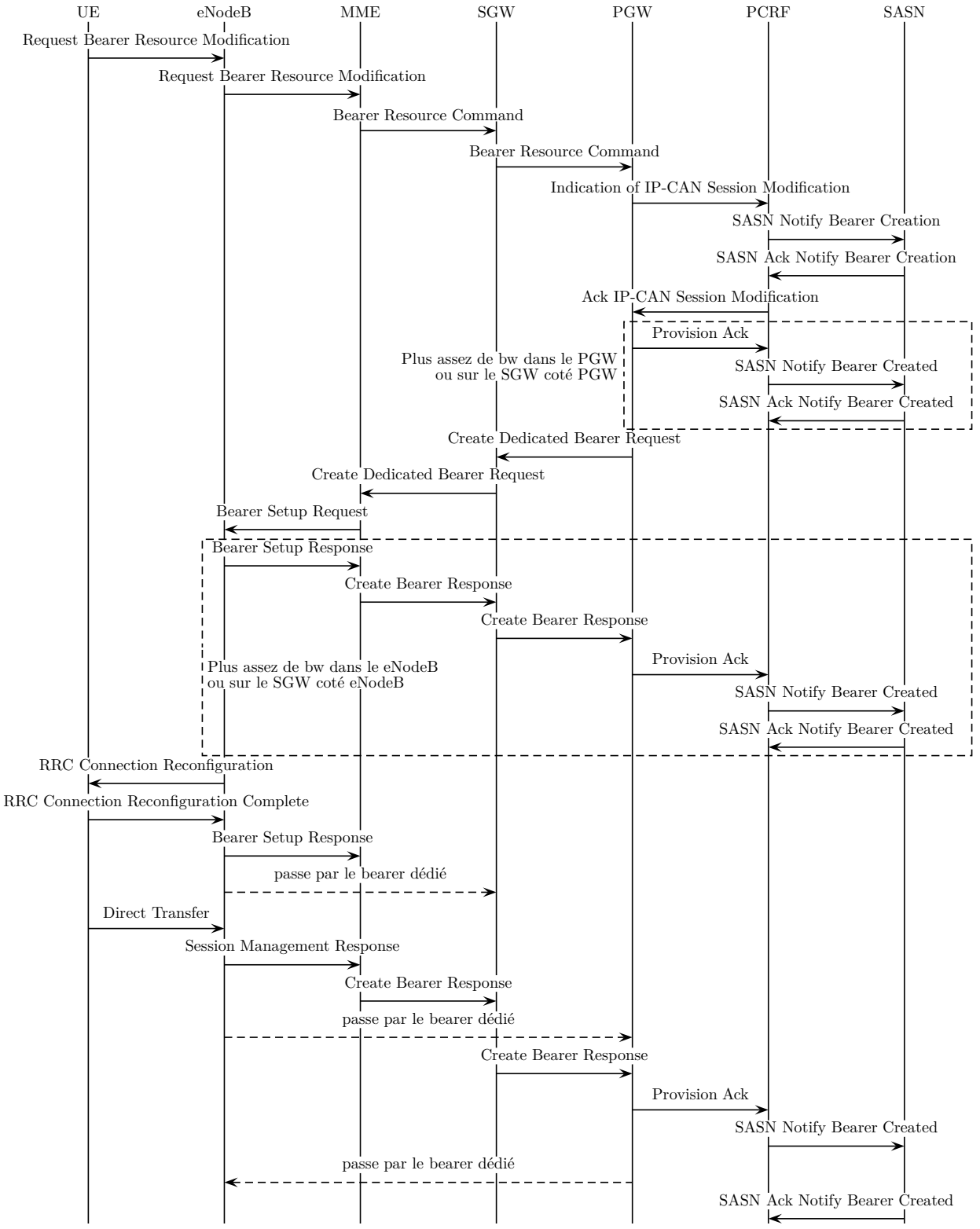


Figure A.6 – Création de *bearer* dédié avec UE QoS *aware*

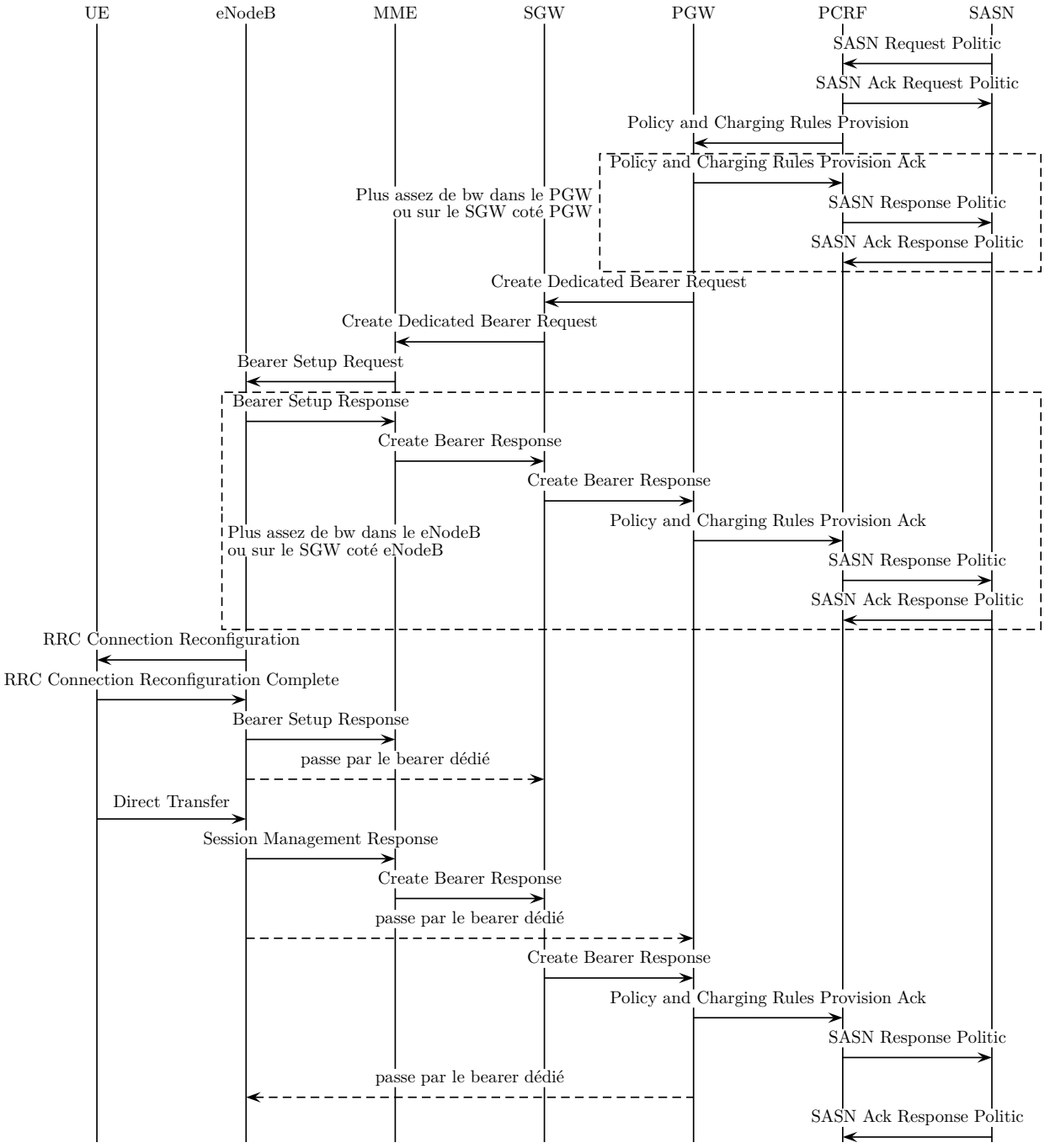


Figure A.7 – Création de *bearer* dédié avec UE QoS *unaware*

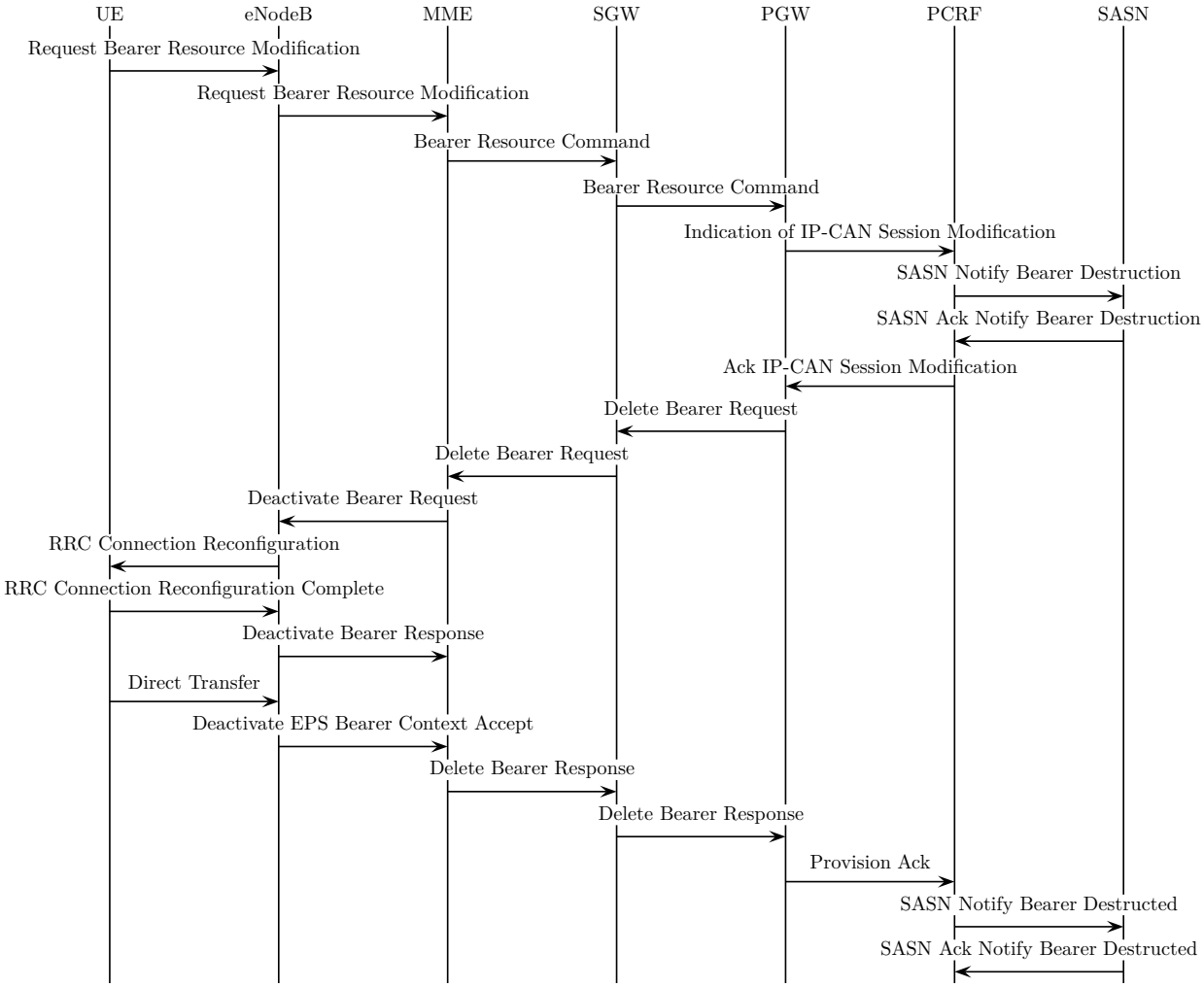


Figure A.8 – Destruction de *bearer* dédié avec UE QoS *aware*

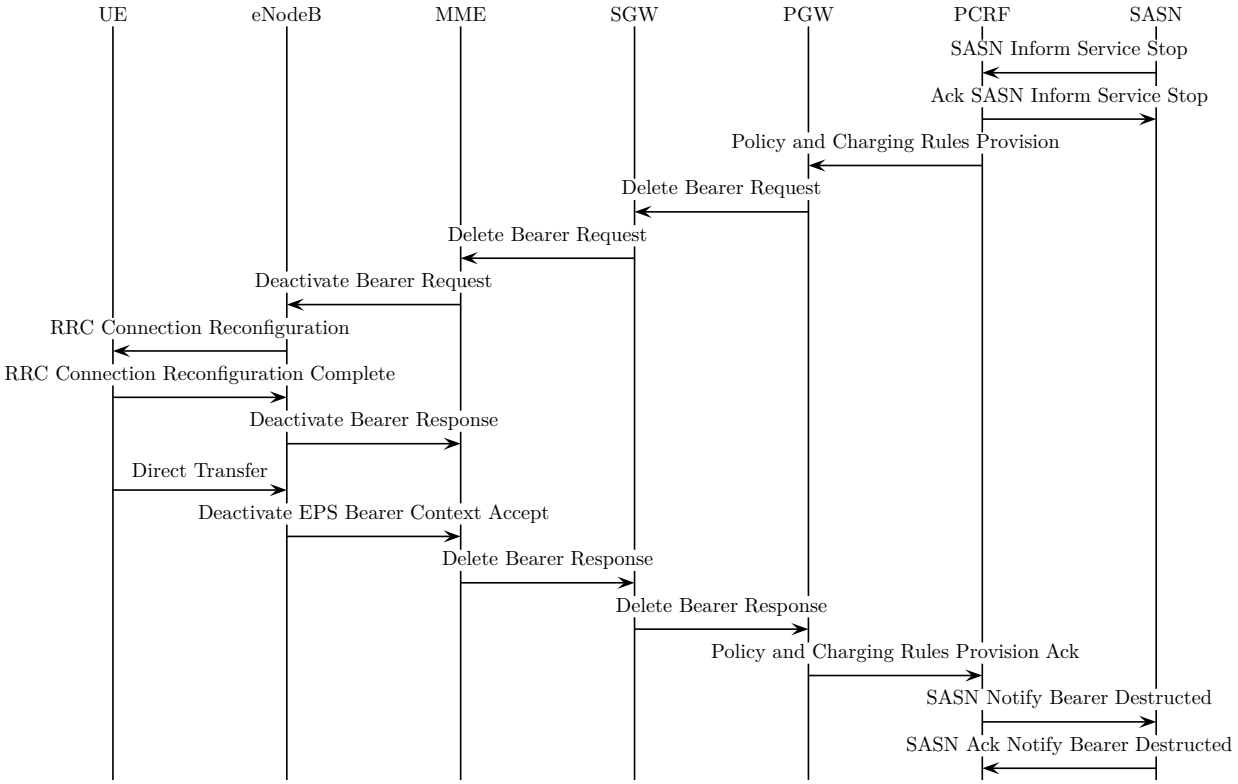


Figure A.9 – Destruction de *bearer* dédié avec UE QoS *unaware*

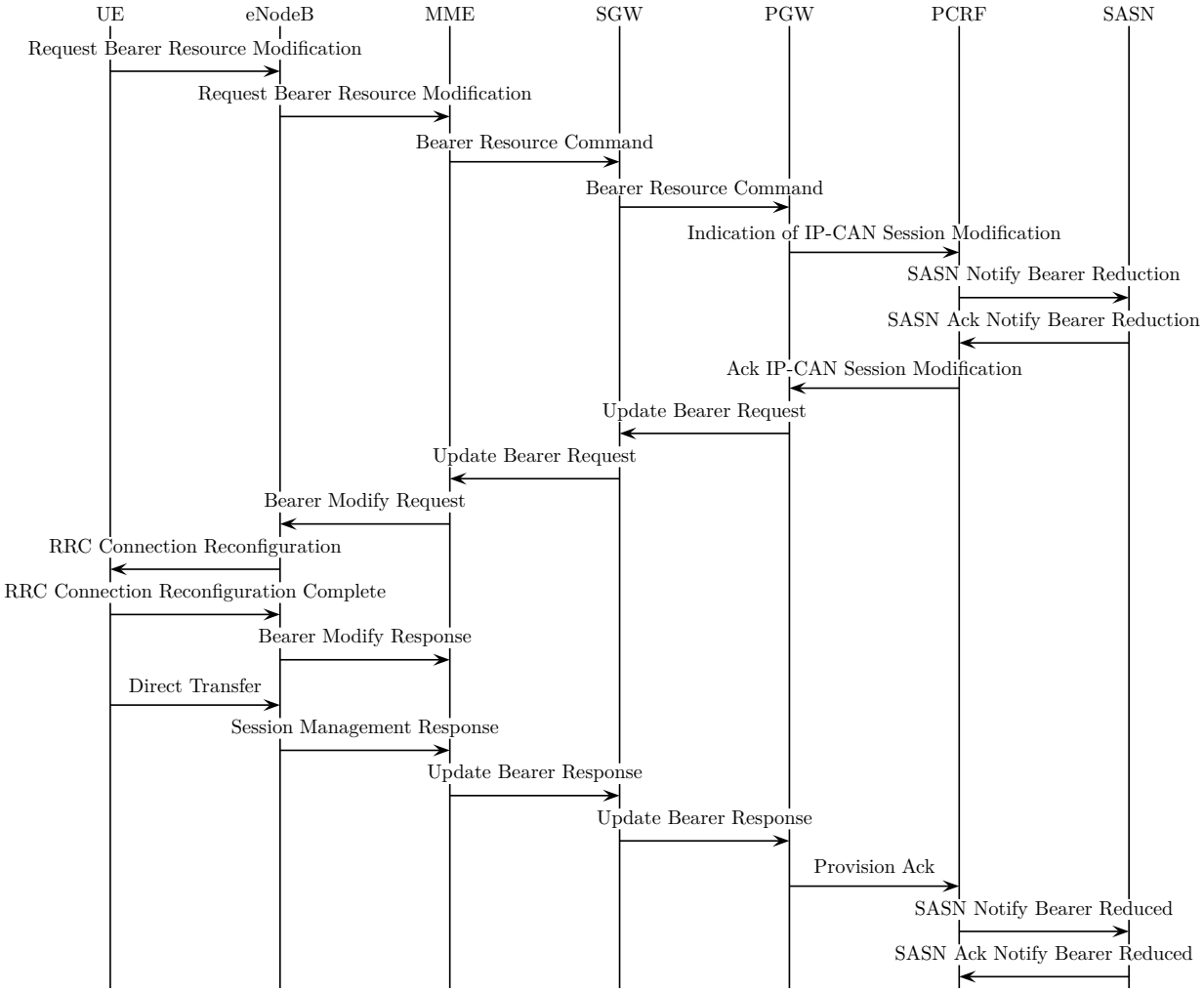


Figure A.10 – Réduction de *bearer* dédié GBR avec UE QoS *aware*

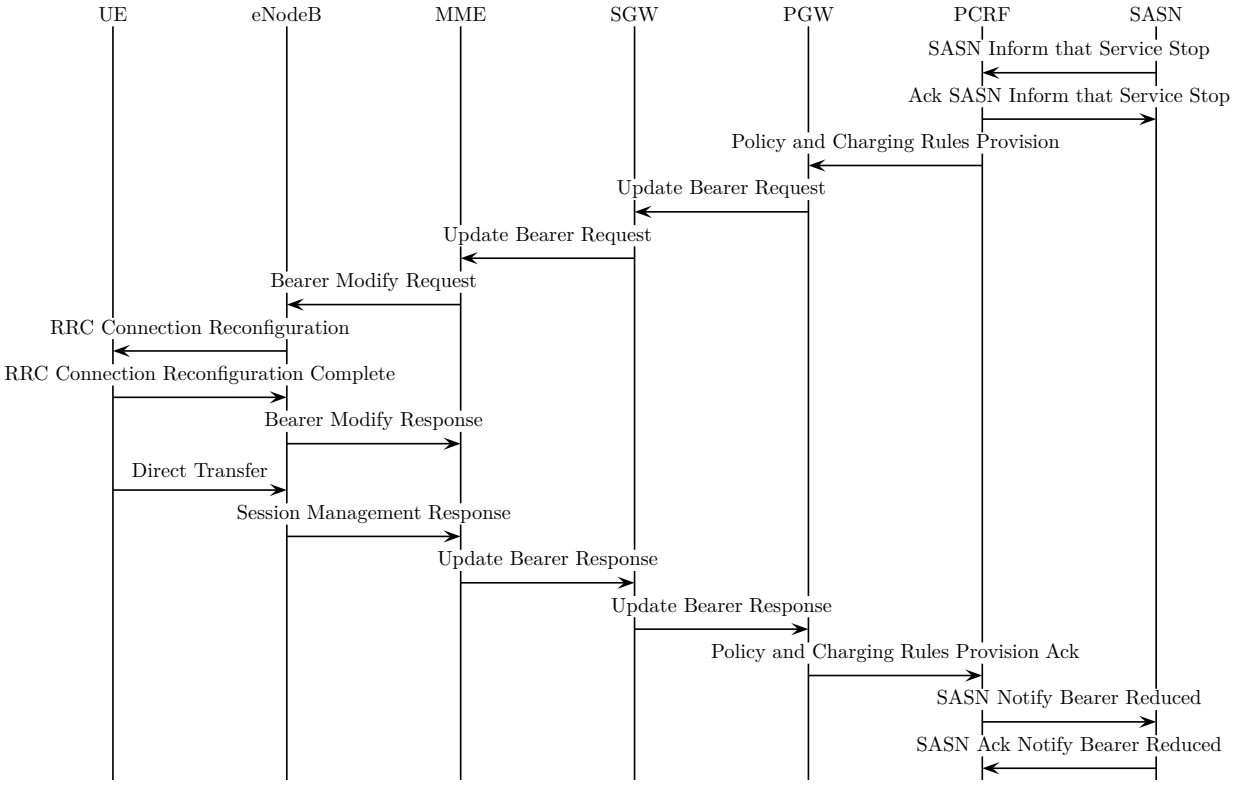


Figure A.11 – Réduction de *bearer* dédié GBR avec UE QoS *unaware*

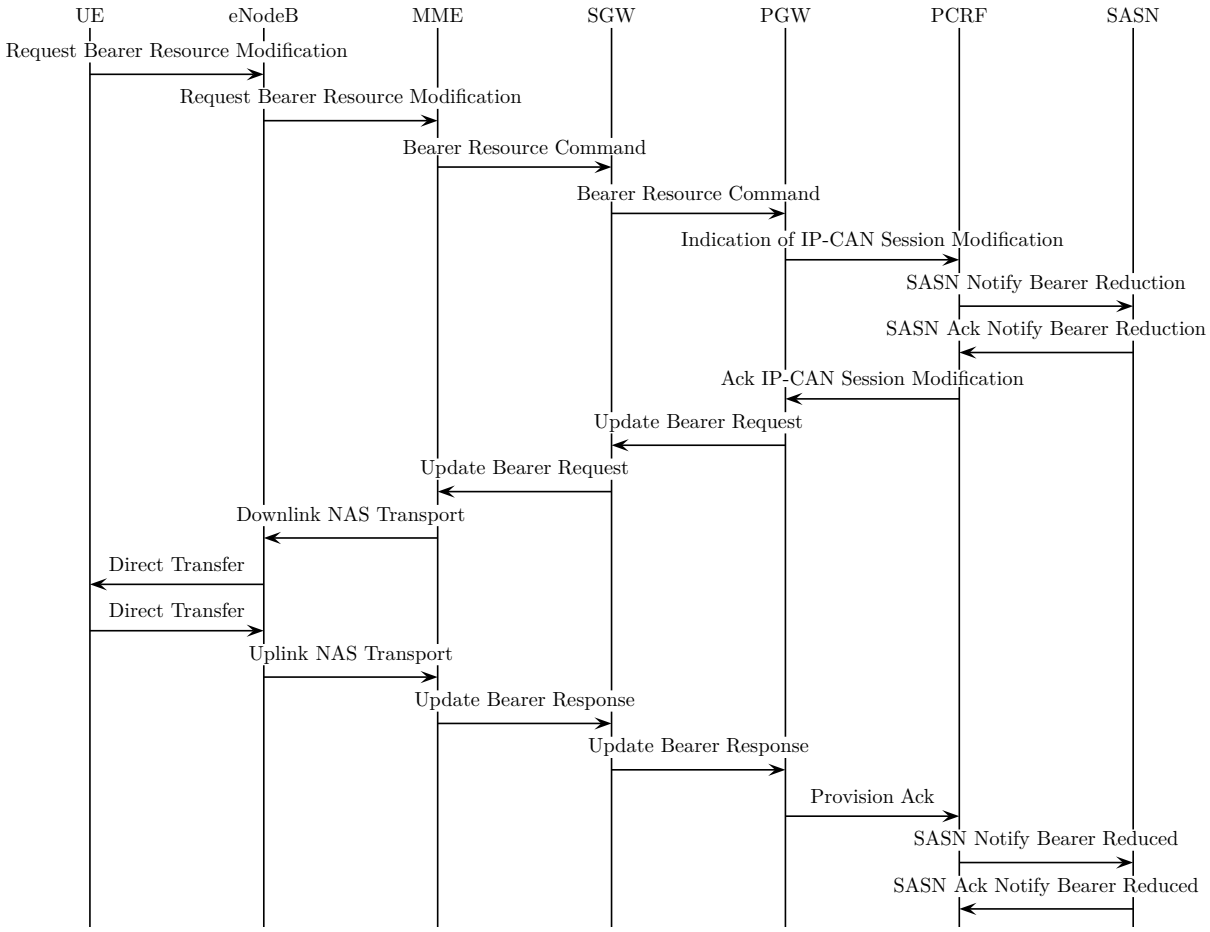


Figure A.12 – Réduction de *bearer* dédié non GBR avec UE QoS *aware*

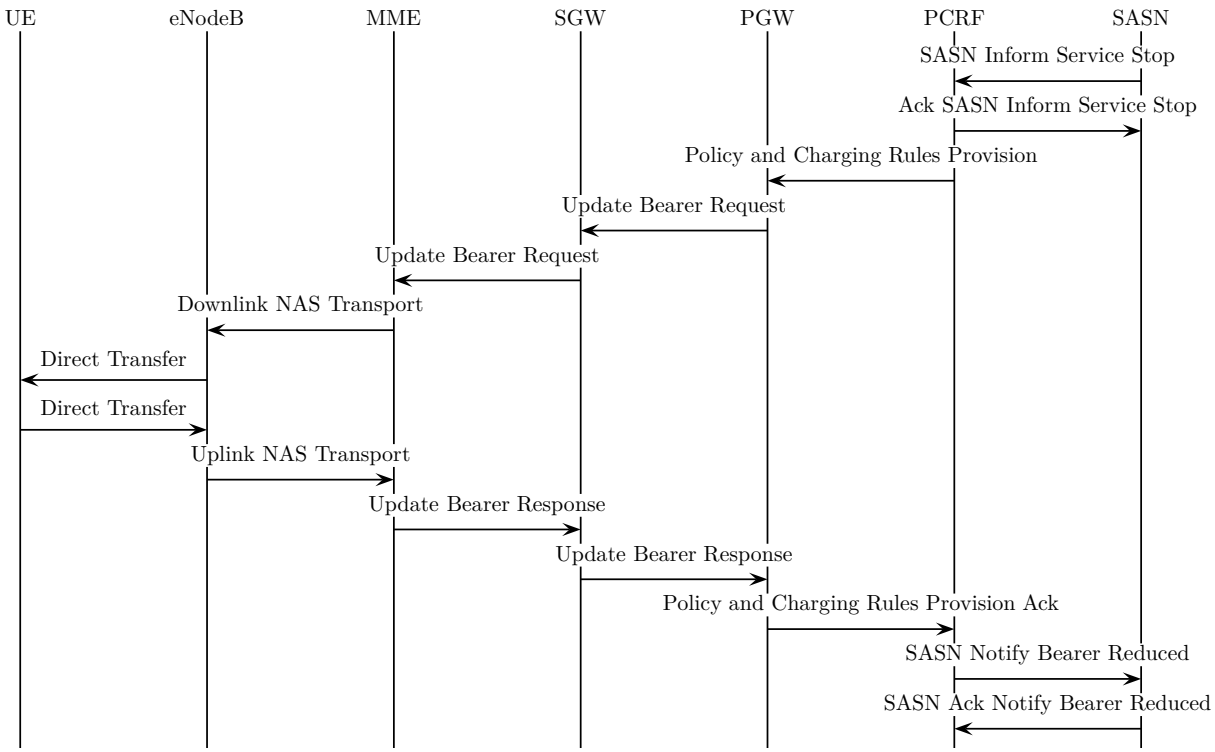


Figure A.13 – Réduction de *bearer* dédié non GBR avec UE QoS *unaware*

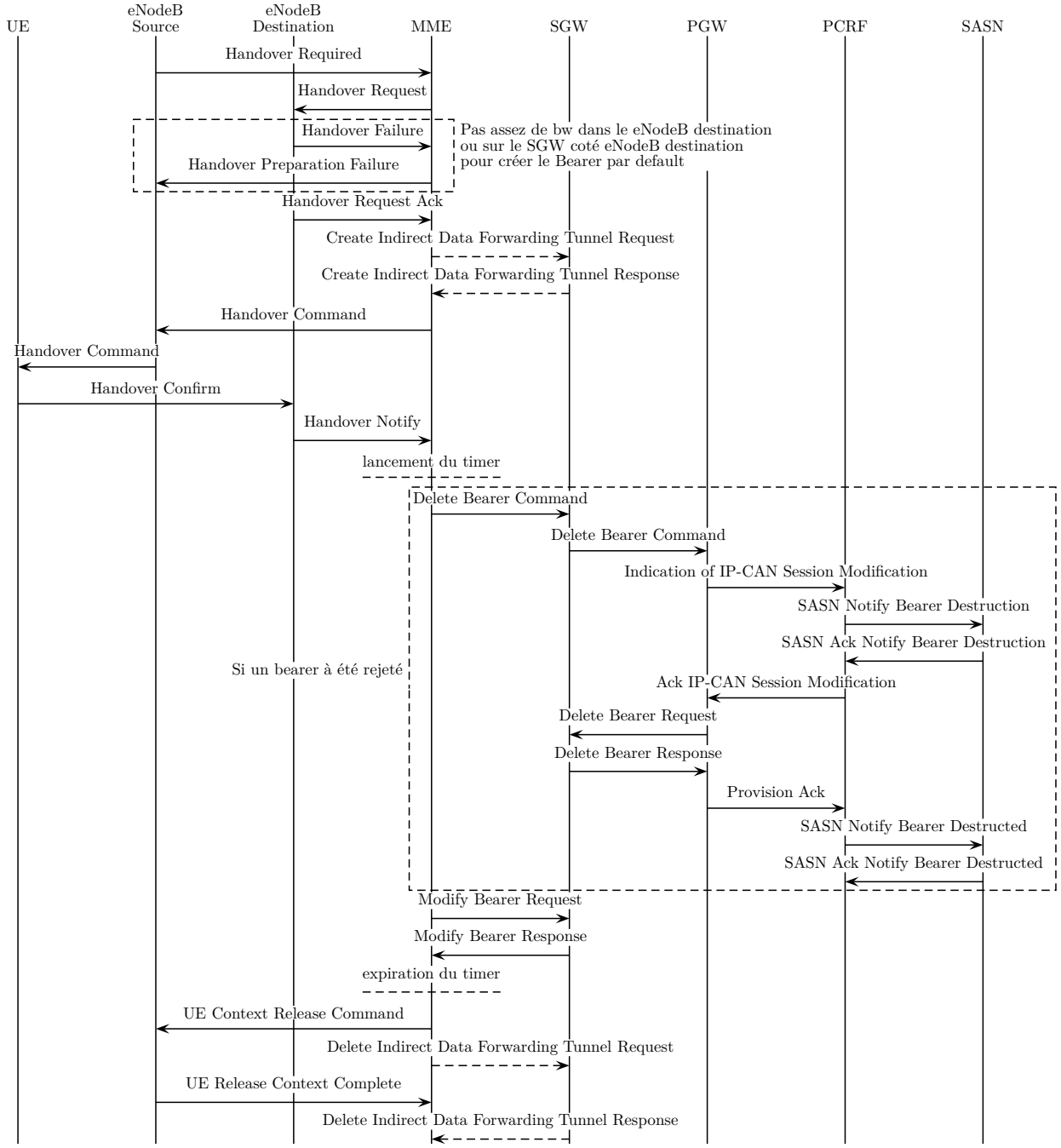


Figure A.14 – S1 Handover

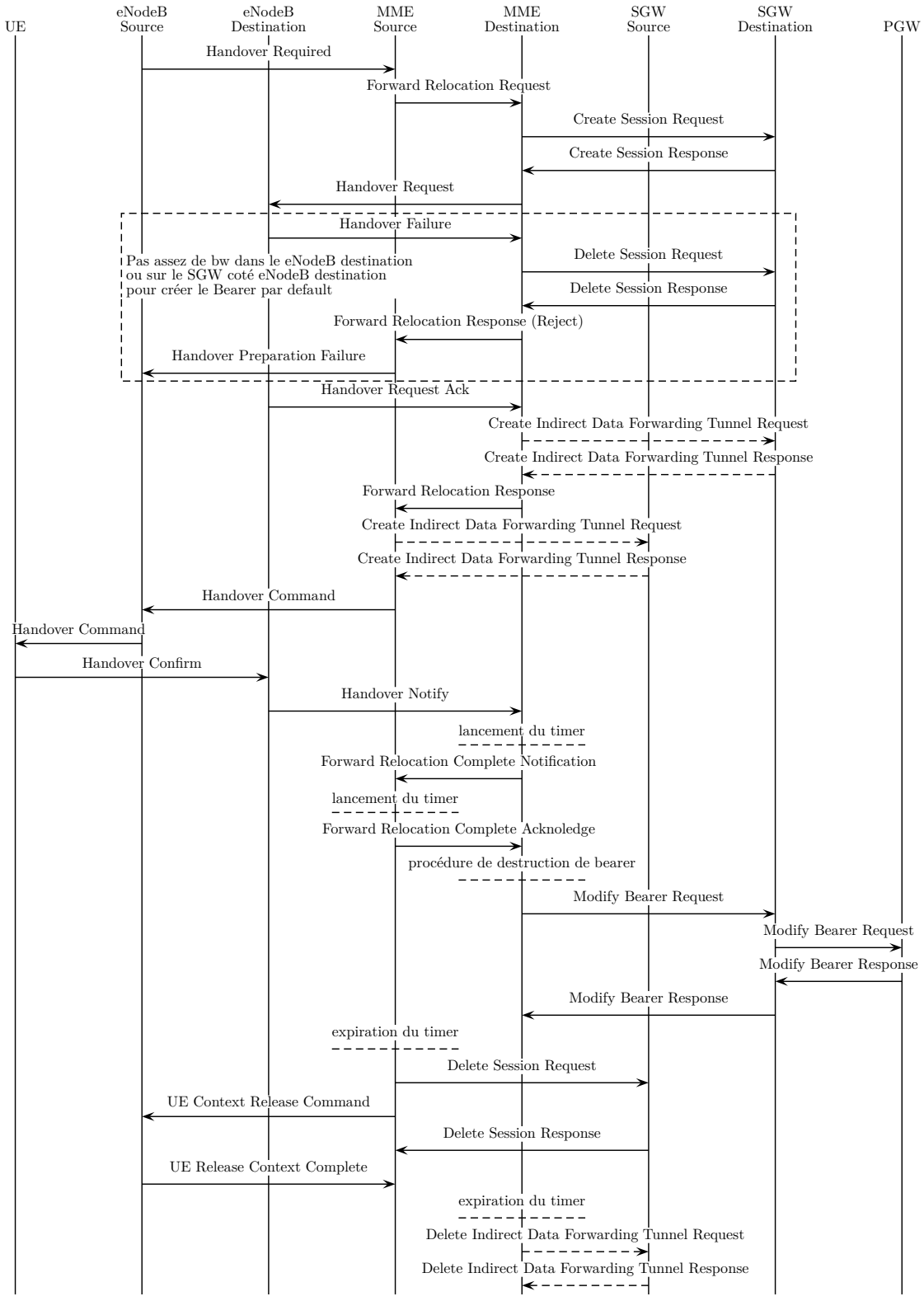


Figure A.15 – S1 Handover avec changement de SGW

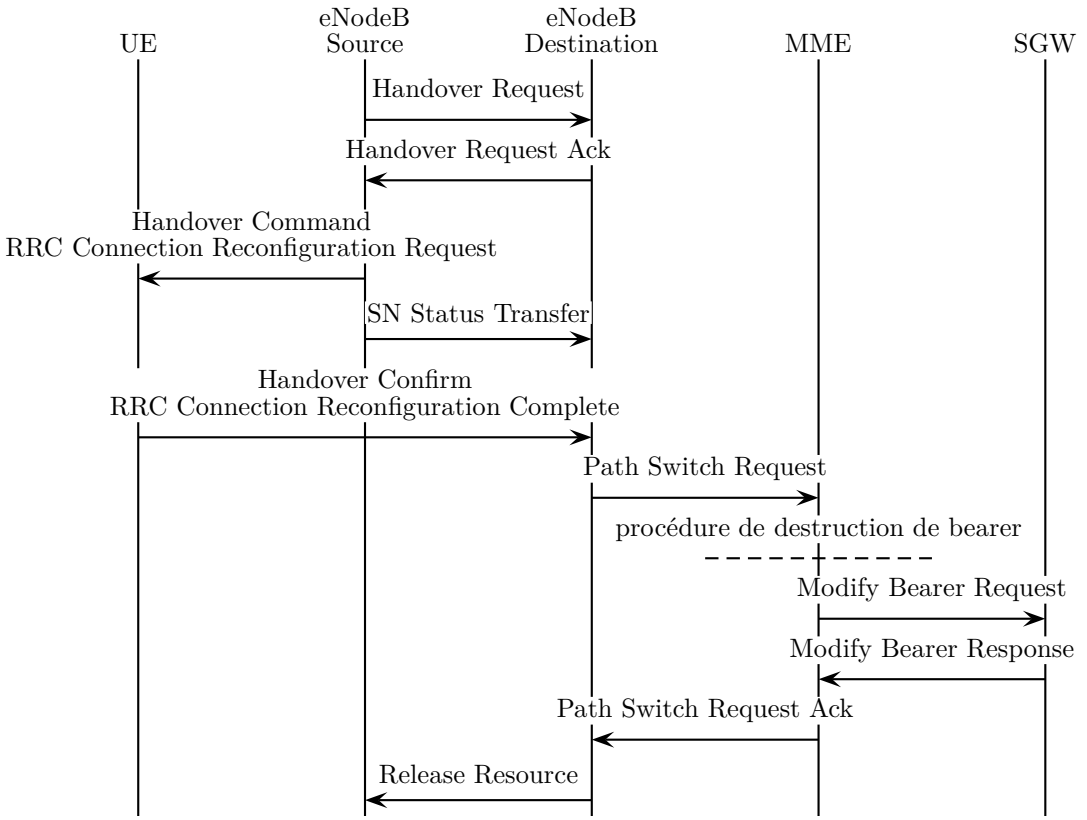


Figure A.16 – X2 Handover

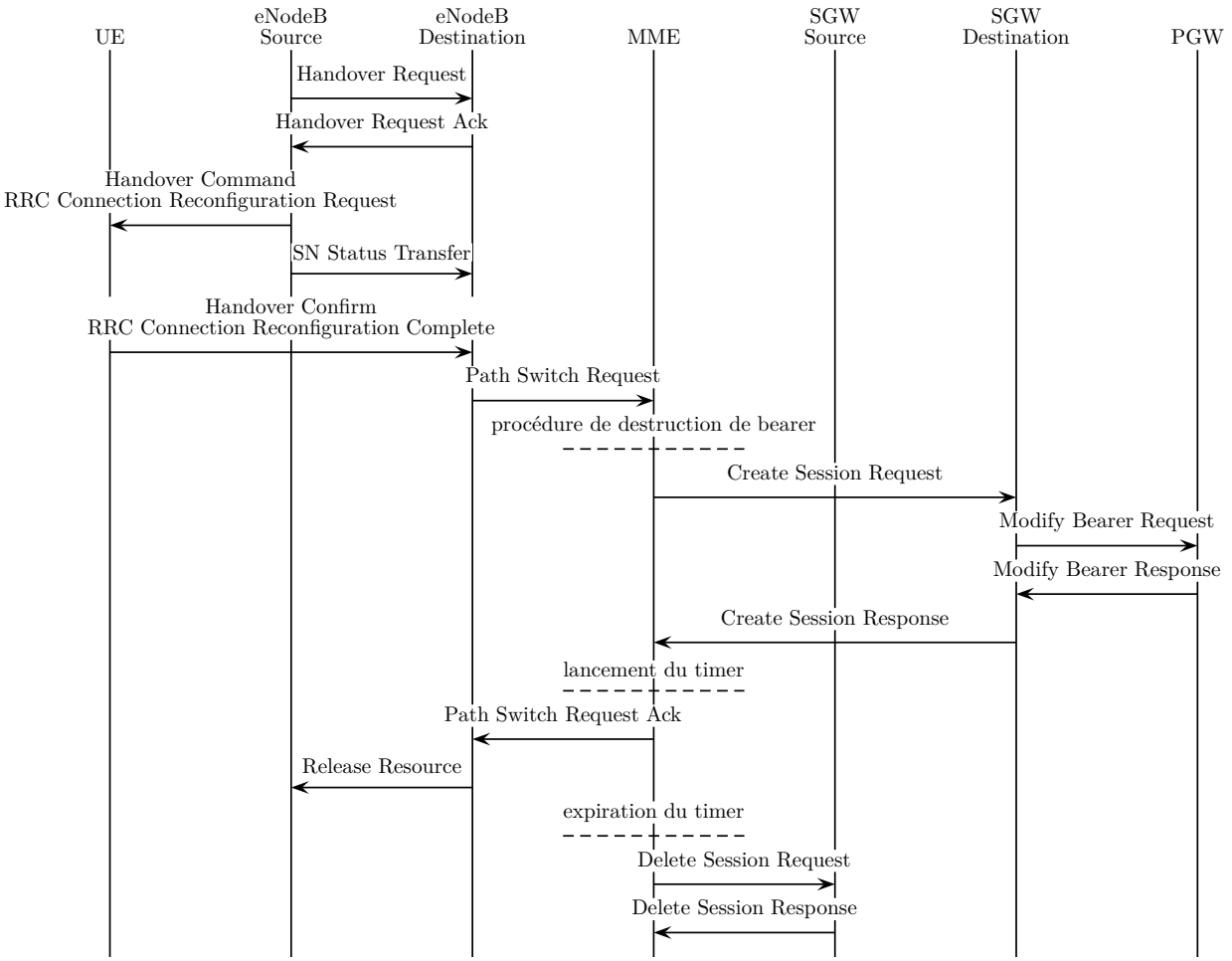


Figure A.17 – X2 Handover avec changement de SGW

ANNEXE B

Diagrammes de séquences de l'architecture 23.402

Les figures suivantes représentent les diagrammes de séquences des principales fonctions de gestion de bearer dans l'architecture 23.402 [9]. Les informations utilisées pour obtenir ces diagrammes sont tirées des documents 23.401 [8], 23.402 [9], 23.203 [7] et 36.300 [29].

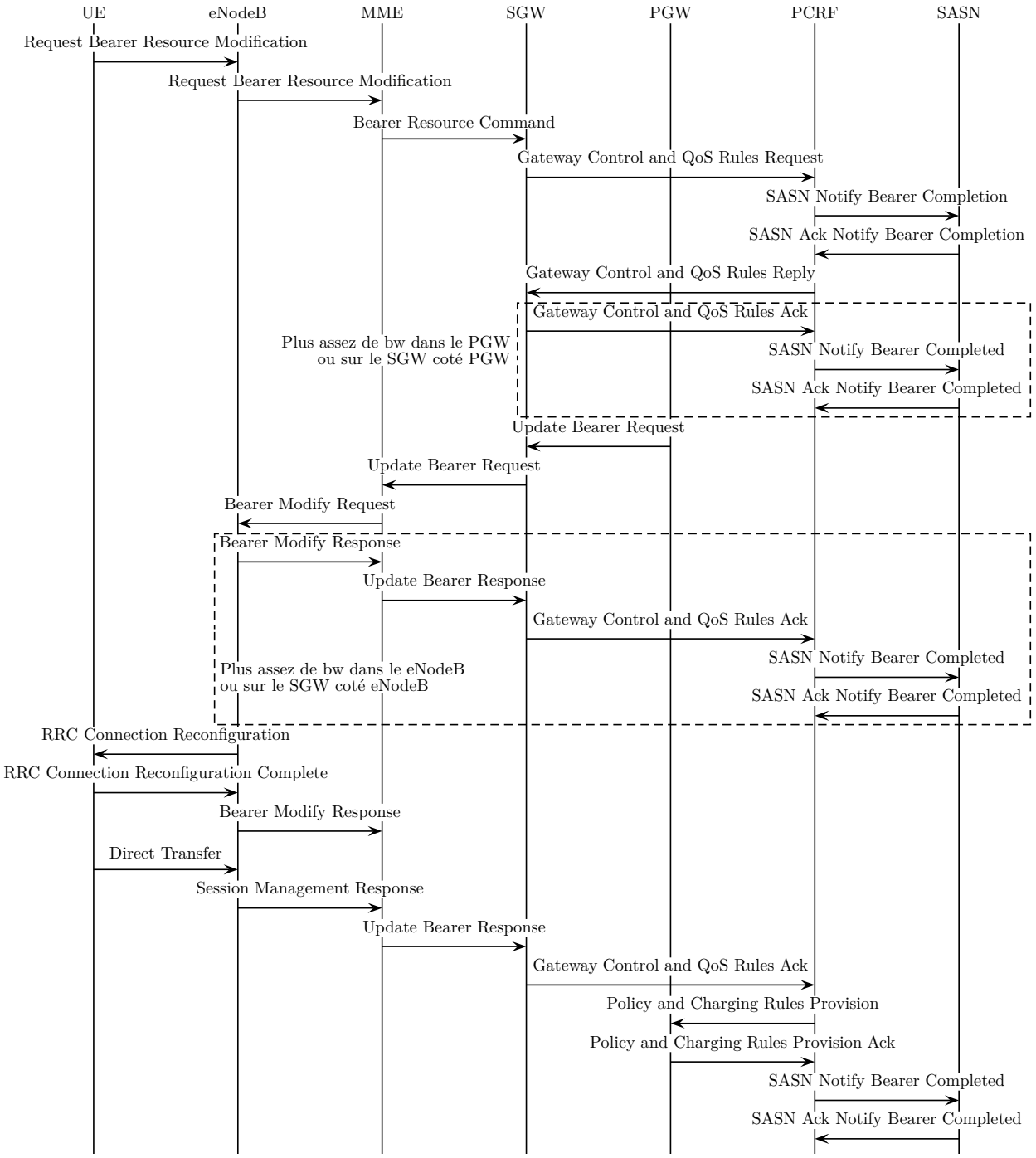


Figure B.1 – Complétion de *bearer* dédié GBR avec UE QoS *aware*

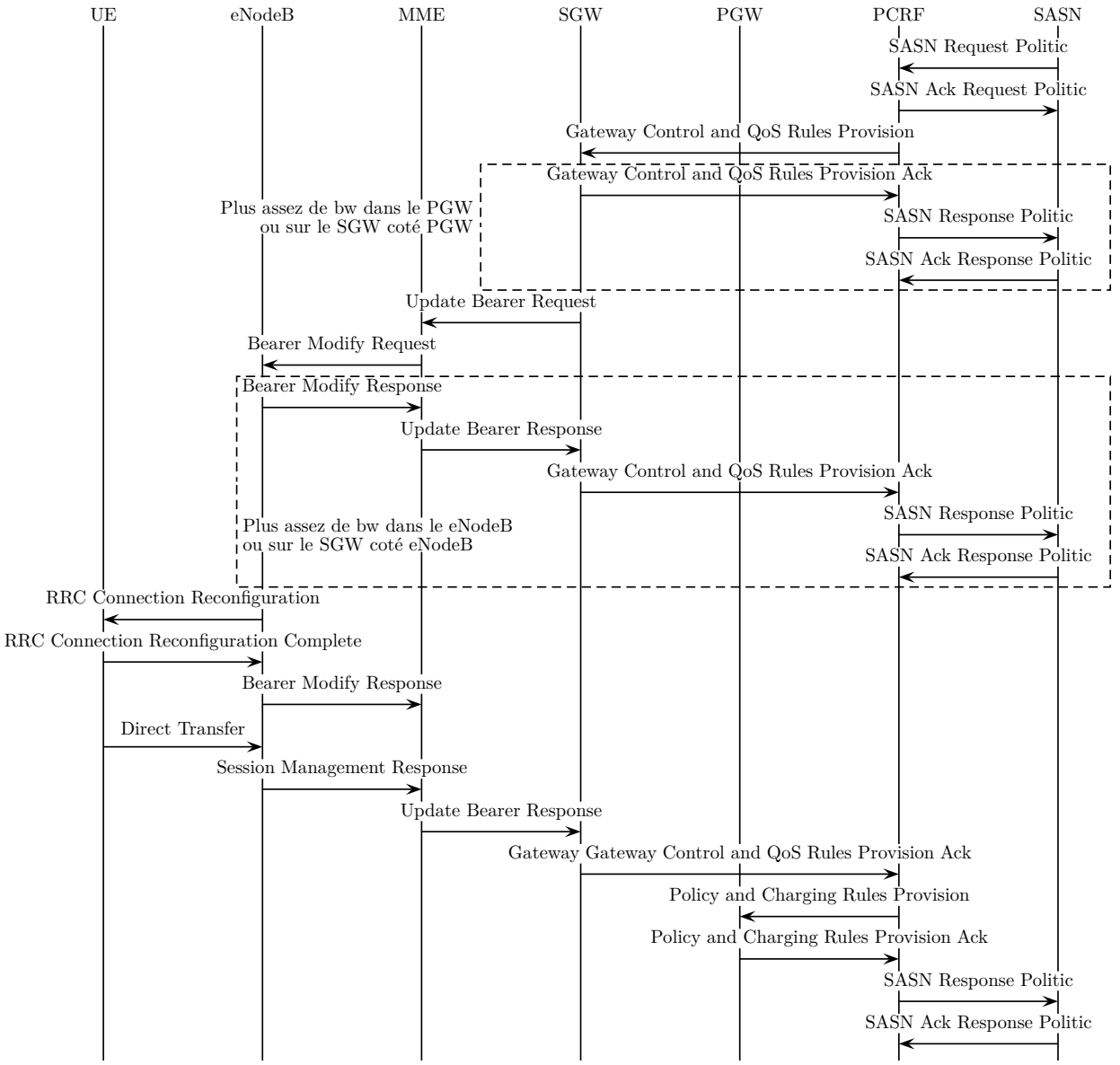


Figure B.2 – Complétion de *bearer* dédié GBR avec UE QoS *unaware*

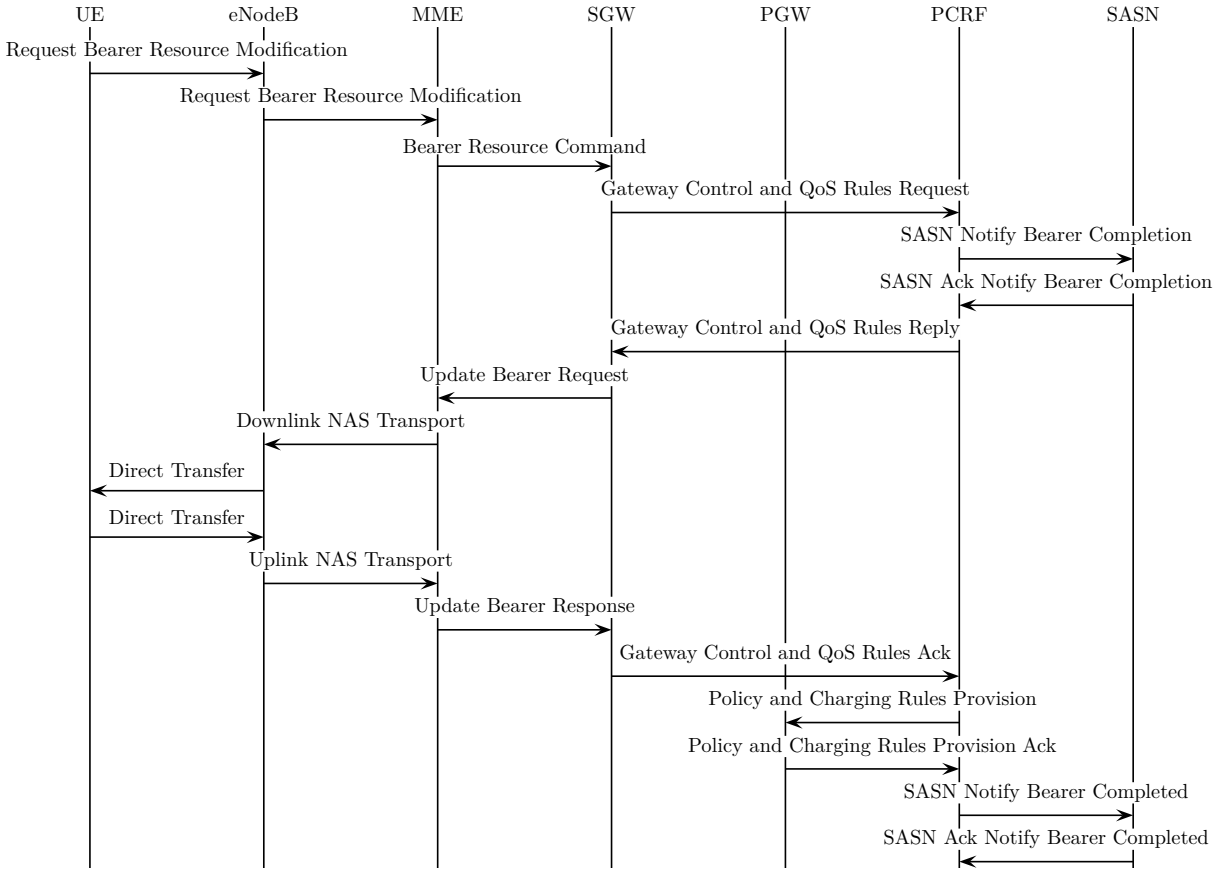


Figure B.3 – Complétion de *bearer* dédié non GBR avec UE QoS *aware*

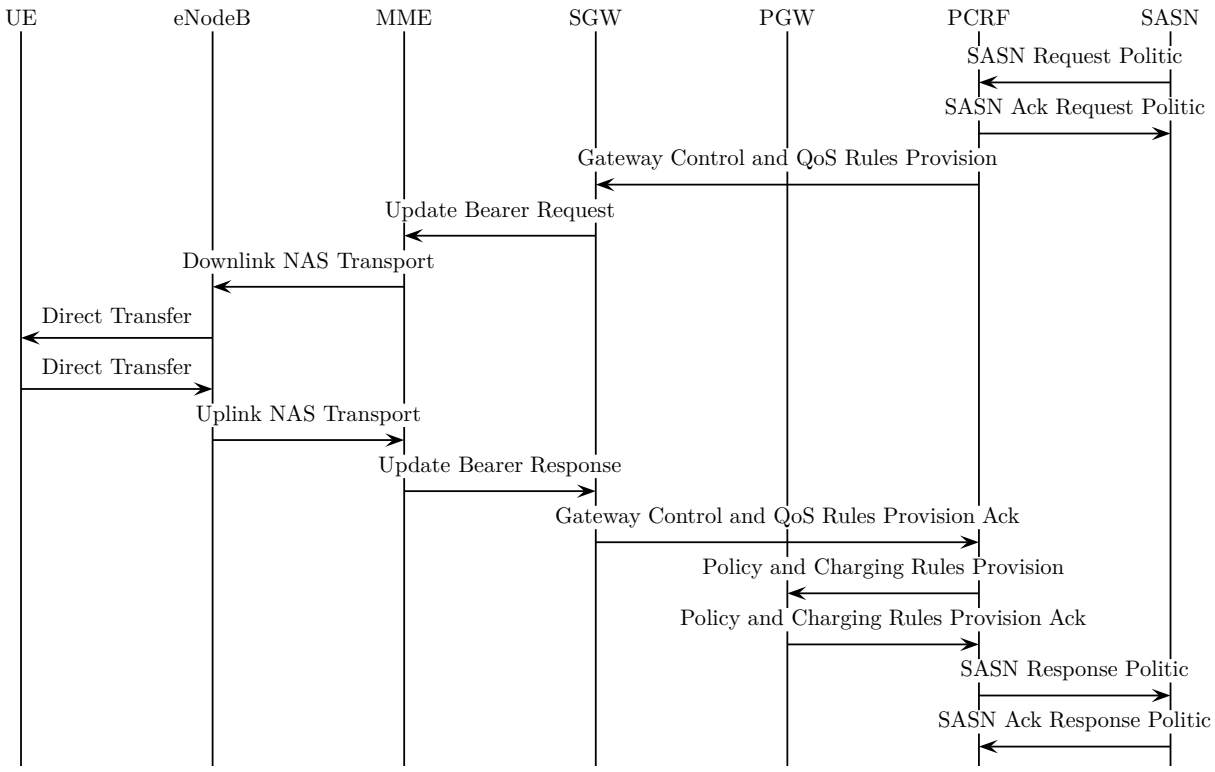


Figure B.4 – Complétion de *bearer* dédié non GBR avec UE QoS *unaware*

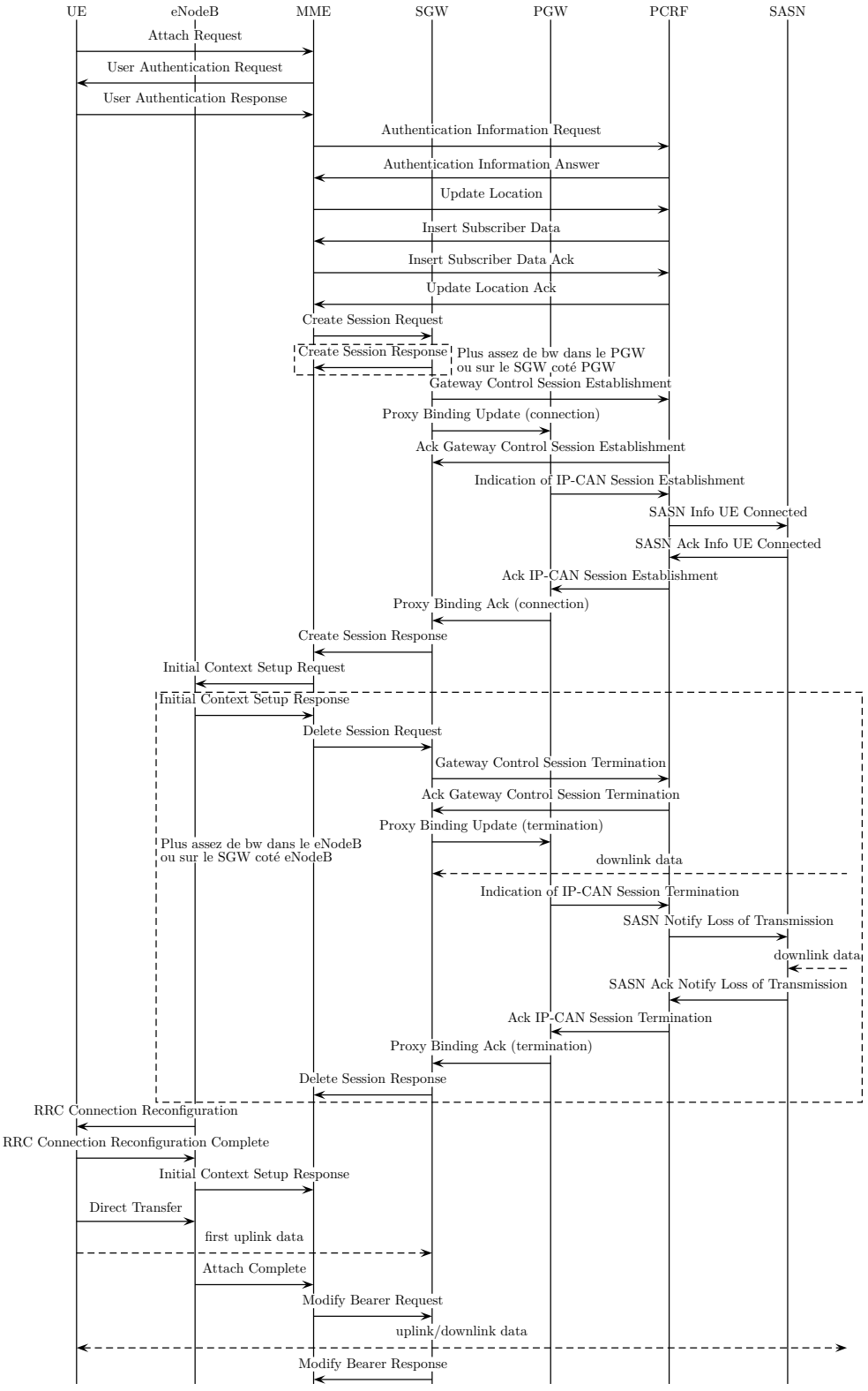


Figure B.5 – Connexion d'un UE

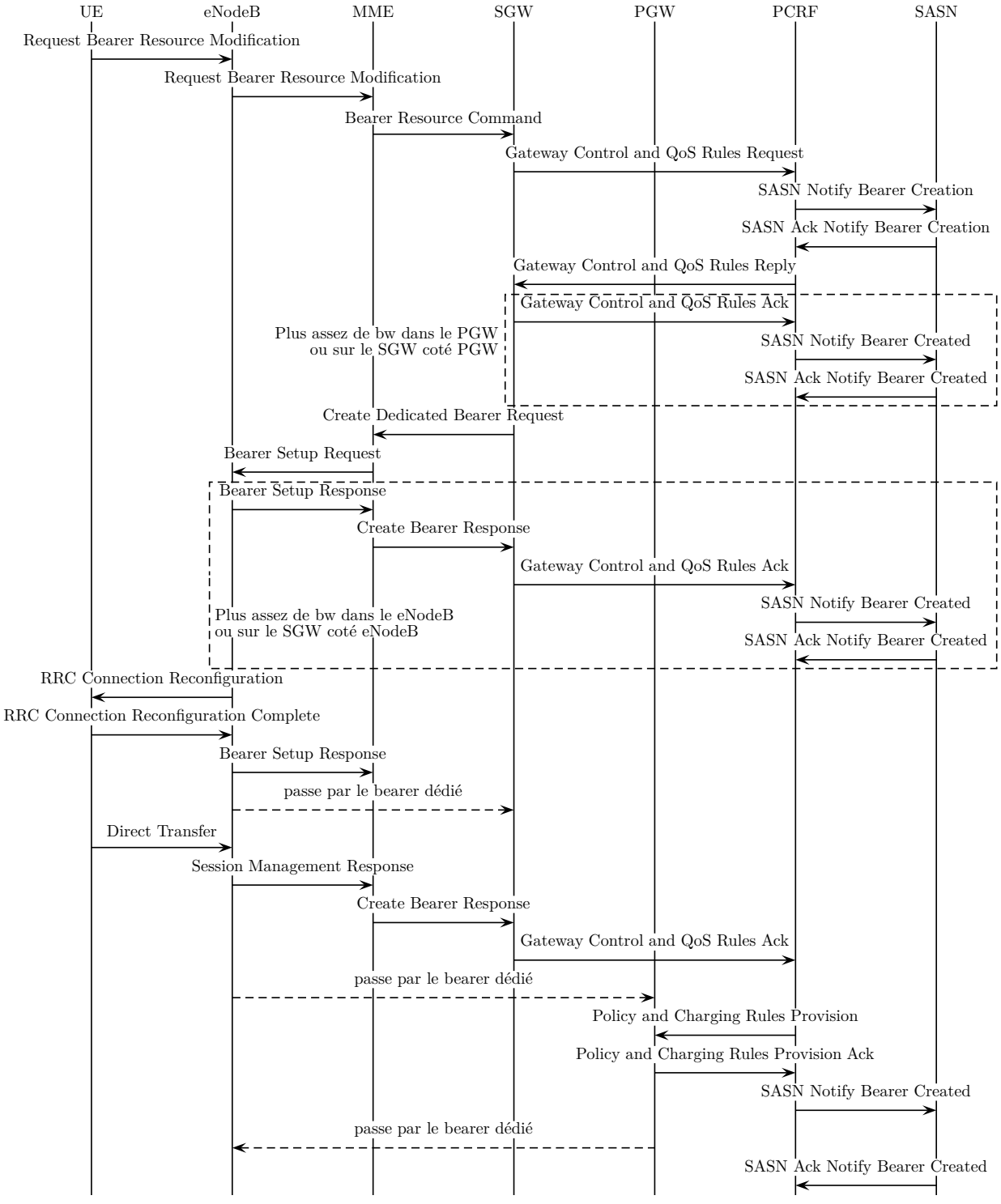


Figure B.6 – Création de *bearer* dédié avec UE QoS *aware*

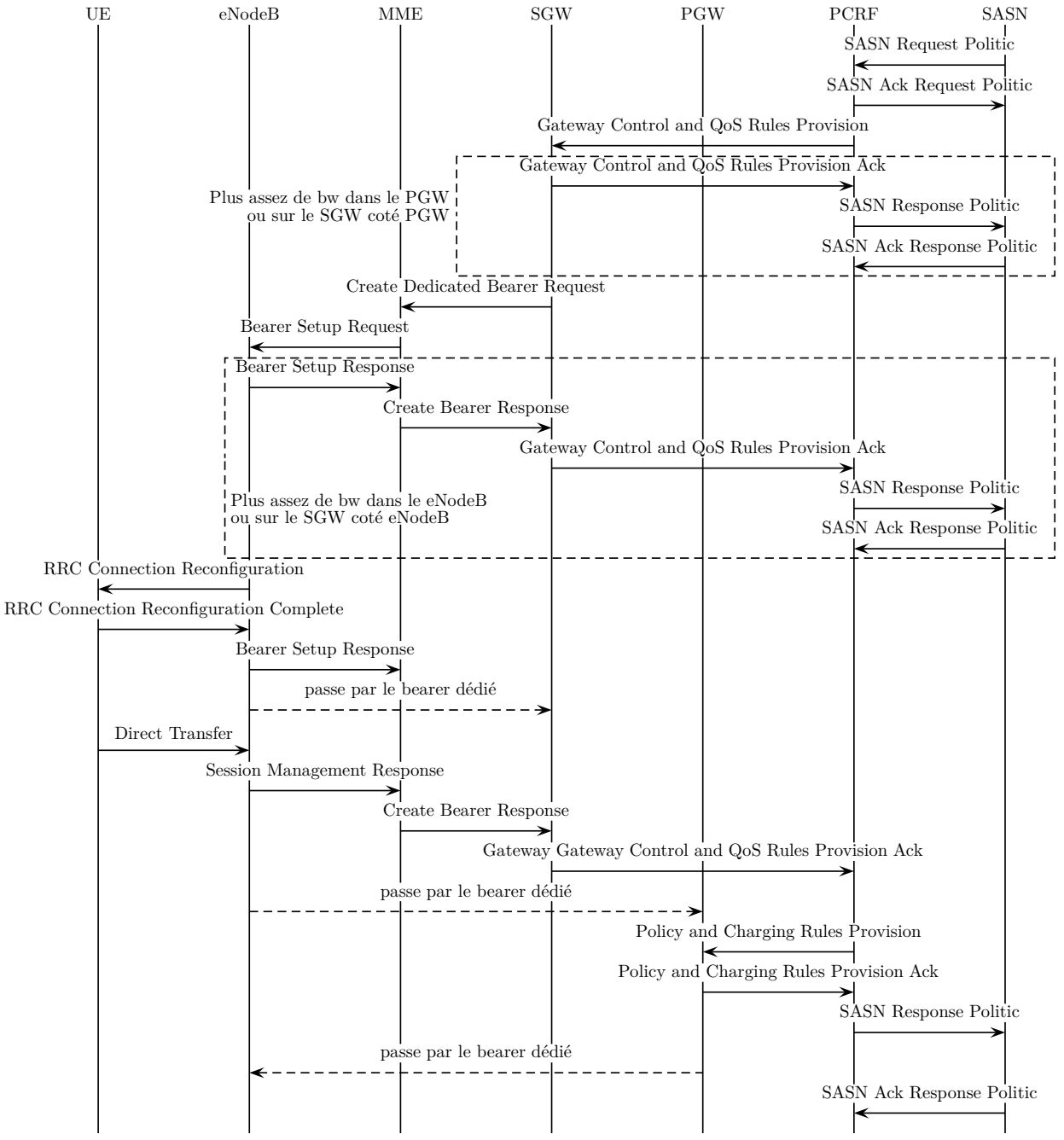


Figure B.7 – Création de *bearer* dédié avec UE QoS *unaware*

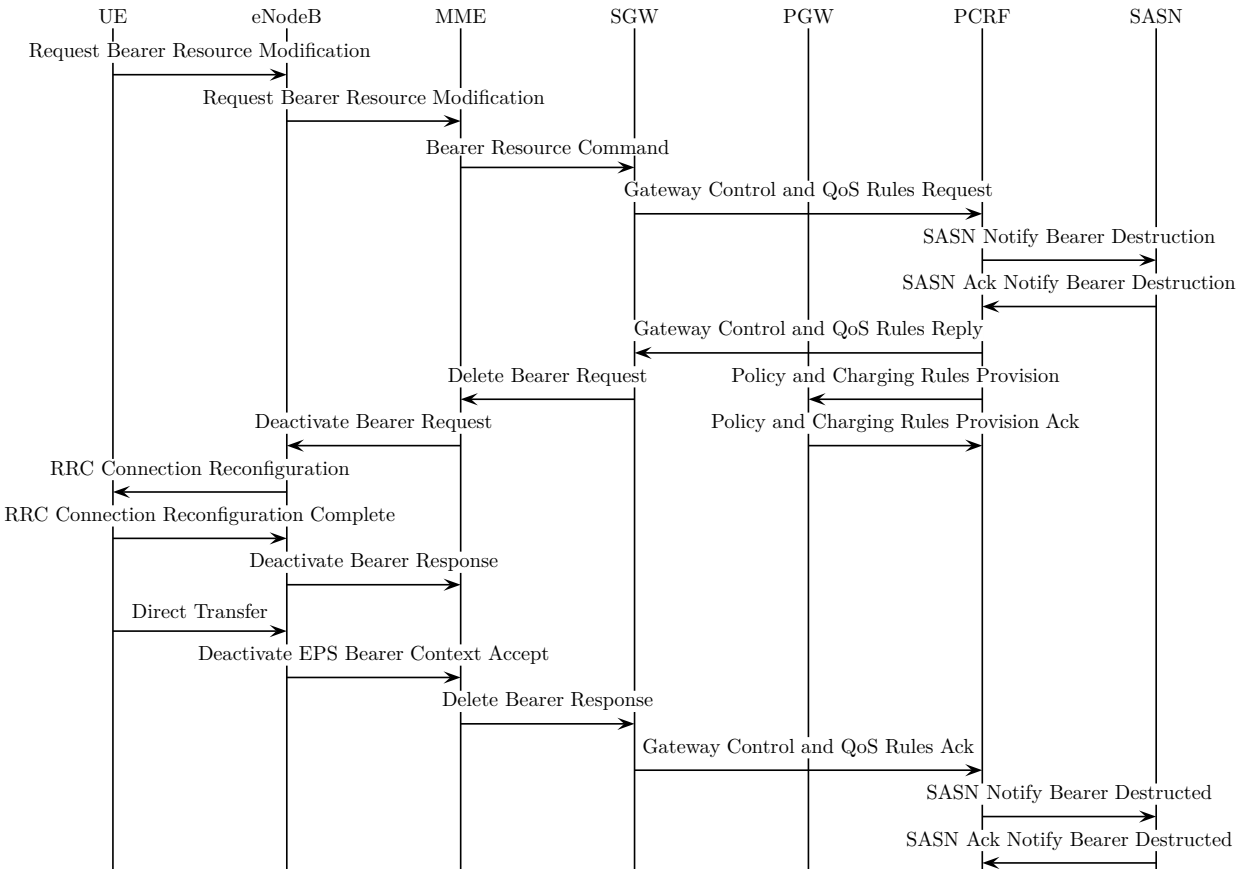


Figure B.8 – Destruction de *bearer* dédié avec UE QoS *aware*

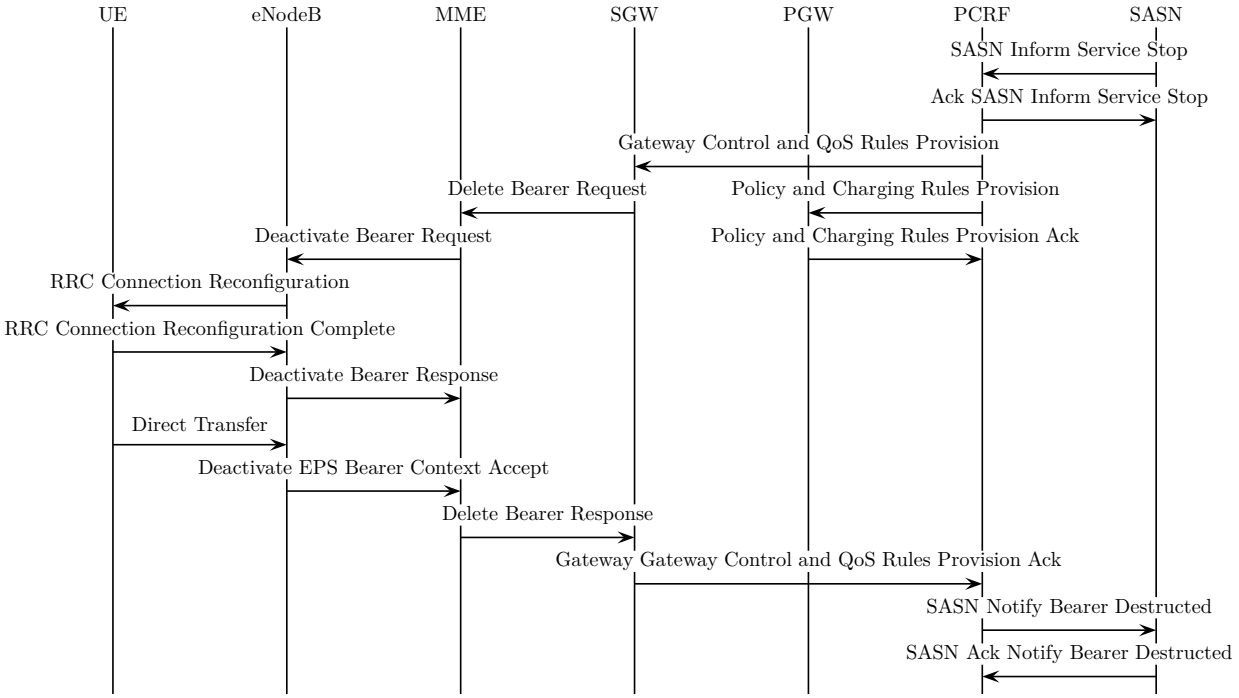


Figure B.9 – Destruction de *bearer* dédié avec UE QoS *unaware*

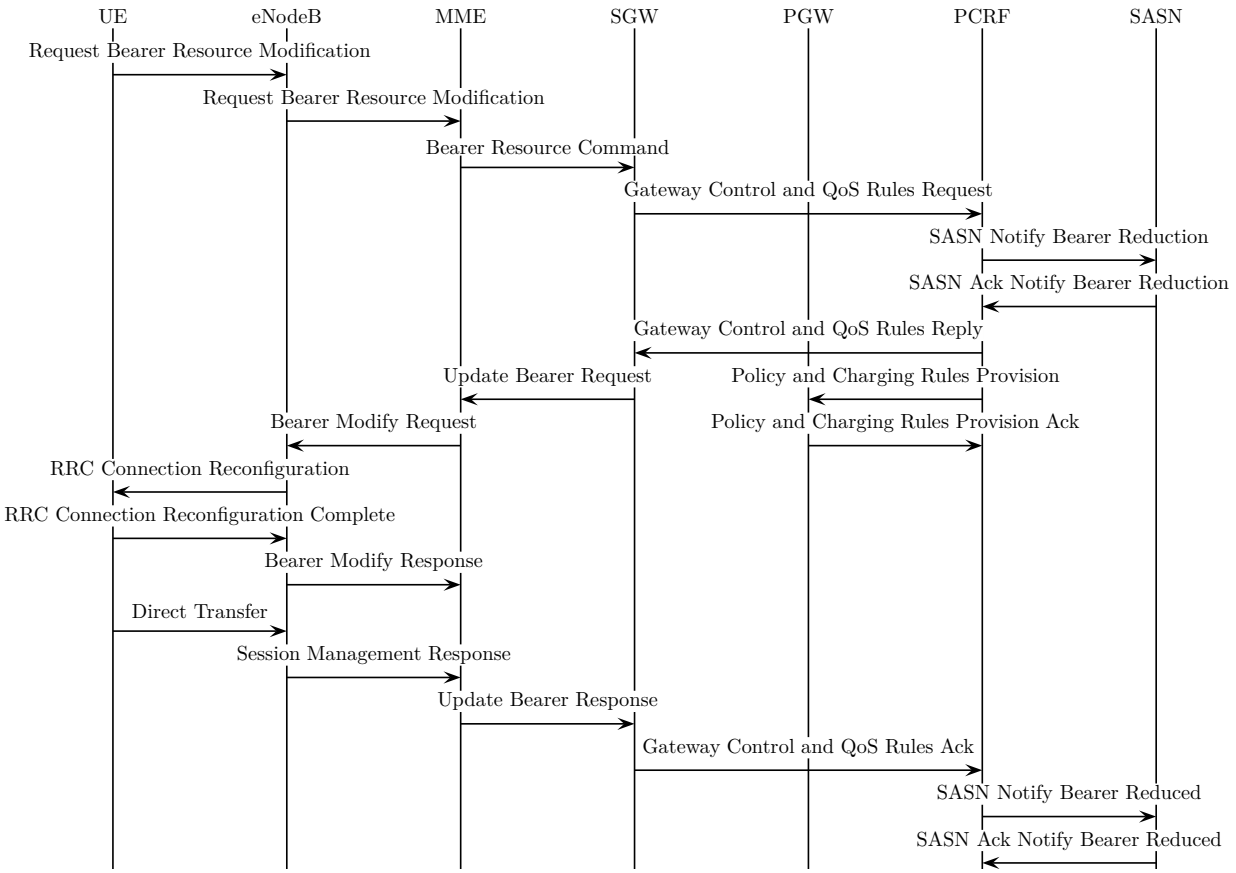


Figure B.10 – Réduction de *bearer* dédié GBR avec UE QoS *aware*

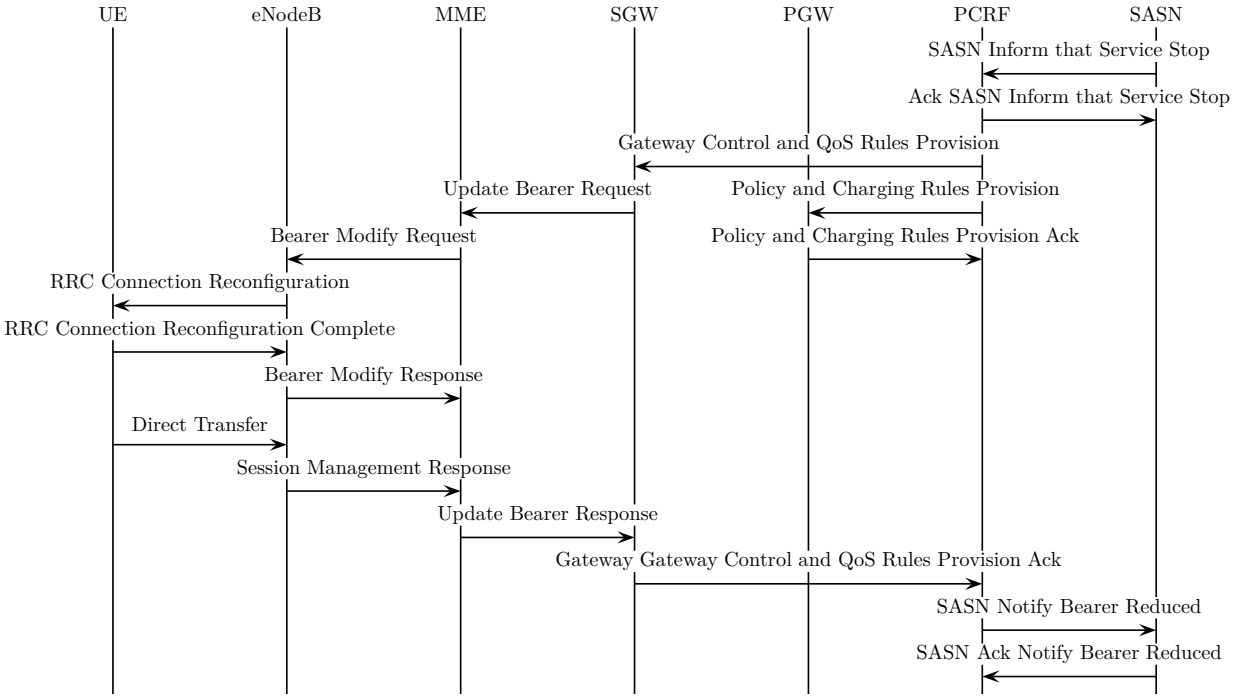


Figure B.11 – Réduction de *bearer* dédié GBR avec UE QoS *unaware*

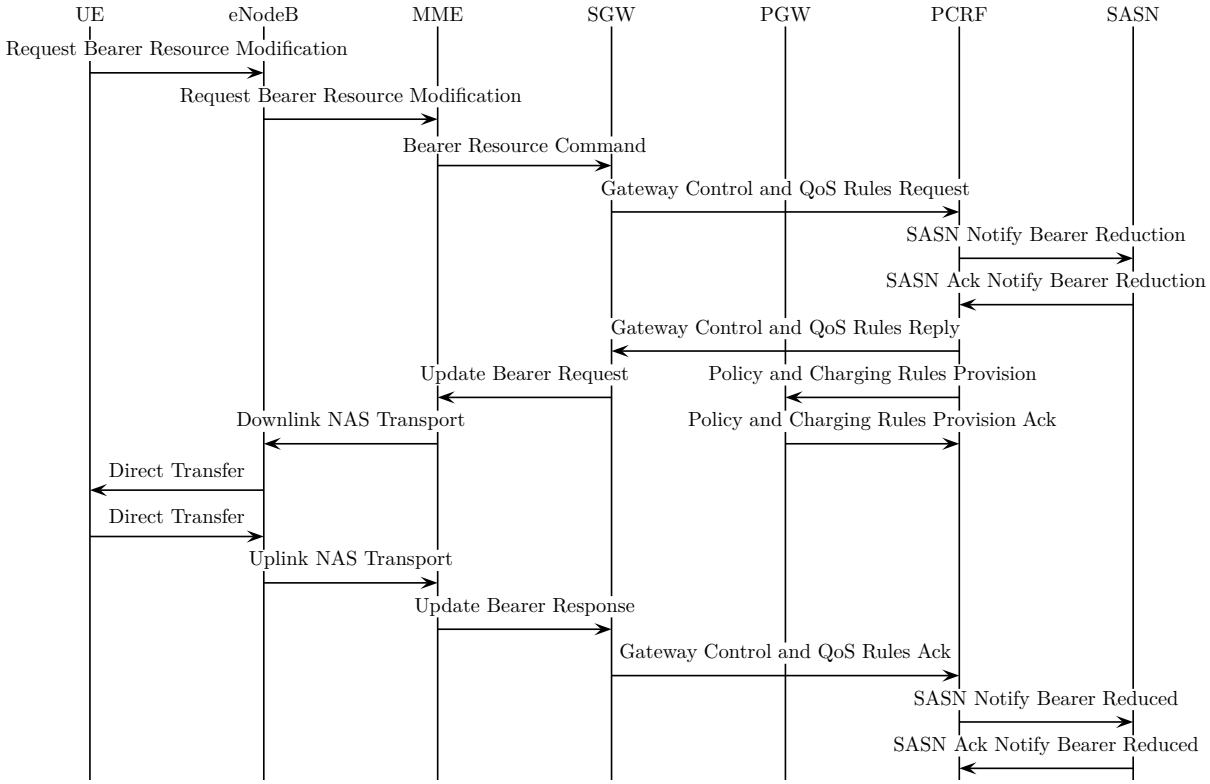


Figure B.12 – Réduction de *bearer* dédié non GBR avec UE QoS *aware*

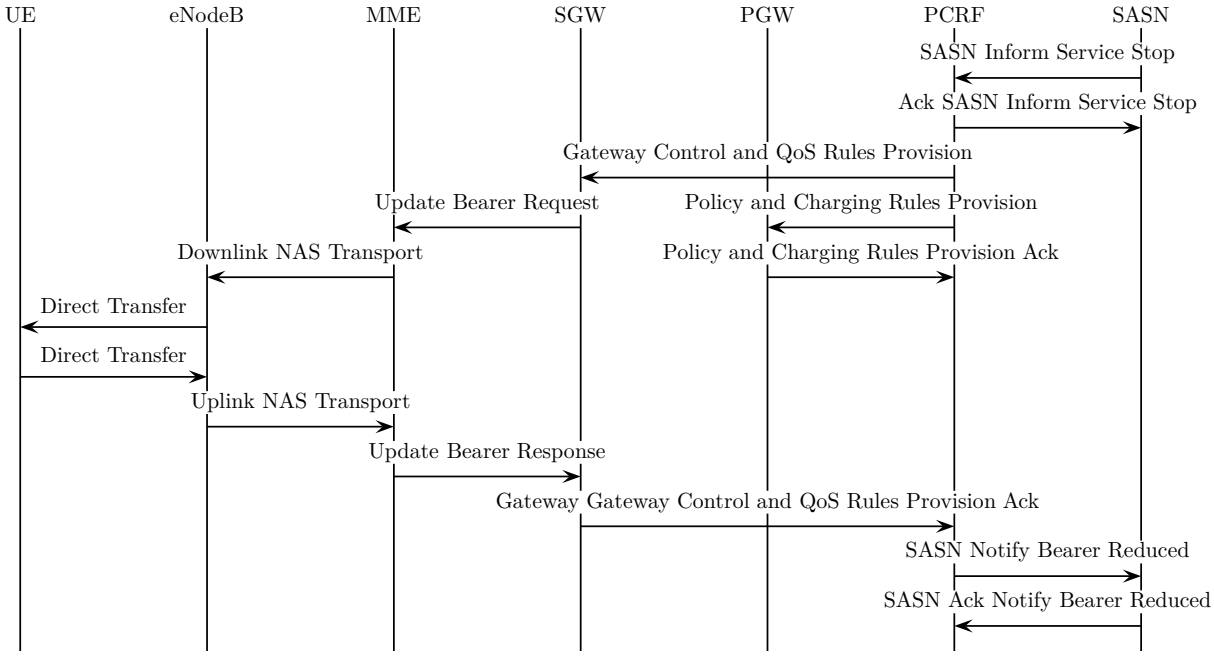


Figure B.13 – Réduction de *bearer* dédié non GBR avec UE QoS *unaware*

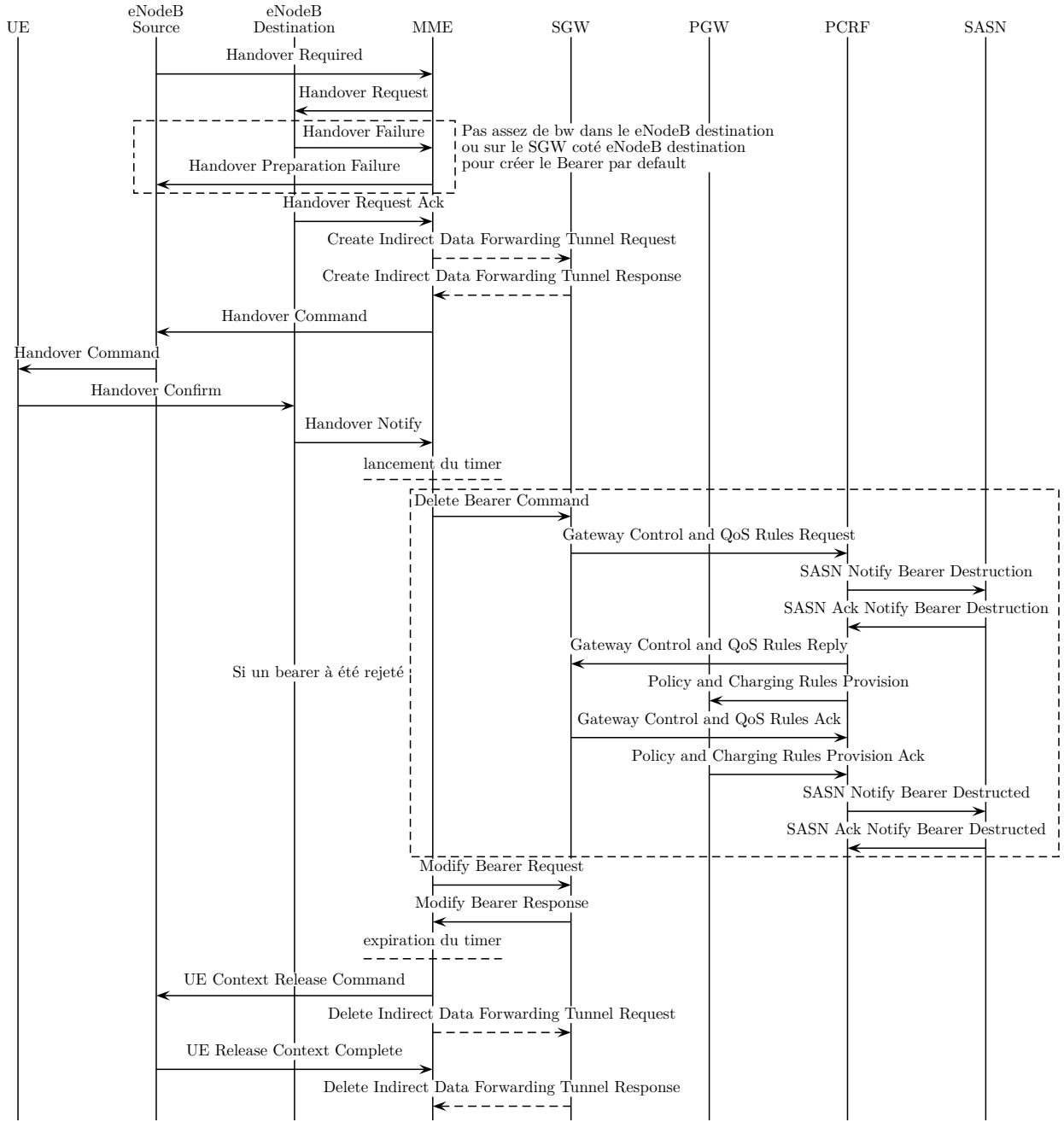


Figure B.14 – S1 Handover

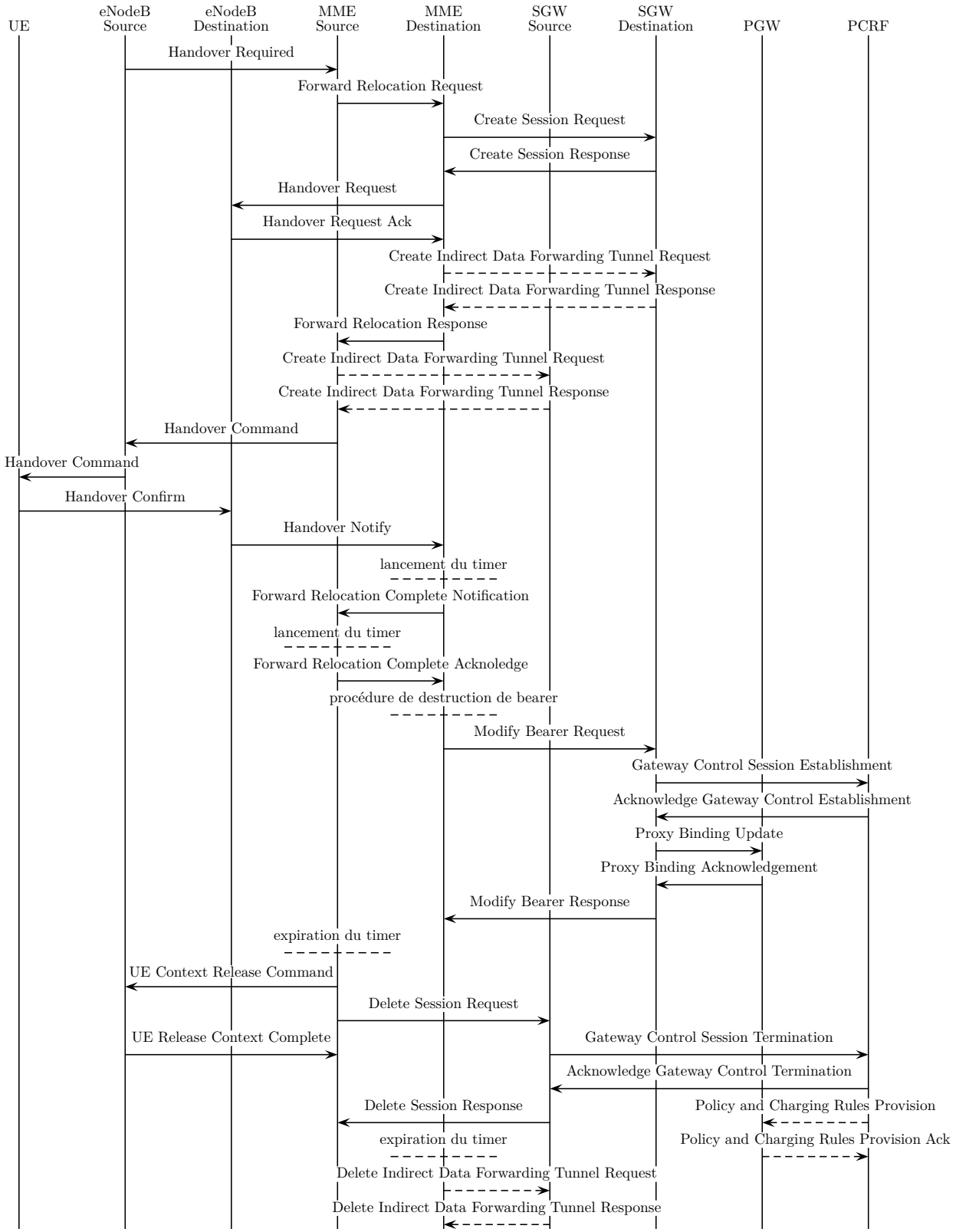


Figure B.15 – S1 Handover avec changement de SGW

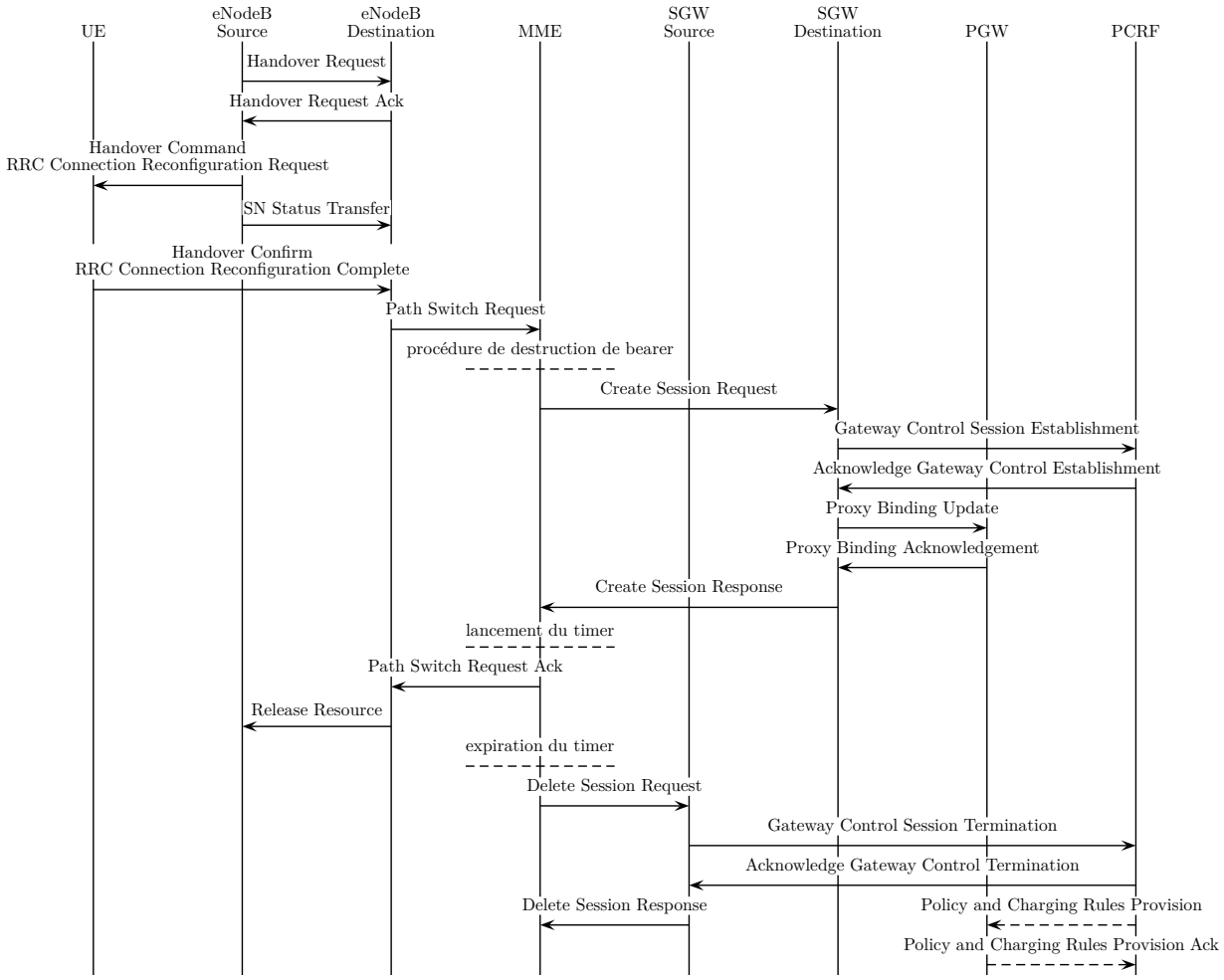


Figure B.16 – X2 Handover avec changement de SGW

ANNEXE C

Diagrammes de séquences de l'architecture 403

Les figures suivantes représentent les diagrammes de séquences des principales fonctions de gestion de bearer dans l'architecture 403, soit l'architecture 23.402 [9] légèrement modifiée sur la partie PMIPv6 pour supprimer le protocole GRE. La procédure de connexion d'un UE est exactement la même que celle de l'architecture 23.402 [9] ; cependant le PBU contient une option *Flow Label* supplémentaire dans architecture. La procédure de relève avec interface X2 et sans changement de SGW est identique à celle de l'architecture 23.402 [9]. Les informations utilisées pour obtenir ces diagrammes sont tirées des documents 23.401 [8], 23.402 [9], 23.203 [7] et 36.300 [29] et inspirées des écrits de Gundavelli *et al.* [65] et Hui *et al.* [70].

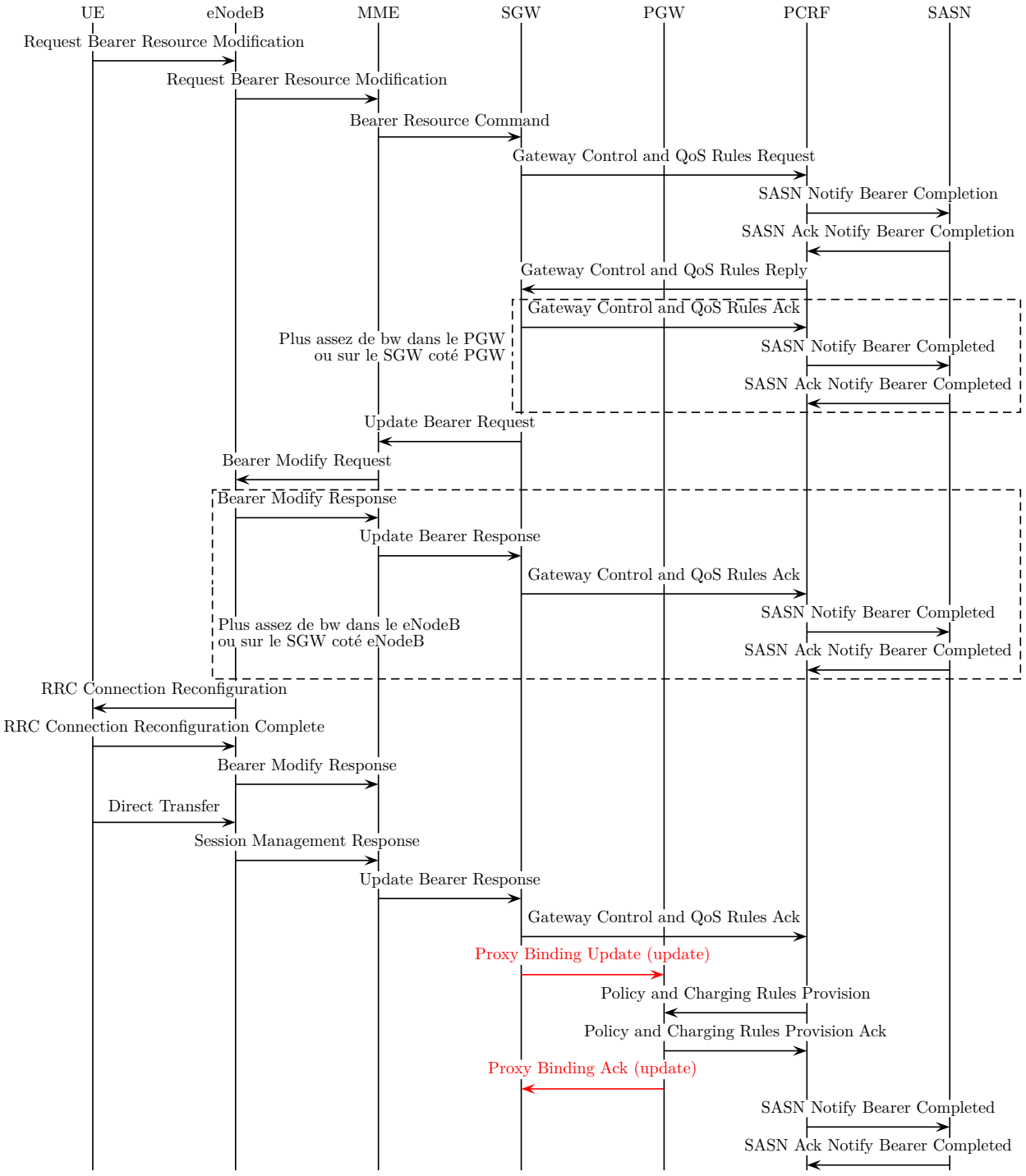


Figure C.1 – Complétion de *bearer* dédié GBR avec UE QoS *aware*

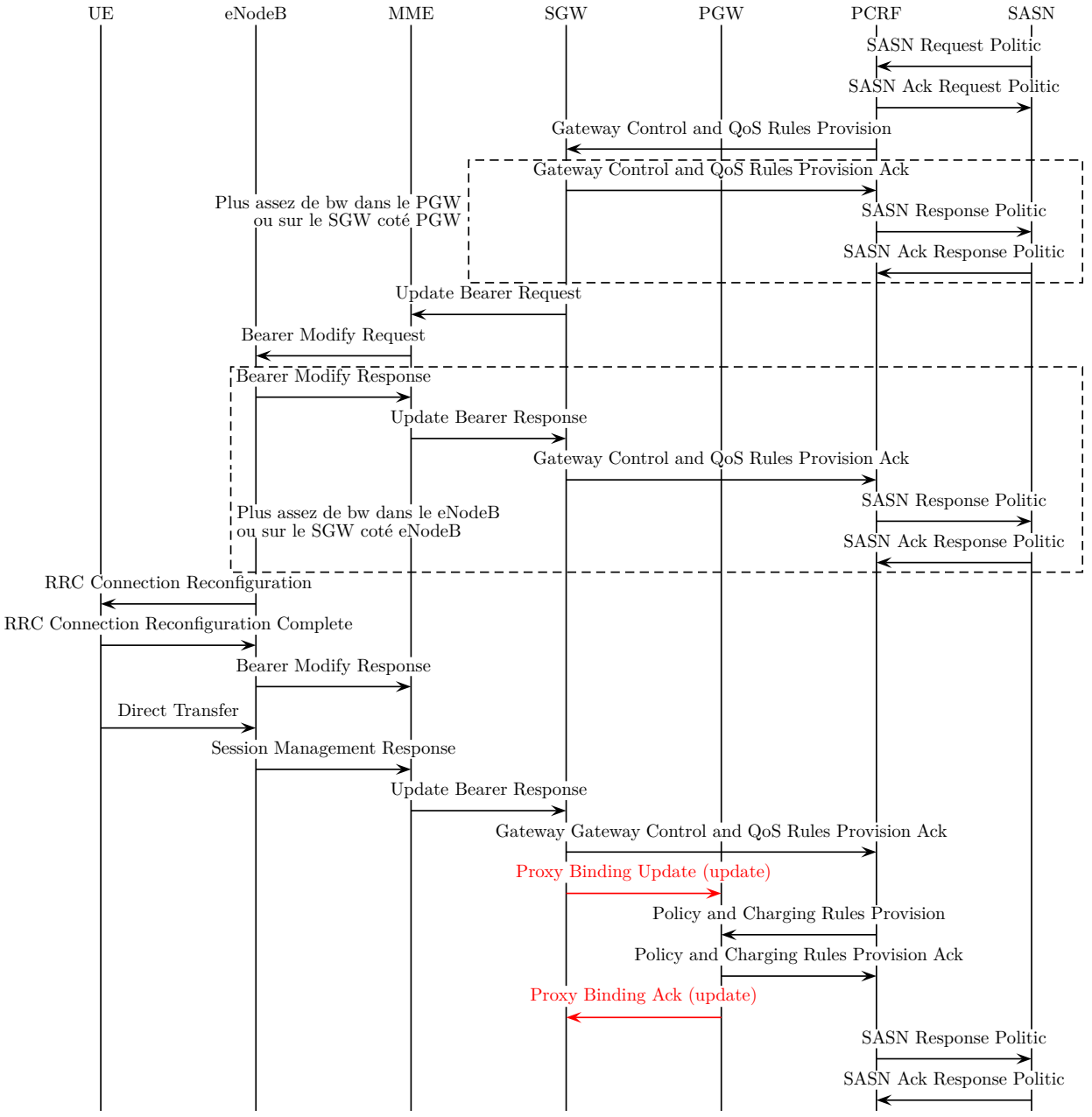


Figure C.2 – Complétion de *bearer* dédié GBR avec UE QoS *unaware*

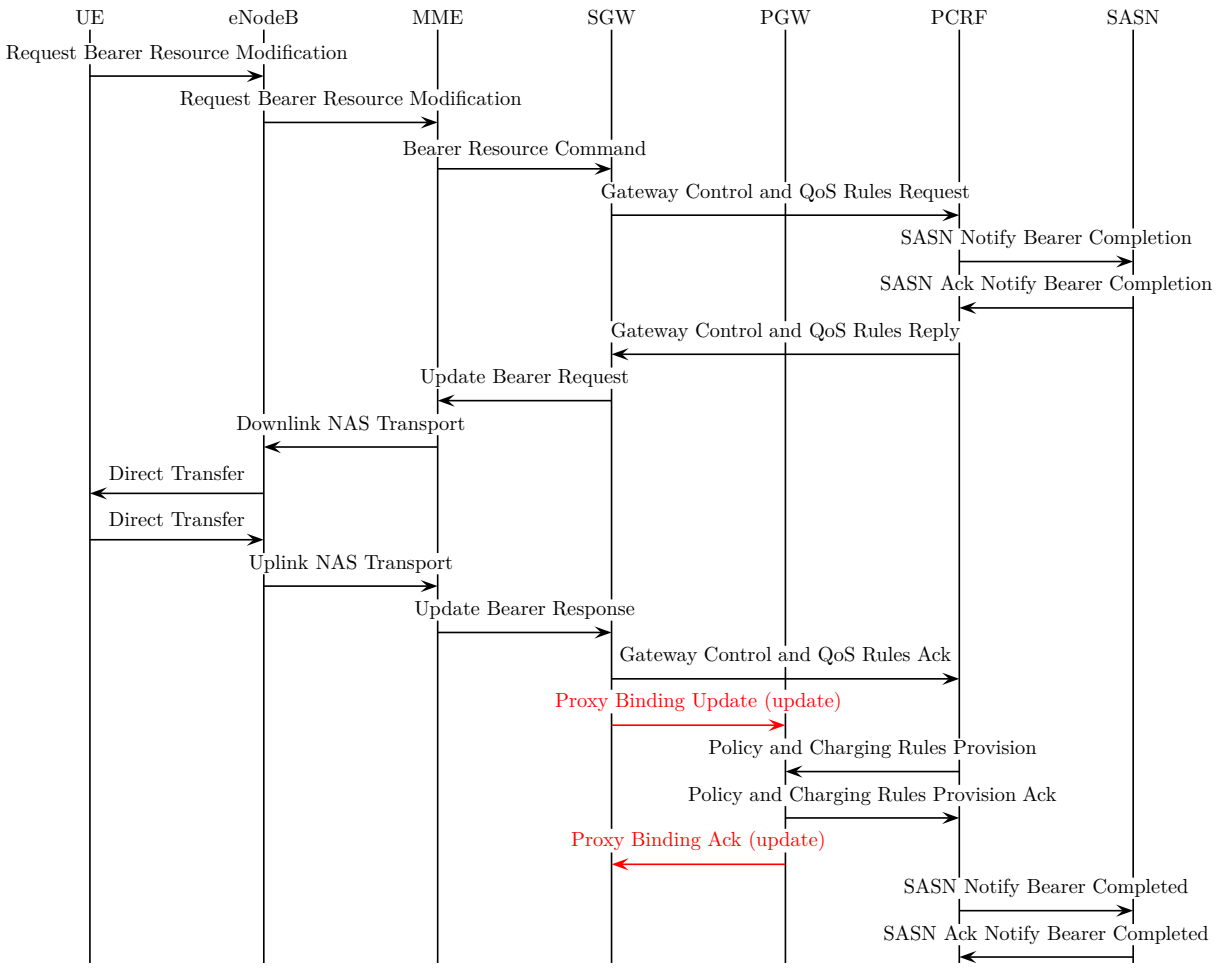


Figure C.3 – Complétion de *bearer* dédié non GBR avec UE QoS *aware*

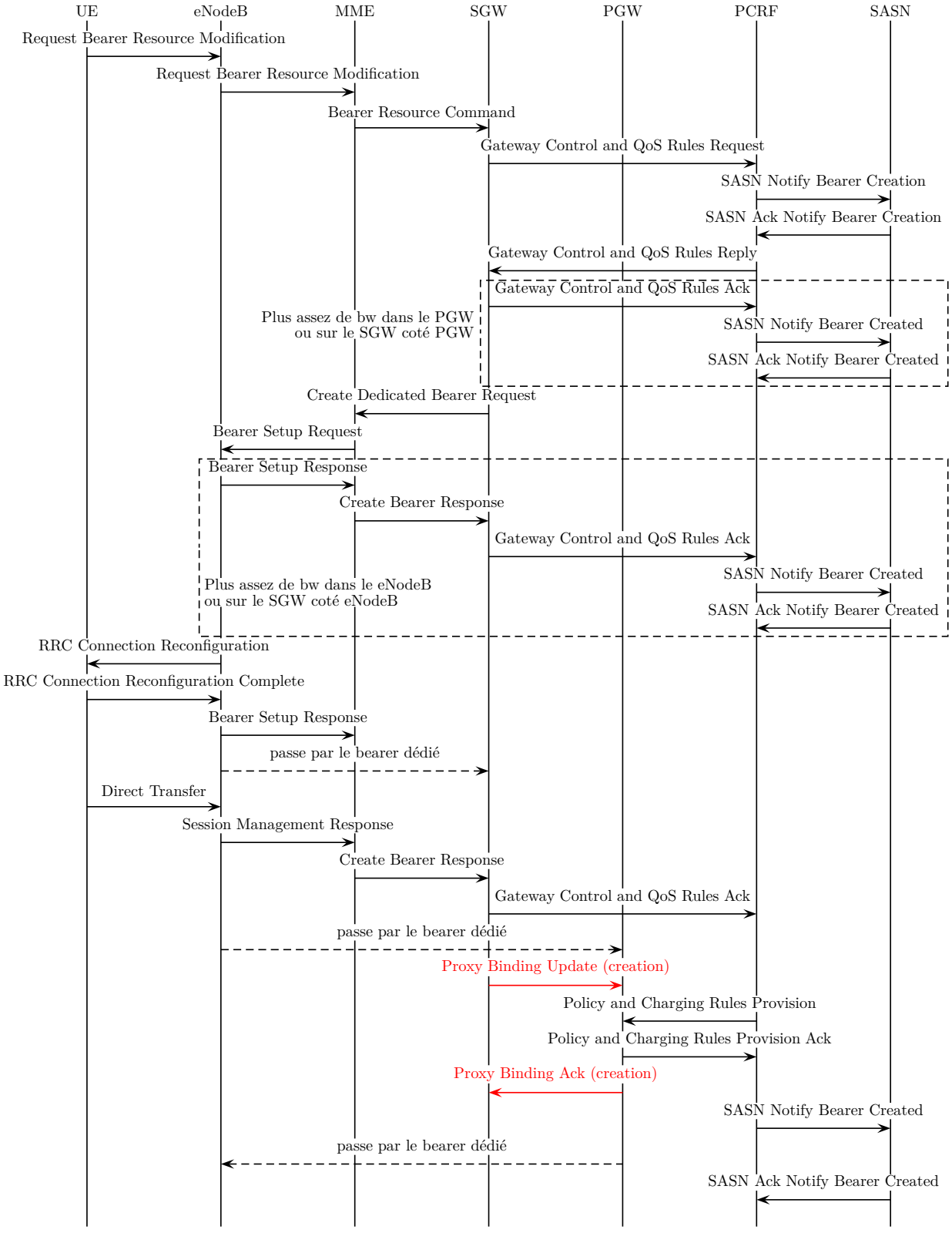


Figure C.4 – Création de *bearer* dédié avec UE QoS *aware*

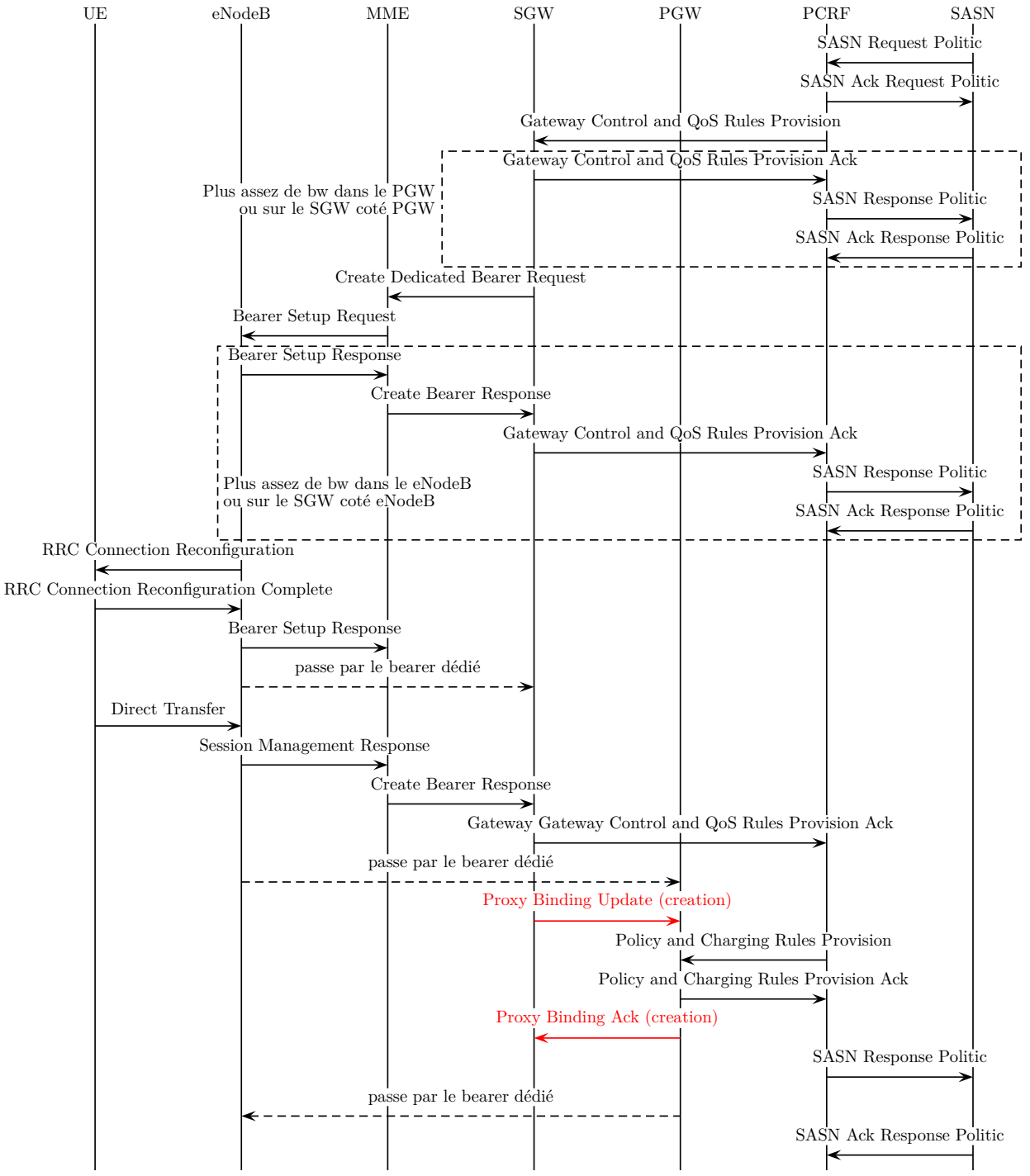


Figure C.5 – Création de *bearer* dédié avec UE QoS *unaware*

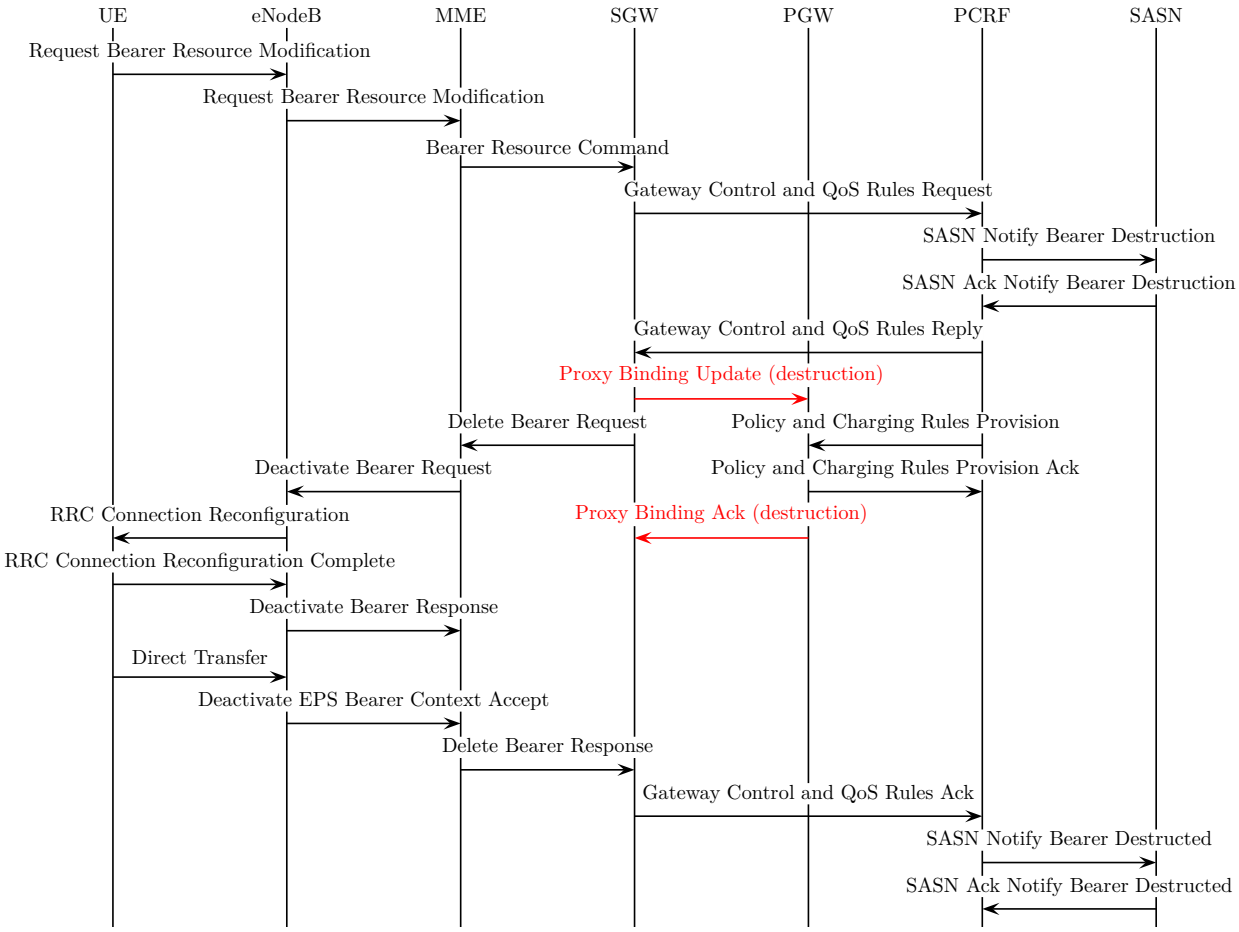


Figure C.6 – Destruction de *bearer* dédié avec UE QoS *aware*

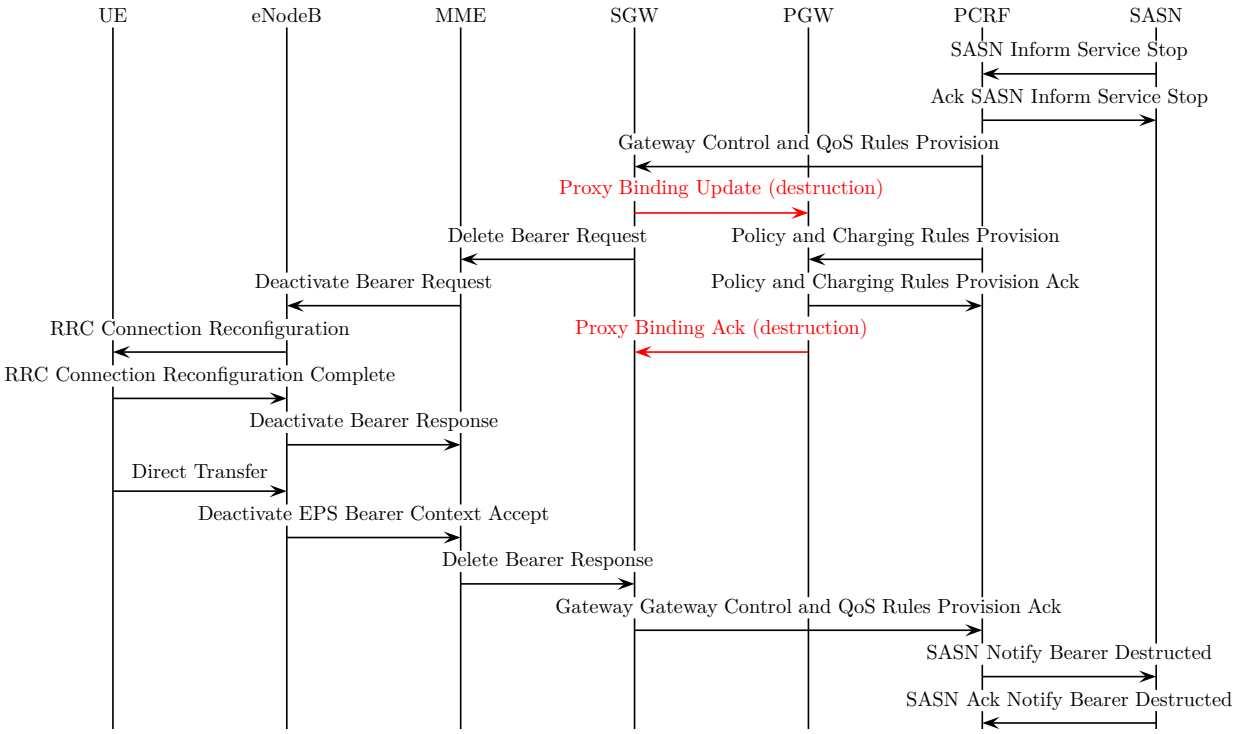


Figure C.7 – Destruction de *bearer* dédié avec UE QoS *unaware*

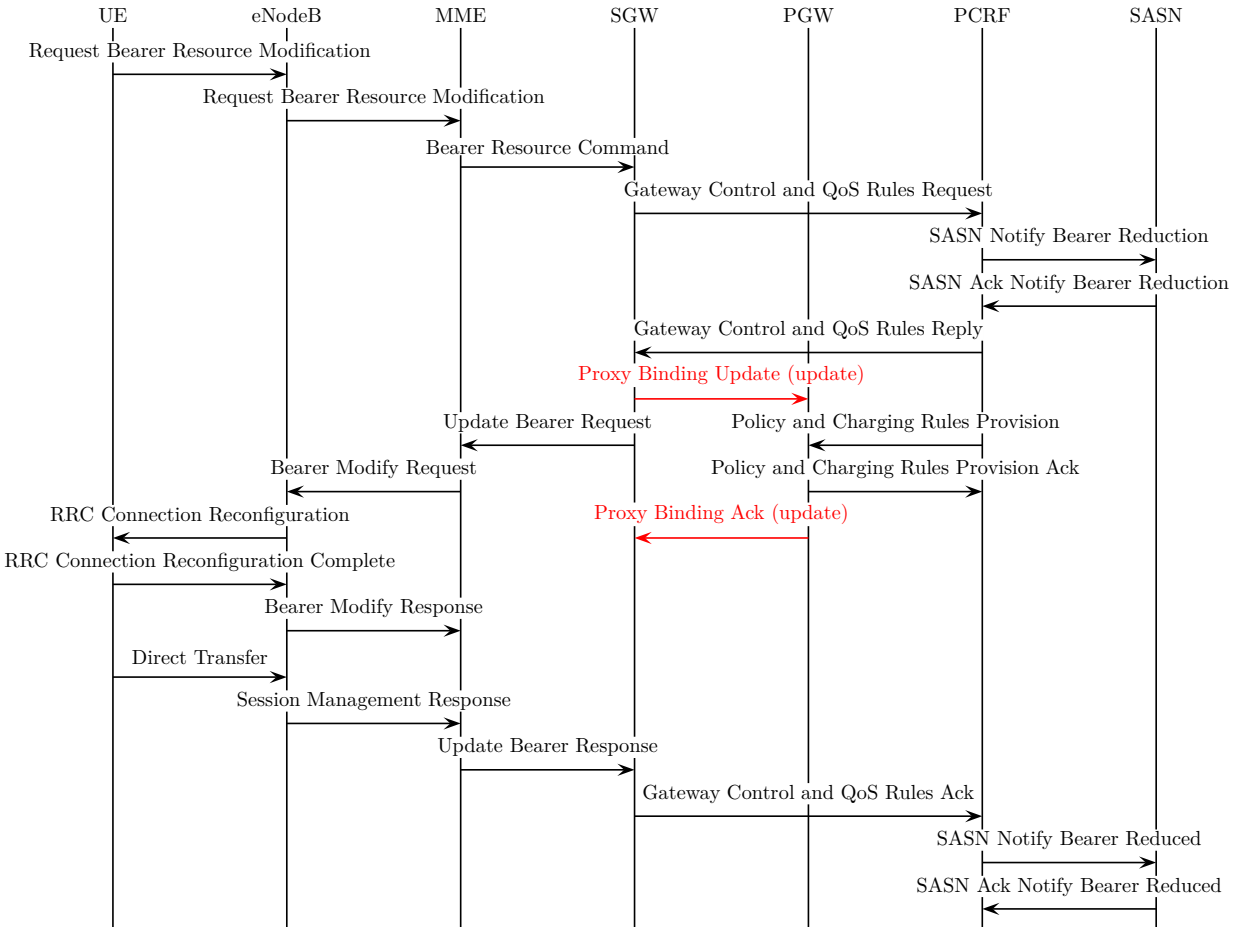


Figure C.8 – Réduction de *bearer* dédié GBR avec UE QoS *aware*

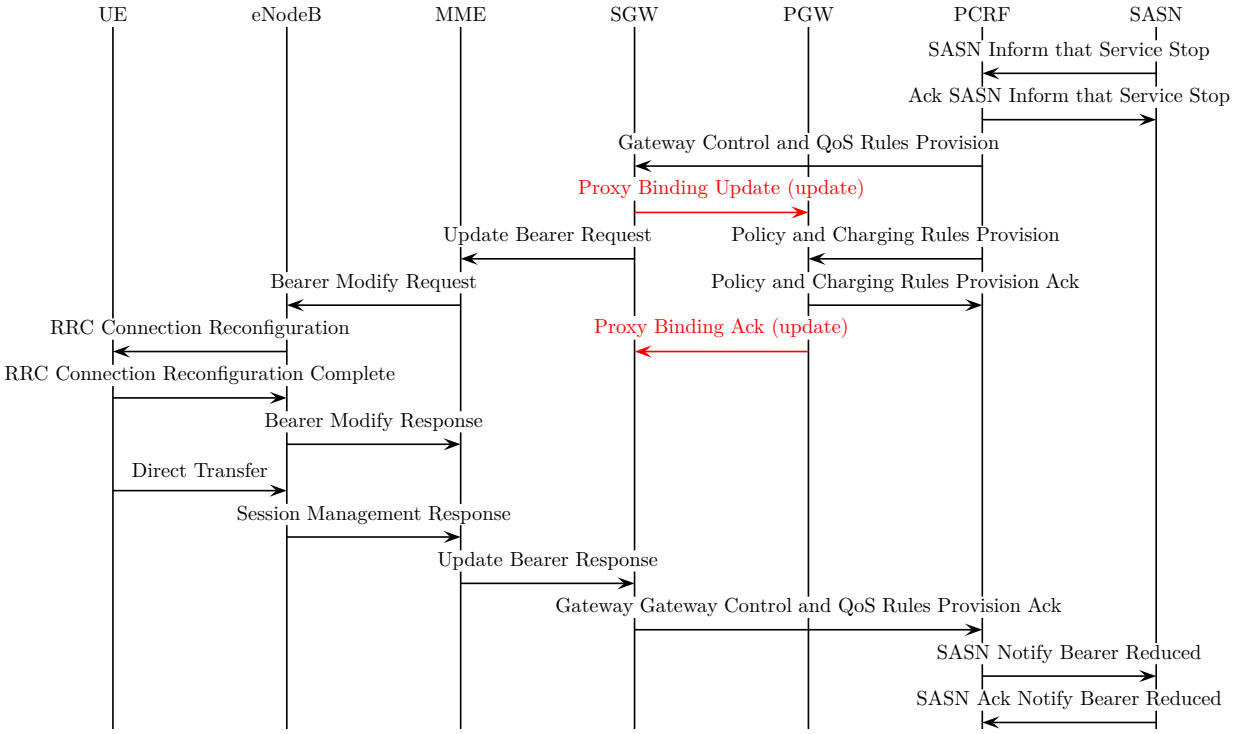


Figure C.9 – Réduction de *bearer* dédié GBR avec UE QoS *unaware*

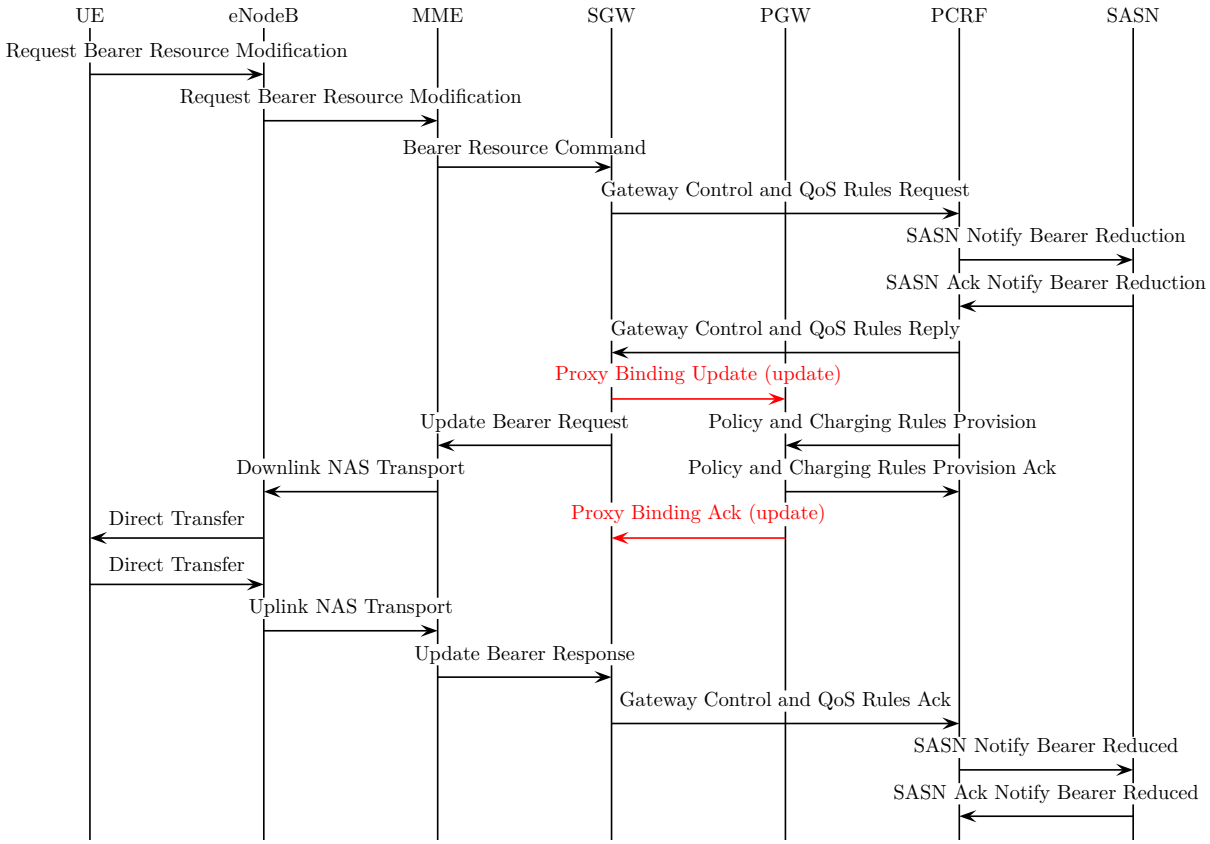


Figure C.10 – Réduction de *bearer* dédié non GBR avec UE QoS *aware*

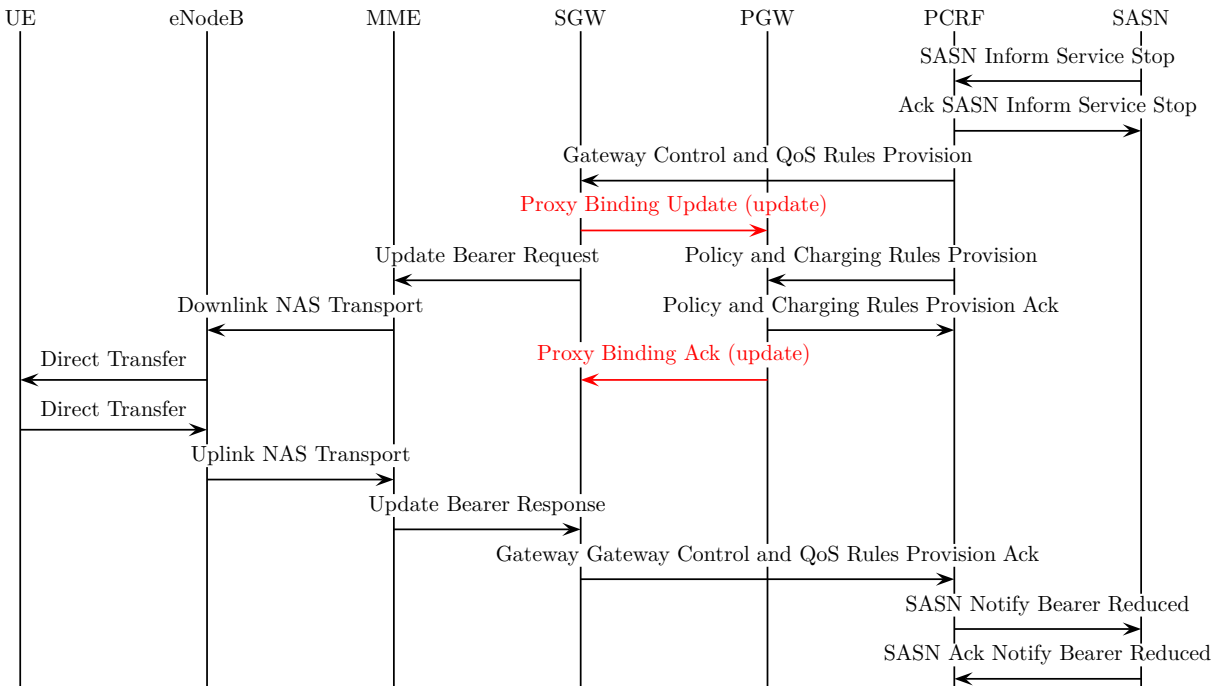


Figure C.11 – Réduction de *bearer* dédié non GBR avec UE QoS *unaware*

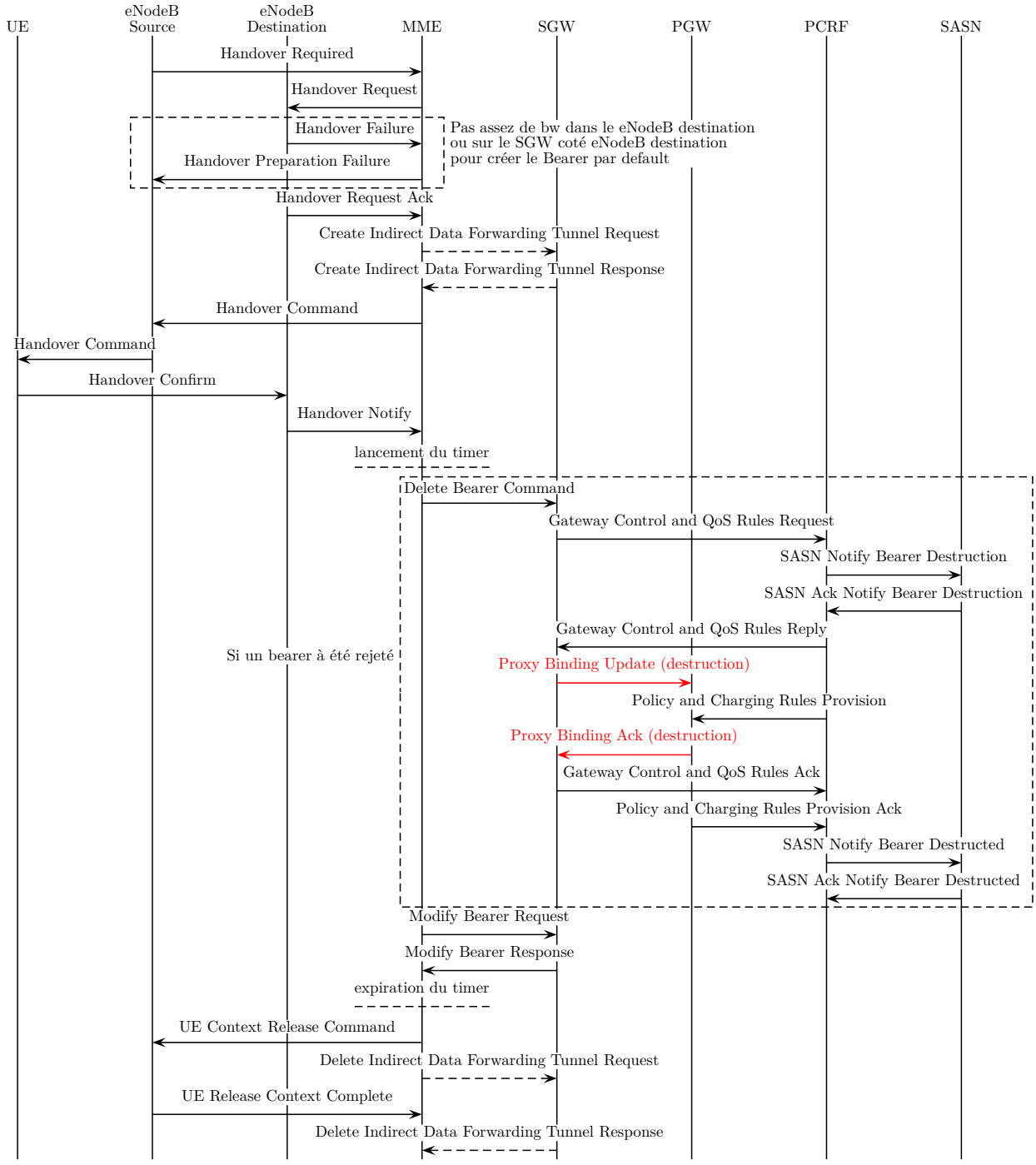


Figure C.12 – S1 Handover

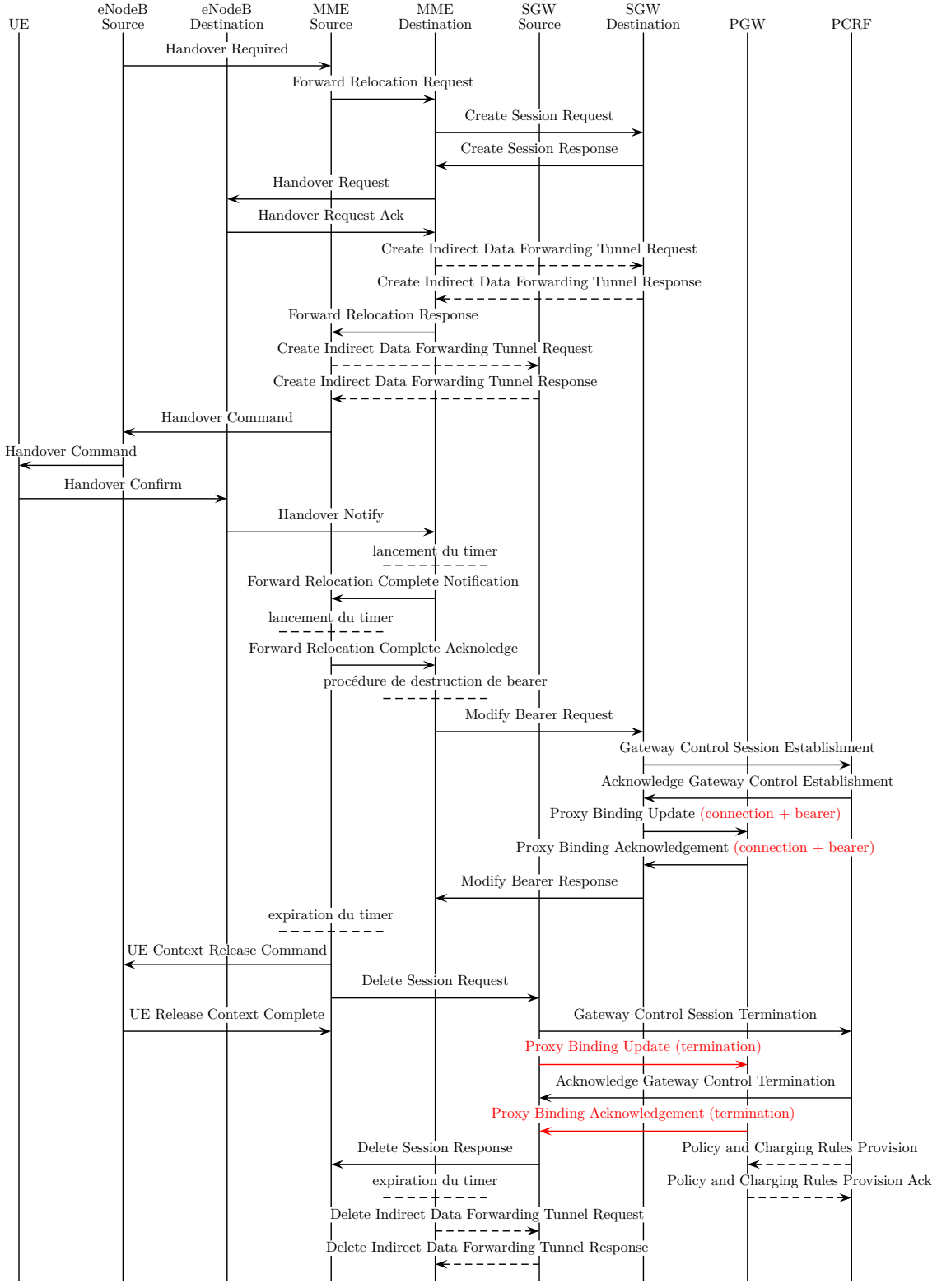


Figure C.13 – S1 Handover avec changement de SGW

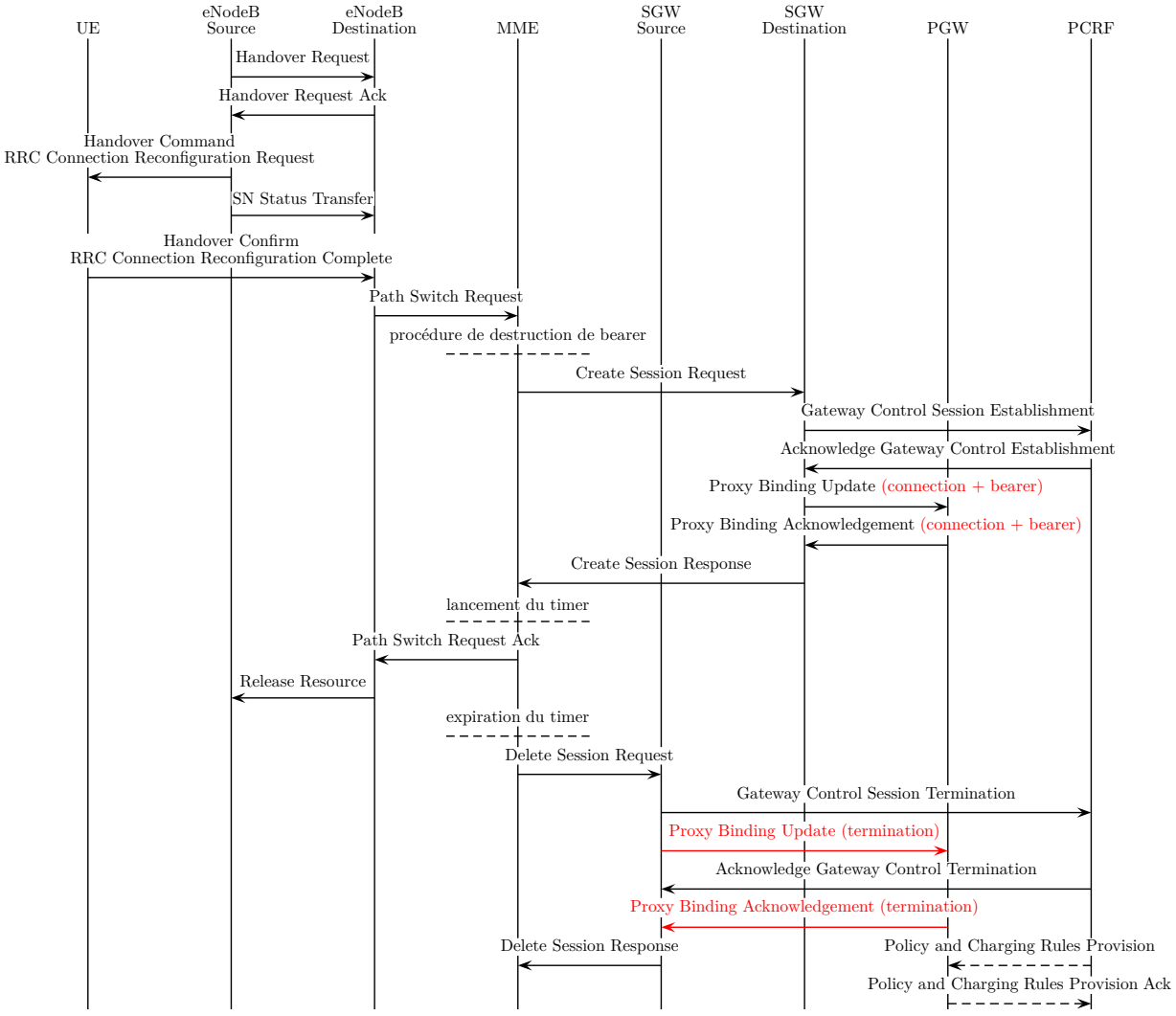


Figure C.14 – X2 Handover avec changement de SGW

ANNEXE D

Diagrammes de séquences de l'architecture 404

Les figures suivantes représentent les diagrammes de séquences des principales fonctions de gestion de bearer dans l'architecture 404, soit l'architecture 403 modifiée pour supprimer les protocoles GTP-U et GRE et intégrer PMIPv6 entre le eNodeB et le SGW. Les informations utilisées pour obtenir ces diagrammes sont tirées des documents 23.401 [8], 23.402 [9], 23.203 [7] et 36.300 [29], ainsi que des écrits de Gundavelli *et al.* [65] et Hui *et al.* [70].

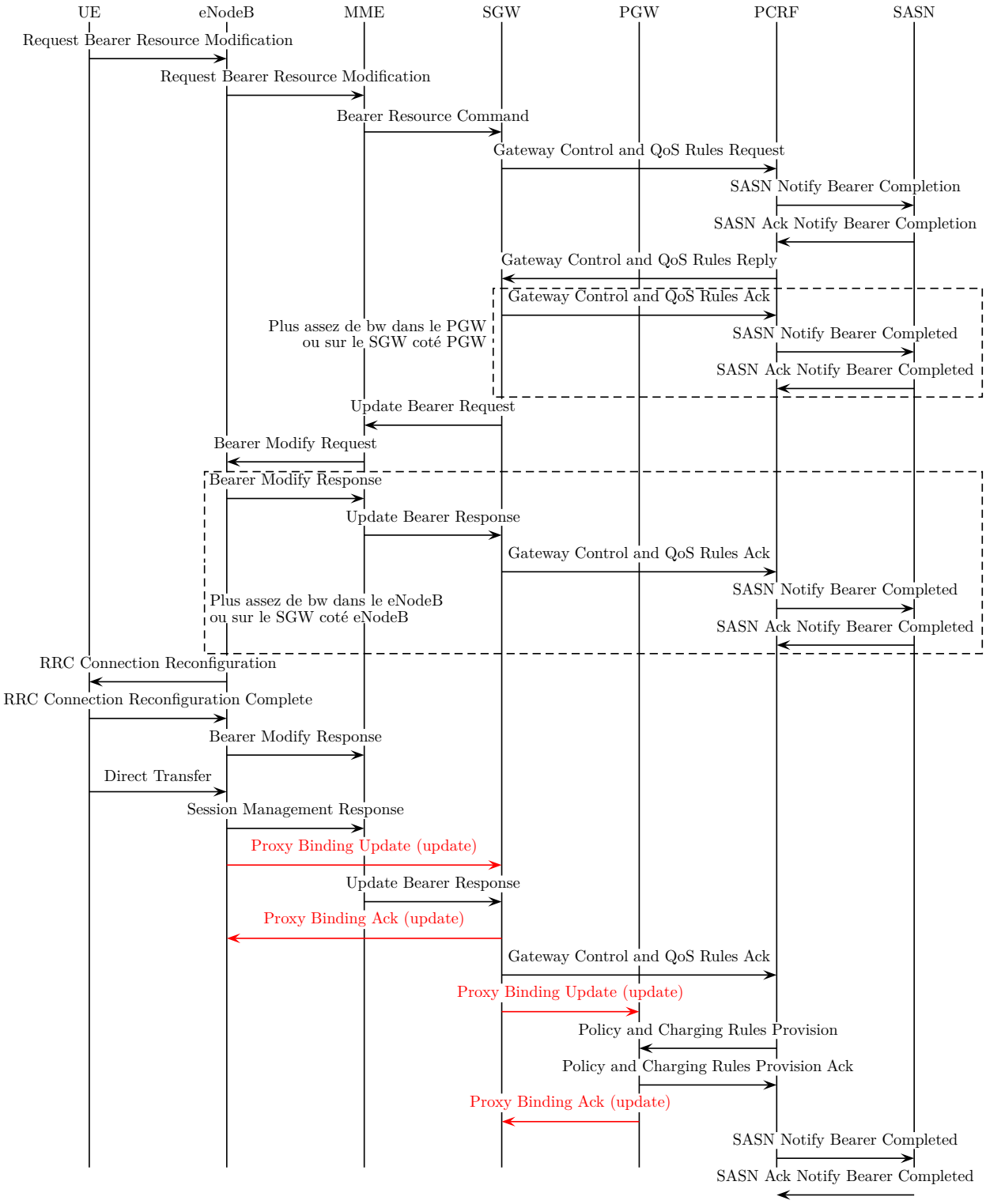


Figure D.1 – Complétion de *bearer* dédié GBR avec UE QoS *aware*

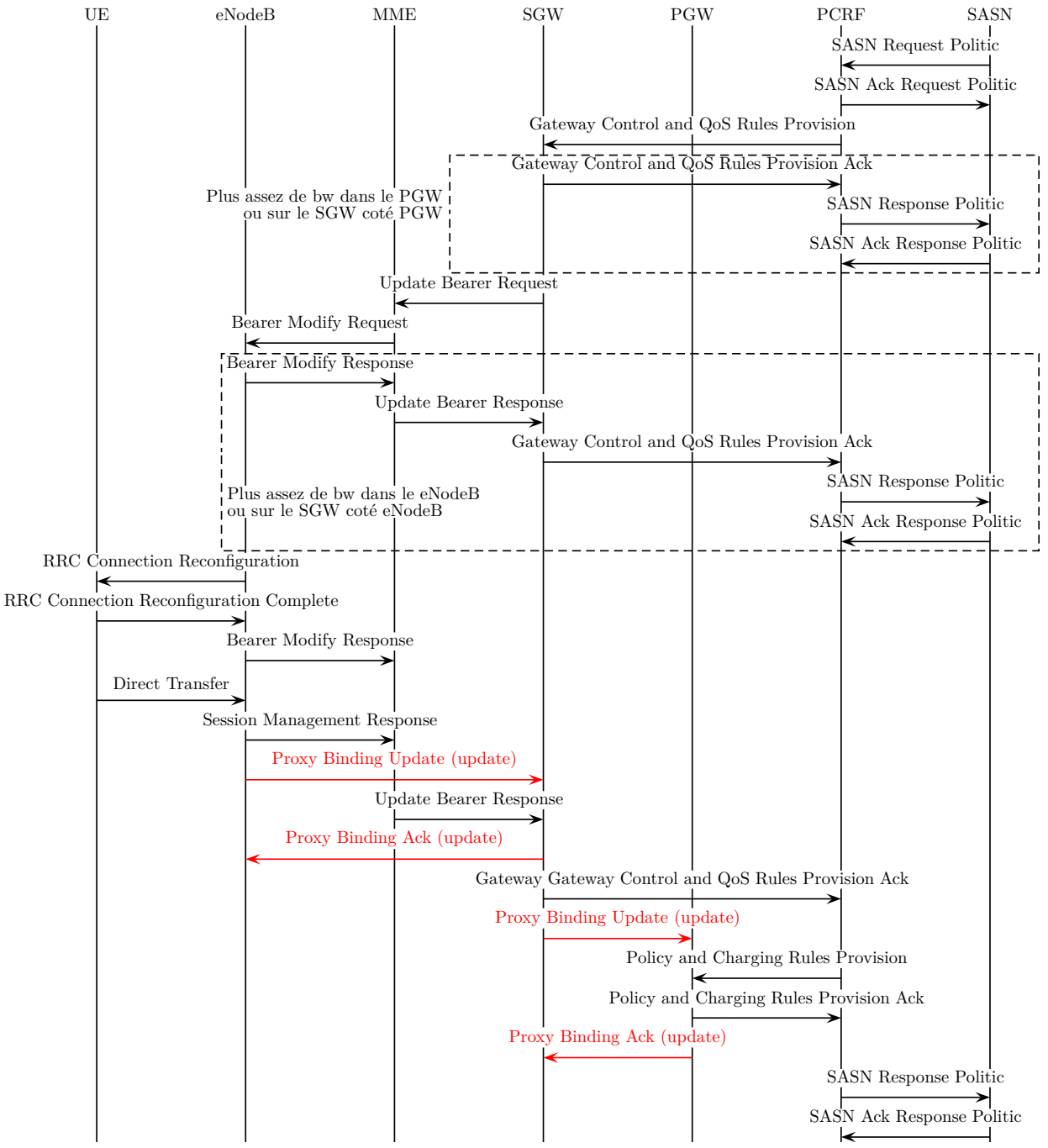


Figure D.2 – Complétion de *bearer* dédié GBR avec UE QoS *unaware*

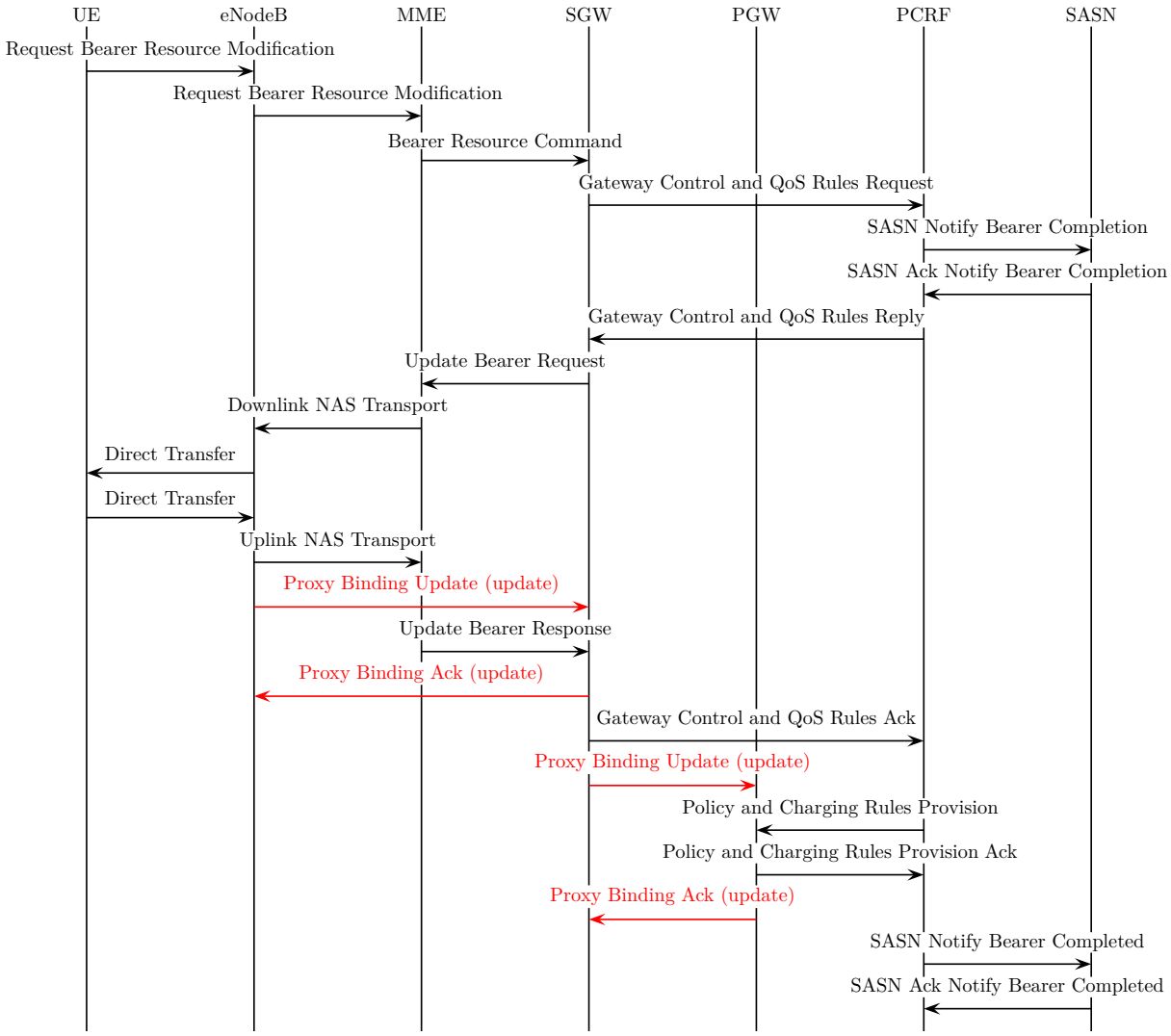


Figure D.3 – Complétion de *bearer* dédié non GBR avec UE QoS *aware*

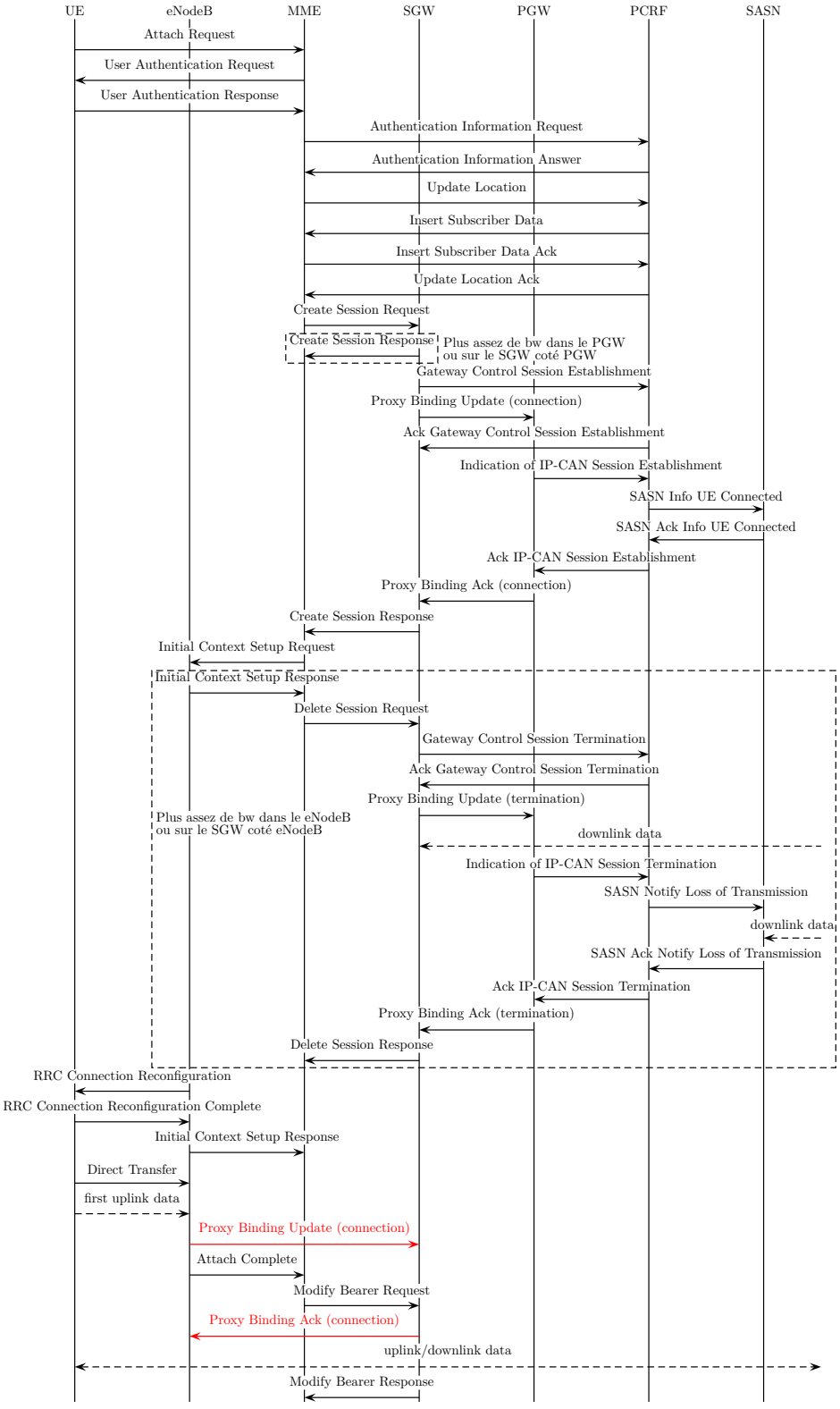


Figure D.4 – Connexion d'un UE

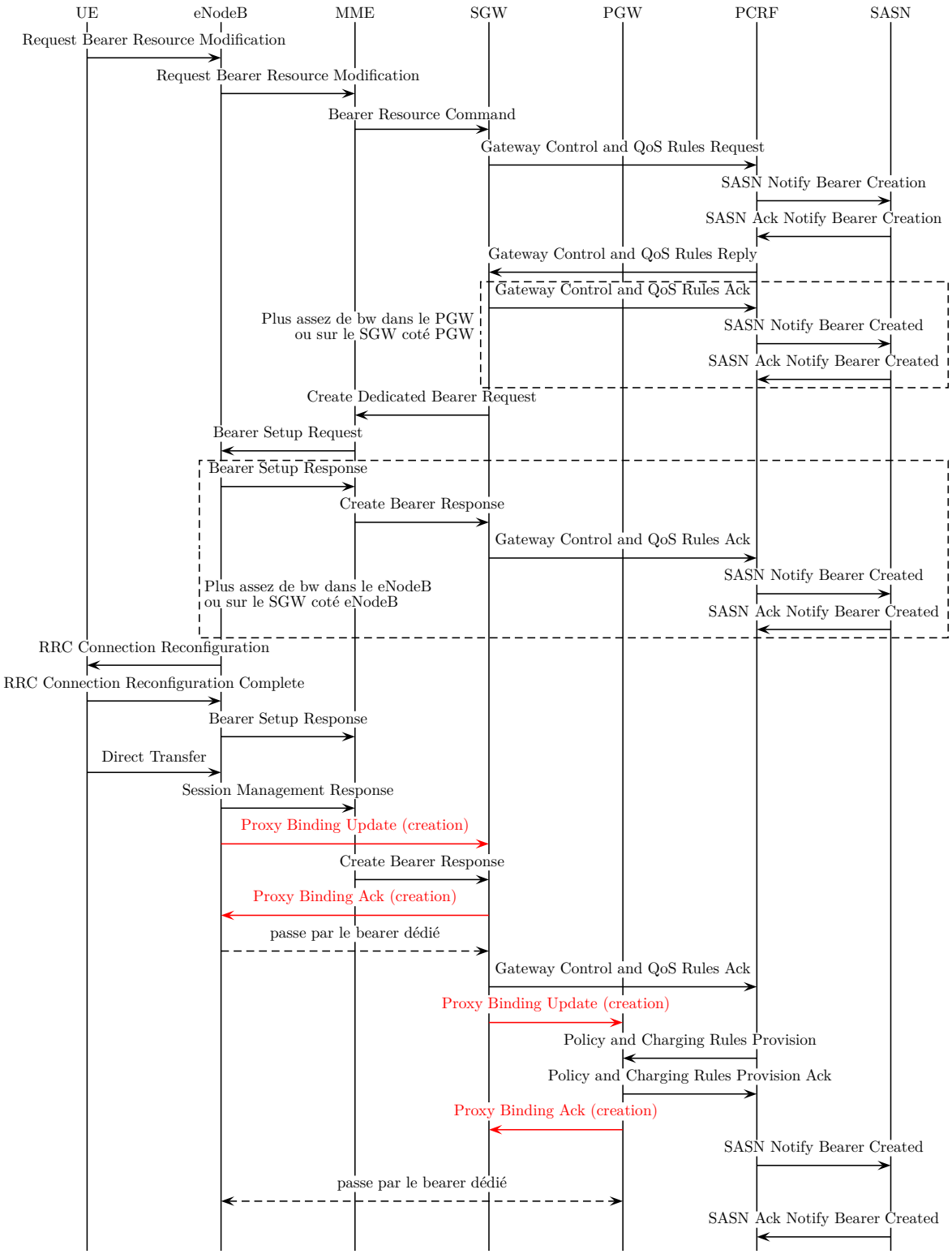


Figure D.5 – Création de *bearer* dédié avec UE QoS *aware*

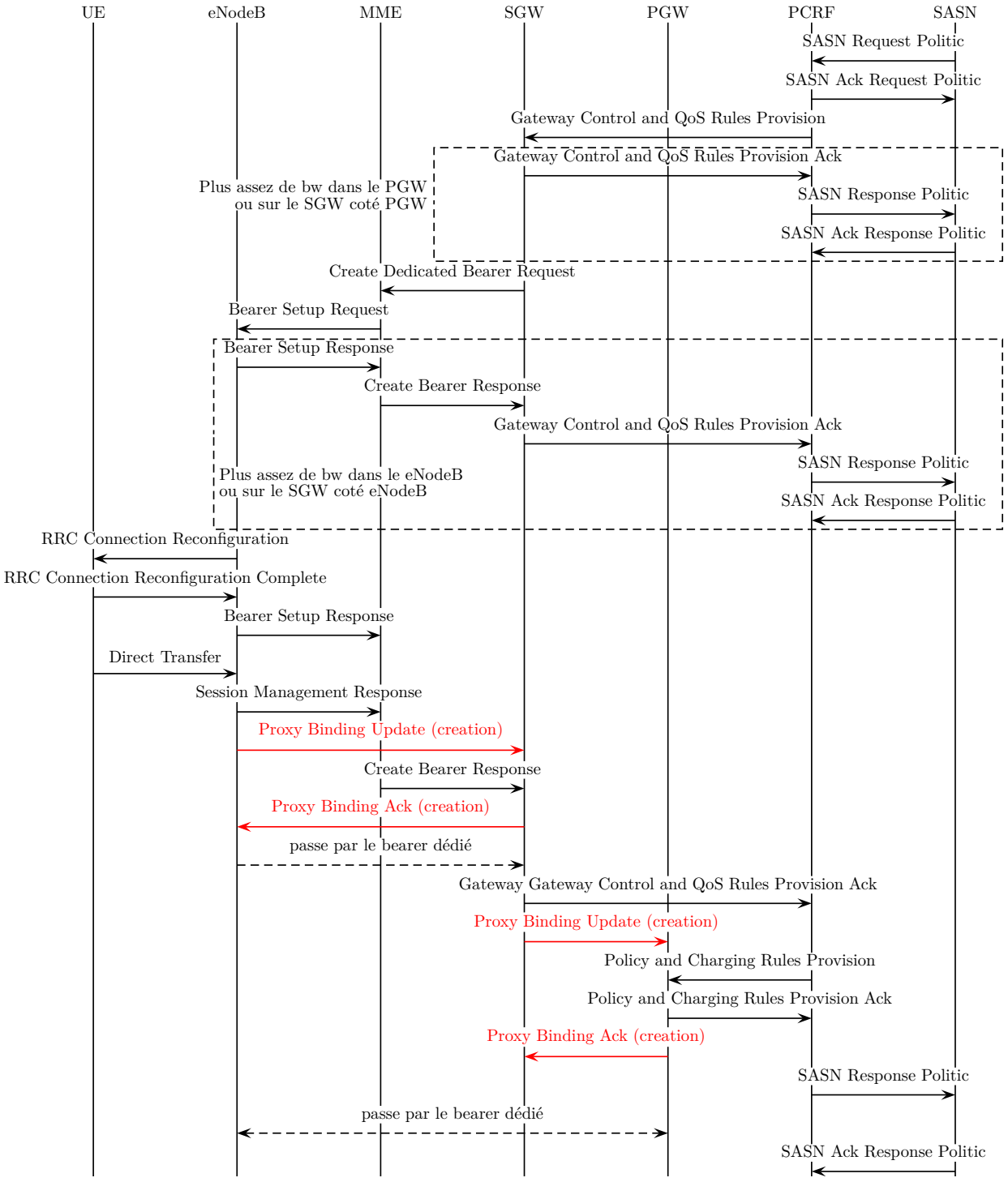


Figure D.6 – Création de *bearer* dédié avec UE QoS *unaware*

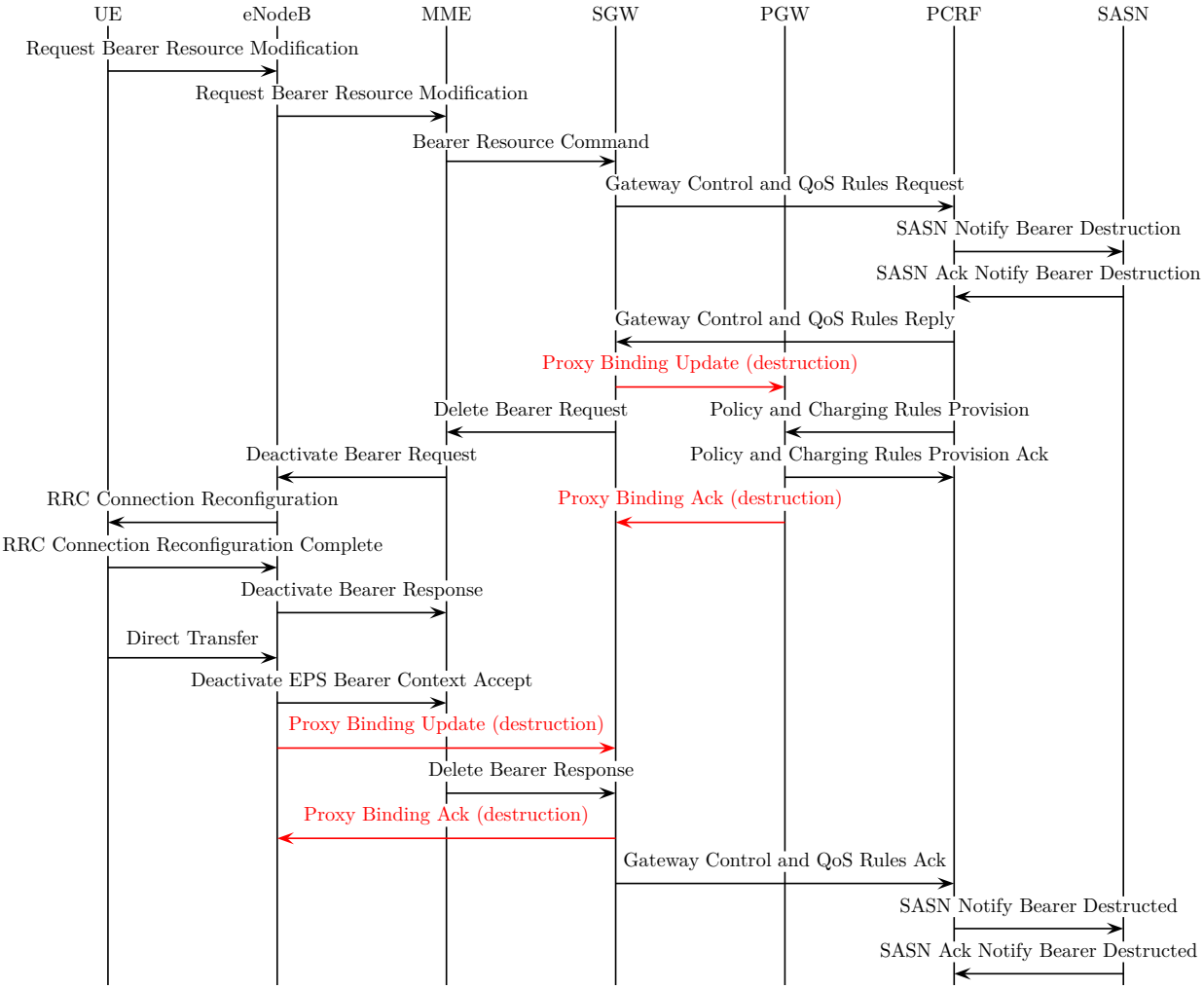


Figure D.7 – Destruction de *bearer* dédié avec UE QoS *aware*

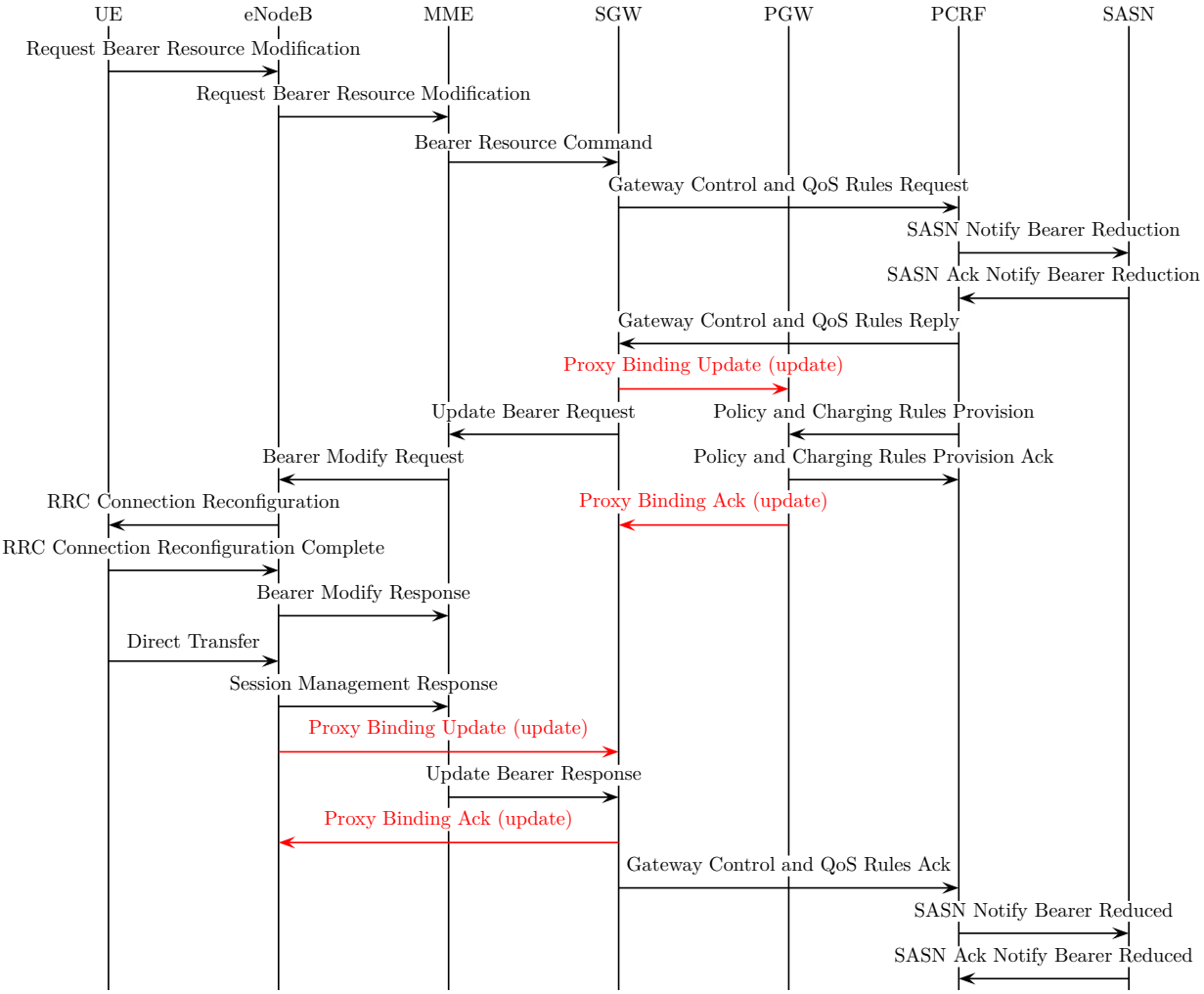


Figure D.8 – Réduction de *bearer* dédié GBR avec UE QoS *aware*

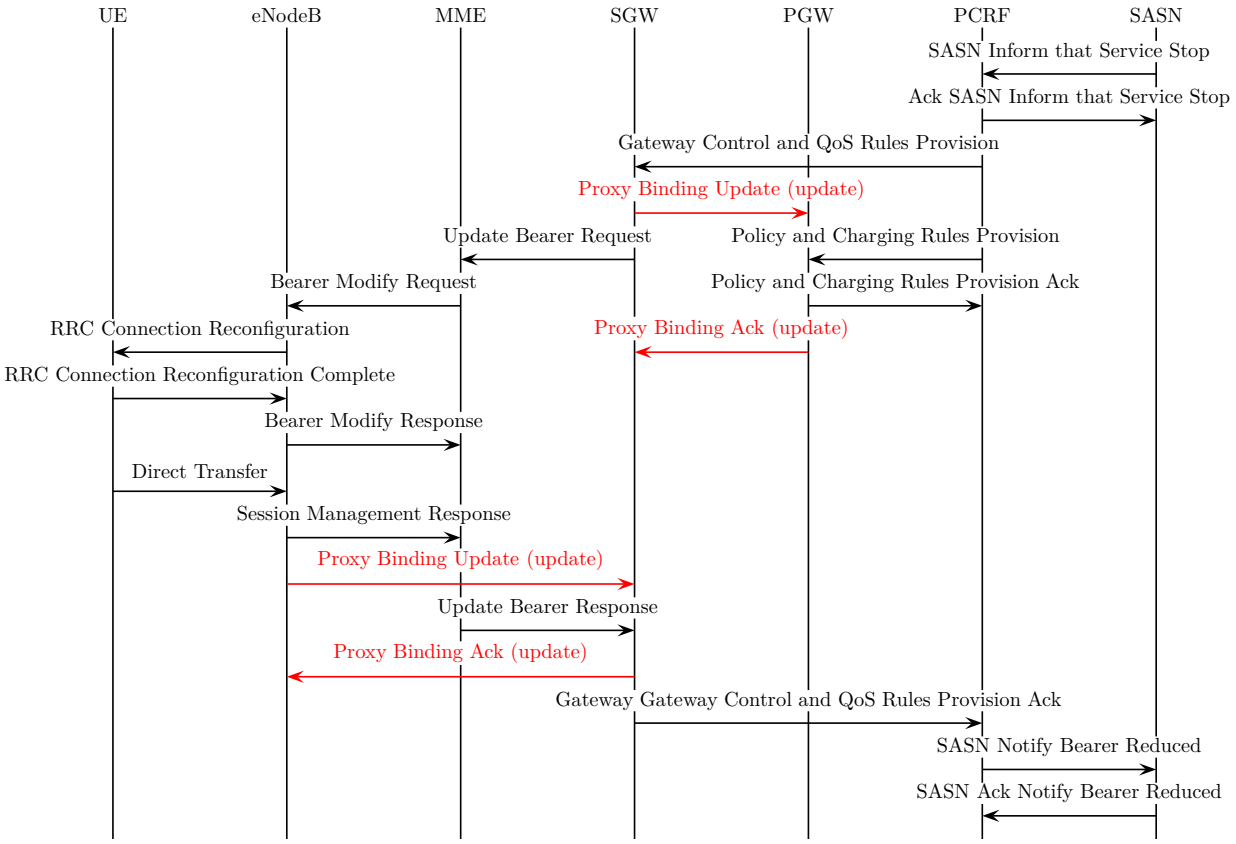


Figure D.9 – Réduction de *bearer* dédié GBR avec UE QoS *unaware*

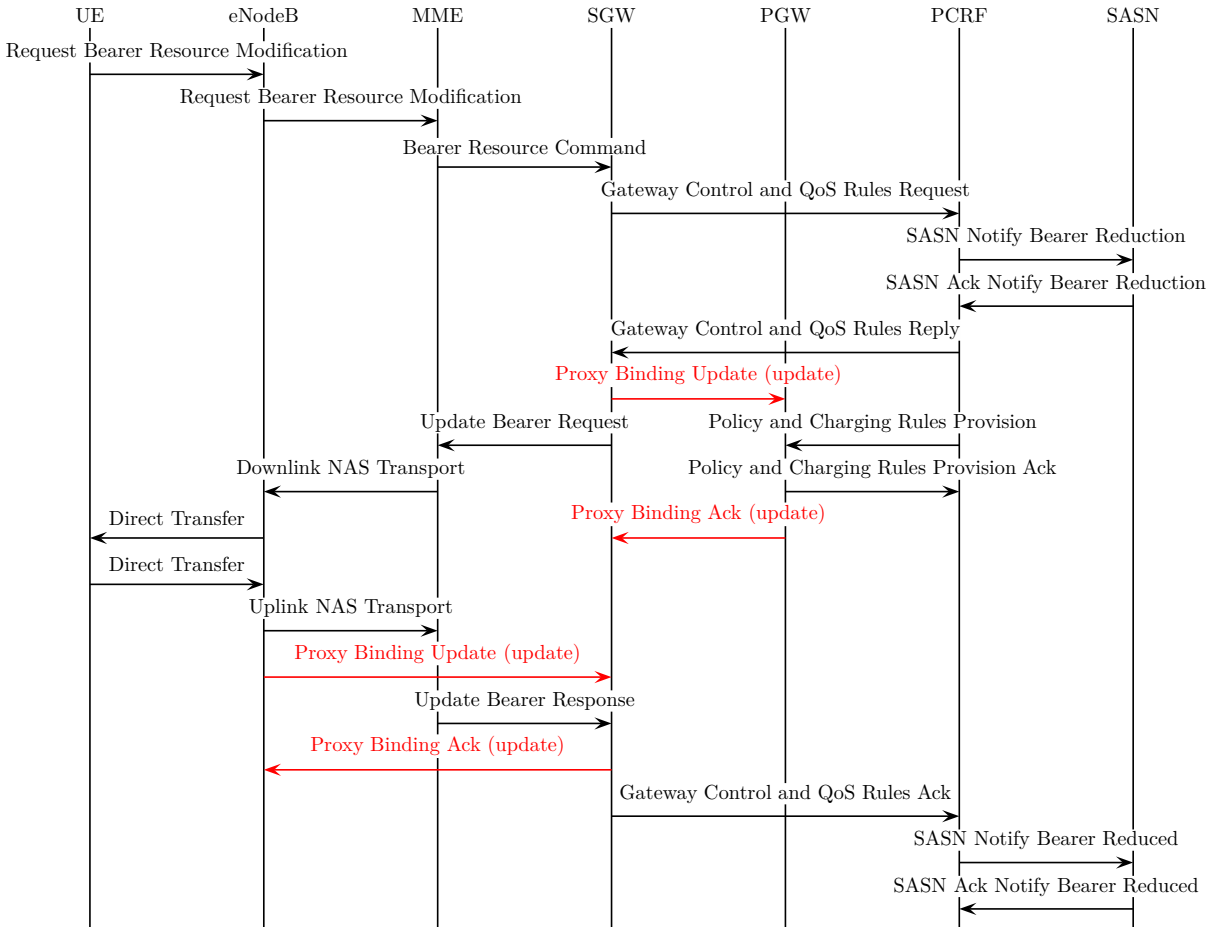


Figure D.10 – Réduction de *bearer* dédié non GBR avec UE QoS *aware*

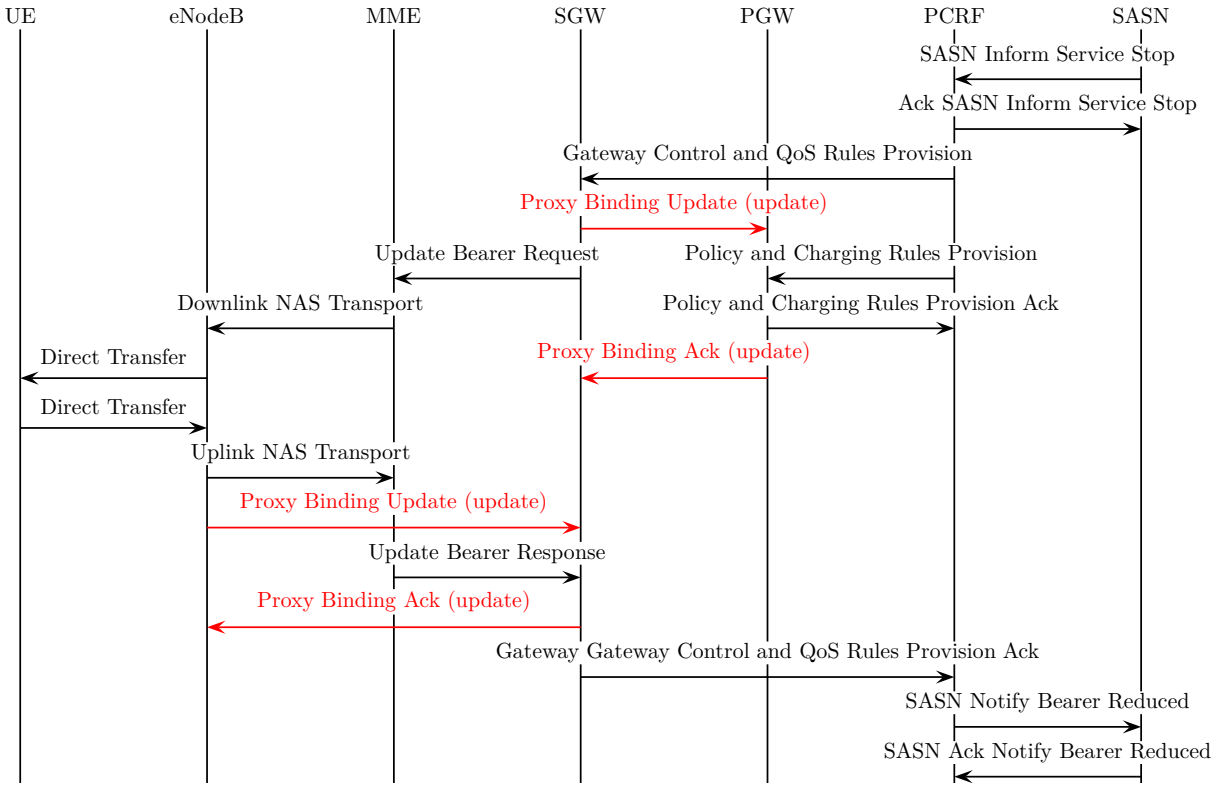


Figure D.11 – Réduction de *bearer* dédié non GBR avec UE QoS *unaware*

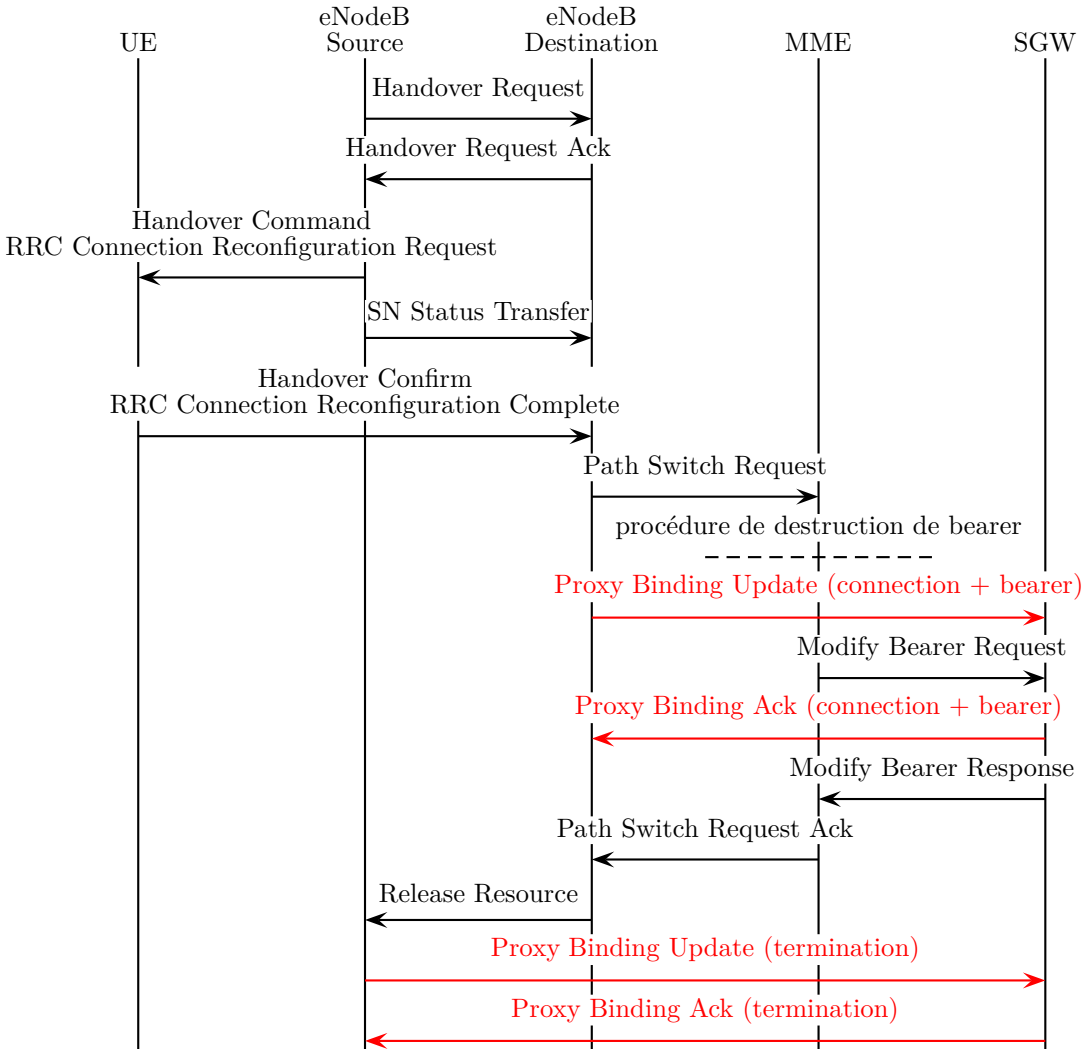


Figure D.12 – X2 Handover

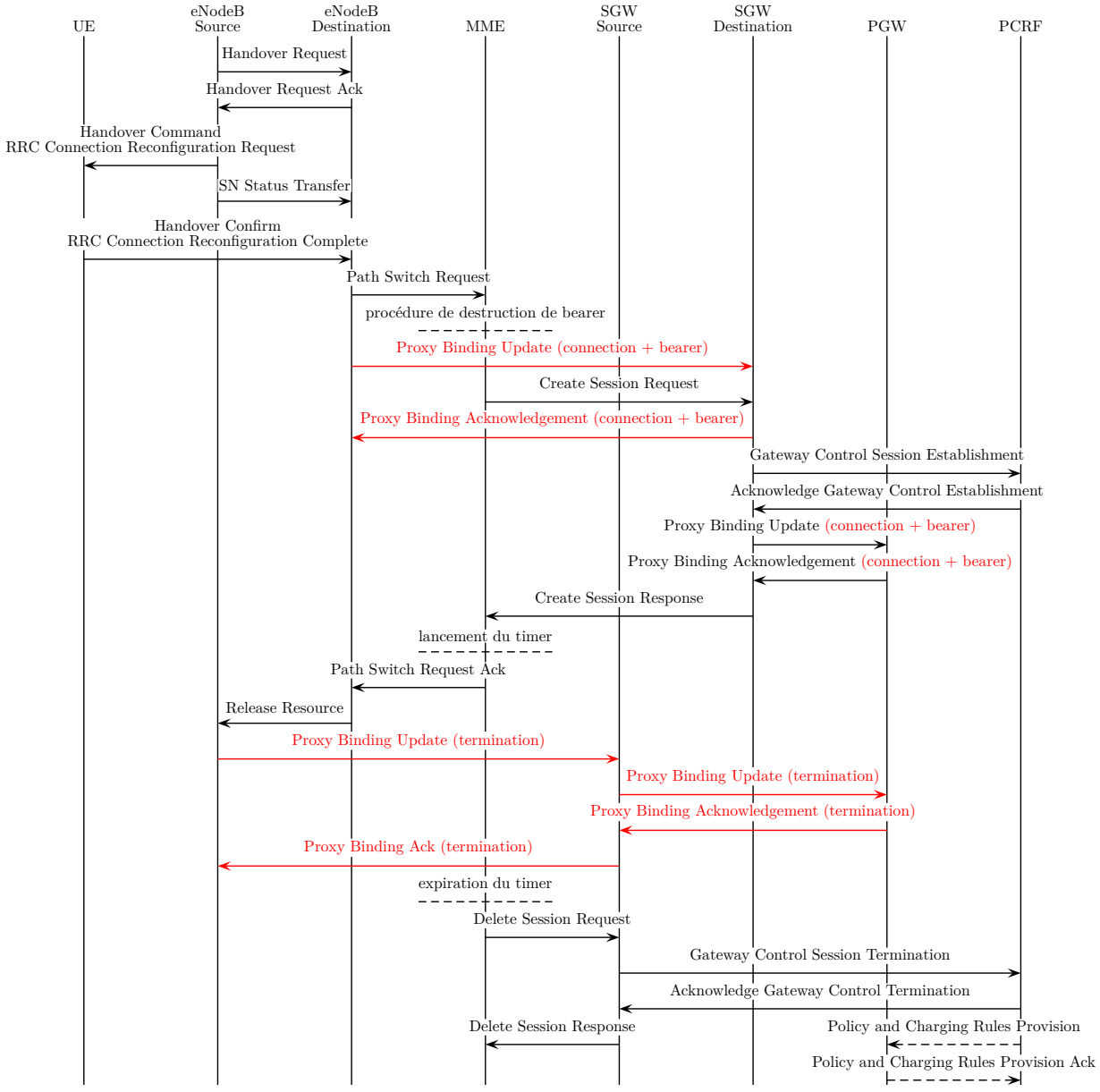


Figure D.13 – X2 Handover avec changement de SGW

ANNEXE E

Résultats des simulations avec un seul UE

Les figures suivantes présentes des comparaisons des résultats obtenus par nos différents simulateurs. Les simulations utilisent toutes le même scénario, un UE se connecte à la première seconde et durant les trois premières secondes après sa connexion il lance des trafics voix, vidéo et web. À la trentième seconde après sa connexion il fait un relève vers l'eNodeB suivant. À la cinquantième seconde il lance son trafic FTP. Chaque connexion (type de trafic) dure cent secondes. Le réseau comporte deux eNodeB.

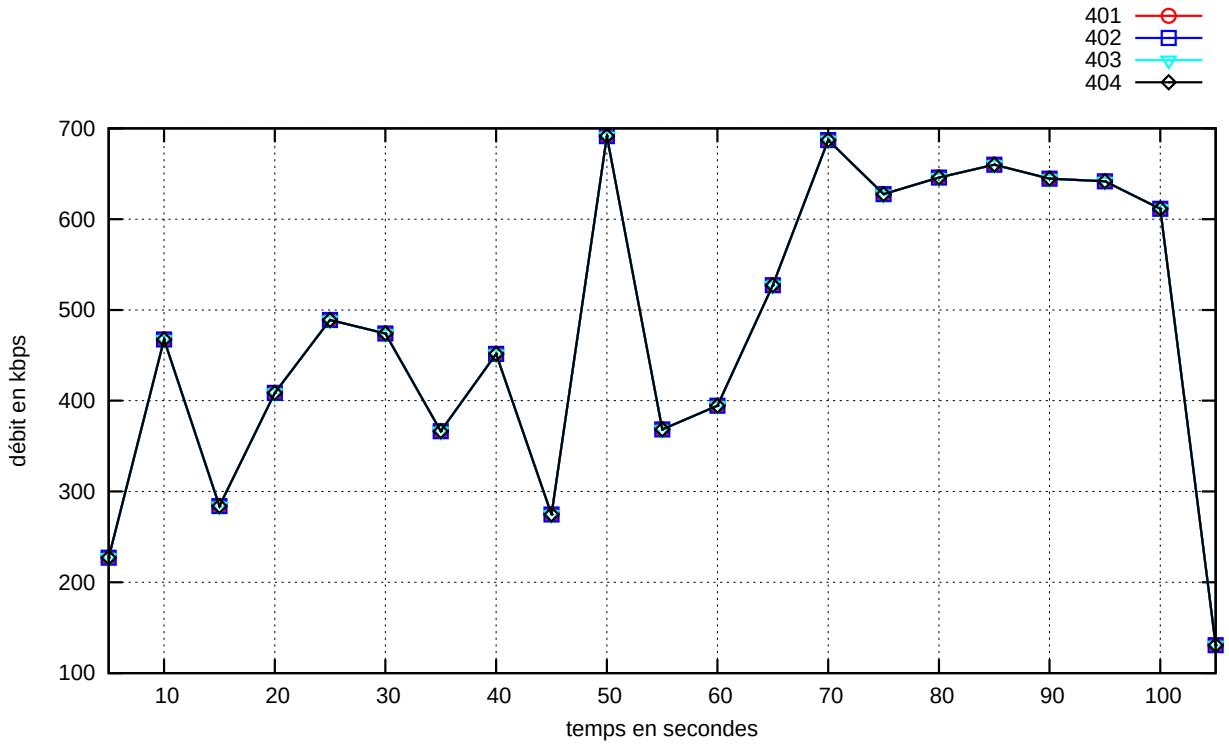


Figure E.1 – Débit du trafic vidéo

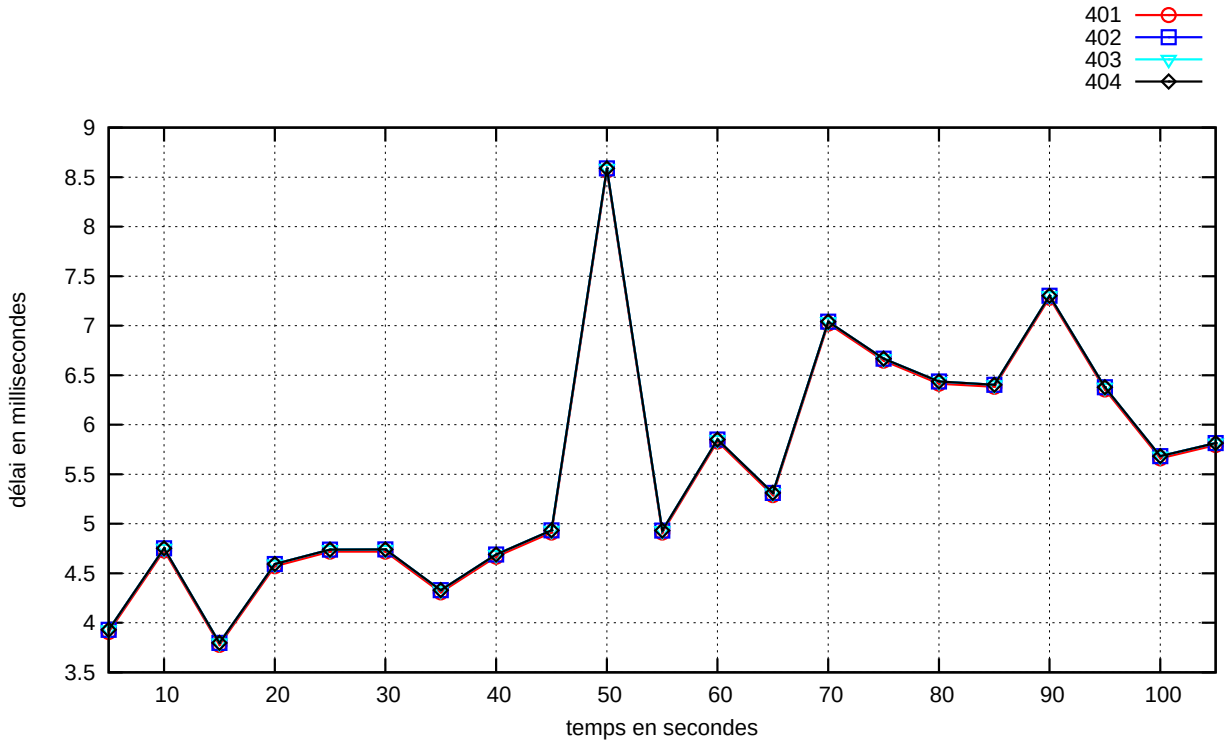


Figure E.2 – Délai moyen du trafic vidéo entre le SASN et le UE

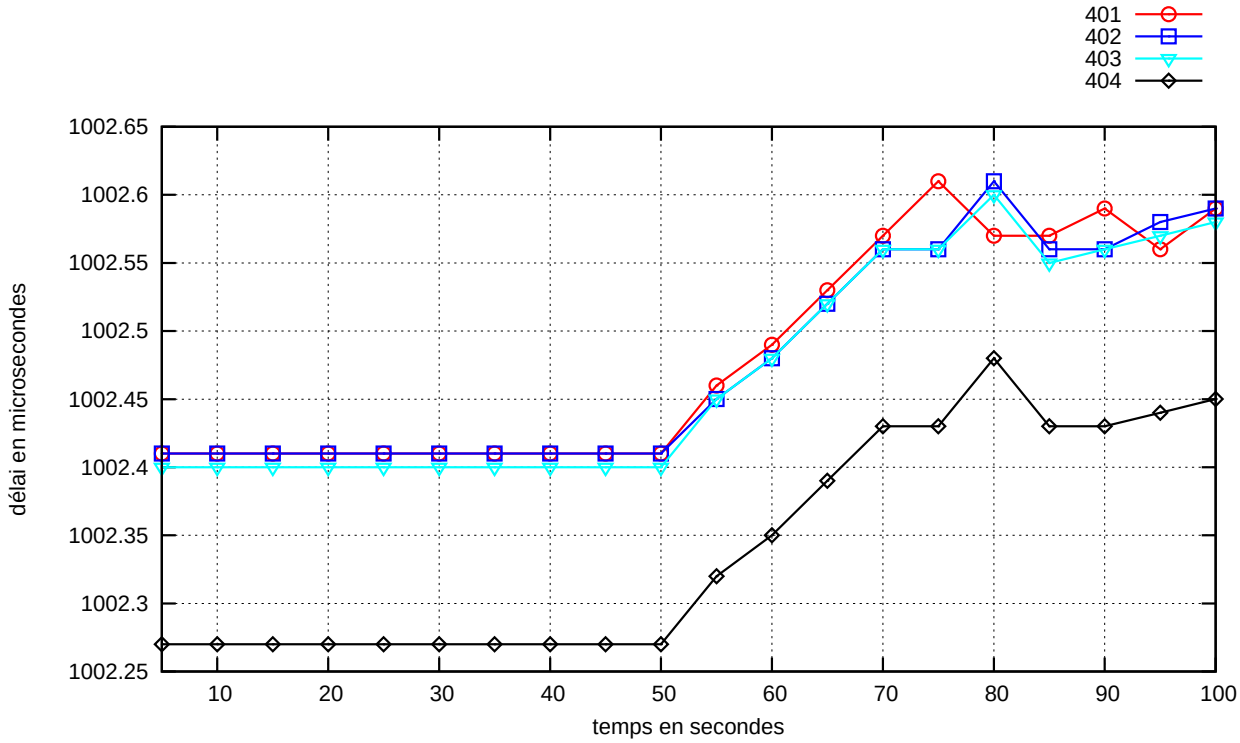


Figure E.3 – Délai moyen du trafic de voix entre le SASN et le eNodeB

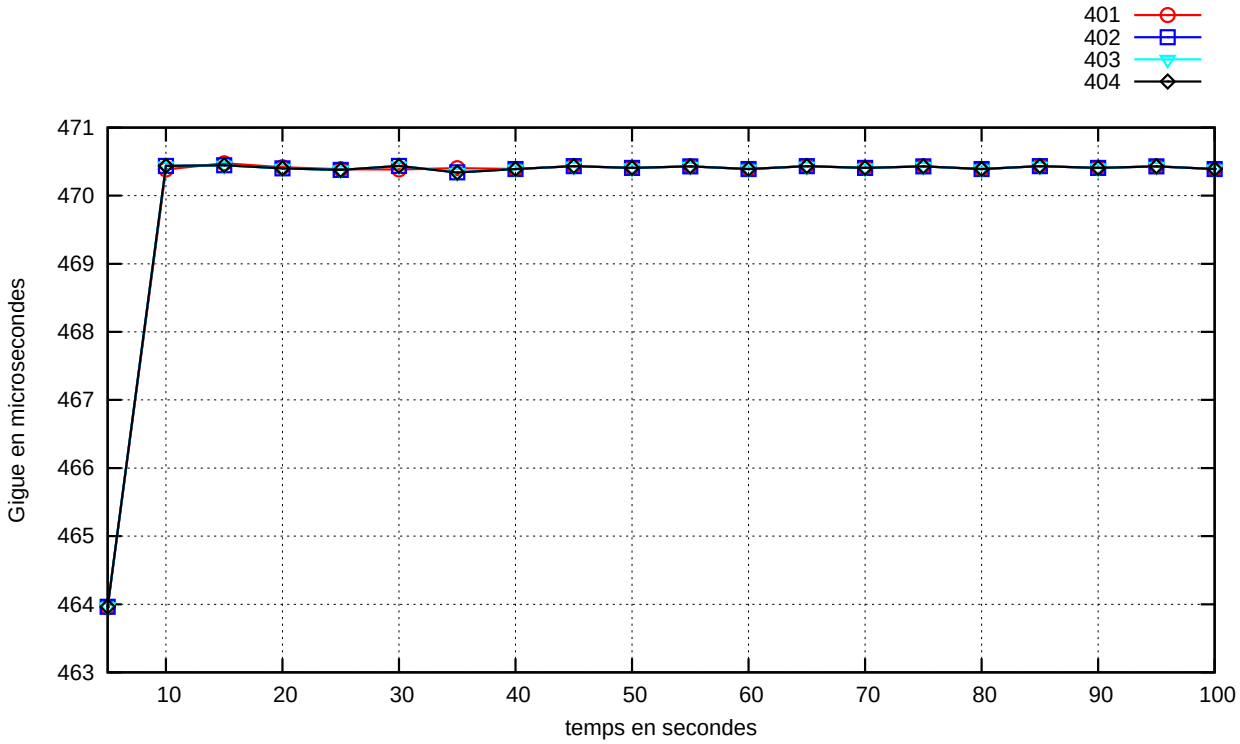


Figure E.4 – Gigue du trafic de voix entre le SASN et le UE

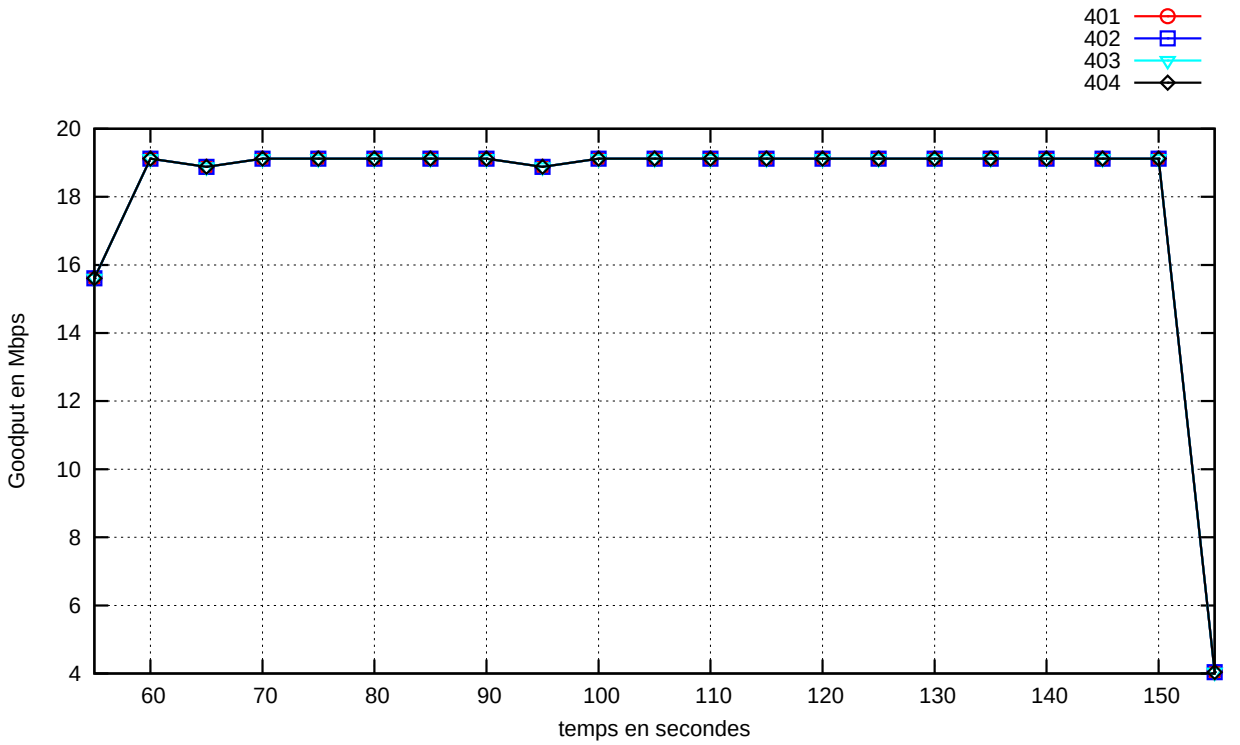


Figure E.5 – Goodput du trafic FTP

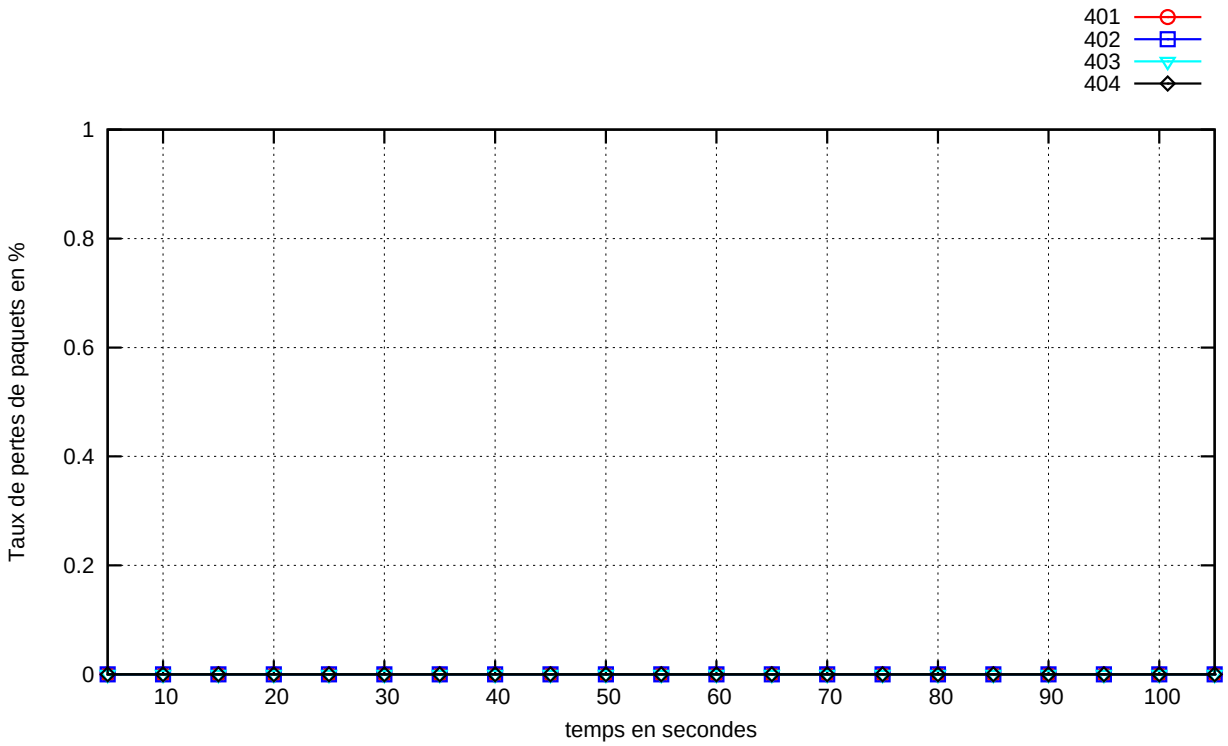


Figure E.6 – Taux de perte de paquets du trafic de voix entre le SASN et le UE

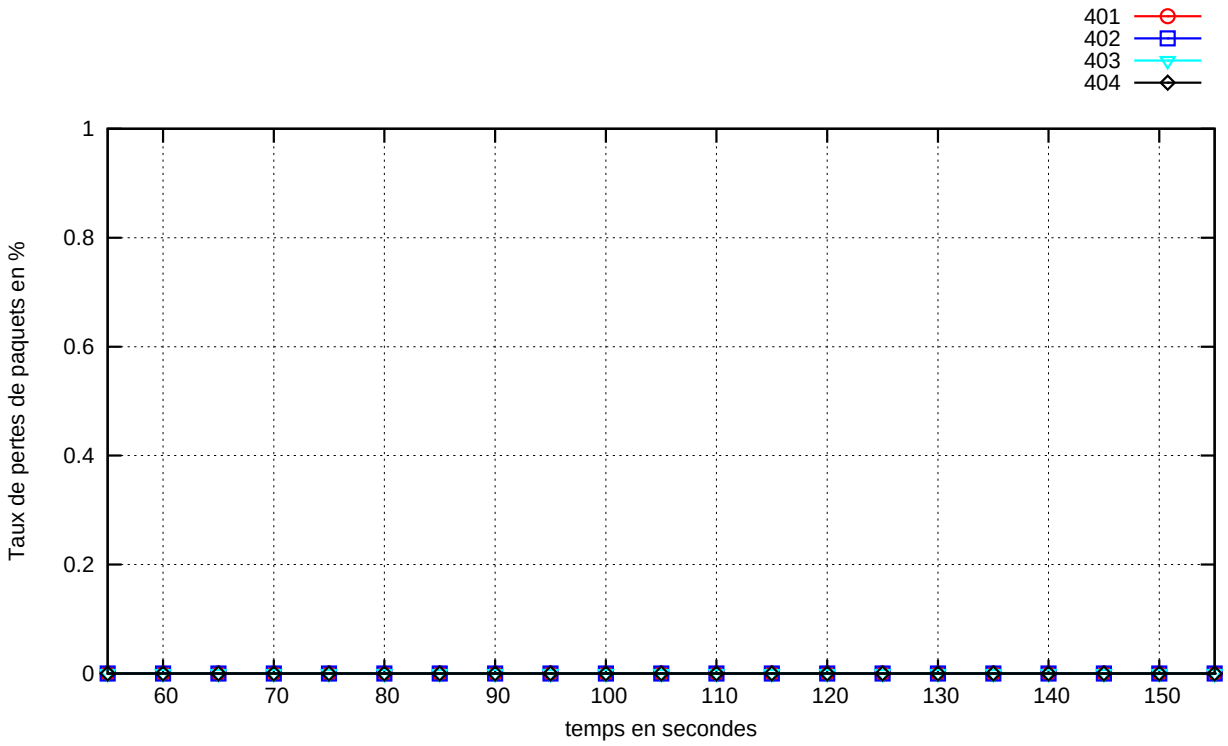


Figure E.7 – Taux de perte de paquets du trafic FTP entre le SASN et le UE

ANNEXE F

Résultats des simulations avec mille UE

Les figures suivantes présentes des comparaisons des résultats obtenus par nos différents simulateurs. Les simulations utilisent toutes le même scénario, un UE se connecte toutes les secondes, durant les trois premières secondes après sa connexion il lance des trafics voix, vidéo et web. À la trentième seconde après sa connexion il fait un relève vers l'eNodeB suivant (dans une liste de eNodeB partagé par tous les UE) celle sur laquelle il est attaché actuellement. À la cinquantième seconde il lance son trafic FTP. Chaque connexion (type de trafic) dure cent secondes. En tout mille UE se connectent, le réseau contenant dix eNodeB.

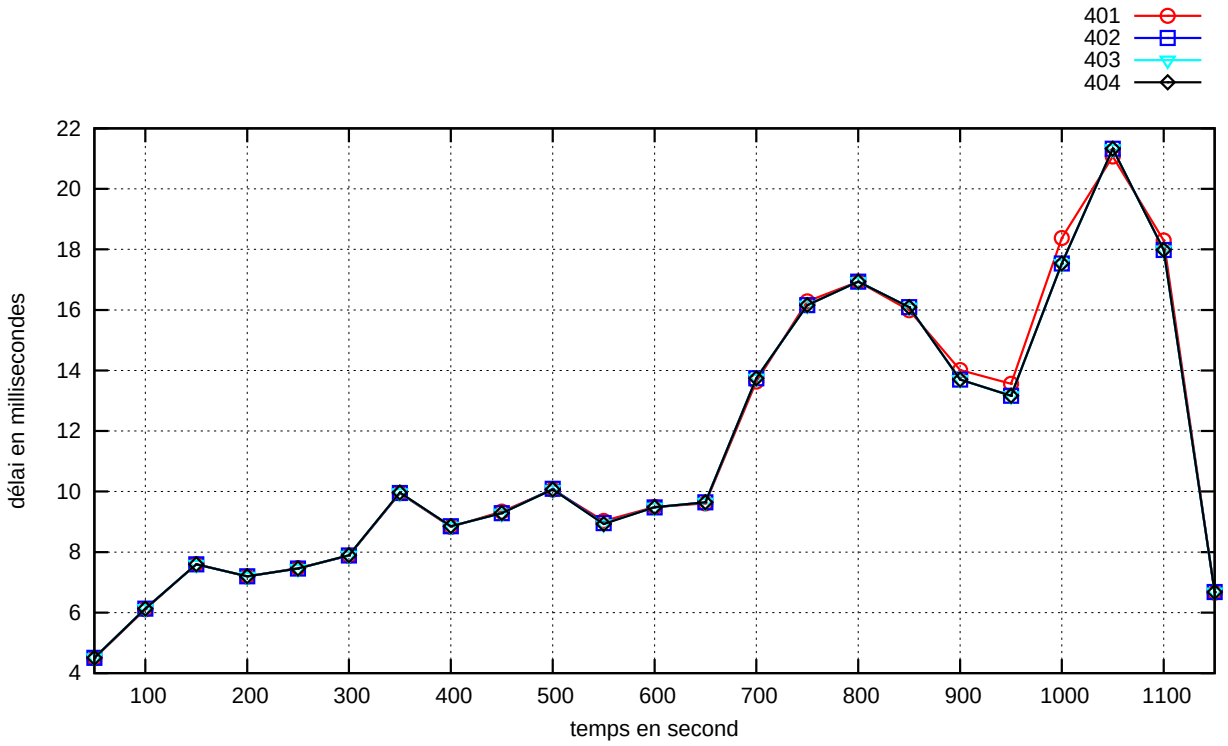


Figure F.1 – Délai moyen du trafic vidéo entre le SASN et les UE

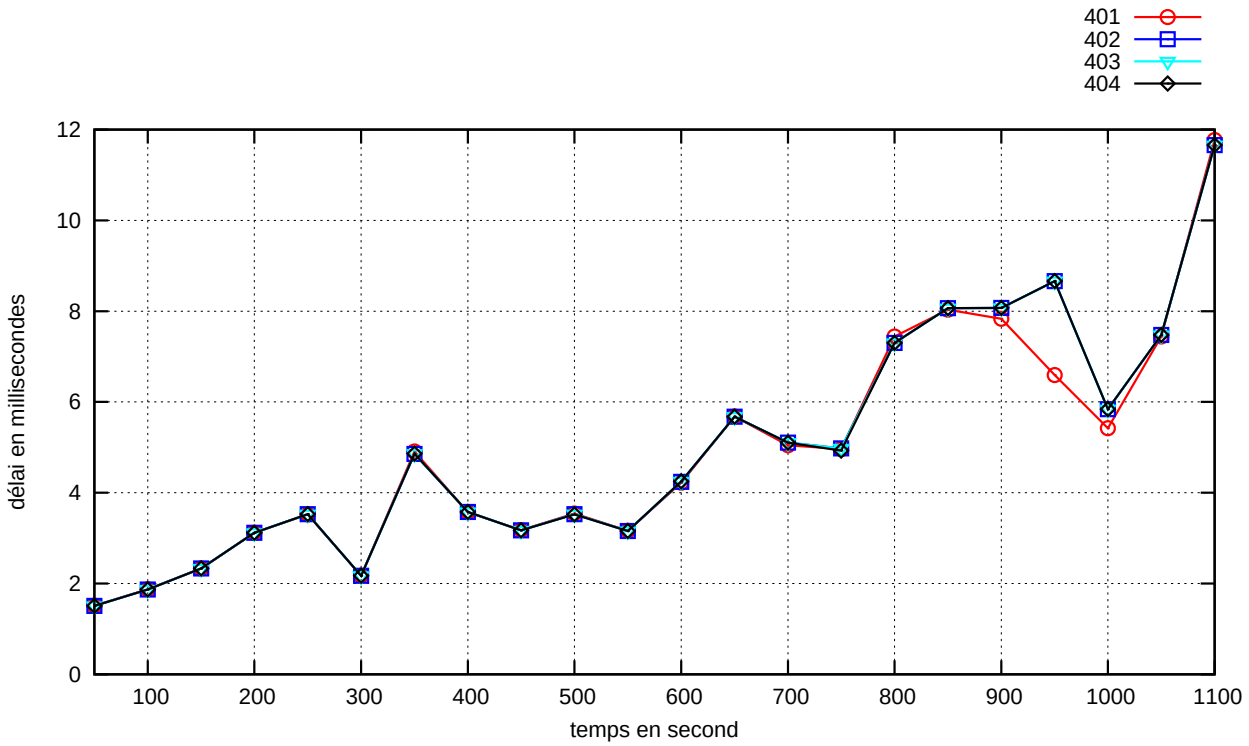


Figure F.2 – Délai moyen du trafic de voix entre le SASN et les UE

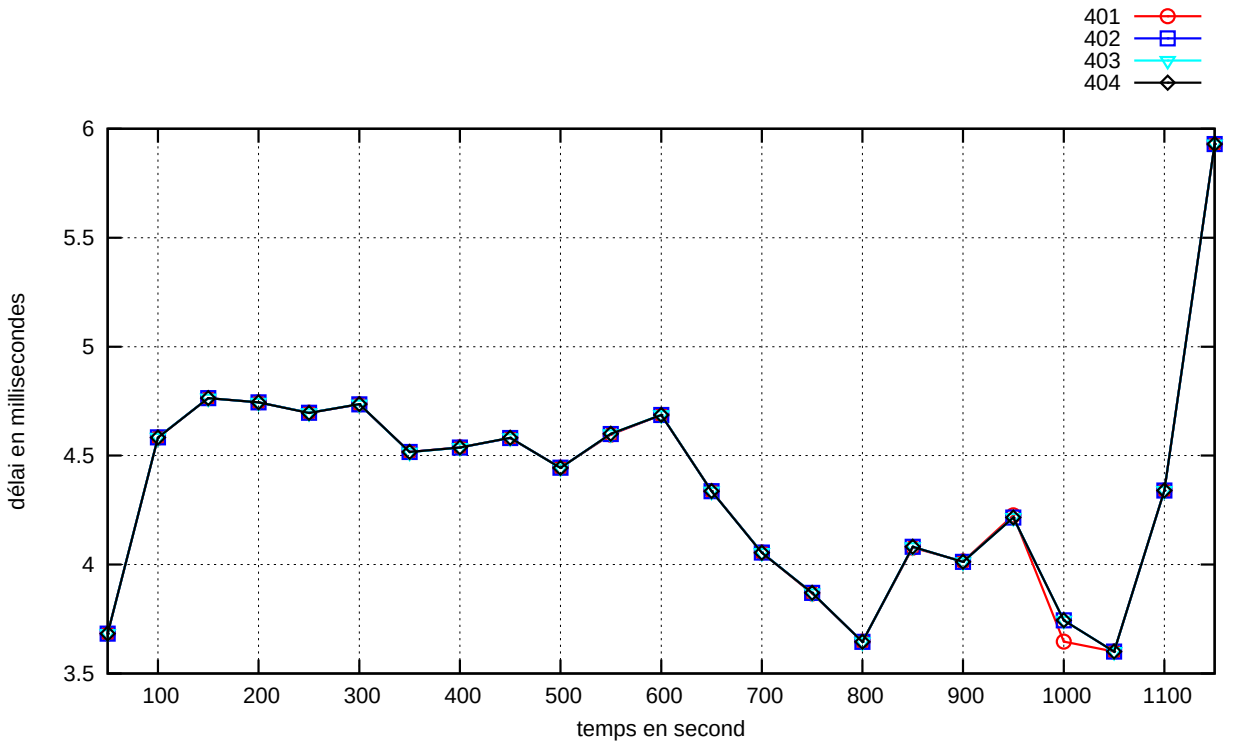


Figure F.3 – Délai moyen du trafic vidéo entre le SASN et les eNodeB

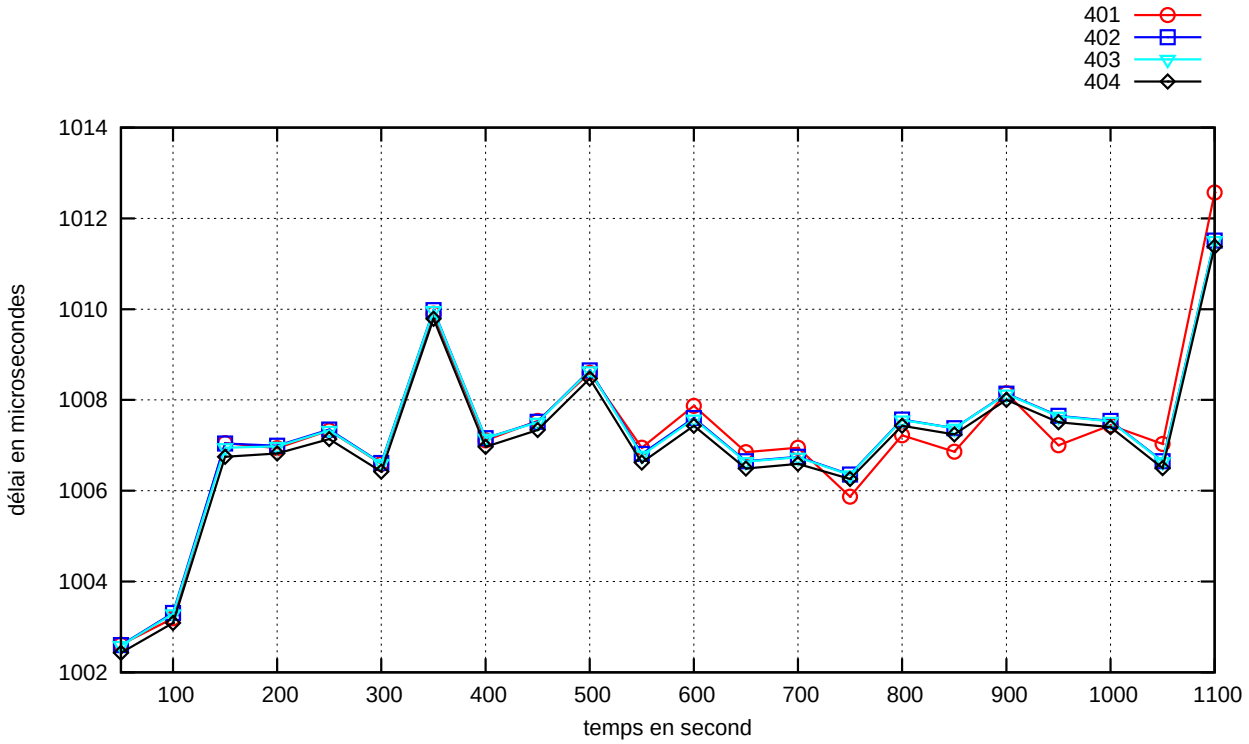


Figure F.4 – Délai moyen du trafic de voix entre le SASN et les eNodeB

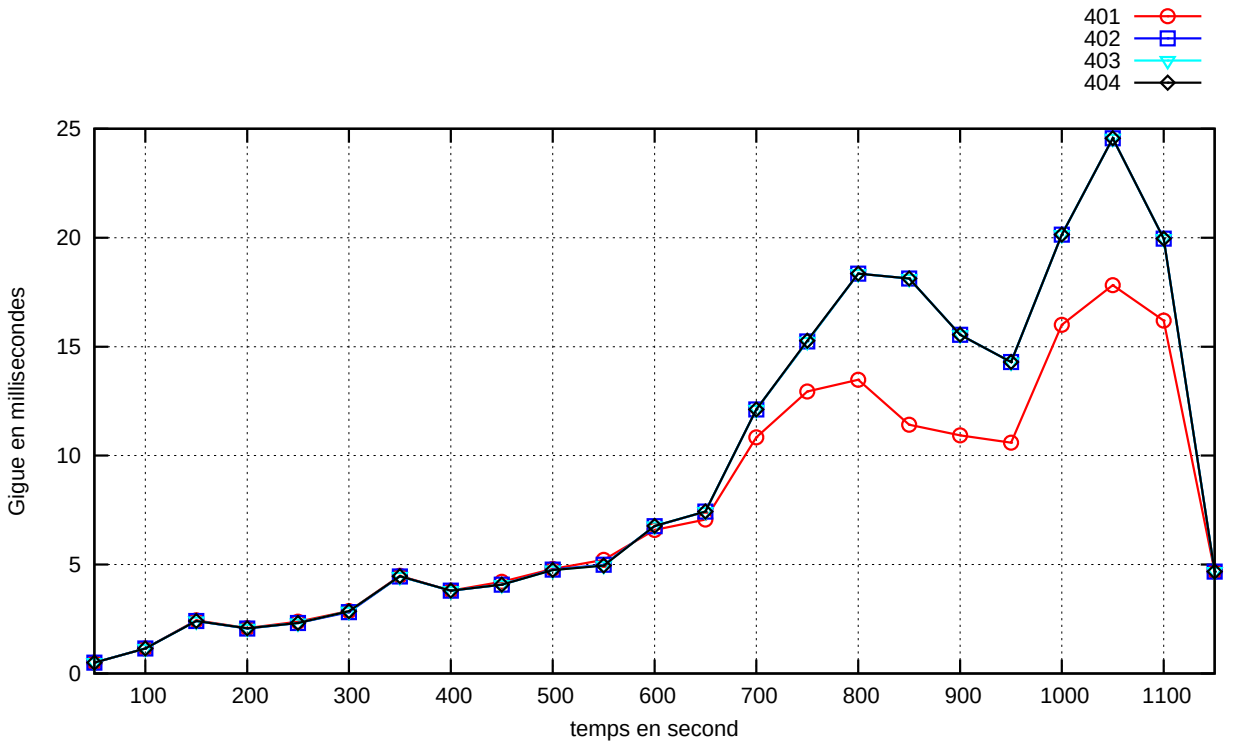


Figure F.5 – Gigue du trafic vidéo entre le SASN et les UE

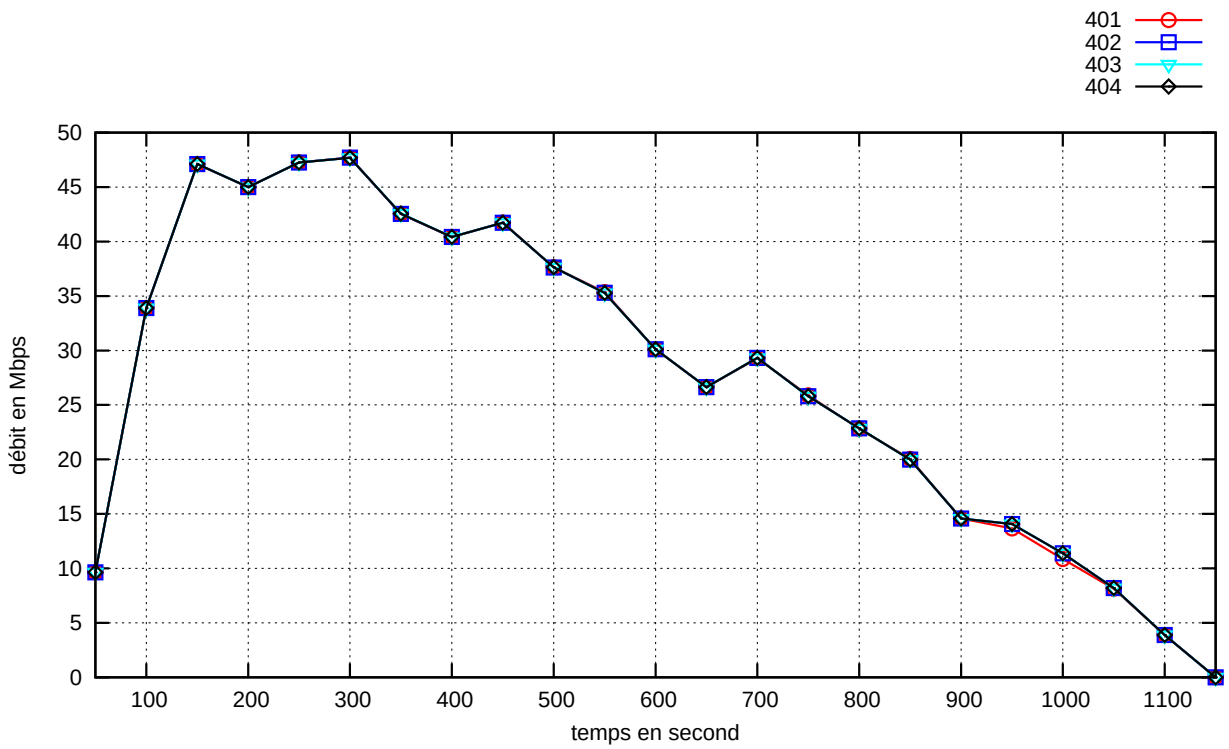


Figure F.6 – Débit global du trafic vidéo

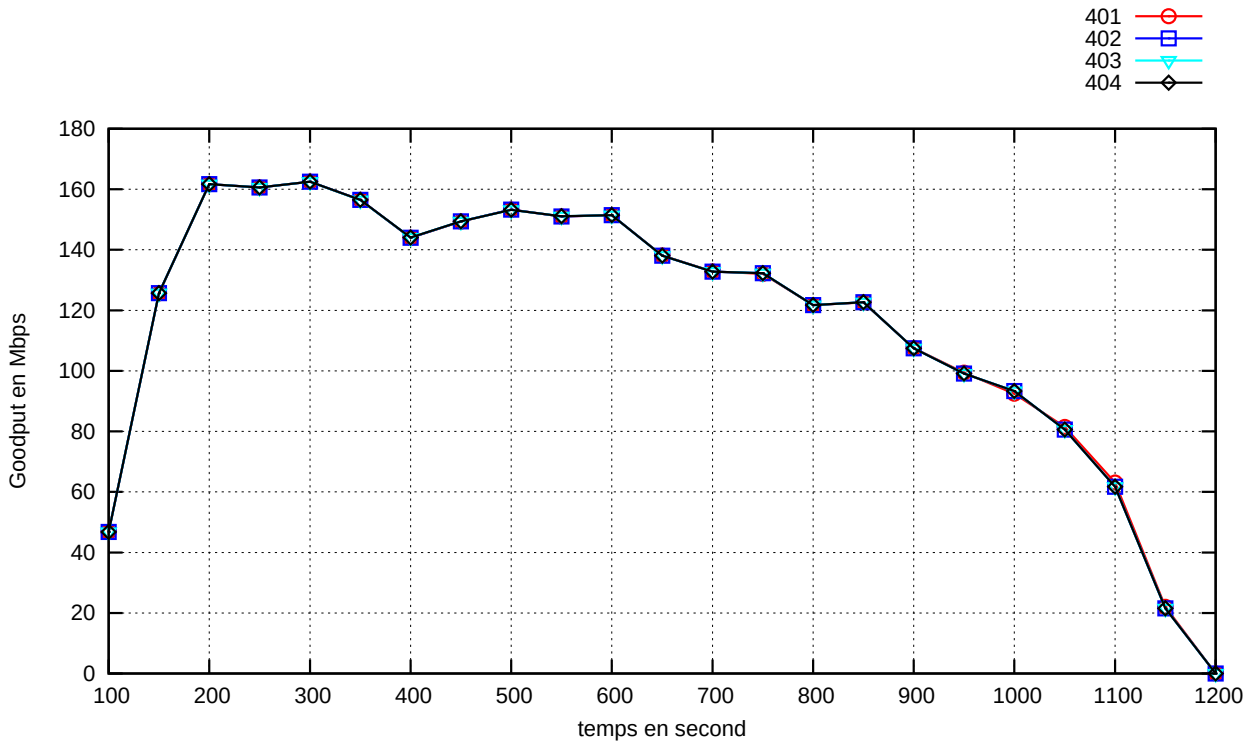


Figure F.7 – Goodput global du trafic FTP

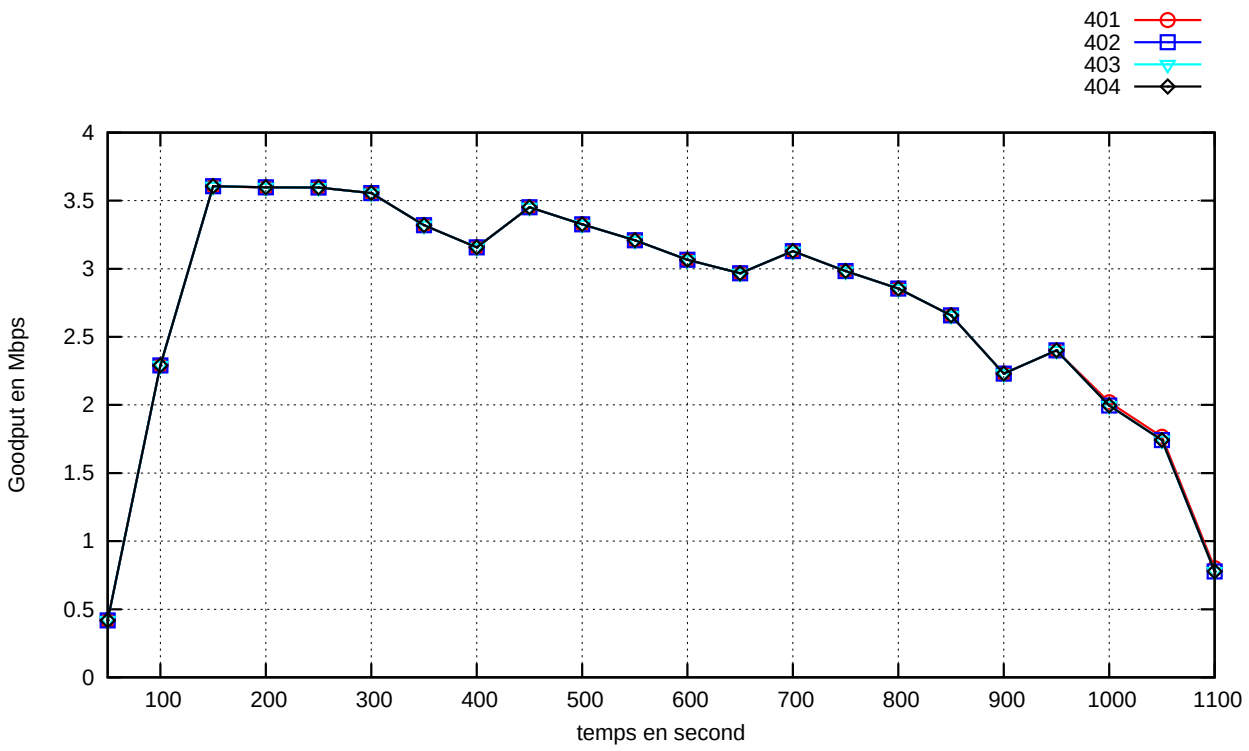


Figure F.8 – Goodput global du trafic web

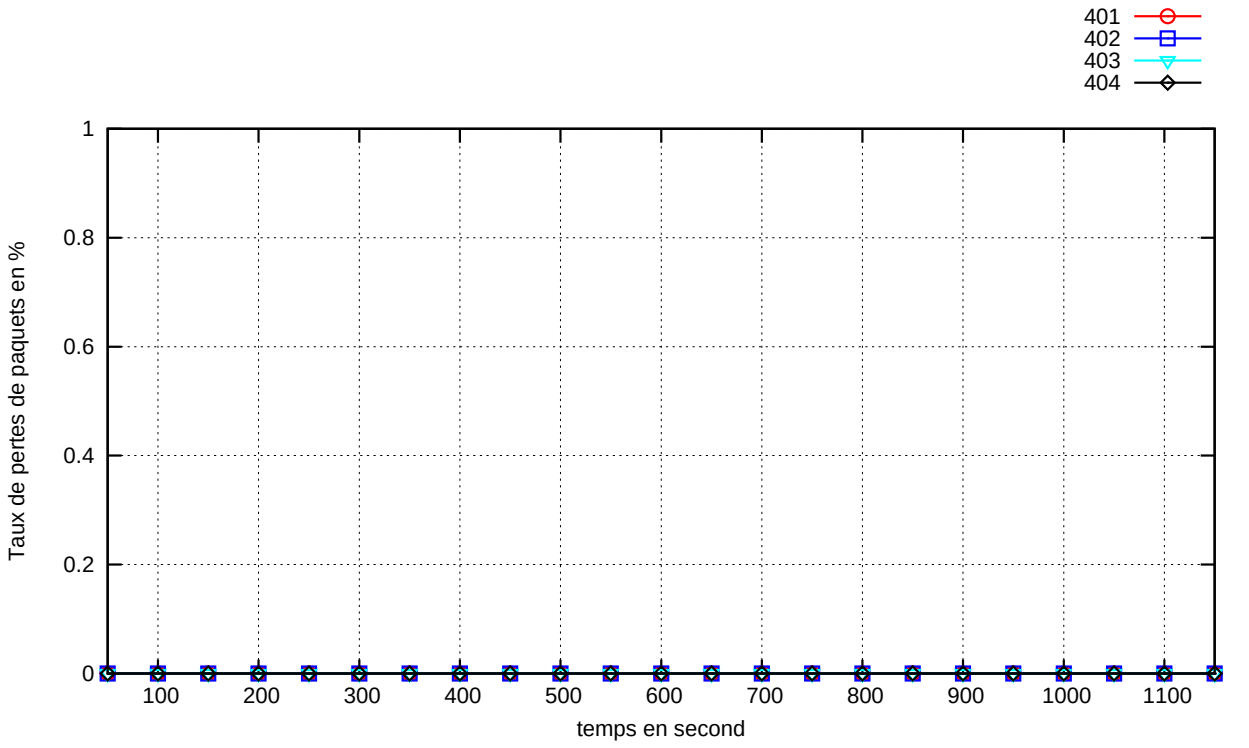


Figure F.9 – Taux de perte de paquets global du trafic vidéo entre le SASN et les UE

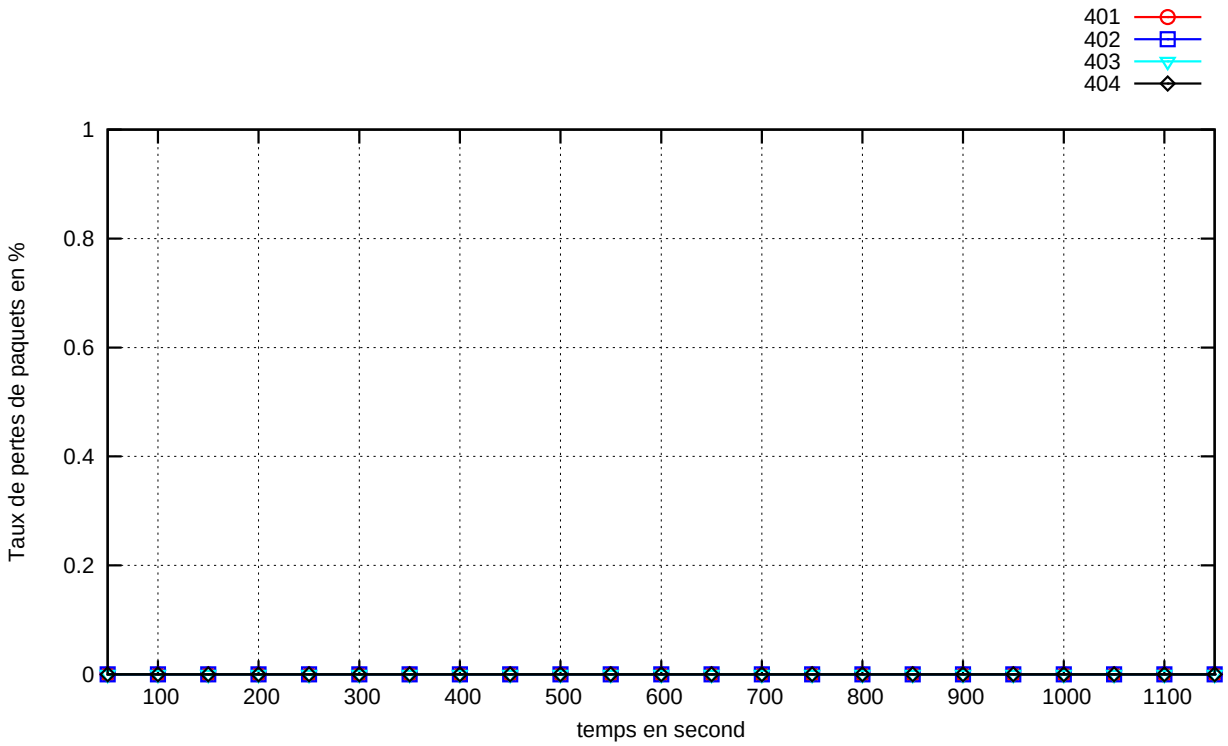


Figure F.10 – Taux de perte de paquets global du trafic web entre le SASN et les UE

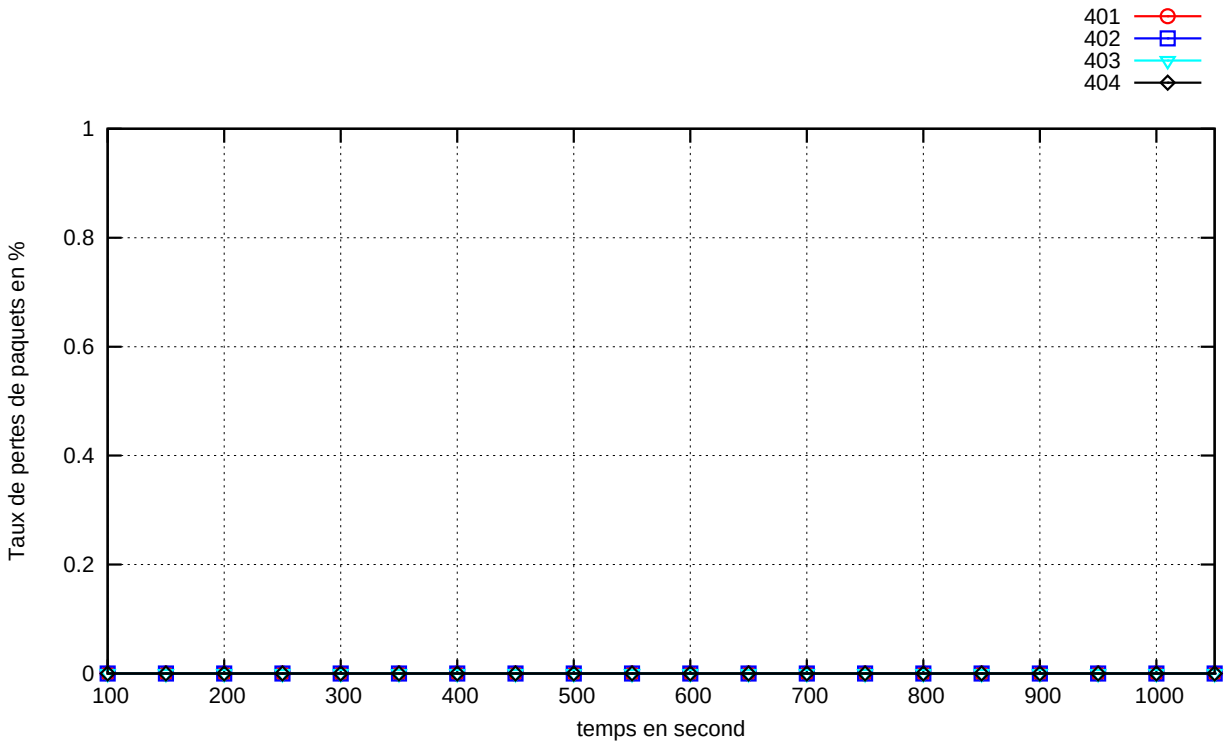


Figure F.11 – Taux de perte de paquets global du trafic FTP entre le SASN et les UE

ANNEXE G

Tailles des messages GTP

Les paragraphes suivants détaillent les tailles des messages du protocole GTP-C (ces tailles incluent les 12 octets de l'en-tête GTP-C) que nous avons utilisé.

Tableau G.1 – Taille des IE du protocole GTP (inspiré de 29.274 [22])

type de IE	taille en octets
AMBR	12
APN	104
APN Restriction	5
Bearer Contexts	4 plus le contenue
Bearer QoS	26
Bearer TFT (voir [10, sec 10.5.6.12])	$4 + 60$ (nombre de SFI)
Cause	10
Charging Characteristics	6
Charging Id	8
Delay Value	5
EPS Bearer ID (EBI)	5
Flow QoS	25
Fully-qualified PDN Connection Set Identifier (FQ-CSID)	22
F-TEID	24
IMSI	12
Indication	6
ME Identity (MEI)	12
MS ISDN Number (MSISDN)	10
Operations and Maintenance Center (OMC) Identity	19
PDN Address Allocation (PAA)	22
PDN Type	5
Procedure Transaction Id (PTI)	5
RAT type	6
Selection Mode	5
Serving Network	7

type de IE	taille en octets
Traffic Aggregate Description (TAD) (voir TFT dans [10])	$4 + 60$ (nombre de SFI)
Trace Reference	10
Trace Type	3
Trigger Id	19
Trace Information	50
User CSG Information (UCI)	12
User Location Information (ULI)	5 plus le contenue

Tableau G.2 – Taille des informations du IE ULI du protocole GTP (inspiré de 29.274 [22])

type d'information	taille en octets
CGI	6
ECGI	6
RAI	6
SAI	6
TAI	4

Create Session Request D'après le tableau G.3 le message GTP-C « Create Session Request » fait 372 octets sur le lien S11 et 394 sur le lien S5/S8.

Create Session Response D'après le tableau G.4 le message « Create Session Response » fait 231 octets sur le lien S11 s'il est envoyé directement du SGW (il ne vient pas du PGW) et que la réponse est négative. Si en revanche il vient du PGW le message fait 253 octets. Si le message, toujours sur l'interface S11 vient du PGW (en passant par le PGW) et que la réponse est positive alors il fait 234 octets. Si le message est sur le lien S5/S8 (il vient forcement du PGW cf. figure A.5) alors il fait 172 octets pour une réponse positive et 191 octets pour réponse négative.

Delete Session Request D'après le tableau G.5 le message « Delete Session Request » fait 33 octets.

Delete Session Response Ce message fait 22 octets suivant le tableau G.6.

Modify Bearer Request D'après le tableau G.7 ce message fait :

- 73 octets s'il est envoyé par le MME sur le lien S11 lors de la connexion d'un UE ;

Tableau G.3 – Create Session Request (interprété de [22])

(a) Create Session Request

IE	type	taille	Interface	conditions
IMSI		12		
MSISDN		10		
ULI incluant ECGI		11		
Serving Network		7		
RAT Type		6		
Indication Flags	Indication	6		
Sender F-TEID for Control Plane	F-TEID	24		
PGW S5/S8 Address for Control Plane	F-TEID	24	S11	
APN		104		
Selection Mode		5		
PDN Type		5		
PAA		22		
Maximum APN Restriction	APN Restriction	5		
APN-AMBR	AMBR	12		
Bearer Contexts to be created	Bearer Context	4 + contenue		
Trace Information		50		
MME-FQ-CSID	FQ-CSID	22		
SGW-FQ-CSID	FQ-CSID	22	S5/S8	
Charging Characteristics		6		

(b) Bearer Contexts to be created

IE	type	taille	Interface	conditions
EBI		5		
S5/S8-U SGW F-TEID	F-TEID	24	S5/S8	
Bearer Level QoS	Bearer QoS	26		

Tableau G.4 – Create Session Response (interprété de [22])

(a) Create Session Response

IE	type	taille	Interface	conditions
Cause		10		
Sender F-TEID for Control Plane	F-TEID	24	S11	
PGW S5/S8 F-TEID	F-TEID	24		
PAA		22		
APN Restriction		5		
Bearer Contexts created	Bearer Context	4 + contenue		
Bearer Contexts marked for removal	Bearer Context	4 + contenue		si c'est une réponse négative
PGW-FQ-CSID	FQ-CSID	22	S5/S8	
			S11	si la reponse vient du S5/S8
SGW-FQ-CSID	FQ-CSID	22	S11	

(b) Inclue dans le Bearer Context created

IE	type	taille	Interface	conditions
EBI		5		
Cause		10		
S1-U SGW F-TEID	F-TEID	24	S11	
S5/S8-U PGW F-TEID	F-TEID	24		
Bearer Level QoS	Bearer QoS	26		
Charging Id		8	S5/S8	

(c) Inclue dans le Bearer Contexts marked for removal

IE	type	taille	Interface	conditions
EBI		5		
Cause		10		

Tableau G.5 – Delete Session Request (interprété de [22])

IE	type	taille	Interface	conditions
Cause		10		
Linked EPS Bearer ID (LBI)	EBI	5		
Indication Flags	Indication	6		

Tableau G.6 – Delete Session Response (interprété de [22])

IE	type	taille	Interface	conditions
Cause		10		

- $40 + 33 \times$ (nombre de *bearers* conservés) $+ 9 \times$ (nombre de *bearers* supprimés) octets s'il est envoyé par le MME sur le lien S11 lors d'un *handover* ;
- $62 + 33 \times$ (nombre de *bearers* conservés) $+ 9 \times$ (nombre de *bearers* supprimés) octets s'il est envoyé par le SGW sur le lien S5/S8 lors d'un *handover*.

Tableau G.7 – Modify Bearer Request (interprété de [22])

(a) Modify Bearer Request

IE	type	taille	Interface	conditions
Indication Flags	Indication	6		
Delay Downlink Packet Notification Request	Delay Value	5		requête de service provenant du UE
Bearer Contexts to be modified	Bearer Context	4 + contenue		tout excepté sur S5/S8 lors d'une requête de service provenant du UE
Bearer Contexts to be removed	Bearer Context	4 + contenue	S11	*handover*, à multiplié par le nombre de *bearer* détruit
MME-FQ-CSID	FQ-CSID	22	S11 et S5/S8	
SGW-FQ-CSID	FQ-CSID	22	S5/S8	

(b) Inclue dans Bearer Contexts to be modified

IE	type	taille	Interface	conditions
EBI		5		
S1 eNodeB F-TEID	F-TEID	24		UE QoS *aware* ou durant un *handover*

(c) Inclue dans Bearer Contexts to be removed

IE	type	taille	Interface	conditions
EBI		5		

Modify Bearer Response Il fait 44 octets lors d'une procédure de connexion et $44 + 43 \times$ (nombre de *bearers* supprimés) lors d'un *handover*. Ces valeurs sont tirée du tableau G.8.

Update Bearer Request D'après le tableau G.9, pour les UE QoS *aware* ce message fait 86+60×(nombre de TFT) octets s'il sort du SGW sur le lien S11 et 64+60×(nombre de TFT) octets s'il sort du PGW sur le lien S5/S8. Dans tout les cas il fait cinq octets de moins pour

Tableau G.8 – Modify Bearer Response (interprété de [22])

(a) Modify Bearer Request

IE	type	taille	Interface	conditions
Cause		10		
Bearer Contexts marked for removal	Bearer Context	4 + contenue		handover
SGW-FQ-CSID	FQ-CSID	22		

(b) Inclue dans Bearer Context marked for removal

IE	type	taille	Interface	conditions
EBI		5		
Cause		10		
S1 SGW F-TEID	F-TEID	24		

les UE QoS *unaware*.

Update Bearer Response D'après le tableau G.10 ce message fait 63 octets s'il sort du MME sur le lien S11 et 85 octets s'il sort du SGW sur le lien S5/S8.

Create Bearer Request D'après le tableau G.11 pour le UE QoS *aware* ce message fait $153 + 60 \times$ (nombre de TFT) octets s'il sort du SGW sur le lien S11 et $115 + 60 \times$ (nombre de TFT) octets s'il sort du PGW sur le lien S5/S8. Dans tout les cas il fait cinq octets de moins pour les UE QoS *unaware*.

Create Bearer Response Il fait 133 octets sur le lien S11. Sur le lien S5/S8 ce message fait 133 octets si la requête « Create Bearer Request » correspondante s'est rendue jusqu'au eNodeB et 89 octets sinon. Ces valeurs sont tiré du tableau G.12.

Bearer Resource Command Tel que représenté au tableau G.13, ce message fait une taille de $57 + 60 \times$ (nombre d'IP) octets si c'est une demande d'ajout de services dans un *bearer* et 26 octets de moins si c'est une demande de suppression de services.

Delete Bearer Request D'après le tableau G.14 pour les UE QoS *aware* ce message fait 66 octets sur le lien S11 et 44 sur le lien S5/S8. Il fait cinq octets de moins pour les UE QoS *unaware*.

Delete Bearer Response D'après le tableau G.15 ce message fait 63 octets sur le lien S11 et 88 sur le lien S5/S8.

Tableau G.9 – Update Bearer Request (interprété de [22])

(a) Update Bearer Request

IE	type	taille	Interface	conditions
Bearer Contexts	Bearer Context	4 + contenue		
PTI		5		si le UE est QoS aware
APN-AMBR	AMBR	12		
PGW-FQ-CSID	FQ-CSID	22		si la requete est passé par le PGW
SGW-FQ-CSID	FQ-CSID	22	S11	si la requete est passé par le SGW

(b) Inclue dans le Bearer Context created

IE	type	taille	Interface	conditions
EBI		5		
TFT	Bearer TFT	4 + 60 (nombre de SFI)		si c'est une modification de TFT
Bearer Level QoS	Bearer QoS	26		si c'est une modification de QoS

Tableau G.10 – Update Bearer Response (interprété de [22])

(a) Update Bearer Response

IE	type	taille	Interface	conditions
Cause		10		
Bearer Context		4 + contenue		
MME-FQ-CSID	FQ-CSID	22	S11	
			S5/S8	si la reponse vient du S11
SGW-FQ-CSID	FQ-CSID	22	S5/S8	

(b) Inclue dans le Bearer Context

IE	type	taille	Interface	conditions
EBI		5		
Cause		10		

Tableau G.11 – Create Bearer Request (interprété de [22])

(a) Create Bearer Request

IE	type	taille	Interface	conditions
PTI		5		le UE est QoS aware
LBI	EBI	5		
Bearer Context		4 + contenue		
PGW-FQ-CSID	FQ-CSID	22		
SGW-FQ-CSID	FQ-CSID	22	S11	

(b) Inclue dans le Bearer Context

IE	type	taille	Interface	conditions
EBI		5		
TFT	Bearer TFT	4 + 60 (nombre de SFI)		
S1-U SGW F-TEID	F-TEID	24	S11	
S5/8-U PGW F-TEID	F-TEID	24		
Bearer Level QoS	Bearer QoS	26		
Charging Id		8	S5/S8	

Tableau G.12 – Create Bearer Response (interprété de [22])

(a) Create Bearer Response

IE	type	taille	Interface	conditions
Cause		10		
Bearer Context		4 + contenue		
MME-FQ-CSID	FQ-CSID	22	S11	
			S5/S8	si la reponse vient du S11
SGW-FQ-CSID	FQ-CSID	22	S11	
			S5/S8	si la reponse vient du S11

(b) Inclue dans le Bearer Context

IE	type	taille	Interface	conditions
EBI		5		
Cause		10		
S1-U eNodeB F-TEID	F-TEID	24	S11	
S1-U SGW F-TEID	F-TEID	24	S11	
S5/8-U SGW F-TEID	F-TEID	24	S5/S8	
S5/8-U PGW F-TEID	F-TEID	24	S5/S8	

Tableau G.13 – Bearer Resource Command (interprété de [22])

IE	type	taille	Interface	conditions
LBI	EBI	5		
PTI		5		
Flow QoS		26		sauf pour resource release
TAD		4 + 60 (nombre de SFI)		
EBI		5		

Tableau G.14 – Delete Bearer Request (interprété de [22])

IE	type	taille	Interface	conditions
EBI		5		
PTI		5		si le UE est QoS aware
PGW-FQ-CSID	FQ-CSID	22		
SGW-FQ-CSID	FQ-CSID	22	S11	
Cause		10		si la requete est du à un handover

Tableau G.15 – Delete Bearer Response (interprété de [22])

(a) Delete Bearer Response

IE	type	taille	Interface	conditions
Cause		10		
Bearer Context		4 + contenue		
MME-FQ-CSID	FQ-CSID	22	S11	
			S5/S8	
SGW-FQ-CSID	FQ-CSID	22	S5/S8	

(b) Inclue dans le Bearer Context

IE	type	taille	Interface	conditions
EBI		5		
Cause		10		

Create Indirect Data Forwarding Tunnel Request D'après le tableau G.16 ce message fait 69 octets.

Tableau G.16 – Create Indirect Data Forwarding Tunnel Request (interprété de [22])

(a) Create Indirect Data Forwarding Tunnel Request

IE	type	taille	Interface	conditions
Bearer Context		4 + contenue		

(b) Inclue dans le Bearer Context

IE	type	taille	Interface	conditions
EBI		5		
eNodeB F-TEID for DL data forwarding	F-TEID	24		
SGW F-TEID for DL data forwarding	F-TEID	24		

Create Indirect Data Forwarding Tunnel Response D'après le tableau G.17 ce message fait 89 octets.

Tableau G.17 – Create Indirect Data Forwarding Tunnel Response (interprété de [22])

(a) Create Indirect Data Forwarding Tunnel Response

IE	type	taille	Interface	conditions
Cause		10		
Bearer Context		4 + contenue		

(b) Inclue dans le Bearer Context

IE	type	taille	Interface	conditions
EBI		5		
Cause		10		
S1-U SGW F-TEID for DL data forwarding	F-TEID	24		
SGW F-TEID for DL data forwarding	F-TEID	24		

Delete Bearer Command D'après le tableau G.18 ce message fait 21 octets.

Delete Indirect Data Forwarding Tunnel Request Ce message ne contient aucun IE sa taille correspond donc à celle de l'en-tête du protocole GTP-C soit 12 octets.

Delete Indirect Data Forwarding Tunnel Response D'après le tableau G.19 ce message fait 22 octets.

Tableau G.18 – Delete Bearer Command (interprété de [22])

(a) Delete Bearer Command

IE	type	taille	Interface	conditions
Bearer Context		4 + contenue		

(b) Inclue dans le Bearer Context

IE	type	taille	Interface	conditions
EBI		5		

Tableau G.19 – Delete Indirect Data Forwarding Tunnel Response (interprété de [22])

IE	type	taille	Interface	conditions
Cause		10		

ANNEXE H

Tailles des messages Diameter

Dans cette annexe nous détaillons la taille des messages Diameter que nous avons utilisés dans nos simulateurs. Ces tailles ont été calculées grâce au tableau H.1 généré grâce à Rigney *et al.* [92], Calhoun *et al.* [48], Loughney [74], Calhoun *et al.* [49] et Hakala *et al.* [66] et des documents 23.003 [2], 23.011 [3], 24.008 [10], 29.002 [15], 29.060 [16], 29.061 [17], 29.212 [18], 29.214 [19], 29.229 [20], 29.272 [21], 29.274 [22], 29.329 [25], 32.298 [26], 32.299 [27], 32.422 [28].

Tableau H.1 – Taille des AVP du protocole Diameter [2, 3, 10, 15, 16, 17, 18, 19, 20, 21, 22, 25, 26, 27, 28, 92, 48, 74, 49, 66]

IE	taille en octets
3GPP-Charging-Characteristics	16
3GPP-MS-TimeZone	16
3GPP-User-Location-Info	24
3GPP-RAT-Type	16
3GPP-SGSN-MCC-MNC	20
3GPP-SGSN-Address	24
3GPP-SGSN-IPv6-Address	36
Access-Restriction-Data	16
Access-Network-Charging-Address	32
Access-Network-Charging-Identifier-Gx	84
Access-Network-Charging-Identifier-Value	24
Acct-Multi-Session-Id	68
AF-Charging-Identifier	24
AF-Signalling-Protocol	16
Age-Of-Location-Information	16
All-APN-Configurations-Included-Indicator	16
Allocation-Retention-Priority	60
AMBR	44
AN-GW-Address	32
APN-Aggregate-Max-Bitrate-DL	16

IE	taille en octets
APN-Aggregate-Max-Bitrate-UL	16
APN-Configuration	500
APN-Configuration-Profile	544
APN (depuit l'IE GTP)	112
APN-OI-Replacement	32
Auth-Application-Id	16
Authentication-Info	412
AUTN	28
AUTS	28
Bearer-Control-Mode	16
Bearer-Identifier	24
Bearer-Operation	16
Bearer-Usage	16
Call-Barring-Infor-List	36
Called-Station-ID	32
CC-Correlation-Id	24
CC-Input-Octets	20
CC-Money	76
CC-Output-Octets	20
CC-Request-Number	16
CC-Request-Type	16
CC-Service-Specific-Units	20
CC-Session-Failover	16
CC-Sub-Session-Id	20
CC-Time	16
CC-Total-Octets	20
CC-Unit-Type	16
CGI	24
Charging-Information	300
Charging-Rule-Base-Name	24
Charging-Rule-Definition	828
Charging-Rule-Install	960
Charging-Rule-Name	24
Charging-Rule-Remove	60

IE	taille en octets
Charging-Rule-Report	340
Check-Balance-Result	16
Confidentiality-Key (CK)	28
CoA-Information	112
CoA-IP-Address	32
Complete-Data-List-Included-Indicator	16
Context-Identifier	16
Cost-Information	100
Cost-Unit	24
Credit-Control-Failure-Handling	16
CSG-Id	16
CSG-Information-Reporting	16
CSG-Subscription-Data	44
Currency-Code	16
Current-Location-Retrieved	16
Default-EPS-Bearer-QoS	88
Destination-Host	36
Destination-Realm	36
Direct-Debiting-Failure-Handling	16
ECGI	20
EPS-Location-Information	356
EPS-Subscribed-QoS Profile	88
EUTRAN-Cell-Global-Identity	20
EUTRAN-Vector	140
Event-Report-Indication	536
Event-Timestamp	16
Event-Trigger	16
Experimental-Result	44
Experimental-Result-Code	16
Expiration-Date	16
Exponent	16
Feature-List	16
Feature-List-ID	16
Filter-Id	20

IE	taille en octets
Final-Unit-Action	16
Final-Unit-Indication	224
Flow-Description	112
Flow-Direction	16
Flow-Information	340
Flow-Label	72
Flow-Number	16
Flows	60
Flow-Status	16
Framed-IP-Address	24
Framed-IPv6-Prefix	36
Geodetic-Information	24
Geographical-Information	20
GERAN-Vector	92
GMLC-Number	24
GMLC-Restriction	16
GPRS-Subscription-Data	232
Granted-Service-Unit	200
G-S-U-Pool-Identifier	16
G-S-U-Pool-Reference	92
Guaranteed-Bitrate-DL	16
Guaranteed-Bitrate-UL	16
HPLMN-ODB	16
ICS-Indicator	16
Integrity-Key (IK)	28
Immediate Response Preferred	16
IP address (depuit GTP)	28
Item-Number	16
IP-CAN-Type	16
KASME	28
Kc	20
LCS-Info	120
LCS-PrivacyException	84
Location-Area-Identity	20

IE	taille en octets
Max-Requested-Bandwidth-DL	16
Max-Requested-Bandwidth-UL	16
Media-Component-Number	16
Metering-Method	16
MIP6-Agent-Info	96
MIP6-Home-Link-Prefix	28
MIP-Home-Agent-Address	28
MIP-Home-Agent-Host	28
MME-Location-Information	164
MN Network Access Identifier (NAI)	64
Monitoring-Key	16
MSISDN	20
Multiple-Services-Credit-Control (for Request)	428
Multiple-Services-Credit-Control (for Answer)	604
Multiple-Services-Indicator	16
Network-Access-Mode	16
Network-Request-Support	16
Notification-To-UE-User	16
Number Of Requested Vectors	16
Offline	16
OMC-Id	24
Online	16
Operator-Determined-Barring	16
Origin-Host	36
Origin-Realm	36
Origin-State-Id	16
Packet-Filter-Content	112
Packet-Filter-Identifier	24
Packet-Filter-Information	324
Packet-Filter-Operation	16
Packet-Filter-Usage	16
PCC-Rule-Status	16
PDN-Connection-ID	24
PDN-GW-Allocation-Type	16

IE	taille en octets
PDN-Type	16
PDP-Address	28
PDP-Context	204
PDP-Type	16
PLMN-Client	16
Precedence	16
Pre-emption-Capability	16
Pre-emption-Vulnerability	16
Primary-Charging-Collection-Function-Name	72
Primary-Event-Charging-Function-Name	72
Priority-Level	16
Proxy-Host	36
Proxy-Info	72
Proxy-State	24
QCI	16
QoS-Information	208
QoS-Negotiation	16
QoS-Rule-Base-Name	24
QoS-Rule-Name	24
QoS-Rule-Report	78
QoS-Subscribed	24
QoS-Upgrade	16
RAT-Frequency-Selection-Priority-ID	16
Rating-Group	16
RAI	24
RAND	28
RAT-Type	16
Re-Auth-Request-Type	16
Redirect-Address-Type	16
Redirect-Host	36
Redirect-Host-Usage	16
Redirect-Max-Cache-Time	16
Redirect-Server	64
Redirect-Server-Address	36

IE	taille en octets
Regional-Subscription-Zone-Code	16
Reporting-Level	16
Requested-EUTRAN-Authentication-Info	88
Requested-Action	16
Requested-Service-Unit	184
Resource-Allocation-Notification	16
Restriction-Filter-Rule	112
Result-Code	16
Re-synchronization-Info	44
Revalidation-Time	16
Roaming-Restricted-Due-To-Unsupported-Feature	16
Route-Record	36
Routing-Area-Identity	24
Rule-Activation-Time	16
Rule-Deactivation-Time	16
Rule-Failure-Code	16
Secondary-Charging-Collection-Function-Name	72
Secondary-Event-Charging-Function-Name	72
Security-Parameter-Index	72
Served-Party-IP-Address	28
Service-Area-Identity	24
Service-Context-Id	48
Service-Identifier	16
Service-Parameter-Info	52
Service-Parameter-Type	16
Service-Parameter-Value	24
Service-Selection	76
Service-Type	60
Service-Type-Identity	16
Session-Id	68
Session-Release-Cause	16
SGSN-Location-Information	180
SRES	16
SS-Code	24

IE	taille en octets
SS-Status	16
STN-SR	28
Subscriber-Status	16
Subscription-Data	1472
Subscription-Id	52
Subscription-Id-Data	24
Subscription-Id-Type	16
Supported-Features	60
TAI	56
Tariff-Change-Usage	16
Tariff-Time-Change	16
Teleservice-List	36
Termination-Cause	16
TFT-Filter	112
TFT-Packet-Filter-Information	300
ToS-Traffic-Class	16
Trace-Collection-Entity	28
Trace-Data	164
Trace-Depth	16
Trace-Event-List	24
Trace-Interface-List	24
Trace-NE-Type-List	16
Trace-Reference	20
TS-Code	24
Tunnel-Header-Filter	112
Tunnel-Header-Length	16
Tunnel-Information	68
ULA-Flags	16
ULR-Flags	16
Unit-Value	48
Usage-Monitoring-Information	476
Usage-Monitoring-Level	16
Usage-Monitoring-Report	16
Usage-Monitoring-Support	16

IE	taille en octets
Used-Service-Unit	200
User-Equipment-Info	52
User-Equipment-Info-Type	16
User-Equipment-Info-Value	24
User-Name	28
UTRAN-Vector	168
Validity-Time	16
Value-Digits	20
Vendor-Id	16
Visited PLMN Id	16
Visited-Network-Identifier	44
VPLMN-Dynamic-Address-Allowed	16
XRES	28

Update Location Request fait 96 octets d'après le tableau H.2.

Update Location Answer fait 1524 octets d'après le tableau H.3.

Authentification Information Request fait 152 octets d'après le tableau H.4.

Authentification Information Answer fait 448 octets d'après le tableau H.5.

Insert Subscriber Data Request fait 1520 octets d'après le tableau H.6.

Insert Subscriber Data Answer fait 392 octets d'après le tableau H.7.

Indication of IP-CAN Session Establishment c'est un message de type « Credit Control Request » pouvant posséder les AVP représentés au tableau H.8. En prenant les AVP nécessaires pour la connexion d'un UE on obtient le tableau H.10 ce qui fait un total de 728 octets.

Acknowledge IP-CAN Session Establishment ce message est de type « Credit Control Answer » pouvant posséder les AVP représentés au tableau H.9. Selon le tableau H.12 ce message fait 1472 octets lorsqu'il est utilisé pour la création d'un nouveau *bearer* (connexion d'un UE).

Indication of IP-CAN Session Modification ce message est de type « Credit Control Request » (voir tableau H.8). Ce message ressemble au « Indication of IP-CAN Session Establishment » mais s'applique aux modifications de session soit une création/ modification/suppression de *bearer*. D'après le tableau H.11 il fait 1264 octets.

Acknowledge of IP-CAN Session Modification ce message est de type « Credit Control Answer » (voir tableau H.9). Il est en tout point identique au message « Acknow-

ledge IP-CAN Session Establishment » (voir tableau H.12) mais ne possède pas l'AVP `Default-EPS-Bearer-QoS` de 88 octets. Il fait donc 1384 octets lors de la création d'un nouveau *bearer* et 2284 octets lors d'une destruction.

Indication of IP-CAN Session Termination est un « Credit Control Request » (voir tableau H.8) sans AVP optionnel, soit 244 octets.

Acknowledge of IP-CAN Session Termination est un « Credit Control Answer » (voir tableau H.9) avec pour seul AVP optionnel un `Charging-Rule-Remove`. Ce message fait donc 328 octets.

Policy and Charging Rules Provision est un message de type « Re-Authentication Request » pouvant posséder les AVP représentés au tableau H.13. En prenant les AVP nécessaires on obtient le tableau H.15 qui nous indique que ce message fait 1432 octets.

Acknowledge Policy and Charging Rules Provision message de type « Re-Authentication Answer » (voir tableau H.14). D'après le tableau H.16 il fait 440 octets.

Tableau H.2 – Update Location Request (interprété de [21])

IE	AVP	taille	Interface	conditions
IMSI	User-Name	28		
Update-Location-Request (ULR) Flags		16		
Visited-PLMN-Id		16		
RAT Type		16		

Tableau H.3 – Update Location Answer (interprété de [21])

IE	AVP	taille	Interface	conditions
Result	Result-Code	16		
ULA-Flags		16		
Subscription-Data		1472		

Tableau H.4 – Authentification Information Request (interprété de [21])

IE	AVP	taille	Interface	conditions
IMSI	User-Name	28		
Requested EUTRAN Authentication Info		88		
Visited PLMN ID		16		

Tableau H.5 – Authentification Information Answer (interprété de [21])

IE	AVP	taille	Interface	conditions
Result	Result-Code	16		
Authentication-Info		412		

Tableau H.6 – Insert Subscriber Data Request (interprété de [21])

IE	AVP	taille	Interface	conditions
IMSI	User-Name	28		
Subscription Data		1472		

Tableau H.8 – Credit-Control-Request (interprété de [66])

type de IE	taille en octets
Session-Id	68
Auth-Application-Id	16
Origin-Host	36
Origin-Realm	36
Destination-Realm	36
CC-Request-Type	16
CC-Request-Number	16
fin des AVP obligatoires	
Destination-Host	36
Origin-State-Id	16
Subscription-Id	52
Supported-Features	60
Network-Request-Support	16
Packet-Filter-Information	324
Packet-Filter-Operation	16
Bearer-Identifier	24
Bearer-Operation	16

Tableau H.7 – Insert Subscriber Data Answer (interprété de [21])

IE	AVP	taille	Interface	conditions
Result	Result-Code	16		
IMS Voice over PS Sessions Supported		16		
EPS-Location-Information		356		

type de IE	taille en octets
Framed-IP-Address	24
Framed-IPv6-Prefix	36
IP-CAN-Type	16
3GPP-RAT-Type	16
RAT-Type	16
Termination-Cause	16
User-Equipment-Info	52
QoS-Information	208
QoS-Negotiation	16
QoS-Upgrade	16
Default-EPS-Bearer-QoS	88
AN-GW-Address	32
3GPP-SGSN-MCC-MNC	20
3GPP-SGSN-IPv6-Address	36
RAI	24
3GPP-User-Location-Info	24
3GPP-MS-TimeZone	16
Called-Station-ID	32
PDN-Connection-ID	24
Bearer-Usage	16
Online	16
Offline	16
TFT-Packet-Filter-Information	300
Charging-Rule-Report	340
Event-Trigger	16
Event-Report-Indication	536
Access-Network-Charging-Address	32
Access-Network-Charging-Identifier-Gx	84
CoA-Information	112
Usage-Monitoring-Information	476
Proxy-Info	72
Route-Record	36

Les AVP suivant ne sont utilisés que dans les simulateurs basés sur les architectures 402, 403 et 404.

Tableau H.9 – Credit-Control-Answer (interprété de [18])

IE / AVP	taille
Session-Id	68
Auth-Application-Id	16
Origin-Host	36
Origin-Realm	36
Result-Code	16
Experimental-Result	44
CC-Request-Type	16
CC-Request-Number	16
fin des AVP obligatoires	
Supported-Features	60
Bearer-Control-Mode	16
Event-Trigger	16
Origin-State-Id	16
Redirect-Host	36
Redirect-Host-Usage	16
Redirect-Max-Cache-Time	16
Charging-Rule-Remove	60
Charging-Rule-Install	960
Charging-Information	300
Online	16
Offline	16
QoS-Information	208
Revalidation-Time	16
Default-EPS-Bearer-QoS	88
Bearer-Usage	16
3GPP-User-Location-Info	24
Usage-Monitoring-Information	476
CSG-Information-Reporting	16
Route-Record	36

Tableau H.10 – Indication of IP-CAN Session Establishment : connexion (interprété de [18])

IE / AVP	taille
Session-Id	68
Auth-Application-Id	16
Origin-Host	36
Origin-Realm	36
Destination-Realm	36
CC-Request-Type	16
CC-Request-Number	16
fin des AVP obligatoires	
Subscription-Id	52
Network-Request-Support	16
Framed-IP-Address	24
Framed-IPv6-Prefix	36
IP-CAN-Type	16
RAT-Type	16
Default-EPS-Bearer-QoS	88
AN-GW-Address	32
3GPP-MS-TimeZone	16
Called-Station-ID	32
PDN-Connection-ID	24
Bearer-Usage	16
Access-Network-Charging-Address	32
Access-Network-Charging-Identifier-Gx	84
APN-Aggregate-Max-Bitrate-DL	16
APN-Aggregate-Max-Bitrate-UL	16

Tableau H.11 – Indication of IP-CAN Session Establishment/Modification : création, modification ou suppression de *bearer* (interprété de [18])

IE / AVP	taille
Session-Id	68
Auth-Application-Id	16
Origin-Host	36
Origin-Realm	36
Destination-Realm	36
CC-Request-Type	16
CC-Request-Number	16
fin des AVP obligatoires	
Packet-Filter-Information	324
Packet-Filter-Operation	16
QoS-Information	208
Charging-Rule-Report	340
Event-Trigger	16
Access-Network-Charging-Address	32
Access-Network-Charging-Identifier-Gx	84

Tableau H.12 – Acknowledge IP-CAN Session Establishment (interprété de [18])

IE / AVP	taille	condition
Session-Id	68	
Auth-Application-Id	16	
Origin-Host	36	
Origin-Realm	36	
Result-Code	16	
Experimental-Result	44	
CC-Request-Type	16	
CC-Request-Number	16	
fin des AVP obligatoires		
Supported-Features	60	seulement pour 23.402 [9]
Charging-Rule-Remove	60	ou Charging-Rule-Install
Charging-Rule-Install	960	ou Charging-Rule-Remove
Charging-Information	300	
QoS-Information	208	
Revalidation-Time	16	
Default-EPS-Bearer-QoS	88	
Bearer-Usage	16	
3GPP-User-Location-Info	24	
Usage-Monitoring-Information	476	
CSG-Information-Reporting	16	

Tableau H.13 – Re-Auth-Request (interprété de [18])

IE / AVP	taille
Session-Id	68
Auth-Application-Id	16
Origin-Host	36
Origin-Realm	36
Destination-Realm	36
Destination-Host	36
Re-Auth-Request-Type	16
fin des AVP obligatoires	
Session-Release-Cause	16
Origin-State-Id	16
Event-Trigger	16
Event-Report-Indication	536
Charging-Rule-Remove	60
Charging-Rule-Install	960
Default-EPS-Bearer-QoS	88
QoS-Information	208
Revalidation-Time	16
Usage-Monitoring-Information	476
Proxy-Info	72
Route-Record	36

Tableau H.14 – Re-Auth-Answer (interprété de [18])

IE / AVP	taille
Session-Id	68
Origin-Host	36
Origin-Realm	36
fin des AVP obligatoires	
Result-Code	16
Experimental-Result	44
Origin-State-Id	16
IP-CAN-Type	16
RAT-Type	16
AN-GW-Address	32
3GPP-SGSN-MCC-MNC	20
3GPP-SGSN-Address	24
3GPP-SGSN-IPv6-Address	36
RAI	24
3GPP-User-Location-Info	24
3GPP-MS-TimeZone	16
Charging-Rule-Report	340
Access-Network-Charging-Address	32
Access-Network-Charging-Identifier-Gx	84
Proxy-Info	72

Tableau H.15 – Policy and Charging Rules Provision (interprété de [18])

IE / AVP	taille	condition
Session-Id	68	
Auth-Application-Id	16	
Origin-Host	36	
Origin-Realm	36	
Destination-Realm	36	
Destination-Host	36	
Re-Auth-Request-Type	16	
fin des AVP obligatoires		
Event-Trigger	16	seulement pour 23.402 [9]
Event-Report-Indication	536	seulement pour 23.402 [9]
Charging-Rule-Install	960	
QoS-Information	208	

Tableau H.16 – Acknowledge Policy and Charging Rules Provision (interprété de [18])

IE / AVP	taille
Session-Id	68
Origin-Host	36
Origin-Realm	36
fin des AVP obligatoires	
Result-Code	16
Experimental-Result	44
IP-CAN-Type	16
RAT-Type	16
AN-GW-Address	32
3GPP-User-Location-Info	24
3GPP-MS-TimeZone	16
Access-Network-Charging-Address	32
Access-Network-Charging-Identifier-Gx	84

Gateway Control Session Establishment est un « Credit Control Request » (voir tableau H.8). D'après le tableau H.17 il fait 932 octets.

Acknowledge Gateway Control Session Establishment est un message de type « Credit Control Answer » (voir tableau H.9). Il est identique au message « Acknowledge IP CAN Session Establishment » utilisé dans l'architecture 23.401 [8] mais contient l'AVP `Supported-Features` en plus. D'après le tableau H.12 il fait donc 1532 octets.

Gateway Control Session Modification est un message de type « Credit Control Request » (voir tableau H.8). D'après le tableau H.18 il fait 886 octets.

Gateway Control Session Modification reply or Acknowledgement ce message est identique à « Acknowledge IP-CAN Session Establishment » mais ne possède pas l'AVP `Default-EPS-Bearer-QoS` de 88 octets. Il fait donc 1444 octets lorsqu'il est utilisé pour une création de *bearer* et 2324 octets lors d'une destruction.

Gateway Control and QoS Rules Provision est un message de type « Re-Authentication Request » (voir le tableau H.13). D'après le tableau H.15 il fait 1984 octets.

Gateway Control and QoS Rules Provision Acknowledgement de type « Re-Authentication Answer » (voir tableau H.14). D'après le tableau H.16 il fait 440 octets.

Gateway Control Session Termination est un message de type « Credit Control Request » (voir tableau H.8) sans AVP optionnel, il fait donc 244 octets.

Acknowledge Gateway Control Session Termination est un message de type « Credit Control Answer » (voir tableau H.9).

Son seul AVP optionnel est un `Charging-Rule-Remove`. Il fait donc 328 octets.

Tableau H.17 – Gateway Control Session Establishment (interprété de [18])

IE / AVP	taille
Session-Id	68
Auth-Application-Id	16
Origin-Host	36
Origin-Realm	36
Destination-Realm	36
CC-Request-Type	16
CC-Request-Number	16
fin des AVP obligatoires	
Subscription-Id	52
Network-Request-Support	16
Framed-IP-Address	24
Framed-IPv6-Prefix	36
IP-CAN-Type	16
RAT-Type	16
User-Equipment-Info	52
QoS-Information	208
Default-EPS-Bearer-QoS	88
AN-GW-Address	32
3GPP-SGSN-MCC-MNC	20
3GPP-User-Location-Info	24
3GPP-MS-TimeZone	16
Called-Station-ID	32
PDN-Connection-ID	24
APN-Aggregate-Max-Bitrate-DL	16
APN-Aggregate-Max-Bitrate-UL	16

Tableau H.18 – Gateway Control Session Modification : création, modification ou suppression de *bearer* (interprété de [18])

IE / AVP	taille
Session-Id	68
Auth-Application-Id	16
Origin-Host	36
Origin-Realm	36
Destination-Realm	36
CC-Request-Type	16
CC-Request-Number	16
fin des AVP obligatoires	
Packet-Filter-Information	324
Packet-Filter-Operation	16
QoS-Information	208
Event-Trigger	16
QoS-Rule-Report	78

ANNEXE I

Exemple d'application de QoS sous GNU/Linux

Ceci est un exemple de fichier de configuration permettant de créer plusieurs classes de QoS différentes sous GNU/Linux. Deux personnes partagent une connexion, elles ont chacun leurs branches ce qui permet de répartir équitablement le trafic entre eux. Dans chacune des branches on trouve trois files permettant de prioriser le trafic Secure Shell (SSH) par rapport au web, lui-même prioritaire par rapport aux autres types de trafic comme le P2P. La seule file partagée est une file de très haute priorité pour le trafic de VoIP.

```
#!/bin/bash
TC="/sbin/tc"
EXTIF="ppp0"

# Nettoyage
${TC} qdisc del dev ${EXTIF} root    &> /dev/null

#                       1: prio
#                       / \
#                  _____/   \_____
#                 |               |
#               1:1             1:2
#                 |               |
#               2: bfifo        3: htb
#                                 |
#                               3:1
#                               / \
#                  ____________/   \____________
#                 |                             |
#               3:2 personne #1             3:3 personne #2
#              / | \                         / | \
#         ____/  |  \____               ____/  |  \____
#        |       |       |             |       |       |
#       3:21    3:22    3:23          3:31    3:32    3:33
#        |       |       |             |       |       |
#      21:     22:     23:           31:     32:     33:
#       sfq     sfq     sfq           sfq     sfq     sfq

UPRATE="440" # Upload Rates en kbps

# par defaut tout va dans 1:2
${TC} qdisc add dev $EXTIF root handle 1: prio bands 2 priomap 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
# creation de la file haute priorite
${TC} qdisc add dev $EXTIF parent 1:1 handle 2: bfifo

# tete du HTB, les paquet ne correspondant a aucun filtre
# vont dans 3:23
${TC} qdisc add dev $EXTIF parent 1:2 handle 3: htb default 23
```

```
# limitation du debit total du HTB
# les classe HTB se partage ce debit
${TC} class add dev $EXTIF parent 3:0 classid 3:1 htb rate ${UPRATE}kbit

# on divise le trafic de maniere equitable entre les 2 personnes
# (si l'un des deux n'utilise pas sont trafic, le reste ira a l'autre)
${TC} class add dev $EXTIF parent 3:1 classid 3:2 htb rate $((UPRATE/2))kbit ceil ${UPRATE}kbit
${TC} class add dev $EXTIF parent 3:1 classid 3:3 htb rate $((UPRATE/2))kbit ceil ${UPRATE}kbit

################################################
## creation des filtre classes feuille du HTB
## ainsi que des files d'attente correspondantes
################################################

########## personne #1
# ssh
${TC} class add dev $EXTIF parent 3:2 classid 3:21 htb rate $((UPRATE/2 * 20/100))kbit ceil ${UPRATE}kbit \
        prio 1 quantum 1500
${TC} qdisc add dev $EXTIF parent 3:21 handle 21: sfq perturb 10
# web
${TC} class add dev $EXTIF parent 3:2 classid 3:22 htb rate $((UPRATE/2 * 60/100))kbit ceil ${UPRATE}kbit \
        prio 2 quantum 1500
${TC} qdisc add dev $EXTIF parent 3:22 handle 22: sfq perturb 10
# autres
${TC} class add dev $EXTIF parent 3:2 classid 3:23 htb rate $((UPRATE/2 * 20/100))kbit ceil ${UPRATE}kbit \
        prio 3 quantum 1500
${TC} qdisc add dev $EXTIF parent 3:23 handle 23: sfq perturb 10

######### personne #2
# ssh
${TC} class add dev $EXTIF parent 3:3 classid 3:31 htb rate $((UPRATE/2 * 25/100))kbit ceil ${UPRATE}kbit \
        prio 1 quantum 1500
${TC} qdisc add dev $EXTIF parent 3:31 handle 31: sfq perturb 10
# web
${TC} class add dev $EXTIF parent 3:3 classid 3:32 htb rate $((UPRATE/2 * 50/100))kbit ceil ${UPRATE}kbit \
        prio 2 quantum 1500
${TC} qdisc add dev $EXTIF parent 3:32 handle 32: sfq perturb 10
# autres
${TC} class add dev $EXTIF parent 3:3 classid 3:33 htb rate $((UPRATE/2 * 25/100))kbit ceil ${UPRATE}kbit \
        prio 3 quantum 1500
${TC} qdisc add dev $EXTIF parent 3:33 handle 33: sfq perturb 10

################################
## creation des filtre dirigeant les paquets dans les classes
################################

######### Filtre 1:0

# trafic avec TOS a EF
${TC} filter add dev $EXTIF parent 1:0 protocol ip prio 1 u32 match ip tos 0xb8 0xff flowid 1:1

# tous le reste va dans 1:2 (donc 3:0)

######### Filtre 3:0
```

```
# les paquets de la personne #2
# utilise un marquage realise par netfilter (dont la configuration est plus souple)
${TC} filter add dev $EXTIF parent 3:0 protocol ip prio 1 handle 3 fw classid 3:3

# les paquets de la personne #1
# tous le reste
${TC} filter add dev $EXTIF parent 3:0 protocol ip prio 3 u32 match ip dst 0.0.0.0/0 flowid 3:2

######### Les filtres relatif a la branche HTB de la personne #2

# file prio 1
# ssh
${TC} filter add dev $EXTIF parent 3:3 protocol ip prio 1 u32 match ip dport 22 0xffff flowid 3:31
${TC} filter add dev $EXTIF parent 3:3 protocol ip prio 2 u32 match ip sport 22 0xffff flowid 3:31

# file prio 2
# http
${TC} filter add dev $EXTIF parent 3:3 protocol ip prio 5 u32 match ip dport 80 0xffff flowid 3:32
# dns
${TC} filter add dev $EXTIF parent 3:3 protocol ip prio 6 u32 match ip dport 53 0xffff flowid 3:32
# https
${TC} filter add dev $EXTIF parent 3:2 protocol ip prio 7 u32 match ip dport 443 0xffff flowid 3:32

# file prio 3
# tout le reste
${TC} filter add dev $EXTIF parent 3:3 protocol ip prio 8 u32 match ip dst 0.0.0.0/0 flowid 3:33

######### Les filtres relatif a la branche HTB de la personne #1

# file prio 1
# ssh
${TC} filter add dev $EXTIF parent 3:2 protocol ip prio 1 u32 match ip dport 22 0xffff flowid 3:21
${TC} filter add dev $EXTIF parent 3:2 protocol ip prio 1 u32 match ip sport 22 0xffff flowid 3:21

# file prio 2
# web
${TC} filter add dev $EXTIF parent 3:2 protocol ip prio 2 u32 match ip dport 80 0xffff flowid 3:22
${TC} filter add dev $EXTIF parent 3:2 protocol ip prio 2 u32 match ip sport 80 0xffff flowid 3:22
${TC} filter add dev $EXTIF parent 3:2 protocol ip prio 2 u32 match ip dport 443 0xffff flowid 3:22
# dns
${TC} filter add dev $EXTIF parent 3:2 protocol ip prio 2 u32 match ip sport 53 0xffff flowid 3:22
${TC} filter add dev $EXTIF parent 3:2 protocol ip prio 2 u32 match ip dport 53 0xffff flowid 3:22
# email
${TC} filter add dev $EXTIF parent 3:2 protocol ip prio 2 u32 match ip dport 25 0xffff flowid 3:22

# file prio 3
# on a specifie a la racine du HTB que tout paquet ne correspondant au aucun filtre va dans cette classe.
```