# Interaction des Mécanismes RLC/MAC et de SCTP dans les Réseaux Mobiles B3G

Mèriem Afif




# THÈSE

Présentée pour obtenir le grade de Docteur en Technologies de l'Information et de Communications
de

l'École Supérieure des Communications de Tunis
et de
l'École Nationale Supérieure des Télécommunications de Paris

Spécialité : Informatique et Réseaux

par

## Mériem AFIF

# Interaction des Mécanismes RLC/MAC et de SCTP dans les Réseaux Mobiles B3G

Soutenue le 12 Novembre 2007 devant le jury composé de :

| | | |
|---|---|---|
| Riadh Robbana | Professeur, EPT-Tunis | Président |
| Tijani Chahed | Professeur, INT-Evry | Rapporteur |
| Zied Choukair | Maître de Conférences, SUPCOM-Tunis | Rapporteur |
| Sihem Guemara | Maître de Conférences, SUPCOM-Tunis | Examinatrice |
| Elie Najm | Professeur, ENST-Paris | Examinateur |
| Sami Tabbane | Professeur, SUPCOM-Tunis | Directeur de Thèse |
| Philippe Martins | Maître de Conférences, ENST-Paris | Directeur de Thèse |

---

Travail effectué, dans le cadre d'une convention de co-tutelle, au sein de l'unité de recherche MEDIATRON de SUP'COM et du département INFRES de l'ENST-Paris

# Remerciements

Cette thèse au long cours s'achève et je tiens ici à remercier toutes les personnes qui m'ont accompagnée, soutenue et bien souvent donnée d'elles-mêmes durant toutes ces années.

Cette thèse n'aurait vu le jour sans la confiance, la patience et la générosité de mes deux directeurs de thèse, Monsieur Sami Tabbane, professeur à l'École Supérieure de Communications de Tunis, et Monsieur Philippe Martins, maître de conférence au département informatique et réseaux de l'École Nationale Supérieure de Télécommunications de Paris. Je voudrais aussi les remercier pour le temps qu'ils m'ont accordé tout au long de ces années, d'avoir cru en mes capacités et de m'avoir aidé à élaborer mes travaux de thèse. De plus, les conseils qu'ils m'ont divulgué tout au long de la rédaction, ont toujours été clairs et succincts, me facilitant grandement la tâche et me permettant d'aboutir à la production de cette thèse. Je les remercie également pour leurs qualités humaines certaines, leur disponibilité et leur écoute qui ne sont jamais démenties, et pour toute l'énergie consacrée.

Je ne sais comment exprimer ma gratitude à ces deux personnes autrement qu'en leur promettant d'agir comme eux avec des étudiants dans ma situation, si un jour l'occasion m'en est donnée.

Je tiens à remercier Monsieur Philippe Godlewski, professeur au département INFRES, pour avoir bien voulu réviser la thèse. Ses commentaires et ses suggestions ont été fort judicieux et appréciés. Nos discussions longues et passionnées et son esprit critique ont largement contribué à m'aquérir rigueur dans mon travail. Je le suis profondément reconnaissante pour ses conseils et critiques qu'il m'a prodigués tout au long de mes séjours à l'ENST.

Je voudrais également remercier Monsieur Riadh Robbana qui m'a fait l'honneur de présider le jury de ma soutenance.

Mes plus chaleureux remerciements s'adressent à Monsieur Tijani Chahed et à Monsieur Zied Choukair d'avoir accepter la lourde tâche de rapporter ce mémoire, de même que de participer au jury. Ils ont également contribué par leurs nombreuses remarques et suggestions à améliorer la qualité et la clarté de ce mémoire, et je leur en suis très reconnaissante. Mes remerciements vont également aux examinateurs qui ont eu l'amabilité d'examiner ma thèse. Je pense, plus particulièrement à Madame Sihem Guemara et Monsieur Elie Najm.

Je souhaite également exprimer ma profonde gratitude à Monsieur Naceur Ammar, directeur de SupCom, et Monsieur Sadok Mabrouk, ancien secrétaire générale de SupCom, pour leurs aides et

encouragements.

J'ai une pensée toute particulière pour toute l'équipe du département INFRES, doctorants et stagiaires, avec qui j'ai partagé mes séjours à l'ENST avec beaucoup de bonheur et parmi lesquels je comptes à présents certains de mes meilleurs amis. Je pense aussi à mes collègues et amis de MEDIATRON et de SupCom qui ont vécu avec moi divers moments tristes ou joyeux de ma vie de doctorante. Je pense plus particulièrement à Manel.

Que soient aussi remerciés les membres de ma famille, je leur remercie pour leurs encouragements et leur soutient constant. Un grand merci à ma chère mère et à ma soeur qui m'ont toujours encouragée à terminer cette thèse. Je les remercie d'avoir su créer autour de moi une bonne ambiance de détente.

Je dédie ce travail
A la mémoire de mon cher et regretté père
A ma chère mère

«En témoignage de ma profonde affection et mon amour indéfectible, qu'ils trouvent ici une récompense aux sacrifices déployés à mon égard depuis ma naissance»

A tous les membres de ma famille
A tous mes amis

*« Il n'y a point de chemin trop long à qui marche lentement et sans se presser ; il n'y a point d'avantages trop éloignés à qui s'y prépare par* ***la patience****. »*

Jean de LA BRUYÈRE

# Résumé de l'étude

Ce rapport est composé de 6 chapitres. L'objectif de ce travail de thèse est de montrer comment tirer profit des interactions inter-couches pour le contrôle de congestion et de flux. Le contexte de notre étude a pour contexte principale les réseaux mobiles de transmission de données (EDGE/EGPRS) et plus particulièrement le *handover data* intra et inter-RAT (*Radio Access Technology*) (EDGE/WLAN). Nous nous intéressons dans ce travail au protocole de transport SCTP (*Stream Control Transmission Protocol*).

Optimiser de manière dynamique la transmission de données, en tenant compte à la fois des besoins de différentes applications et des caractéristiques très variables du médium, nécessite de nouvelles interactions entre différentes couches de la pile protocolaire. Les solutions liant les réactivités des couches transport aux changements des conditions de transmission sur l'interface radio, requièrent le choix du protocole de transport. Nous avons considéré le SCTP dans nos études afin de profiter de ses caractéristiques innovantes par rapport aux protocoles de transport usuels (TCP et UDP). Notre analyse de couche transport se base, dans un premier temps, sur le mécanisme *multistreaming* de SCTP. Les résultats de simulations montrent que les performances obtenues avec SCTP avec *multistreaming* sont meilleures que celles obtenues avec TCP, supportant une application de type HTTP1.0 et opérant sur une interface radio de type EDGE/EGPRS. SCTP avec *multistreaming* permet d'éviter les problèmes de *head of Line Blocking* (HoL), résultat déjà établi par plusieurs études sur les réseaux filaires.

La deuxième contribution de cette thèse porte sur la conception d'un nouveau mécanisme permettant de remonter les mesures sur l'interface radio à la couche SCTP. Nous proposons ainsi une extension de SCTP basée sur le *multihoming* et la remontée des mesures de qualité de lien radio. Nous créons un nouveau *chunk*, appelé *QoS_Measurement_Chunk*. C'est grâce à ce *chunk* que SCTP met à jour ses paramètres de contrôle de congestion (cwnd, ssthresh) en fonction des mesures (MCS, BLER) qui y sont contenues. Ce concept, testé dans des situations de dégradations de la qualité sur l'interface radio, montre une meilleure performance que le SCTP standard. Le *QoS_Measurement_Chunk*, basé sur une association du *multihoming* à la mobilité, prouve une amélioration des performances de SCTP particulièrement dans une situation de *handover data* EDGE/EDGE et EDGE/WLAN.

Notre troisième contribution concerne l'élaboration de modèles analytiques permettant de lier le comportement au niveau SCTP aux conditions de propagation au niveau de la couche liaison de EDGE. Nous mettons en évidence ainsi une modélisation inter-couches entre la couche liaison et la couche transport. Des simulations, nous ont permis de relever des mesures sur le débit utile disponible au niveau SCTP ainsi qu'au niveau liaison. Plus précisément, les résultats obtenus nous ont permis de représenter le débit utile au niveau SCTP en fonction du débit utile au niveau liaison de EDGE. Nous pouvons ainsi modéliser les réactivités de la couche transport en fonction des paramètres qualité de la couche liaison sur une connexion évaluée de bout en bout. Au moyen des outils mathématiques, des équations théoriques paramétrables ont été obtenues. En fonction du couple (MCS, BLER), une fonction analytique quadratique permet de lier l'évolution de la taille de la fenêtre de congestion au débit utile sur l'interface radio. Cette modélisation introduit une modification du mécanisme de contrôle de congestion consistant ainsi en une restriction à la phase *slow start*. La fenêtre de congestion n'évolue plus de façon exponentielle, ses variations sont plutôt conditionnées par les conditions de transmission sur l'interface radio qui lui sont remontées dans des *chunks* de contrôle ; les SACKs par exemple. Ceci permet d'avoir un contrôle continu de la connexion. Nous avons procédé par des tests, au moyen des outils statistiques, afin de choisir judicieusement les fonctions mathématiques proposées pour chaque (MCS, BLER). Cette optimisation *cross-layer* permet ainsi l'évitement de congestion au niveau transport déclenchée par des dégradations sur l'interface radio. Notre but est de fournir un moyen permettant de modéliser de façon optimale l'interaction entre la couche transport (SCTP) et la couche liaison (RLC/MAC EDGE).

# Abstract

The main goal of this work is to propose an efficient adaptive congestion and flow control mechanisms using cross-layer interactions in wireless networks. The context of our studies includes mainly wireless data transmission over networks such as (EDGE/EGPRS) and aiming more particularly the intra and inter-RAT (Radio Access Technology) data handover (EDGE/WLAN). We are interested in this work on SCTP transport protocol (Stream Control Transmission Protocol).

To optimize data transmission according to both data characteristics and channel condition variations, a cross-layering approach becomes necessary. In order to propose solutions that adapt transport layer reactivities to radio interface transmission conditions change, an efficient transport protocol is of primary importance. We use SCTP protocol in order to benefit from its features, compared to the ordinary transport protocols (TCP and UDP). Our first study is focused on SCTP's multistreaming mechanism. Our simulations results show that the obtained performances with SCTP's multistreaming are better than those obtained with TCP, supporting an HTTP1.0 application over EDGE/EGPRS radio interface. SCTP's multistreaming mechanism contributes to the avoidance of Head of Line Blocking (HoL) matters, which has been proven by several studies with wired networks.

The second contribution of this thesis relates to a new mechanism design that reports radio interface measurements to SCTP layer. Thus, we propose an SCTP extension based on multihoming and radio link quality measurements report. We create a new SCTP control chunk, called QoS_Measurement_Chunk. This extension contributes to improve the protocol performance in variable transmission conditions on radio environment. The SCTP QoS_Measurement_Chunk extension is a cross-layer mechanism that uses the control chunk aspect of SCTP. This chunk serves to inform the network about the transmission conditions variations on the radio interface. According to information supported by this chunk, sent by the mobile terminal to the GGSN, the network adapts its transmission flow by modifying the congestion control parameters (cwnd and ssthresh). This concept, activated in different radio interface continuous quality degradation scenarios, shows better performance than standard SCTP. These degradations are considered in order to trigger a data handoff that activates the multihoming SCTP's feature. QoS_Measurement_Chunk, based on a multihoming/mobility combination, particularly insures an improvement of SCTP performances in data handover situations : EDGE/EDGE and EDGE/WLAN.

Our third contribution relates on an analytical model proposition that expresses the SCTP layer behavior depending on the EDGE link layer propagation conditions. Thus, we highlight a cross-layer modeling between link layer and transport layer. Simulations, give us measurements on both SCTP bandwidth and link layer bandwidth. More precisely, the obtained results enabled us to represent the SCTP bandwidth variations according to EDGE link layer bandwidth variations, which offer a model of the transport layer reactivities depending on the link layer quality parameters for the end by end network connection. By means of mathematical tools, tunable theoretical equations were obtained. According to the couple (MCS, BLER), a quadratic analytical type function relates the evolution of the congestion window size to the radio interface bandwidth. This modeling introduces a modification of the congestion control mechanism consisting on restriction to the slow start phase. The congestion window doesn't progress exponentially. Its variations are rather limited by the radio interface transmission conditions change which can be reported on a control chunk as SACKs chunks. This provides a continuous connection control. We proceeded by tests, by means of statistic tools, in order to judiciously choose the mathematical functions suggested for each (MCS, BLER). This cross-layer optimization avoids congestion control triggering at transport layer caused by radio interface quality degradations. Our goal is to provide means that give an optimal interaction model between transport layer (SCTP) and link layer (RLC/MAC of EDGE).

# Table des matières

# Table des Figures

# Table des Tableaux

# Acronymes

**A**

| | |
|---|---|
| ACK | Acknowledgement |
| AMM | Address Management Module |
| AMPS | Advanced Mobile Phone Service |
| APN | Access Point Name |
| ARQ | Automatic Repeat reQuest |
| A-RWND | Advertised Receiver Window Credit |
| ASCONF | Address Configuration Change Chunk |
| ASCONF-Ack | Address Configuration Acknowledgement |

**B**

| | |
|---|---|
| BDP | Bandwidth Delay Product |
| BLER | BLock Error Rate |
| BSN | Block Sequence Number |
| BSS | Base Station Subsystem |
| BSSGP | Base Station System GPRS Protocol |
| BSS-PFC: | Base Station Subsystem – Packet Flow Context |

**C**

| | |
|---|---|
| $C^3$-TCP | Cross-layer Congestion Control- TCP |
| CCM | Congestion Control Module |
| CN | Correspondant Node |
| CS | Coding Scheme |
| CSCTP | Cellular SCTP |
| CWND | Congestion Window |
| CWR | Congestion Window Reduced |

**D**

| | |
|---|---|
| DF | Don't Fragment |
| DHCP | Dynamic Host Configuration Protocol |
| DL | Down Link |
| DNS | Domain Name Server |
| DoS | Denial of Service |

**E**

| | |
|---|---|
| EC | Error Counter |

| | |
|---|---|
| ECNE | Explicit Congestion Notification Echo |
| ECSD | Enhanced Circuit Switched |
| | Enhanced Data rates for Global Evolution |
| EDGE | |
| EGPRS | Enhanced General Packet Radio Service |
| **F** | |
| FCFS | First Come First Served |
| FTP | File Transfert Protocol |
| **G** | |
| GERAN | Gsm Edge Radio Access Network |
| GGSN | Gateway Gprs Support Node |
| GLL | Generic Link Layer |
| GMM | Gprs Mobility Management |
| GMSK | Gaussian Minimum Shift Keying |
| GPRS | General Packet Radio Service |
| GSM | Global System for Mobile communication |
| GTP | Gprs Tunneling Protocol |
| **H** | |
| HA | Host Agent |
| HB | HeartBeat |
| HLR | Home Location Register |
| HoL | Head of Line Blocking |
| HSCSD | High Speed Circuit Switched Data |
| HTML | Hypertext Markup Language |
| HTTP | Hypertext Transport Protocol |
| **I** | |
| IETF | Internet Engineering Task Force |
| IP | Internet Protocol |
| IPAC | IP address Acquisition Completion |
| IR | Incremental Redandancy |
| ITU | International Telecommunication Union |
| **L** | |
| L2HC | Layer 2 Handover Completion |
| L2SS | Layer 2 Signal Strength |
| LA | Link Adaptation |
| LL | Link Layer |
| LLC | Link Layer Control |

| | |
|---|---|
| LQC | Link Quality Control |
| **M** | |
| MAC | Message Authentication Code |
| MAC | Medium Access Control |
| MAXIN | Interface with Maximum Signal Strength |
| MBWA | Mobile Broadband Wireless Access Networks |
| MCS | Modulation Coding Scheme |
| mIP | Mobile IP |
| MM | Mobility Management |
| MN | Mobile Node |
| MS | Mobile Station |
| mSCTP | Mobile SCTP |
| MSS | Maximum Segment Size |
| MTU | Maximum Transmission Unit |
| **N** | |
| NACK | Negative Acknowledgement |
| NIC | Network Interface Card |
| NS2 | Network Simulator 2 |
| NSAPI | Network layer Service Access Point Identifier |
| **O** | |
| OSI | Open System Interconnection |
| **P** | |
| PBA | Partial Bytes Ack |
| PCU | Packet Control Unit |
| PDCH | Packet Data CHannel |
| PDCP | Packet Data Convergence Protocol |
| PDP | Packet Data Protocol |
| PDU | Packet Data Unit |
| PEL | Protocol Engineering Lab |
| PFC | Packet Flow Context |
| PFI | Packet Flow Identifier |
| PMM | Path Management Mechanism |
| PMTU | Path MaximumTransmission Unit |
| PR-SCTP | Partial Reliability SCTP |
| 8-PSK | 8- Phase shift Keying |
| PSTN | Public Switched Telephone Network |
| **Q** | |

| | |
|---|---|
| QoS | Quality of Service |
| **R** | |
| RAT | Radio Access Technology |
| RLC | Radio Link Control |
| RR | Round Robin |
| RTO | Retransmission Time Out |
| RTT | Round Trip Time |
| RTTVAR | Round Trip Time Variation |
| RWND | Receiver Window |
| **S** | |
| SACK | Selective Acknowledgement |
| SCTP | Stream Control Transmission Protocol |
| SDU | Service Data Unit |
| SGSN | Serving Gprs Support Node |
| SID | Stream Identifier |
| SIGTRAN | Signalling Transport |
| SM | Session Management |
| SN | Sequence Number |
| SNDCP | Sub Network Dependent Convergence Protocol |
| SNR | Signal to Noise Ratio |
| SNS | Sequence Number Space |
| SRTT | Smoothed Round Trip Time |
| SS7 | Signalling System n°7 |
| SSN | Stream Sequence Number |
| SSTHRESH | Slow Start Threshold |
| **T** | |
| TBF | Temproray Block Flow |
| TCB | Transmission Control Block |
| TCP | Transmission Control Protocol |
| TFT | Trafic Flow Template |
| TI | Transaction Identifier |
| TL | Transport Layer |
| TLLI | Temproray Logical Link Identity |
| TLV | Type Length Value |
| TOM | Tunnelling Of Messages |
| TS | Time Slot |
| TSN | Transmit Sequence Number |

**U**

| | |
|---|---|
| UDP | User Datagram Protocol |
| UL | Up Link |
| ULP | Upper Layer Protocol |
| UMTS | Universal Mobile Telecommunications System |

**V**

| | |
|---|---|
| VoIP | Voice over IP |

**W**

| | |
|---|---|
| WLAN | Wireless Local Area Network |
| WS | Window Size |

# Chapitre 1 : Introduction Générale

L'objectif visé par nos travaux, présentés dans ce rapport, est d'étudier l'interaction entre les protocoles couche transport et les mécanismes couche liaison de données dans le cas des réseaux radio mobiles de transmission de données (EDGE/EGPRS). Pour l'atteindre nous nous intéressons au protocole de transport SCTP (*Stream Control Transmission Protocol*) qui, comparé aux protocoles de transport usuels, présente une meilleure fiabilité grâce à ses principales caractéristiques : le *Multihoming,* le *Multistreaming* et la sécurisation d'une connexion. Plus exactement, nos études mettent en avant l'importance d'un dialogue inter-couches (Transport/Liaison) afin d'améliorer les performances du protocole de transport engendrant ainsi une amélioration de la qualité de service offert en environnement radio mobile et particulièrement dans le cas d'une mobilité inter différents réseaux d'accès. Nos proposons une extension possible de SCTP lui permettant d'acquérir des informations sur la qualité du lien radio contribuant ainsi à adapter le flux de transmission au niveau 4 à celui au niveau 2. Une telle adaptation nous conduit à proposer une nouvelle politique du mécanisme de contrôle de congestion. Notre contribution se base principalement sur une combinaison de la mobilité et du *multihoming* dans un cadre de *handover* inter-RAT (*Radio Access Technology*). Nous proposons également une modélisation analytique permettant de lier les variations du débit utile au niveau transport à celui au niveau liaison dans le cas d'une connexion de bout en bout. Basées sur le principe *cross-layer*, les solutions proposées permettent d'améliorer les performances au niveau transport en liant les réactivités des couches supérieures aux variations des conditions de propagation et aux adaptations de la couche liaison. Ce chapitre introductif exploite d'abord l'état de l'art du protocole SCTP, noeud principale de nos études, en passant par une brève historique. Ensuite, une formulation de la problématique et un éclaircissement des domaines de nos motivations et contributions sont présentés. Enfin, une organisation du présent rapport est fournie.

## 1.1. État de l'art de SCTP

Le protocole SCTP est défini dans la RFC 2960 [RFC2960], et un texte d'introduction est fourni dans la RFC 3286 [RFC3286]. Le premier standard de SCTP a été réalisé en octobre 2000 par le groupe de travail SIGTRAN (*SIGnaling TRANsport*) de l'IETF (*Internet Engineering Task Force*). Initialement, SCTP était destiné au transport de messages de signalisation PSTN (*Public Switched Telephone Network*) sur des réseaux IP (VoIP). En tant que protocole de transport, SCTP est beaucoup plus flexible comparé à TCP (*Transmission Control Protocol*) et plus fiable que UDP (*User Datagram Protocol*). En effet, il fournit des services similaires à TCP, assurant la remise en

ordre des séquences et le contrôle de congestion. De plus, SCTP fournit un concept de transmission de données complètement neuf, basé sur des *streams*. Au début une conception possible, de ce nouveau protocole, doit respecter les exigences suivantes [RFC3286] :

- La transmission des données en messages,
- La possibilité d'assembler plusieurs messages dans une seule trame,
- La livraison de paquets en séquence ou non,
- La fiabilité de transmission,
- Plusieurs adresses IP pour une association,
- Plusieurs flots (*Stream*) de messages dans une association.

SCTP est apparu relativement bien adapté au transport de données en général, notamment pour les applications multimédia demandant une certaine qualité de service. Il est devenu un protocole d'utilisation générale du niveau transport, comme TCP et UDP. Il a été standardisé dans la RFC 2960, dont les principales caractéristiques s'articulent sur les points suivants [RFC2960] :

- Initiation d'une association sécurisée,
- Utilisation du mécanisme «*cookie»* contre les attaques de type DoS (*Denial of Service*),
- Fermeture d'une association sécurisée (la procédure évite des connexions semi-ouvertes),
- *Multihoming* : possibilité d'utiliser plusieurs adresses Ipv4 ou/et Ipv6 pendant une association,
- *Multistreaming* : SCTP utilise des *streams* de messages pour la transmission de données. Le protocole permet d'avoir plusieurs *streams* indépendants dans une association

Le protocole SCTP [RFC2960], est un protocole de transport fiable se déployant sur un service paquet de niveau réseau sans connexion, le protocole IP par exemple. Une connexion SCTP est appelée association dans la terminologie SCTP. Contrairement à TCP, SCTP est orienté message. Dans le cas où un hôte dispose de plusieurs adresses IP, les adresses sont échangées lors de l'établissement de la session. Le *multihoming* est une des particularités de SCTP qui consiste au fait que plusieurs adresses IP peuvent être activées au sein d'une même association. Un mécanisme de contrôle d'erreur est implémenté dans SCTP et permet de détecter les pertes, la rupture de séquences ou la duplication de paquets. Un schéma de retransmission est utilisé pour corriger ces erreurs. SCTP utilise le principe d'acquittement sélectif pour acquitter la réception des données. SCTP offre un service de multiplexage/démultiplexage entre flux, c'est le *multistreaming*. En effet, une application multimédia peut être découpée en plusieurs flots pouvant avoir chacun des schémas de remise des données différents.

Nous exploitons d'abord le *multihoming*. La façon de définir une connexion TCP comme l'ensemble de quatre paramètres (une adresse IP d'émetteur, un port source, une adresse IP du récepteur et un port destination), détermine deux adresses stables et inéchangeables pendant la connexion. Une fois les adresses définies à l'établissement d'une connexion, elles sont maintenues jusqu'à sa terminaison. Ainsi une connexion TCP est établie entre deux adresses IP. Quand à SCTP, il propose une logique différente c'est qu'une association est établie entre deux entités appelées point terminal. Un point terminal est un ensemble d'adresses de destination auxquelles sont envoyés des

paquets ou un ensemble d'adresses sources desquelles des paquets sont également acceptés. Un point terminal comprend aussi un numéro de port, qui doit être unique et ne peut pas être utilisé par un point terminal différent. Pour ce qui est des adresses, un point terminal peut utiliser une liste d'adresses Ipv4, IPv6 ou des noms de machines [RFC2960]. Un chemin est ainsi créé pour chaque adresse IP déclarée par le destinataire pendant la phase d'initialisation. Il contient une adresse IP destination et une adresse IP source en plus des paramètres supplémentaires comme les paramètres de contrôle de congestion, des compteurs de retransmission etc.

Ensuite, la capacité de gérer plusieurs *streams* indépendants au sein d'une même association, est une caractéristique qui rend SCTP différent de tous les protocoles de transports usuels. Celle ci est rendue possible par l'utilisation d'un champ présent dans chaque *chunk* de données indiquant le numéro du *stream* approprié. En fonction de ce numéro, l'émetteur décide dans quel *stream* le message est transmis et le récepteur peut reconnaître à quel *stream* le message appartient. Comme la gestion d'acquittement est basée sur des *streams*, la transmission de chaque *stream* est indépendante. Les applications qui peuvent bénéficier du *multistreaming* sont principalement :

- Les protocoles de signalisation dans les réseaux téléphoniques publiques. Ce qui est important, dans ce cas, est l'indépendance des *streams* dans une association et la possibilité de livraison de messages en hors séquence.
- Les navigateurs web, en effet, le navigateur peut télécharger une page entière, au cours d'une association, dont tous les objets (fichier html, images, sons, styles etc.) peuvent être téléchargés en même temps dans des *streams* indépendants. Cela permet d'économiser des ressources tant du côté du client que du côté du serveur, ce que nous prouvons par simulations dans le troisième chapitre du présent rapport.

Enfin, comme les adresses sont maintenues tout au long de la durée de vie d'une association, une extension de SCTP a été proposée [STE05] permettant la configuration dynamique des adresses durant une association. La fonctionnalité de configuration dynamique est particulièrement utile. En fait, le *multihoming* dynamique permet de modifier la configuration des interfaces réseaux sans déranger des connexions réseau déjà établies. De plus, dans le cas d'un point terminal possédant plusieurs interfaces réseau, une mobilité peut être supportée au moyen du *multihoming* dynamique. C'est qu'une interface réseau sera utilisée pour assurer l'établissement d'une association. En même temps, l'autre interface se connectera à un autre réseau et une fois connectée elle sera ajoutée à l'association (adressage dynamique). La transmission de données au cours de l'association peut être maintenant assurée par la deuxième interface.

## 1.2. Problématique

Dans les réseaux filaires la congestion est déclenchée suite à une perte ou une saturation des paquets au niveau des *buffers*. La fenêtre de congestion doit donc être en mesure de s'adapter à l'état courant des *buffers*, sans pour autant influer négativement sur le débit de la connexion. Alors que le caractère mobile du réseau implique que deux points terminaux puissent à tout moment être déconnectés, créant ainsi de nombreuses pertes de paquets, notamment d'acquittements, qui pour SCTP ou TCP vont engendrer de nombreux hors temps consécutifs provoquant ainsi une congestion. Le problème réside dans la transition qui doit s'effectuer lorsqu'un mobile change de position. Ce problème connu sous le nom de *handoff* intervient à plusieurs niveaux réseau.

L'ensemble de congestion prend alors sa source dans toutes les couches réseaux inférieures à la couche transport. Les voies sur lesquelles s'articule notre vision d'amélioration de performances de SCTP en environnement mobiles, sont :

- Face aux nombreux hors temps et acquittements dupliqués générés, nous pouvons souhaiter différencier leurs causes et modifier ainsi les algorithmes de contrôle de congestion,
- Nous pouvons supposer que SCTP transmet sur le réseau un nombre élevé de paquets à la fois. L'idée est alors de minimiser le nombre de paquets dans le réseau afin de diminuer les problèmes d'accès, et donc les pertes.

Il s'agit donc de concevoir **un nouveau algorithme de régulation de la fenêtre de congestion en fonction de la capacité du réseau**. Pour optimiser la fenêtre de congestion nous devons étudier d'abord les causes principales de dégradation de la qualité du lien radio. C'est dans cette optique que nous nous intéressons particulièrement à une situation de *handover*. En fait, la procédure de *handover* d'un terminal mobile se décompose en trois étapes. D'abord, certaines mesures doivent être effectuées et rassemblées dans un rapport de mesures. Ensuite une décision de *handover* est prise en fonction du rapport. Enfin, le *handover* est exécuté si la décision de *handover* est positive. En effet, la première étape consiste à mesurer certains paramètres requis pour analyser le statut de la connexion existante entre le terminal, la cellule utilisée et le statut de la qualité d'autres interfaces réseaux disponibles. Les mesures peuvent être effectuées par le terminal ou par le réseau. Pratiquement, le terminal mobile participe toujours à la prise de mesures [PAH00]. Nous nous intéressons principalement aux paramètres dynamiques dans le cadre de nos études. Ils comprennent la surveillance et analyse des paramètres d'accès réseau tels que la puissance de réception, le taux d'erreur binaire, le taux d'erreur bloc etc. Quand les mesures sont effectuées, elles sont rassemblées dans un rapport de mesures et envoyées à l'entité de décision du *handover* (le réseau).

Après avoir identifier les paramètres qui sont l'origine de déclenchement d'une congestion au niveau transport, nous nous intéressons à ce stade à l'architecture protocolaire réseau. Comme au niveau des architectures usuelles aucun échange d'informations sur l'état de l'interface radio n'est mis en place entre les différentes couches protocolaires, de la physique au transport.

L'architecture actuelle des protocoles réseaux, que ce soit le modèle OSI (*Open System Interconnection*) ou le modèle TCP/IP, repose sur un ensemble de modules devant chacun fournir un service précis et ce de façon autonome. Ces modules ont été organisés hiérarchiquement dans une pile, chaque module se trouvant au dessus du service dont il a besoin pour fournir le sien.

Cependant, dès l'apparition d'influences entre deux couches, l'utilité de dialogue entre elles afin qu'elles puissent communiquer entre elles un ensemble de variables, statistiques ou dynamiques, qui aideront à la résolution d'un problème de congestion par exemple, s'avère indispensable. C'est ce que proposent les architectures inter-couches (*cross-layers*).

Au final, l'idée dans les réseaux sans fil devraient s'orienter vers une architecture comportant un module *cross-layer* auquel les évènements de chaque couche seraient liés, et qui aurait pour rôle de base de données des variables du réseaux.

Dans nos études, nous nous inspirons du principe *cross-layer* pour concevoir une nouvelle extension de SCTP permettant d'améliorer ses performances en environnement mobile. Notre idée est de concevoir un mécanisme *cross-layer* sans pouvoir ajouter un module spécifique entre la couche liaison de données et la couche transport. Nos contributions sont basées principalement sur

les fonctionnalités offertes par SCTP, et visent une combinaison entre le *multihoming* et la mobilité. Le but d'adapter le flux de transmission sur l'interface radio à celui au niveau transport, est d'éviter une congestion au niveau SCTP déclenchée suite à des dégradations de qualité engendrées par la mobilité.

## 1.3. Motivation et contributions

Premièrement, pour justifier notre choix du type de protocole de transport dans nos travaux de recherche, nous nous intéressons à l'amélioration de performances de SCTP par rapport à TCP. Plusieurs études ont été développées visant la comparaison des deux protocoles, montrant ainsi les avantages de SCTP par rapport à TCP dans les réseaux filaires. En ce qui concerne cette thèse, l'environnement sur lequel nous contribuons pour améliorer la qualité de service est plutôt celui des réseaux mobiles. Nous développons ainsi nos simulations sur un réseau de type EDGE/EGPRS sous une plateforme NS2 (*Network Simulator*). Une application interactive de type HTTP1.0 est mise au dessus d'une couche SCTP ou TCP. Dans une première étape nous montrons que les deux protocoles ont un comportement similaire, en désactivant le *multistreaming* et le *multihoming* pour SCTP. Ce qui est évident puisque les deux protocoles ont le même algorithme de contrôle de congestion. Le *multistreaming* est activé pour SCTP, dans une seconde étape. SCTP apporte une amélioration notable par rapport à TCP.

La deuxième contribution de cette thèse concerne l'élaboration de mécanismes de sélection d'une technologie d'accès radio, en utilisant des interactions entre la couche liaison et la couche transport. En effet, les réseaux 2G, 2.5G, 2.75G, 3G exploitent plusieurs technologies d'accès radio pour lesquels il faudra définir des procédures de *handover* à différents niveaux du réseau. En plus du *handover data* inter-systèmes au niveau liaison, un *handover data* inter-systèmes est défini au niveau transport basé sur des politiques de *multihoming* et de remontée de mesures dynamiques sur l'interface radio aux entités fixes du réseau à savoir le SGSN (*Serving GPRS Support Node* ) et le GGSN ( *Gateway GPRS Support Node*). Cette contribution met en avant une corrélation du *multihoming* à la mobilité. Nous définissons des mécanismes de *handover data* inter-systèmes permettant de maintenir la même session applicative au cours du basculement du système et améliorant ainsi les performances du processus de *handover*. Nous concevons alors un nouveau *chunk* de contrôle SCTP que nous appelons *QoS_Measurement_Chunk*, ainsi qu'une modification du mécanisme de contrôle de congestion en SCTP est introduite.

Enfin, la dernière contribution de cette thèse porte sur une optimisation inter-couches (liaison/transport) (*cross-layer*) mettant en évidence la technologie EDGE/EGPRS et le protocole SCTP. Plus précisément, nous élaborons une modélisation mathématique permettant de lier les réactivités au niveau transport aux changements des conditions de transmission au niveau liaison de données. Cette approche permet une adaptation du flux de transmission au niveau 4 à celui au niveau 2 en fonction des mesures sur l'interface radio remontées à la couche transport. Cette adaptation est assurée par le calcul de nouvelles tailles de fenêtre de congestion en fonction du débit utile disponible sur l'interface radio, au moyen des fonctions mathématiques, que nous proposons dans cette thèse, paramétrables en fonction de changements des conditions de transmission sur l'interface radio (e.g. MCS : *Modulation Coding Scheme*, BLER : *BLock Error Rate*, etc.). Des tests statistiques, tels que KHI2 et calcul des coefficients de variation sont exploités afin de justifier les modélisations théoriques sélectionnées.

## 1.4. Organisation du rapport

Notre rapport de thèse est réparti sur plusieurs chapitres présentant l'enchaînement suivi pour résoudre la problématique posée ci-dessus, au moyen des contributions proposées. Les principaux chapitres sont énumérés ci-dessous.

Le deuxième chapitre comporte une présentation technique du protocole SCTP. Nous nous intéressons à ses fonctionnalités de base telles que l'ouverture, la fermeture d'une association et la procédure de transfert de données, ainsi qu'une description de ses caractéristiques fondamentales est développée. Nous décrivons SCTP par comparaison aux protocoles de transport usuels (TCP et UDP) afin de montrer l'apport de concevoir un nouveau protocole de transport. En particulier, nous nous intéressons aux extensions mobiles SCTP afin de fixer le cadre de nos contributions. Il s'agit de l'adressage dynamique défini dans [STE05] et qui permet d'associer le *multihoming* à la mobilité. Nous étudions de manière critique les solutions proposées dans ce contexte telles que le mSCTP (*Mobile SCTP*) [RIE04], le cSCTP (*Cellular SCTP*) [AYD03], etc. Ces alternatives visent l'implémentation de SCTP, auquel des modifications ont été introduites, en environnement mobile en particulier dans une situation de *handover data*.

Dans le troisième chapitre, une étude de performances de SCTP sur une interface radio EDGE/EGPRS pour un trafic HTTP est développée. Nous décrivons les simulateurs exploités dans nos travaux. Nous nous intéressons à ce niveau à l'apport de la caractéristique de *multistreaming* de SCTP par rapport aux solutions existantes avec les protocoles de transport usuels. Nous expliquons dans cette optique l'avantage d'introduire une telle fonctionnalité pour éviter les problèmes de *head of line blocking* (HOL), handicap majeur de TCP avec les services interactifs. Nous testons d'abord le comportement des deux protocoles TCP et SCTP sous les mêmes conditions standards. Autrement dit, nous désactivons le *multihoming* et le *multistreaming* en SCTP. Un comportement quasi identique est noté, puisque les deux protocoles ont un même mécanisme de contrôle de congestion. Ensuite, nous examinons un scénario consistant à activer le *multistreaming* de SCTP en ouvrant 3 *streams* entre le client et le serveur pour une session web comportant des pages formées de trois objets en moyenne. Enfin, nous comparons le comportement de SCTP avec *multistreaming* au TCP standard pour une application HTTP1.0. Les résultats que nous présentons décrivent l'évolution du temps de réponse en fonction du taux d'erreurs bloc sur l'interface radio EDGE/EGPRS dans un réseau chargé. Nous visualisons également les variations des numéros de séquence au niveau RLC/MAC (*Radio Link Control/Medium Access Control*). Nos résultats sont accompagnés d'une explication détaillée du problème de HOL et comment le *multistreaming* en SCTP permet de l'éviter.

Le chapitre quatre examine en particulier le *multihoming* en SCTP. Nous nous intéressons à une corrélation du *multihoming* avec la mobilité. Nous proposons une modification de SCTP basée sur le *multihoming* et la remontée des mesures sur l'interface radio à la couche transport. Notre extension consiste à créer un nouveau *chunk* de contrôle appelé *QoS_Measurement_Chunk*, qui remonte les mesures à partir du mobile au SGSN et au GGSN. Cette solution est inspirée d'une étude critique des approches proposées avec le mSCTP *(mobile SCTP*), le cSCTP (*cellular SCTP*) et également la GLL (*Generic Link Layer*). Ce mécanisme est fiable dans le sens d'éviter une congestion au niveau transport causée par des dégradations de qualité sur l'interface radio et d'assurer la continuité d'une session applicative déclenchée avant un *handover* inter-système. Le

principe de la proposition est d'adapter le flux de transmission au niveau transport à celui au niveau liaison. Un changement du mécanisme de contrôle de congestion est indispensable. Nous simulons notre extension de SCTP dans différents contextes de *handover data* inter-systèmes. En effet, nous simulons d'abord un *handover data* intra-RAT pour des variations déterministes des conditions de transmission radio (schéma de codage, BLER) ensuite pour des variations aléatoires. Enfin, un *handover data* inter-RAT est mis en avant prouvant ainsi l'efficacité de SCTP avec *QoS_Measurement_Chunk* comparée au SCTP standard.

Le chapitre cinq, en complémentarité avec le chapitre 4, étudie une formalisation mathématique, permettant de lier la taille de fenêtre de congestion au débit utile disponible au niveau de l'interface radio. Il s'agit d'une modélisation analytique mettant en avant un aspect inter-couches (*cross-layer*) (SCTP/RLC-MAC EDGE). Nous abordons dans ce chapitre la problématique résolue par une modélisation *cross-layer* et nous traitons les solutions existantes dans la littérature dans ce domaine afin de positionner notre contribution. Notre proposition consiste à modifier le mécanisme de contrôle de congestion de sorte à nous restreindre à la phase de *slow start* sans pour autant avoir une évolution exponentielle de la fenêtre de congestion. Un ensemble de simulations est considéré en régime permanent permettant de lier le débit utile au niveau transport à celui au niveau liaison par des fonctions mathématiques paramétrables selon le couple (MCS, BLER). Nous considérons des approximations par des fonctions linéaires et quadratiques que nous validons par la suite par des calculs statistiques de KHI2 (variable de K.Pearson) et du coefficient de variation.

Le dernier chapitre de ce rapport conclut les résultats que nous avons obtenus et l'apport de nos contributions, ainsi que des perspectives futures de ce travail sont fournies.

# Chapitre 2 : Fonctions de bases de SCTP et Extensions mobiles de SCTP

## 2.1. Introduction

Ce chapitre présente quelques aspects et fonctionnalités du protocole de transport SCTP (*Stream Control Transmission Protocol*). Cela permet de situer le cadre de nos contributions. Le SCTP est un nouveau protocole de couche transport qui a été conçu par l'IETF (*Internet Engineering Task Force*) pour fournir un transport fiable et sécurisé d'une variété d'applications, principalement la signalisation du réseau de téléphonie publique, sur les réseaux IP (*Internet Protocol*). Malgré que TCP (*Transport Control Protocol*) soit le protocole de transport le plus populaire dans le monde de l'IP, SCTP fournit en plus des fonctionnalités offertes par TCP, d'autres caractéristiques supplémentaires importantes ce qui fait la différence entre les deux protocoles. Parmi les contributions apportées par SCTP nous identifions, dans nos travaux, principalement le *Multihoming* et le *Multistreaming*.

Nous nous intéressons ici en particulier à la problématique de couplage de la mobilité avec le *multihoming* dans une situation de changement d'interfaces (*handover*). Introduire l'aspect de *multihoming* pour une meilleure gestion de la qualité de service, dans les réseaux mobiles de transmission de données, reste un défi majeur et encore largement ouvert.

Le but de ce chapitre est d'analyser et comprendre le protocole SCTP et les améliorations qu'il peut apporter dans les réseaux de transmission de données, principalement mobiles. D'abord nous mettons l'accent sur une brève étude comparative entre les protocoles de transport existants (TCP et UDP: *User Datagram Protocol*) et SCTP. Cette étude sera développée de façon plus détaillée et illustrée par des simulations dans une étape plus avancée de ce rapport. Ensuite, différents aspects fonctionnels de SCTP tels que les étapes d'établissement d'une association, les types de messages échangés ainsi qu'un détail de la procédure de transfert de données avec SCTP et le principe du mécanisme de contrôle de congestion en SCTP par référence à celui en TCP, sont abordés. Enfin, nous présentons les différentes extensions de SCTP liées aux *multihoming* et à la mobilité, qui ont été proposées pour améliorer le basculement d'interfaces dans un réseau à multi-accès radio, ainsi que leurs insuffisances fonctionnelles.

## 2.2. Limitations des protocoles de transport existants

Les protocoles usuels de transport des données dans les réseaux IP sont TCP et UDP. Pour répondre aux nouveaux besoins des télécommunications en terme des exigences de fiabilité et de synchronisation liées à la signalisation, un nouveau protocole a été élaboré par le groupe de travail SIGTRAN (*Signaling Transport*) de l'IETF appelé SCTP [RFC2960].

Deux fonctionnalités majeures ont été introduites pour le design de SCTP, à savoir le *multihoming* et le *multistreaming*. Les différences principales avec TCP sont le *multi-homing* et le concept de flux (*Stream*) à l'intérieur d'une même connexion (appelée association dans la terminologie SCTP). Alors que dans TCP un flux fait référence à une séquence d'octets, un flux SCTP fait référence à une séquence de messages (*chunk*).

De plus, des fonctionnalités qui sont optionnelles en TCP ont été incluses dans la spécification de base de SCTP, telles que le *Selective Acknowledgment* (SACK), permettant d'acquitter la réception de datagrammes erronés ou dupliqués.

Les attaques simples de type *SYN attack* qui affectent TCP ne sont plus possibles avec SCTP. Ce nouveau protocole inclut aussi des mécanismes protégeant les applications contre le *Head of Line Blocking* (HOL) grâce à sa caractéristique de *multistreaming*.

Le protocole SCTP présente des caractéristiques très intéressantes en tant que moyen de transport pour la signalisation SS7 (*Signaling System N°7*), et aussi comme alternative aux protocoles TCP ou UDP dans des applications standards. L'avantage d'un nouveau protocole tel que SCTP est que tous les aspects optionnels des protocoles TCP ou UDP ont été intégrés dans la conception de base de SCTP [FRA03]. Ce nouveau protocole peut se positionner entre TCP et UDP en offrant une QoS (*Quality of Service*) ajustable à l'application tout en évitant les contraintes de TCP qui contrarient les applications multimédia.

La conception de hôtes à connexions-multiples (*multihomed*) permet à une seule association SCTP d'utiliser des chemins multiples à travers un seul port. Actuellement, plusieurs connexions sont nécessaires, souvent de types différents (UDP, TCP), ce qui implique l'utilisation de plusieurs ports et des difficultés de gestion du réseau.

Malgré que la motivation principale de SCTP est le transport des messages de signalisation sur IP sur le réseau commuté de téléphonie publique (PSTN, VoIP (*Voice over Internet Protocol*)), SIGTRAN assure que SCTP est fiable pour d'autres applications ayant des exigences similaires [RQX02]. Le tableau 2.1 récapitule l'apport de SCTP par rapport à TCP et UDP.

Tableau 2.1. Comparaison entre SCTP-TCP et UDP[RQX02] .

| | **UDP** | **TCP** | **SCTP** |
|---|---|---|---|
| **Démarrage** | Aucun établissement de connexion | Établissement de la connexion en 3 temps | Établissement de la connexion en 4 temps avec échange de *cookies*. L'échange de données peut être effectué déjà dès le 3ième message d'établissement. |
| **Assurance de la livraison (fiabilité)** | Aucun acquittement des messages et aucune assurance de livraison | Acquittement des messages pour assurer la transmission. L'acquittement sélectif est optionnel dans TCP | Acquittement des messages pour assurer la transmission. **L'acquittement est sélectif** ; seuls les messages erronés sont retransmis. |
| **Ordre de livraison** | Aucun mécanisme mis en oeuvre pour assurer l'ordre des messages | Numérotation des messages pour assurer l'ordre | Les messages peuvent être en séquence ou non. Même les messages qui ne sont pas en séquence, gardent une assurance de livraison. |
| **Contrôle de congestion** | Aucun mécanisme mis en oeuvre | Mise en oeuvre des mécanismes de contrôle de congestion (*Additive Increase, Multiplicative Decrease)* | Mise en oeuvre des mécanismes de contrôle de congestion (*Additive Increase, Multiplicative Decrease)* |
| **Procédure de fermeture** | Aucune (protocole orienté non connexion) | Fermeture de la connexion spécifiée. Le mode «*half closed*»est possible. | Fermeture de la connexion spécifiée. Le mode «*half closed*» n'est pas possible. |
| ***SYN-Attack*** | Insensible (aucune connexion effectuée) | C'est un des problèmes de TCP | **Résolu avec le cookie** |
| ***Head of Line Blocking*** | Insensible | Partiellement résolu en ouvrant plusieurs connexions | **Résolu car SCTP permet plusieurs *Streams* (non bloquant entre eux dans une même association)** |
| ***Multihoming*** | Non supporté | Non supporté | **Supporté** |

## 2.3. Présentation générale du protocole SCTP

### 2.3.1 Généralités

SCTP (*Stream Control Transmission Protocol*) est un protocole de transport proposé par l'IETF [RFC2960]. Il a été conçu pour palier certaines limitations inhérentes à TCP lors du transport de signalisation téléphonique sur IP (problème du *head of line blocking*). SCTP apporte un service de transport fiable de messages issues des applications utilisateurs (e.g. protocoles de signalisation). Il est orienté connexion. Une connexion SCTP est appelée association (cf. figure 2.1). Il est possible de multiplexer plusieurs flux de messages au sein d'une même association (service de *multistreaming*). Ceci se traduit au niveau paquet par la possibilité de transporter plusieurs messages de signalisation dans un même paquet SCTP. Les messages sont encapsulés dans des structures de données appelés *chunks*. Les *chunks* sont eux mêmes encapsulés dans des paquets SCTP.

SCTP permet également de mettre en oeuvre le *multihoming* en introduisant la possibilité d'associer plusieurs adresses IP à un même port SCTP (cf. figure 2.1). Plusieurs chemins sont alors disponibles pour mettre en relation deux noeuds SCTP distants. A un instant donné, seul un chemin est actif (i.e. est utilisé pour transporter les données) les autres chemins sont utilisés comme sauvegarde au cas où le chemin actif deviendrait indisponible.

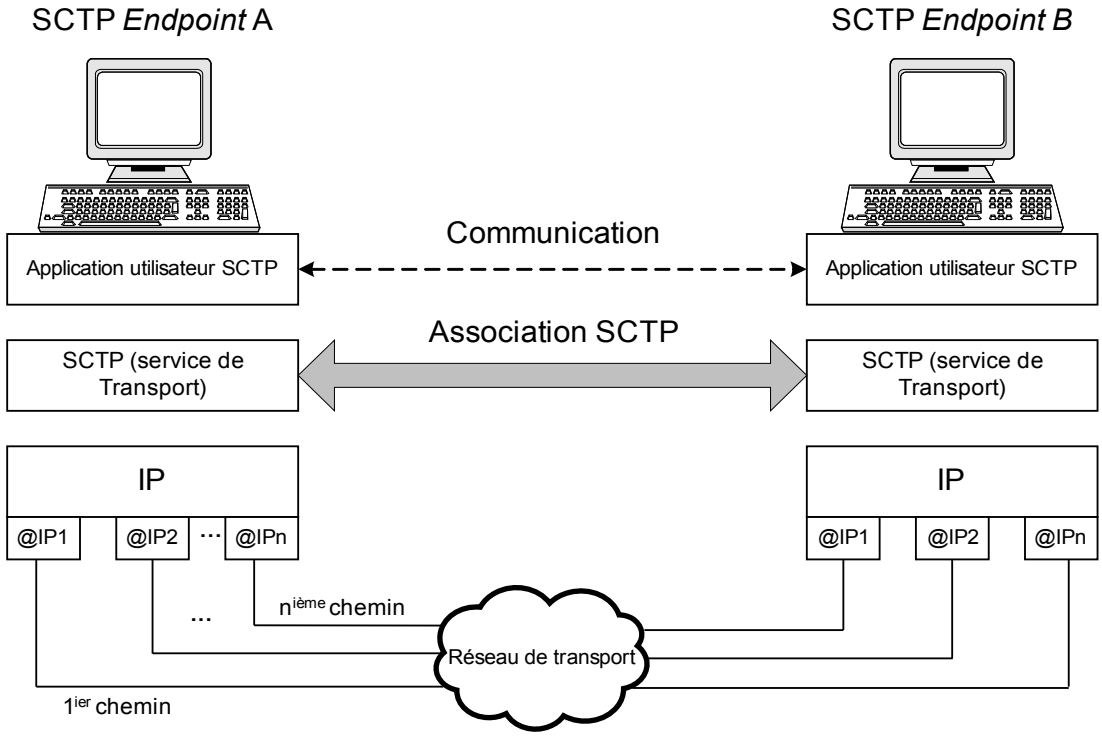



**Figure 2.1.** Schéma d'une association SCTP.

Une association SCTP est établie à la demande d'une application. L'établissement de l'association se fait en deux phases impliquant l'échange de quatre messages (4-*way Handshake* ,cf. figure 2.6). Durant la première phase d'établissement de l'association, un mécanisme de sécurité est mis en place afin d'éviter les attaques de *Deny of Service* (ce mécanisme qui utilise des *Cookie*, est décrit dans la RFC 2522 [RFC2522]). La deuxième phase est celle qui établit effectivement l'association en réservant les ressources associées aux sockets[1] de l'association sur chaque point terminal. Il est également possible au cours de cette deuxième phase d'envoyer des données utilisateurs dans les paquets d'établissement (*chunk Cookie Echo* et *Cookie Ack*).

Le service de transport SCTP peut être décomposé en un ensemble de fonctions (c.f. figure 2.2) dont le descriptif de chacune d'elles est donné par [RFC2960].

Une des caractéristiques du protocole SCTP est qu'en cas de nécessité, il permet de fragmenter les messages utilisateurs afin de s'assurer que les paquets SCTP soient passés aux couches inférieures conformément au PMTU (*Path Maximum Transmission Unit*) spécifié. En réception les fragments sont rassemblés en messages complets avant qu'ils ne soient dirigés vers le nœud SCTP de destination. Les différents fragments d'un même message portent le même SSN (*Stream Sequence Number*). SCTP attribue un TSN (*Transmission Sequence Number*) à chaque fragment de données utilisateur. Le protocole SCTP utilise une procédure d'acquittement sélectif. La retransmission de paquet est conditionnée par les procédures de contrôle de congestion décrites ultérieurement dans ce chapitre.

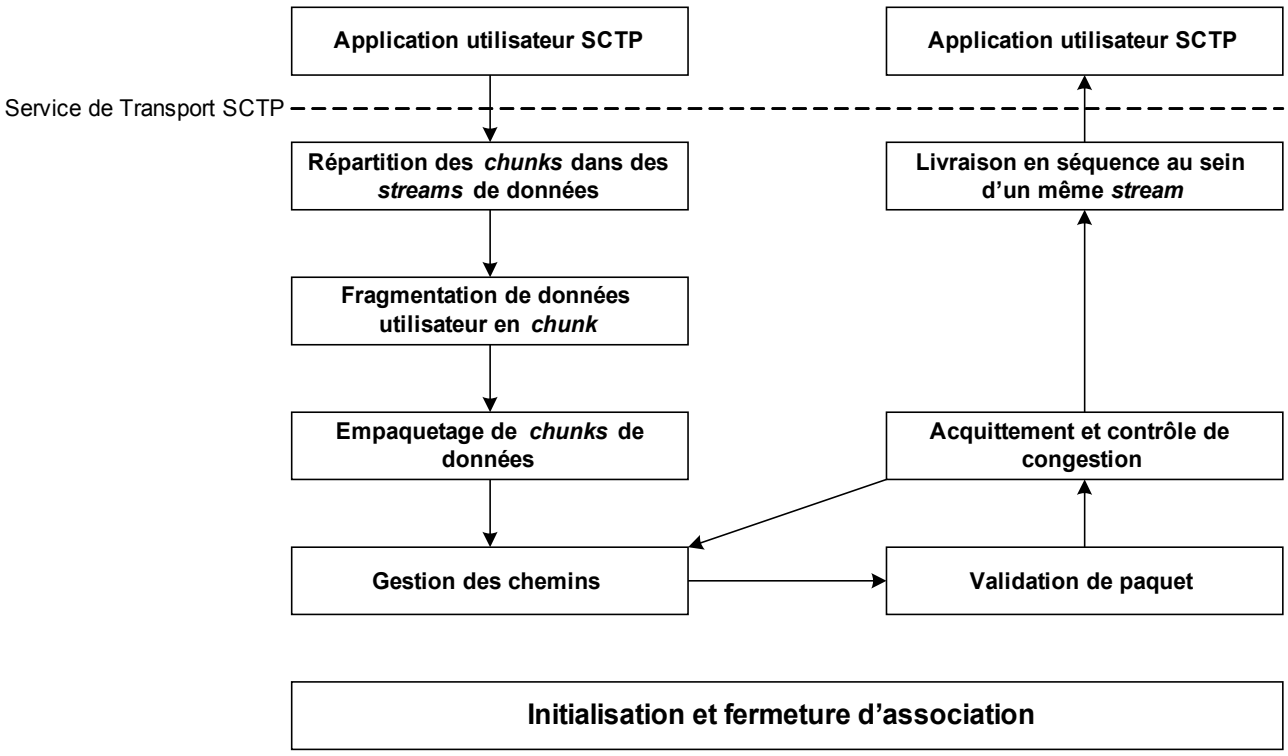
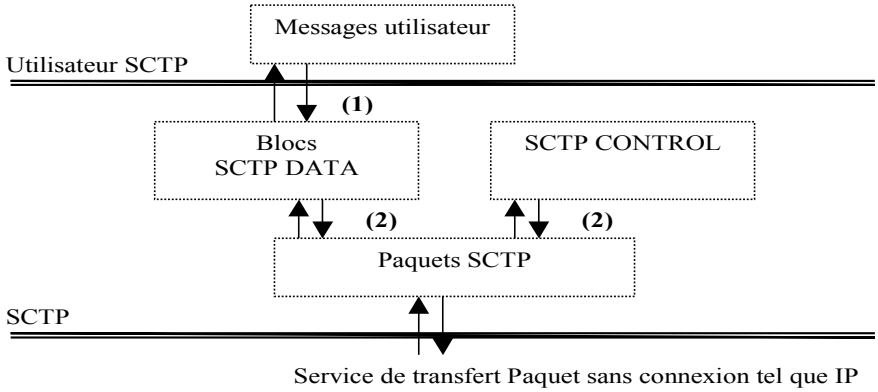



**Figure 2.2.** Fonctions du service de transport SCTP [RFC2960]

1 Si on note @p : port d'un nœud SCTP alors on appelle socket de ce nœud la paire (@p, @IP) où @IP fait partie d'un ensemble d'adresses IP qui constituent les adresses de transport de ce nœud

### 2.3.2 Format générale d'un paquet SCTP

Un paquet SCTP (cf. figure 2.4), constituant la charge utile d'un paquet IP, est constitué d'un entête commun suivi d'un ou plusieurs *chunks*. Les *chunks* étant des blocs de données de taille variable qui peuvent appartenir à des *streams* différents. Chaque *chunk* (cf. figure 2.5) peut contenir aussi bien des informations de contrôle (nous parlons alors de *chunk* de contrôle) que des données utilisateur (nous parlons alors de *chunk* de données (*Data*)) (cf. figure 2.3). Les *chunks* de contrôle précèdent toujours des *chunks* de données dans un paquet SCTP.




(1) Lors de la conversion des messages utilisateur en chunks de données, un nœud SCTP fragmentera les messages de taille supérieure à celle du *MTU Path* en plusieurs chunks de données. Le récepteur des données rassemblera le message fragmenté à partir des chunks de données avant de les livrer à l'utilisateur.

(2) Plusieurs chunks de données et de contrôle peuvent être rassemblés par l'émetteur dans un seul paquet pour qu'il soit transmis ultérieurement (*bundling*), tant que la taille du paquet final n'excède pas le *MTU Path*.

**Figure 2.3.** Transfert de données utilisateur [RFC2960]

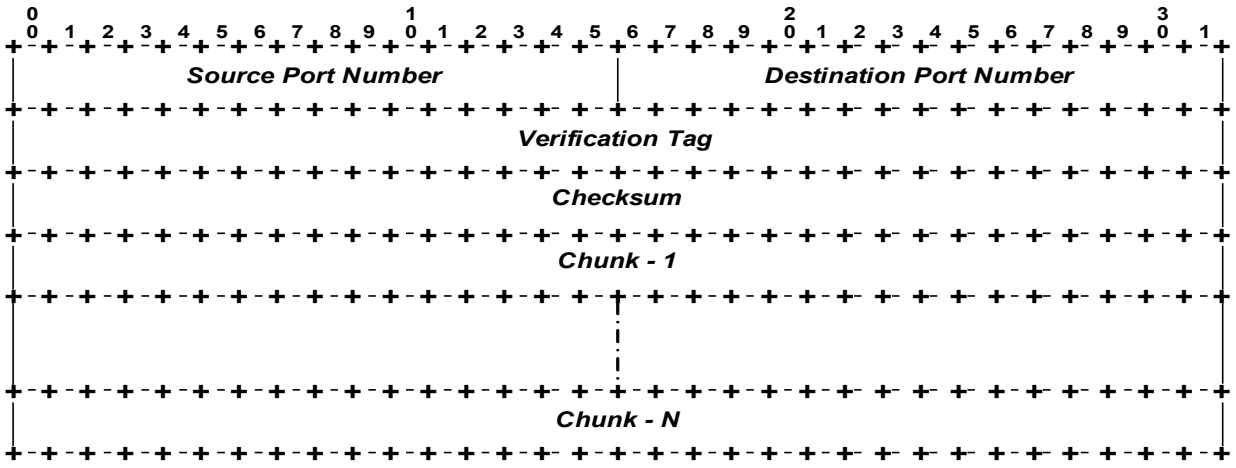



**Figure 2.4**.Format du Paquet SCTP[RFC2960].

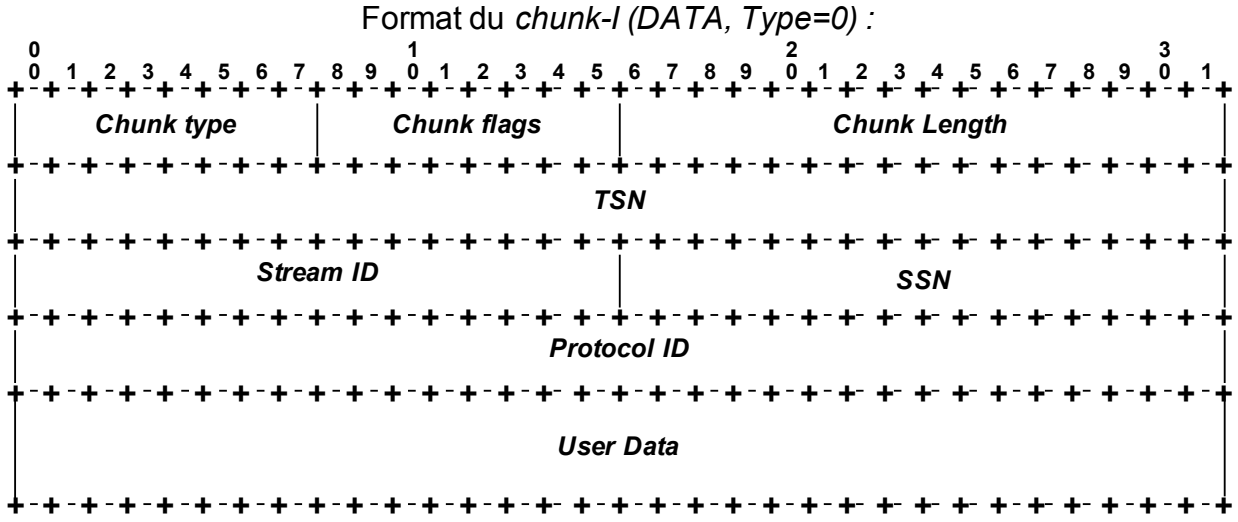



**Figure 2.5**.Format d'un *chunk* SCTP[RFC2960].

L' entête commun d'un paquet SCTP est de taille 12 octets. Pour la détection des erreurs de transmission chaque paquet SCTP est protégé par un *Checksum* de taille 32 bit (Suivant l'algorithme Adler-32). Le *checksum* sert pour la détection des erreurs. Il est plus robuste que les 16 bits de TCP et UDP [RAN 01]. L'entête commun contient également un champ appelé «*Verification Tag*» de taille 32 bit. Ce champ identifie l'association au niveau d'un noeud SCTP (il identifie l'émetteur). Le *Verification Tag* est choisi aléatoirement[2] par chaque noeud SCTP lors de l'ouverture de l'association. L'entête commun est suivi des différents *chunks* de contrôle et de données.

Chaque *chunk* est formé également d'un entête suivi de la charge utile de longueur variable. Il s'agit des données transmises de l'émetteur vers le récepteur. L'entête de chaque *chunk* contient les champs suivants :

- ***Chunk Type*** (8 bits) : permet de déterminer le type de la charge utile du *chunk* (données ou informations de contrôle...),
- ***Chunk Flag*** (8 bits) : utilisé différemment selon le type de *chunk.* À ce niveau le champ *Flag* est plus précis c'est qu'il contient des indicateurs du type de livraison en séquence ou dans le cas de séquencement de données non requis. Le tableau 2.2 résume les états de fragmentation possibles pour un message utilisateur en fonction des bits du champ *Flag*. De plus ce champ contient des bits réservés (*Reserved* 5bits) qui doivent être mis à "0" et ignorés par le récepteur.

2 Il est important que la valeur de INITIATE-Tag soit aléatoire afin de se protéger contre les attaques de type «*man in the middle*» et «*sequence number*». De plus, la valeur du *Verification-Tag* utilisée par un nœud SCTP pour une association donnée ne doit pas changée durant toute la durée de vie de celle ci. Une nouvelle valeur du *Verification-Tag* doit être utilisée chaque fois qu'un nœud termine puis établisse une association avec le même nœud.

Tableau 2.2. Description des bits du champ *chunk flag.*

| ***U (1 bit): Unordered*** | ***B (1 bit): Begining*** | ***E (1 bit): Ending*** | ***Description*** |
|---|---|---|---|
| 1 | X | X | Ce bit indique au récepteur d'ignorer le numéro de séquence du *chunk* |
| 0 | 1 | 0 | Premier *chunk* du message fragmenté |
| | 0 | 0 | *Chunk* intermédiaire du message fragmenté |
| | 0 | 1 | Dernier *chunk* du message fragmenté |
| | 1 | 1 | Message non fragmenté |

- ***Chunk Length*** (16 bits) : indique la taille en octets des données transmises. Ce paramètre représente la taille du *chunk* de données en octets incluant les champs *chunk type, chunk flags, chunk length* et *chunk value*. Si le *chunk* ne contient pas des informations à transmettre (i.e le champ *chunk value* est de longueur nulle), le champ *length* sera affecté à 4 [RFC2960]. Le champ *chunk length* ne compte pas les bits de bourrage (*padding*). En effet, la longueur d'un *chunk* de données doit être un multiple de 4 octets. Si elle ne l'est pas, l'émetteur doit remplir le *chunk* avec des zéros (en octets) [RFC2960] et ce bourrage n'est pas inclut dans le champ *chunk length*. L'émetteur ne peut jamais remplir plus que 3 octets. Le récepteur doit ignorer les octets de bourrage.

L'entête n'est pas suivi uniquement de charge utile à transmettre, mais aussi des champs de contrôle qui varient en fonction du type de *chunk.* Dans ce qui suit nous abordons seulement le type 0 relatif au *chunk Data* (cf figure 2.5), les autres types sont détaillés dans [RFC2960] et [RQX 02]. Les champs de contrôle sont principalement :

- Le numéro de séquence **TSN** (*Transmit Sequence number*) sur 32 bits : numéro de séquence des fragments. Le TSN permet de reconstruire un message fragmenté.
- L'identité du flot (***stream ID***) (sur 16 bits) : permet d'identifier un flot (*stream*). Ce qui offre la possibilité d'avoir plusieurs *streams* dans un même paquet SCTP.
- Le numéro de message **SSN** (*Stream Sequence Number*) dans le flot, sur 16 bits : il s'agit du numéro de séquence du *chunk* dans le *stream*. Le SSN permet le réordonnancement des données par le récepteur.
- Le champ **Protocol_ID**, codé sur 32 bits, sert à indiquer le type d'information contenue dans les *chunks* de données. C'est une couche non utilisée par SCTP mais par la couche application.

Enfin la charge utile est contenue dans le champ *User Data* . C'est un champ de taille variable, il s'agit des données appartenant au *stream* échangées entre deux noeuds SCTP (émetteur/récepteur).

### 2.3.3 Établissement d'une association

La procédure d'ouverture d'une association SCTP tel qu'elle est décrite dans [RFC2960] est donnée par la figure 2.6.

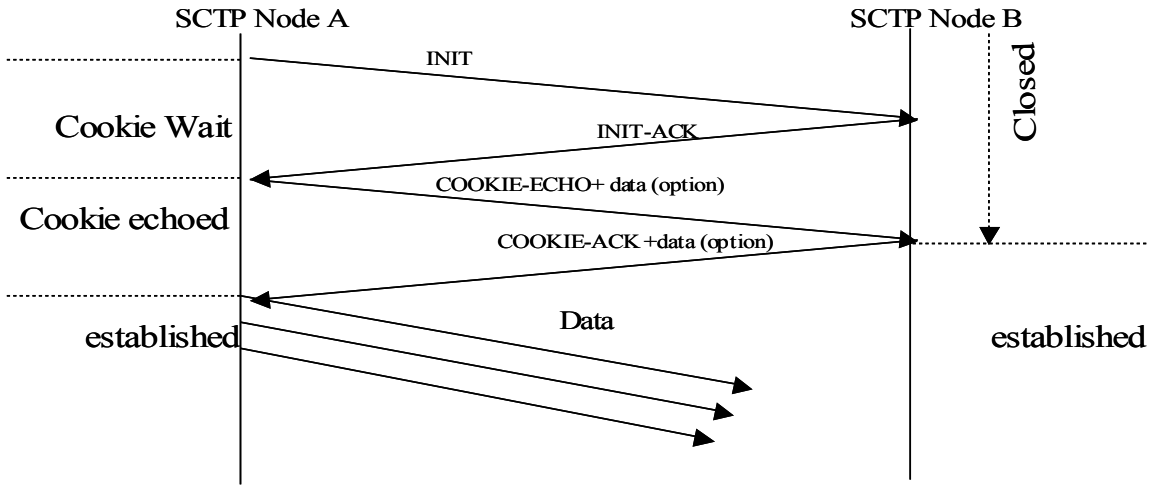



**Figure 2.6.** Phase d'initialisation : échange quadruple [RFC2960].

L'établissement d'une association s'effectue en quatre temps (alors que en 3 temps en TCP ). Cela peut paraître comme inconvénient pour le temps mis pour le déclenchement, mais des données peuvent être transmises au 3ième et 4ième temps du «*4-way startup handshake*» [RFC2960]. De plus, ce mécanisme permet au serveur SCTP de se protéger de plusieurs types d'attaques. Dans ce qui suit nous décrivons les quatre étapes de l'établissement d'une association SCTP entre un noeud A (client) et un noeud B (serveur).

**Étape n°1**

L'association étant à l'état **CLOSED** du côté du serveur (le noeud B) qui attend un message du nœud *A* avec un *chunk* INIT, dont le format est donné par la figure 2.7. Ce *chunk* signale la requête qu'un client (noeud A) demande la création d'une association. Le client A envoie un *chunk* INIT à destination du serveur B et déclenche un temporisateur lui permettant de réémettre ce paquet (contenant le *chunk* INIT) en l'absence de réponse. Ensuite le client A se met en attente du paquet de réponse contenant le *cookie*.

| Type=1 | Chunk flags | Chunk length |
|---|---|---|
| Initiate Tag | | |
| Advertised Received Window Credit (a_rwnd) | | |
| Number of Outbound Streams | Number of Inbound Streams | |
| Initial TSN | | |
| Optional/Variable –Length Parameters | | |

**Figure 2.7.** Format du *Chunk* INIT [RFC2960]

Dans ce *chunk*, le client *A* copie son *Verification Tag* (nous le notons par Tag-A dans la suite de ce chapitre) dans le champ *Initiate Tag* du *chunk* INIT. Le Tag-A est un nombre choisi aléatoirement par le client *A* entre 1 et $(2^{32}-1)$. Après avoir émis le *chunk* INIT, le client *A* déclenche le compteur T1-INIT (*Timer*) et l'association, du côté client A, aura un état **COOKIE-WAIT**. Il est important que la valeur initiale du Tag (*Initiate-Tag*) soit attribuée aléatoirement afin de se protéger contre les attaques de type «*man in the middle*» «*sequence number*» ([RFC2960]). De même pour le champ *initial TSN*.

Deux champs de 16 bits chacun (*Number of Inbound Streams* et *Number of Outbound Streams*) servent à indiquer le nombre maximal de *streams* entrants et sortants que le client et le serveur peuvent supporter au niveau de l'association en cours d'établissement.

**Étape n°2**

Une fois le *chunk* INIT reçu, le serveur B génère un *cookie* regroupant les informations permettant d'établir la connexion avec le client A. Ensuite le serveur B crée une signature numérique [RQX 02] du *cookie* appelée MAC (*Message Authentication Code*). Le serveur B inclut le MAC au cookie et envoie l'intégralité au client A dans un *chunk* INIT-ACK dont le format est donné par la figure 2.8. L'adresse IP de destination, utilisée par le serveur B, doit être celle de l'adresse IP source du *chunk* INIT. L'association garde l'état CLOSED du côté serveur B.

| Type=2 | Chunk flags | Chunk length |
|---|---|---|
| Initiate Tag | | |
| Advertised Received Window Credit (a_rwnd) | | |
| Number of Outbound Streams | Number of Inbound Streams | |
| Initial TSN | | |
| Optional/Variable –Length Parameters | | |

**Figure 2.8.** Format du *Chunk* INIT-ACK[RFC2960]

Dans sa réponse, le serveur *B* doit recopier la valeur de Tag-A dans le champ *Verification-Tag* et fournir son propre *Verification-Tag* (Tag-B) au niveau du champ *Initiate Tag*. Le *chunk* INIT-ACK contient (dans le champ *State COOKIE Parameter*) en plus du *cookie* et du MAC, une indication sur l'instant de création du *cookie*, sa durée de vie, ainsi que toutes les informations qui lui sont nécessaires afin d'établir l'association.

Pour générer un *cookie* certaines étapes doivent être considérées par le serveur B [RFC2960] :

- Créer une association TCB (*Transmission Control Block*) en utilisant les informations du *chunk* INIT reçu et du *chunk* INIT-ACK à émettre,
- Dans le TCB on conserve la date de création du *cookie* et sa durée de vie contenues dans le paramètre «*valid cookie life*»,
- A partir du TCB, on collecte un minimum d'informations utilisé pour recréer le TCB, et on génère un MAC qui est un hash du *cookie* chiffré avec la clé privée du serveur B,
- Générer le *cookie* en combinant ces informations et le MAC résultant.

Après l'émission du *chunk* INIT-ACK avec les paramètres du *cookie*, le serveur B doit supprimer le TCB et toute information liée à la demande d'établissement de la nouvelle association avec le client A. Toutes les données nécessaires à la connexion étant envoyées dans le *cookie,* aucun *buffer* de mémorisation n'est alloué à l'association tant que la connexion n'est pas totalement établie. Cet aspect protège le protocole SCTP de certaines attaques de type «*SYN Attacks*»[INW03].

**Étape N°3**

Après la réception de INIT-ACK du serveur B, le client A doit arrêter son compteur T1-INIT *Timer* et quitter l'état COOKIE-WAIT. Le client A doit alors :

a)- Émettre le cookie qu'il a reçu dans un *chunk* COOKIE-ECHO (cf. figure 2.9),

b)- Déclencher son compteur T1-COOKIE *Timer* et

c)- Entrer dans l'état de **COOKIE-ECHOED**.

| Type=10 | Chunk flags | Chunk length |
|---|---|---|
| Cookie | | |

**Figure 2.9.** Format du *chunk* COOKIE-ECHO [RFC2960]

A ce stade de la procédure d'initialisation de l'association, le serveur B vérifie la signature électronique MAC du *cookie* contenue dans le *chunk* COOKIE-ECHO venant du client A, et qui a été retournée telle quelle. Cette vérification permet au serveur B de contrôler qu'il s'agit bien du *cookie* qu'il l'a déjà envoyé au client A et que les données de connexion n'ont pas été modifiées.

Le client A, de son côté, peut ajouter au paquet envoyé, contenant le *chunk* COOKIE-ECHO, des *chunks* de données mais il doit respecter l'ordre des *chunks* dans le paquet (les *chunks* de contrôle sont placés les premiers devant les *chunks* de données). Cependant, le client A ne doit émettre aucun

autre paquet vers le serveur B jusqu'à la réception de la confirmation d'établissement de l'association de la part du serveur B.

**Étape N°4**

Suite à la réception du *chunk* COOKIE-ECHO, le serveur B doit répondre avec un *chunk* COOKIE-ACK (cf. figure 2.10) après la construction du TCB. Le serveur B, après vérification du *cookie* et du MAC, signale au client A que l'association est établie par l'envoie d'un paquet contenant un *chunk* COOKIE-ACK et l'association, du côté serveur, passe à l'état **ESTABLISHED**. Un *chunk* COOKIE-ACK peut être empaqueté avec des *chunks* de données ou de contrôle (SACK *chunks* par exemple).

Le client A à la réception d'un *chunk* COOKIE-ACK, conclut l'ouverture de l'association SCTP et informe son ULP (*Upper Layer Protocol*) du succès de l'établissement de l'association avec une primitive C*ommunication UP Notification*[3]. C'est à partir de cet instant que le client A commence à émettre des paquets à destination du serveur B. L'association du côté client A passe donc à l'état **ESTABLISHED**.

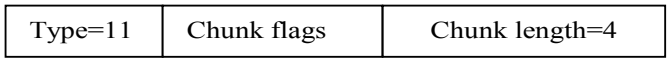



**Figure 2.10.** Format du *chunk* COOKIE-ACK [RFC2960]

**Remarques :**

- Lorsque le TCB est crée, chaque point terminal doit affecter son *Cumulative TSN Ack Point* interne à la valeur de son *Initial TSN* transmis auquel on retranche 1 [RFC2960].
- Les paquets SCTP contenant les *chunks* COOKIE-ECHO et COOKIE-ACK peuvent tous comporter des *chunks* DATA. Ces données ne seront traitées par le serveur que si l'association est établie.
- Si un nœud SCTP recevant soit un *chunk* INIT, INIT-ACK ou COOKIE-ECHO décide de ne pas établir la nouvelle association suite à un manque de paramètres obligatoires dans les *chunks* INIT ou INIT-ACK reçus, ou à des valeurs de paramètres invalides ou à un manque de ressources locales, il doit répondre par un *chunk* ABORT (cf. figure 2.11). Il doit spécifier aussi la cause de ce message ABORT, en incluant les paramètres causant l'erreur dans le *chunk* ABORT (par exemple le type des paramètres obligatoires manquants).

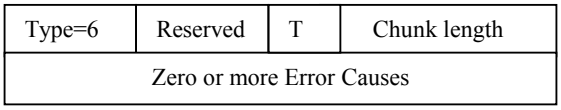



**Figure 2.11.** Format du *Chunk* ABORT [RFC2960]**.**

3*Communication UP Notification* : est utilisée quand SCTP devient prêt à émettre/recevoir des messages utilisateur, ou lorsque une communication perdue vers un point terminal est rétablie.

### 2.3.4 Terminaison d'une association

Pour accomplir sa fiabilité, SCTP doit assurer une procédure de fermeture de connexion. La terminaison normale d'une association SCTP est basée sur une procédure à trois temps (cf. figure 2.12). Cette fermeture normale de l'association est similaire à TCP à la différence que SCTP ne supporte pas l'état de «semi-connecté[4]» [SHA 04]. Chaque nœud SCTP attend la confirmation de la réception des *chunks* de données en cours de transmission avant de libérer l'association. Il existe dans SCTP une autre manière de terminer une association entre le client et le serveur, c'est une méthode plus brutale. Il s'agit d'une procédure ABORT, dont un exemple est présenté ci-dessus, qui signale uniquement qu'un des noeuds terminaux de l'association s'est retiré. Cette fermeture brutale se produit lorsqu'un arrêt immédiat est requis par les applications (situation d'exception, occurrence d'erreurs irrécupérables).

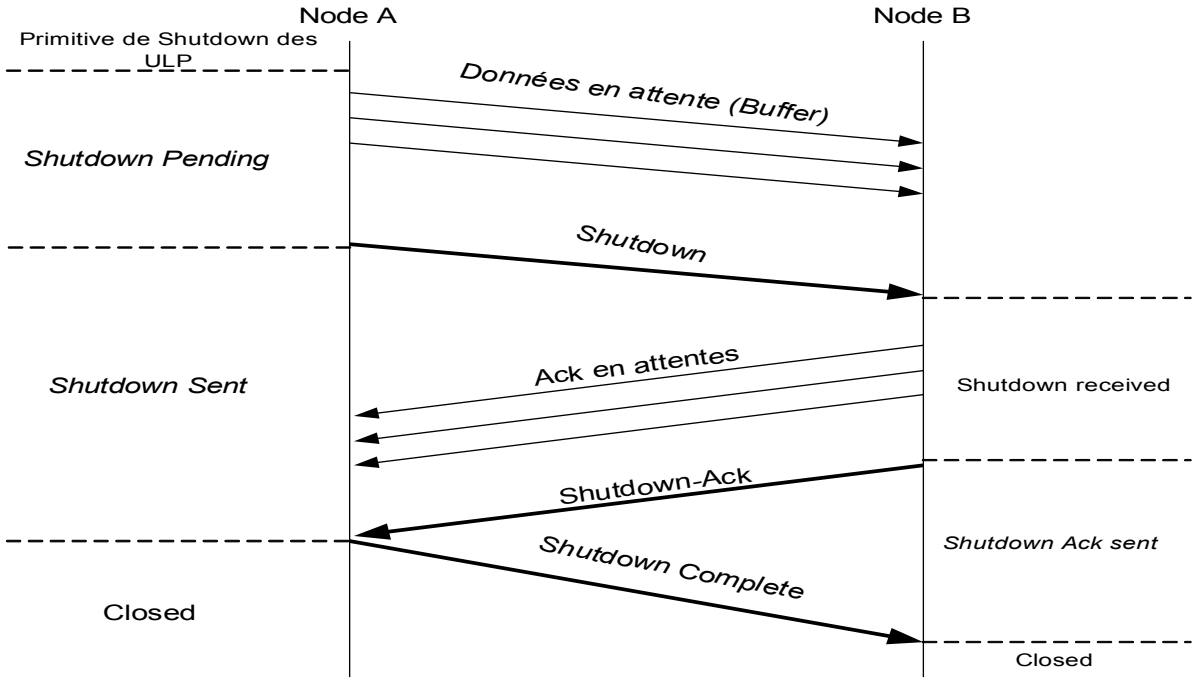



**Figure 2.12.** Terminaison d'une association**.**

**1ère Phase :**

Le client A désire mettre fin à l'association suite à la réception d'une primitive SHUTDOWN de sa couche supérieure. À cet instant l'association du côté client A change d'état en **SHUTDOWN–PENDING**. Le client A n'accepte plus de données à émettre de sa couche supérieure et assure le transfert de tous les messages qui sont stockés dans le *buffer* à destination du serveur *B* avant d'envoyer le *chunk Shutdown* vers le serveur *B*. Le client A vérifie enfin que tous les acquittements

4 SCTP ne supporte pas la fonction d'une connexion «*half-open*» comme TCP, où un noeud peut indiquer qu'il n'a plus de données à émettre, mais l'autre nœud peut continuer à émettre des données. SCTP suppose qu'une fois la procédure d'arrêt est déclenchée, les deux parties stoppent l'émission de nouvelles données à travers l'association.

des paquets SCTP précédemment envoyés ont été reçus. Cependant, le client A peut retransmettre des données qui ont été perdues et signalées par le serveur B et l'association du côté du client passe à l'état **SHUTDOWN-SENT**.

**2ème Phase :**

Ensuite, le client *A* doit émettre un *chunk* SHUTDOWN (cf. figure 2.13) vers son nœud de destination (le serveur B). Le client A précise dans ce *chunk* le dernier TSN reçu du serveur et également les paquets perdus. Lors de l'émission du SHUTDOWN le client *A* doit déclencher un compteur *T2-shutdown Timer* . Dans le cas où ce compteur expire, le message SHUTDOWN doit être réémis avec la mise à jour du dernier TSN reçu en séquence. À la réception d'un *chunk* SHUTDOWN l'association du côté du serveur B change d'état en **SHUTDOWN-RECEIVED**. À son tour le serveur B n'accepte plus de nouveaux paquets en provenance du client A et arrête de transmettre de nouvelles données de ses couches supérieures (ULP). Le serveur B fini par émettre les données stockées dans son buffer de transmission et vérifie que tous les paquets précédemment transmis ont été reçus au moyen de la valeur de TSN fournie dans le *chunk* SHUTDOWN. Pour chaque paquet reçu, le client A acquitte les paquets reçus par un *chunk* SACK, envoie un *chunk* SHUTDOWN et réinitialise son temporisateur.

| Type=7 | Chunk flags | Length =8 |
|---|---|---|
| Cumulative TSN Ack | | |

*Chunk flags* : 8 bits, mis à zéro lors de la transmission et ignoré en réception

*Length* : 16 bits (*unsigned integer*), indique la longueur du paramètre mis à 8,

*Cumulative TSN Ack* : 32 bits (*unsigned integer*), ce paramètre contient le TSN du dernier *chunk* reçu en séquence avant toute pertes (*gap)*.

**Figure 2.13.** Format du *Chunk Shutdown* [RFC2960].

Une fois que le serveur B a atteint l'état SHUTDOWN-RECEIVED, il doit ignorer toute demande de *shutdown* de l'association provenant de son ULP.

Enfin, une fois tous les paquets en attente envoyés et reçus par le client A, le serveur B émet un *chunk* SHUTDOWN-ACK (cf. figure 2.14) et déclenche son temporisateur *T2-shutdown Timer*. C'est à partir de là que l'association du côté serveur change d'état en **SHUTDOWN-ACK-SENT**. Si son temporisateur expire, le serveur B doit réémettre le *chunk* SHUTDOWN-ACK.

| Type=8 | Chunk flags | Chunk length=4 |
|---|---|---|

**Figure 2.14**.Format du *Chunk* Shutdown-ack [RFC2960]**.**

**3ème Phase** :

Suite à la réception d'un SHUTDOWN-ACK, le client A, s'assure qu'il n'a pas eu de perte de données de côtés des deux intervenants dans l'association, arrête son temporisateur *T2-shutdown Timer* et envoie un *chunk* SHUTDOWN-COMPLETE (cf. figure 2.15) au serveur B. Le client A peut alors quitter l'association et supprimer toute information liée à la connexion et l'état de l'association passe définitivement à **CLOSED**.

De son côté le serveur B est garanti de la bonne réception des données au cours de l'association courante en recevant un *chunk* SHUTDOWN-COMPLETE. À cet instant, le serveur *B* vérifie qu'il est dans l'état SHUTDOWN-ACK-SENT, si ce n'est pas le cas le *chunk* sera ignoré. S'il est bien dans l'état SHUTDOWN-ACK-SENT, le serveur B arrête son *T2-shutdown Timer* et supprime l'association (et ainsi l'association entre dans l'état **CLOSED**).

| Type=14 | Reserved | T | Length =4 |
|---|---|---|---|

*Reserved* :7 bits (mis à 0 lors de transmission et ignoré en réception)

T : 1bit, T=0 si l'émetteur a un TCB qui a été détruit sinon T=1.

**Figure 2.15.** Format du *Chunk* SHUTDOWN-COMPLETE[RFC2960]**.**

## 2.3.5 Transfert des données utilisateurs (gestion des acquittements)

D'après la RFC2960 [RFC2960] la transmission de données en SCTP doit se produire uniquement lors des états, décrits précédemment, suivants :

- ESTABLISHED
- SHUTDOWN-PENDING
- SHUTDOWN-RECEIVED

L'exception de ceci, est lorsque les *chunks* de données sont autorisés d'être transmis avec un *chunk* de contrôle de type COOKIE-ECHO quand l'association est dans l'état COOKIE-WAIT. Un récepteur SCTP doit être capable de recevoir au minimum un paquet SCTP de taille 1500 octets, c'est à dire qu'un nœud SCTP ne doit pas indiquer moins de 1500 octets dans son envoie initial (a-rwnd = 1500 *byte*) de INIT ou INIT-ACK [RQX02]. Pour assurer l'efficacité de transmission, SCTP définit des mécanismes pour l'empaquetage des messages utilisateurs de petites tailles et la fragmentation de ceux de tailles importantes (cf page 67 de [RFC2960]).

Le protocole SCTP, étant «*TCP Friendly*» [RFC2960], possède des mécanismes de contrôle de flux et de congestion. Il permet, grâce à sa caractéristique de *Multistreaming*, l'envoie de données correspondant à plusieurs flots (*streams*) dans un même paquet grâce à un mécanisme d'identification de *stream* dans chaque *chunk* comme c'est décrit précédemment dans le paragraphe 2.3.2. Lors de transfert de données, si un message utilisateur est de taille supérieure au PMTU (*Path Maximum Transmission Unit*) d'une association, il sera fragmenté en différents *chunks*. Les *chunks* formant un même message ont des TSNs successifs, et sont identifiés par le même SSN. L'identification de l'ordre des différents morceaux d'un message utilisateur est assurée au moyen

des flags B/E (premier *chunk*, les *chunks* intermédiaires, dernier *chunk*). Pour la transmission des données, SCTP assure un acquittement sélectif au moyen du *SACK chunk.* Ce dernier est utilisé par le récepteur SCTP qui l'envoie en retour afin d'acquitter les *DATA chunks* reçus. Le *SACK chunk* permet à l'émetteur d'identifier la fenêtre avertie de réception, les *chunks* qui ont été perdus, les *chunks* qui ont été reçus dupliqués, etc. Ainsi toutes les informations nécessaires à d'éventuelles retransmissions ou à la gestion de *streams*.

Lorsque un *chunk* de données (*DATA chunk*) (cf. figure 2.16) atteint un récepteur SCTP, ce dernier doit envoyer un SACK (cf. figure 2.16) afin d'acquitter sa réception. Un champ *Cumulative TSN Ack* est utilisé pour acquitter la réception de tous les *chunks* reçus en séquence sans erreurs. Ce champ indique jusqu'à quelle valeur de TSN des données ont été correctement reçues. Un SACK contient aussi les *Gap Ack chunks* qui sont utilisés pour acquitter des plages de *chunks* de données reçues séparément. Le *Gap Ack chunk Start* et *Gap Ack chunk End* sont codés sur 16 bits. Ils indiquent la position relative (par rapport au *Cumulative TSN Ack)* des différentes plages de *chunks* isolés correctement reçus. Le SACK comporte également une indication sur les TSNs dupliqués, donnée par les champs *Number of duplicate TSNs* et *Duplicate TSN*. Le champ *Number of duplicate TSNs,* codé sur 16 bits, indique le nombre de *chunks* dupliqués. Le champ *Duplicate TSN* (sur 32 bits) donne le nombre de fois qu'un TSN a été reçu de façon dupliquée à partir du dernier SACK transmis. Un TSN dupliqué apparaît dans le SACK autant de fois que le récepteur l'a reçu de façon redondante.

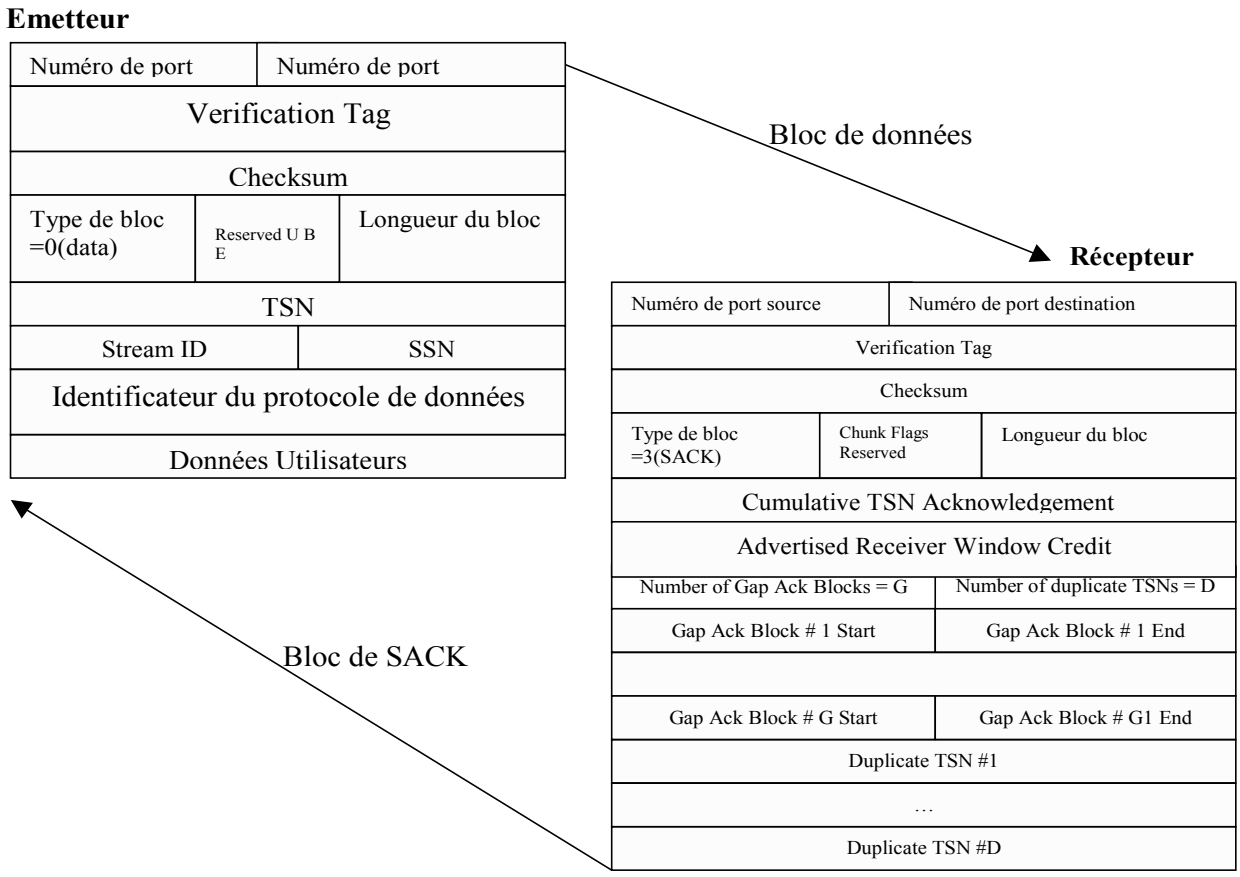



**Figure 2.16.** Transmission de données [RFC2960]**.**

L'exploitation des données transmises par le *chunk* SACK, permet à l'émetteur SCTP de réémettre les *DATA chunks* qui n'ont pas été reçus et qui sont identifiés par leur numéro TSN. Cette retransmission s'effectue après l'expiration du temporisateur ou la réception de 4 *chunks SACK* indiquant la perte du même *DATA chunk* (en TCP c'est 3 non pas 4) [RFC2960].

Nous constatons que SCTP acquitte les paquets reçus non pas les octets comme c'est la cas de TCP. S'il y a eu un problème lors de la transmission des données, et qu'une retransmission de ces dernières a eu lieu, les *DATA chunks* hors séquence sont acquittés à l'aide du champ *Gap Ack Blocks*, qui permet l'acquittement d'une ou plusieurs séquences de TSN. Notons que comparativement à TCP, SCTP permet d'avoir de plus de *GAP Blocks* dans un *chunk* SACK [SHA05]. Il offre un espace réservé de taille $2^{16}$ octets comme le spécifie le champ *chunk length*.

**Exemple illustratif pour *Gap Ack chunks* :**

Supposant qu'un point terminal SCTP (serveur) vient de recevoir les paquets de données suivants (cf. figure 2.17) :

| TSN1 | TSN2 | Gap1 | TSN4 | TSN4 | Gap2 | TSN6 |
|---|---|---|---|---|---|---|

**Figure 2.17.** Numéros des paquets SCTP reçus.

Le SACK à émettre sera le suivant (cf. figure 2.18) :

| Cumulative TSN Ack = 2 | |
|---|---|
| a_rwnd = X | |
| G= 2 | D= 1 |
| # 1 start =2 | # 1 end =2 |
| # 2 start =4 | # 2 end =4 |
| TSN = 4 | |

**Figure 2.18.** Le *chunk SACK* transmis.

Ce SACK indique la bonne réception des paquets dont le TSN vérifie l'une des conditions suivantes :

- $TSN \leq$ cumulative TSN Ack : TSN1 et TSN2,
- $n^\circ 1\ start + cumulative\ TSN\ Ack \leq TSN \leq n^\circ 1\ end + cumulative\ TSN\ Ack$ soit $2+2 \leq TSN \leq 2+2$ : TSN4,

- $n \circ 2\ start + cumulative\ TSN\ Ack \leq TSN \leq n \circ 2\ end + cumulative\ TSN\ Ack$ soit $2+4 \leq TSN \leq 2+4$ : TSN6.

Les TSN 3 et TSN 5 réfèrent aux paquets perdus. Ce SACK indique également la réception dupliquée du paquet dont le TSN est égale à 4.

## 2.4. Contrôle de Congestion en SCTP

SCTP utilise les mêmes modes de contrôle de congestion que TCP en se basant sur la norme [RFC2581], à savoir le *slow start* et le *congestion avoidance*. La principale différence avec TCP réside dans le contrôle adapté au *multihoming*. Les paramètres de congestion étant gérés indépendamment du chemin qu'il soit primaire ou alternatif. Le contrôle de congestion dans SCTP est toujours appliqué à l'association entière et non pas à des flots (*stream*) individuels. Pour contrôler les *chunks* reçus, SCTP attribue un TSN à chaque *chunk* de données utilisateur. Ce TSN est indépendant de tous les SSNs attribués au niveau du flot de données (*stream*).

Dans d'un réseau, de manière générale, la congestion peut se produire à deux niveaux : au niveau du récepteur (taille du buffer de stockage assez faible) ou au niveau du réseau (bande passante de la liaison saturée). Dans le premier cas, le problème est résolu avec le champ *Advertised Receiver Window Credit (a_rwnd)*, qui se trouve dans les *chunks* de contrôle de type INIT, INIT-ACK et SACK. Le deuxième cas, plus complexe à gérer, est résolu à l'aide de plusieurs algorithmes. Ces algorithmes sont les mêmes que ceux utilisés dans TCP [RFC2960], et utilisent les deux variables suivantes pour chaque adresse de réception d'une association :

- *Congestion window* (cwnd) : qui limite le nombre d'octets que le noeud émetteur des données peut avoir en cours de transmission. Il s'agit du nombre d'octets pouvant être transmis sans provoquer la congestion du réseau.
- *Slow Start Threshold* (ssthresh) : c'est une valeur seuil permettant de choisir le bon algorithme de congestion selon les événements du réseau.

Les algorithmes utilisés sont le *Slow Start Algorithm*, le *Fast Retransmit* et le *Fast Recovery* [FRA03]. Nous représentons les différentes phases de contrôle de congestion dans la figure 2.19. Dans la suite de cette section nous focalisons notre étude sur la différence entre le mécanisme de contrôle de congestion en SCTP avec celui de TCP. Malgré qu'ils paraissent avoir un même mécanisme, certaines différences fonctionnelles distinguent le contrôle de congestion dans les deux protocoles. Ensuite nous développons les modes de contrôle de congestion en SCTP.

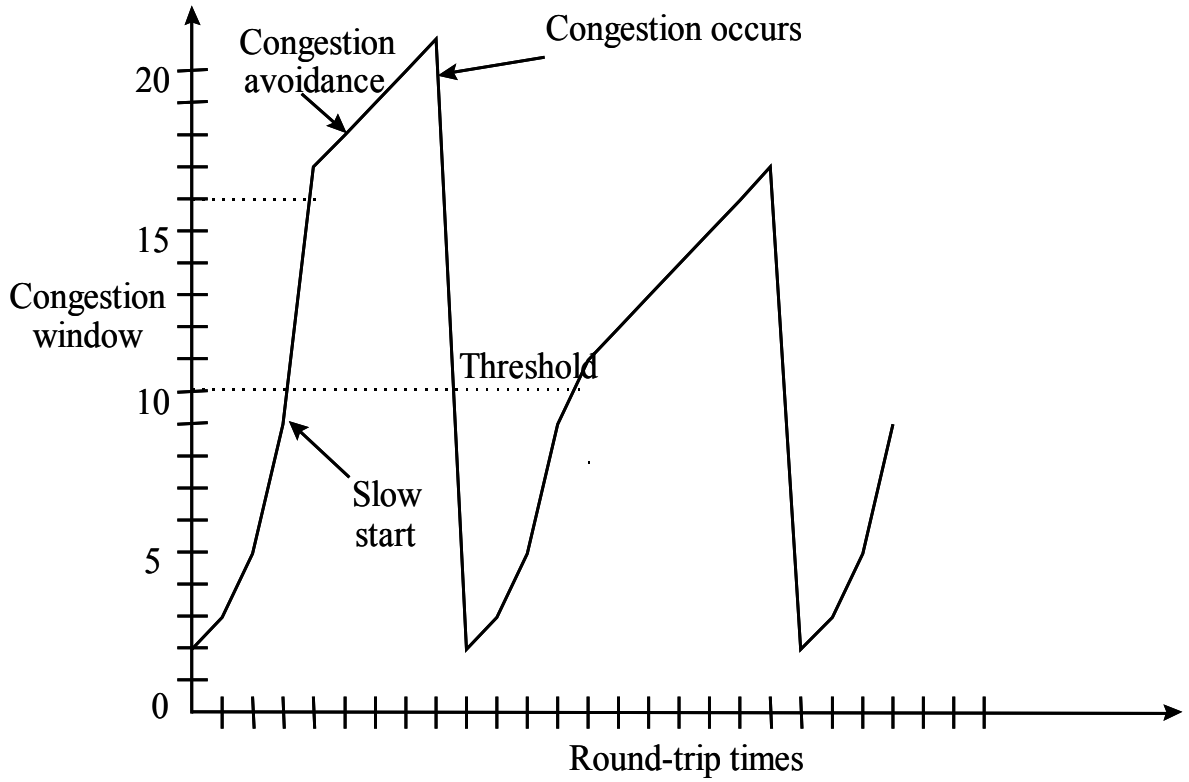



**Figure 2.19.** Différents modes de contrôle de congestion.

### 2.4.1 Différences entre contrôle de congestion en TCP et SCTP

Les algorithmes de contrôle de congestion TCP sont basés sur deux variables *cwnd* et *ssthresh* associées à chaque connexion TCP. SCTP possède une paire (*cwnd, ssthresh*) par adresse de transport SCTP destination (*multihoming* de SCTP).

Dans les spécifications IETF [RFC2960], le protocole SCTP considère les informations supportées par le *chunk* SACK uniquement comme des informations consultatives. En SCTP n'importe quel bloc de données qui a été acquitté par un *chunk* SACK, incluant les données reçues hors séquence, n'est pas considéré comme étant livré (à la couche supérieure) jusqu'à ce qu'il soit acquitté par un *Point Cumulative TSN Ack* (ie. les blocs de données doivent être acquittés par le champ *Cumulative TSN Ack* dans le *chunk* SACK).

Les algorithmes de contrôle de congestion en SCTP, tels qu'ils sont spécifiés par la [RFC2960], considèrent les hypothèses suivantes :

- Un émetteur utilise la même adresse destination jusqu'à l'arrivée d'une instruction de changement d'adresse de la couche supérieure.
- L'émetteur garde pour chacune des adresses destination, vers lesquelles il peut émettre, un ensemble de paramètre de contrôle de congestion (non pas pour chaque paire source-destination mais pour chaque destination). Les paramètres doivent s'effacer si l'adresse n'a pas été utilisée durant une période considérée importante [RFC2960].

- Pour chacune des adresses destination, un nœud SCTP commence la transmission vers cette adresse avec l'algorithme *Slow-Sart*.

TCP garantit la livraison en séquence des données aux couches supérieures, tout au long d'une seule session TCP. Ceci signifie que lorsque TCP remarque un trou (*gap)* dans le numéro de séquence reçu, il attend que ce trou soit recouvert avant la livraison des données aux couches supérieurs. SCTP peut livrer, si le *flag* U est à 1 (cf. tableau 2.2), des données aux couches supérieures même s'il y a un vide dans le TSN. Cette livraison est possible uniquement si les SSNs sont en séquence pour un *stream* particulier et donc que les *chunks* de données perdus font partie d'autres *stream*. Le mécanisme de remontée en hors séquence aux couches supérieures n'affecte pas le paramètre cwnd mais il affecte le calcul de la taille de fenêtre de réception (rwnd).

Comme en TCP, un point terminal SCTP utilise les trois variables de contrôle suivantes pour régler son débit de transmission :

- Le récepteur indique la taille de fenêtre de réception (**rwnd**, en octet), qui est ajustée par le récepteur en se basant sur l'espace de stockage disponible pour les paquets entrants. Cette variable est maintenue pour l'association entière.
- La fenêtre de contrôle de congestion (**cwnd**, en octet) qui est ajustée par l'émetteur en se basant sur les conditions observées du l'état du réseau. Cette variable est maintenue à base d'une adresse par destination. Le cwnd limite le nombre d'octets que l'émetteur de données peut transmettre en attente d'acquittement.
- Le seuil *Slow-Start* (**ssthresh**, en octet) qui est utilisée par l'émetteur pour distinguer les phases de **slow start** et *congestion avoidance*. Cette variable est maintenue pour chaque adresse de destination.

SCTP exige aussi une autre variable de contrôle supplémentaire : ***partial-bytes-acked (pba)***, qui est utilisée durant la phase de *congestion avoidance* afin de faciliter l'ajustement de cwnd. Cette variable comptabilise toutes les données reçues par le récepteur en présence de *chunk* manquants dans la fenêtre de réception.

### *2.4.2 Slow Start*

Le mode *slow start* correspond au mode de démarrage lent. Le déclenchement de transmission dans un réseau, exige au protocole SCTP d'explorer le réseau afin de déterminer la capacité disponible. L'algorithme *Slow Start* est utilisé, pour cette raison, au lancement du transfert ou après la correction de pertes détectées par le temporisateur de retransmission. Une association SCTP, étant en mode *slow start*, doit respecter les règles suivantes :

1- Initialement avant la transmission de données ou après une période de silence (*Idle*) suffisamment importante on doit avoir : **cwnd = 2*MTU,** (*Maximum Transmission Unit*)

2- Le **cwnd** initial, après l'expiration du temps de retransmission ne doit pas être supérieure à **1*MTU**

3- La valeur initial de **ssthresh** peut être arbitrairement élevée (par exemple, les

implémentations peuvent utiliser la taille de fenêtre indiquer par le récepteur)

4- Chaque fois que **cwnd > 0,** le nœud SCTP est autorisé d'avoir cwnd octets de données en attente d'acquittement sur cette adresse de transport.

5- Si **cwnd ≤ ssthresh**, un nœud SCTP doit utiliser l'algorithme *Slow Start* pour augmenter cwnd (supposant que la fenêtre de congestion actuelle est entièrement utilisée). Si un *chunk* SACK entrant incrémente le *Cumulative TSN Ack Point*, cwnd doit être incrémentée par au plus le minimum de :

a- La taille totale des blocs de données précédemment acquittés

b- Le *path MTU* de destination

Dans le cas où le point terminal de réception est *multihomed*, si l'émetteur reçoit un *chunk* SACK qui incrémente son *Cumulative TSN Ack Point*, alors il devrait mettre à jour sa cwnd (ou cwnds) répartie(s) sur les adresses destination vers lesquelles il a transmis les données acquittées. Cependant, si le *chunk* SACK reçu n'incrémente pas le *Cumulative TSN Ack Point*, le point terminal d'émission ne doit pas ajuster la cwnd.

Quand le nœud SCTP ne transmet pas de données sur une adresse de transport donnée (qui correspond à l'adresse primaire et actuellement utilisée comme destination), le **cwnd** de cette adresse doit être ajusté au **max(cwnd/2, 2*MTU)** par **RTO**.

En conclusion, le *slow start* démarre avec une taille de la fenêtre de congestion égale à deux paquets SCTP. La fenêtre de congestion augmente ensuite au fur et à mesure de la réception des acquittements. Au démarrage, l'anticipation se fait sur deux paquets [RFC2960], à chaque acquittement le nombre courant de paquets anticipés est augmenté d'un paquet SCTP, la croissance de la fenêtre d'anticipation sera donc exponentielle. L'objectif de base de cet algorithme est que la vitesse d'émission est égale à la vitesse de réception qui est reflétée par les acquittements. Par contre, après une première absence d'acquittement détectée par un acquittement dupliqué, le mécanisme dit d'évitement de congestion (*congestion avoidance*) se déclenche.

### 2.4.3 *Congestion Avoidance*

Le mécanisme d'évitement de congestion correspond au régime permanent d'une association SCTP. Au cours de cette phase la taille de la fenêtre de congestion est incrémentée de façon linéaire au delà du seuil de détection de congestion (ssthreshold). Le seuil de détection de congestion est déterminé à la détection de perte d'un segment et correspond à la moitié de la taille de la fenêtre de congestion [RFC2581].

L'évolution algorithmique du mécanisme de *congestion avoidance* se résume dans les étapes suivantes :

- Si cwnd > ssthresh, cwnd ← cwnd +1*MTU par RTT, si l'émetteur a "*cwnd*" ou plus d'octets de données en attente d'acquittement pour l'adresse de transport correspondante.

En pratique une implémentation peut assurer cette évolution de la fenêtre de congestion de la façon suivante :

1- **partial-bytes-acked =0,**

2- Chaque fois que **cwnd > ssthresh**, à chaque réception d'un *chunk* SACK on incrémente *Cumulative TSN Ack Point.* Le *partial-bytes-ack* est incrémenté par le nombre total d'octets de tous les nouveaux blocs acquittés par le *chunk* SACK reçu incluant les blocs acquittés par le nouveau *Cumulative TSN Ack* et par les blocs *Gap-Ack,*

3- Si **partial-bytes-ack ≥ cwnd**, et qu'avant l'arrivée du *chunk* SACK l'émetteur a *cwnd* ou plus d'octets de données en attente d'acquittement[5], on augmente cwnd d'un MTU et on remet partial-bytes-acked à $pba = partial.bytes.acked - cwnd$ [RFC2960],

4- Comme en *Slow Start*, quand l'émetteur ne transmet pas de données sur une adresse de transport donnée, le *cwnd* de cette adresse de transport doit être ajusté au max(cwnd/2, 2*MTU) par RTO.

5- Quand toutes les données transmises par l'émetteur ont été acquittées par le récepteur, *partial-bytes-acked* est ré-initialisé à 0.

### 2.4.4 Contrôle de congestion

SCTP comme TCP considère que toute perte est générée par une congestion et diminue donc la taille de la fenêtre de congestion. Ce mécanisme n'est pas adapté aux liens à taux d'erreur élevés, notamment des liens radios GSM, GPRS, EDGE ou WLAN. L'augmentation du taux de perte et celle du délai sont souvent liées. Ce qui constitue le problème que nous supposons résoudre dans le présent mémoire en proposons des modifications du mécanismes de contrôle de congestion pour qu'il s'adapte à ce type d'environnement.

En fait, suite à la détection de pertes de paquets à partir du *chunk* SACK, un point terminal SCTP doit réagir de la façon suivante :

$$ssthresh = max(cwnd/2, 2*MTU)$$

$$cwnd = ssthresh$$

Essentiellement un paquet perdu cause la diminution par la moitié du *cwnd*. Quand le temporisateur *T3-rtx* [RFC2960] expire pour une adresse, SCTP doit exécuter *Slow Start* comme suit :

$$ssthresh = max(cwnd/2, 2*MTU)$$

$$cwnd = 1*MTU$$

Ensuite, SCTP doit s'assurer qu'il n'y a pas plus qu'un paquet SCTP qui soit en cours de transmission pour cette adresse jusqu'à ce que le point terminal SCTP reçoive l'acquittement pour une livraison réussie des données vers cette adresse.

5 C'est à dire avant l'arrivée du *chunk* SACK, la taille des données en cours de transmission est supérieure ou égale à *cwnd.*

### 2.4.5 Processus de retransmission rapide : *Fast Retransmit on Gap Reports*

En absence de perte de données un point terminal SCTP exécute les acquittements retardés. Cependant, chaque fois qu'un point terminal SCTP constate un saut dans le séquencement de TSN qu'il reçoit, il doit commencer à émettre un *chunk* SACK chaque fois qu'un paquet arrive portant des données jusqu'à la correction de ce saut (vide de TSN).

Chaque fois qu'un point terminal SCTP reçoit un *chunk* SACK, indiquant certains TSNs manquants, il doit attendre 3 nouvelles indications de pertes (au moyen des *chunks* SACKs suivants) sur le(s) même(s) TSN(s) avant la prise de l'action en ce qui concerne la retransmission rapide [RFC2960].

Lorsque un TSN est déclaré comme perdu dans le quatrième *chunk* SACK reçu consécutivement, l'émetteur SCTP de données doit :

1- Marquer les *chunks* de données perdus afin qu'ils soient retransmis

2- Ajuster le *ssthresh* et *cwnd* pour les adresses destinations vers lesquelles les *chunks* de données perdus ont été émis selon ce qui est indiqué en 2.4.4,

3- Déterminer combien de *chunks* de données[6] classés pour la retransmission et les mettre dans un seul paquet SCTP tout en respectant les contraintes de Path MTU et de l'adresse de transport de destination vers laquelle le paquet sera envoyé.

4- Déclencher le temporisateur T3-rtx [RFC2960] uniquement si le dernier *chunk* SACK acquitte le plus petit TSN suspendu envoyé vers cette adresse, ou le point terminal SCTP retransmet le *chunk* de données suspendu transmis vers cette adresse.

Notons qu'avant de faire les ajustements indiquer ci-dessus, si le *chunk* SACK reçu acquitte aussi les nouveaux *chunks* de données et incrémente le *Cumulative TSN Ack Point*, les règles d'ajustement de cwnd définies en 2.4.2 et 2.4.3 doivent être appliquées en premier.

Une implémentation de ce qu'est ci-dessus maintient un compteur pour chaque vide de TSN déclaré par le *chunk* SACK. Ce compteur s'incrémente pour chaque *chunk* SACK reçu consécutivement indiquant le vide de TSN. Après avoir atteindre 4 la procédure de retransmission rapide est déclenchée. Le compteur est alors remis à 0, car le cwnd en SCTP limite indirectement le nombre de TSNs suspendus [RQX02].

Enfin, l'effet du rétablissement rapide (*Fast Recovery*) du TCP est réalisé automatiquement sans aucun ajustement de la taille de fenêtre de contrôle de congestion (pages 90-91 de [RFC2960]).

### 2.4.6 Temporisateur de retransmission : *T3-rtx*

Un point terminal SCTP utilise un compteur de retransmission *T3-rtx* pour assurer la livraison de

6 On tient compte de priorité de retransmission c'est que la priorité est accordée aux *chunks* de données les plus anciens. C'est à dire les *chunks* de données qui ont le plus petit TSN sont prioritaires pour une retransmission.

données en absence de défauts de lien avec son nœud destinataire. La durée de ce compteur est mentionnée comme *RTO* (*Retransmission TimeOut*), c'est le hors temps de retransmission. Quand le nœud destinataire est *multi-homed*, le nœud SCTP calculera le *RTO* de façon indépendante pour chaque adresse de transport de destination du récepteur de l'association.

Le mode de fonctionnement du temporisateur de retransmission est géré par les évènements suivants :

- Chaque fois qu'un *chunk* de données est émis vers n'importe qu'elle adresse (incluant une retransmission), si le compteur *T3-rtx* de cette adresse n'est pas déclenché, on l'enclenche de sorte qu'il expirera au bout du RTO relatif à cette adresse.
- Lorsque toutes les données émises vers une adresse ont été acquittées, on arrête le compteur T3-rtx de cette adresse.
- A la réception d'un SACK acquittant les *chunks* de données avec le plus récent TSN pour cette adresse, on relance le compteur T3-rtx pour cette adresse avec son RTO actuel (s'il y a encore de données à émettre vers cette adresse).
- Si un TSN qui a été, précédemment, acquitté par un *Gap Ack Block* ne figure pas dans le *chunk* SACK suivant, alors T3-RTX de l'adresse destination concernée est déclenché.

### 2.4.7 *Path MTU Discovery:* PMTU *Discovery*

Le MTU (*Maximum Transmission Unit*) est un paramètre indispensable pour limiter l'évolution de la taille de fenêtre de congestion, une procédure d'estimation du MTU s'avère primordiale. Il s'agit du *Path MTU Discovery.*

Le *Path MTU Discovery* permet à un point terminal SCTP de maintenir une estimation du MTU le long d'un chemin Internet donné et fragmente les paquets en transmission dont la taille dépasse le MTU découvert. Il y a 4 façons avec lesquelles SCTP diffère du TCP en ce qui concerne la description d'application du PMTU *discovery* :

1- Les associations SCTP peuvent recouvrir plusieurs adresses, c'est le principe de *multihoming*. Un point terminal SCTP doit maintenir des évaluations de la valeur du MTU séparées pour chacune des adresses de transport du point terminal SCTP de destination.

2- Contrairement à TCP, SCTP n'a pas de notion de «*Maximum Segment Size»*. Par conséquent, le MTU relatif à chaque adresse destination doit être initialisé à une valeur qui ne soit pas supérieure au lien MTU lié à l'interface locale vers laquelle les paquets seront routés.

3- Puisque la transmission des données en SCTP est naturellement structurée en termes de TSNs plutôt qu'en octets (comme c'est le cas en TCP), la transmission de nouveaux datagrammes IP doit avoir le flag DF (*Don't Fragment*) positionné.

4- L'émetteur doit avoir un PMTU par association qui sera le plus petit PMTU découvert pour toutes les adresses de transport du point terminal SCTP destinataire. Lors de fragmentation des messages en plusieurs parties, le PMTU d'association doit être utilisé pour calculer la taille de chaque fragment.

## 2.5. *Multistreaming* et Association

Le protocole SCTP est orienté message alors que TCP est orienté octet. Chaque message est associé à un flux, appelé *stream*. Contrairement à TCP, une même connexion SCTP permet la transmission de plusieurs *streams* (ou encore multiplexage de flux [FRA03]), c'est pour cela qu'on utilise le terme association dans SCTP (cf figure 2.20). Les *streams* ont été définis pour permettre de séparer le contrôle d'erreurs de la remise en séquence des messages aux couches supérieures [RFC2960].

Lors de l'établissement d'une association, le client et le serveur s'informent mutuellement du nombre de *streams* possibles dans chaque direction. Ensuite un numéro de séquence (*Stream Sequence Number*) est attribué aux *chunks* de données émis, indépendamment pour chaque flux. De plus, SCTP différencie le transport fiable avec la livraison ordonnée des données, offrant un choix adapté aux besoins des applications. C'est que certaines applications peuvent seulement avoir besoin d'une remise en ordre partielle des datagrammes alors que d'autres pourraient se satisfaire d'un transfert fiable qui ne garantisse aucun ordre de transmission. En effet, le protocole SCTP introduit une indépendance entre la fiabilité de transmission de données et la livraison de celles-ci aux couches supérieures, contrairement au TCP qui relie le transfert fiable de données avec la livraison ordonnée de celles ci. Chaque *chunk* de données utilise deux niveaux de numérotation [RFC2960] :

- Le TSN (*Transmission Sequence Number*) qui numérote les *chunks* au niveau de l'association. Il est utilisé pour le contrôle de la fiabilité des échanges (détection des *chunks* perdus, acquittements)
- La paire (*Stream ID ,* SSN (*Stream Sequence Number*)) : Le *Stream* ID identifie une application de destination. Le SSN sert à la reconstruction du message utilisateur à partir des *chunks* (si il y a eu fragmentation du message utilisateur en plusieurs *chunks*) et à la remise en séquence des messages à l'application utilisateur.

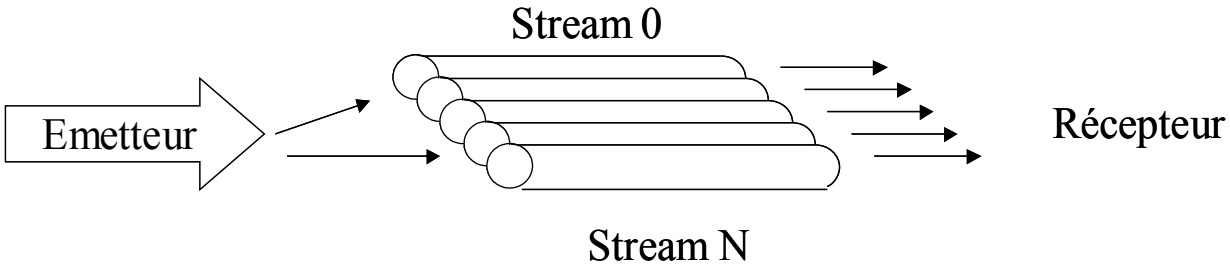



**Figure 2.20.**Principe du *Multistreaming* en SCTP.

Les *chunks* de données sont identifiés par leurs TSN. En SCTP le TSN compte les *chunks* de données émis et non pas les octets transportés par ces *chunks* comme dans le cas du SN (*Sequence Number*) de TCP. Deux *chunks* de données SCTP consécutifs ont deux TSN consécutifs.

Le principe de *multistreaming* dans les paquets SCTP forme un avantage très intéressant dans l'utilisation quotidienne d'applications diverses (comme : navigateur Web, client FTP, application de visio-conférence…) chacune constituant un flux (*stream*) au sein d'une même association alors

qu'avec TCP elles nécessitent l'ouverture d'une connexion distincte pour chacune d'elles. Ceci entraîne la multiplication des ports que les *firewalls* doivent gérer d'où parfois des échecs de connexion avec TCP. Des études effectuées dans le domaine satellitaire [SHA05] trouvent que l'utilisation de *multistreaming* en SCTP est particulièrement bénéfique dans les environnements susceptibles aux erreurs comme les communications par satellite, du fait que le m*ultistreaming* peut réduire le *HOL Blocking* au niveau récepteur, et réduit les exigences de stockage du côté du récepteur.

## 2.6. Multihoming

Le *multihoming* est une propriété essentielle de SCTP. Il permet à une association SCTP d'être associée à plusieurs adresses sources et destination. Chaque terminal peut ainsi être atteint via plusieurs adresses IP. SCTP apporte uniquement un mécanisme de sauvegarde de chemin [RFC2960]. Un seul chemin est actif à la fois ; le partage de charge ne fait pas partie de la spécification actuelle du protocole [RFC2960, RQX02]. Chaque noeud peut être accessible par plusieurs adresses de transport (cf. figure 2.21) fixées au moment de l'ouverture de l'association. Les adresses de transports (c'est-à-dire, liste d'adresses IP + port SCTP) sont échangées lors de la phase d'initialisation d'une association SCTP (*chunks* INIT et INIT-Ack).

Les transmissions vers des nœuds *multihomed* peuvent ainsi gagner en robustesse contre les défaillances du réseau ou les problèmes de congestion en choisissant dynamiquement une alternative, à la condition que des chemins différents soient constitués pour chaque adresse IP du nœud. SCTP voit les adresses IP d'un client *multihomed* comme «différents chemins» possibles pour l'atteindre [IVA02].

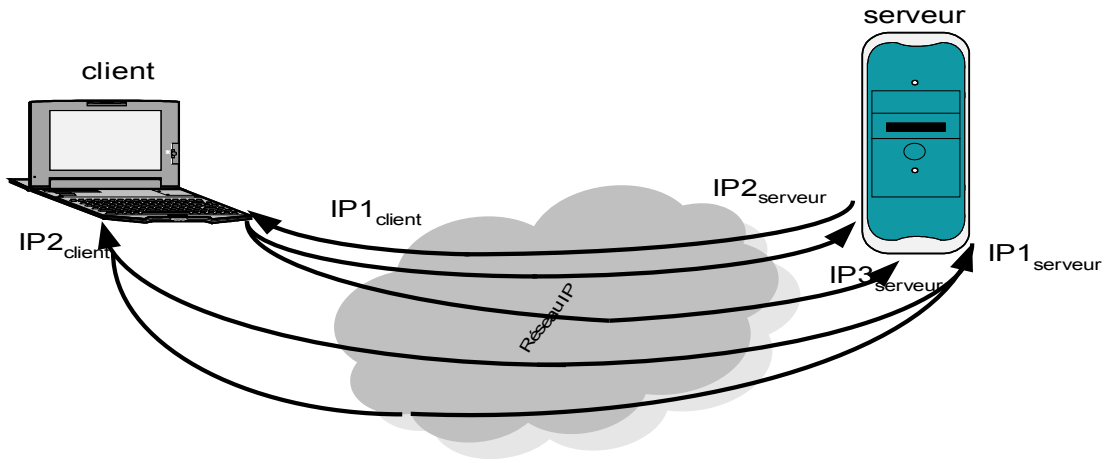



**Figure 2.21.** Exemple de noeuds SCTP *multihomed*.

### 2.6.1 Gestion des adresses IP

Les protocoles des couches supérieures (ou ULP *Upper Layer Protocol*) déterminent toutes les adresses transport et fixent le chemin primaire (chemin par lequel transite le trafic utilisateur en condition normale). Pour effectuer une transmission vers un nœud *mulihomed*, l'émetteur choisit préalablement une des adresses possibles qui correspondra au chemin primaire (lors de l'établissement de l'association). L'émetteur ne doit par la suite envoyer des données que par ce chemin primaire. De plus, pour acquitter des paquets SCTP d'un nœud *multihomed*, l'émetteur des *chunks* SACK devrait utiliser le même chemin que celui emprunté par les *chunks* de données reçus [RFC2960]. Cependant, lorsque le récepteur obtient plusieurs *chunks* dupliqués, il peut supposer que ses *chunks* SACK empruntent un chemin défaillant, il serait alors peut être judicieux d'utiliser une autre adresse secondaire.

Lors de retransmissions il serait également intéressant d'utiliser une autre adresse secondaire possible, la perte de paquets pouvant être liée à un problème de l'adresse primaire, et permet ainsi de réduire les risques de congestion. Lorsque le chemin primaire devient inaccessible (suite à une congestion, ou à une perte de données importante), SCTP basculera tout le trafic sur un des chemins secondaires de l'association considérée (vers une adresse de transport secondaire du noeud de destination). La fonction de *multihoming* est uniquement utilisée à des fins de sauvegarde. Le partage de charge entre plusieurs routes n'est pas supporté.

### 2.6.2 Contrôle des adresses empruntées (ou des chemins)

SCTP se doit de contrôler régulièrement les différentes adresses d'un nœud *multihomed*. Pour cela il considère deux états possibles pour chacun des chemins possibles : actif ou inactif. Le chemin primaire étant considéré actif, la disponibilité des alternatives est contrôlée par l'envoi régulier de *chunks HEARTBEAT Request*, qui doivent être acquittés par un *chunk* HEARTBEAT ACK. Si une adresse ne répond pas après plusieurs HEARTBEAT infructueux, le chemin est considéré inactif.

La décision sur le fait qu'une adresse est accessible ou non est prise selon un mécanisme d'écoute appelé *heartbeating*. Un chemin primaire est supposé inactive, si l'émetteur ne peut plus y accéder suite à l'occurrence de plusieurs expirations de *timer RTO* (*Retransmission Time Out*) consécutives. Dans ce cas les paquets SCTP seront routés vers une autre adresse supposée active. Le choix de l'adresse IP secondaire de destination à activer se fait en contactant le noeud destination via un *chunk Heartbeat Request* (c.f. Figure 2.22). Le nœud SCTP distant doit répondre avec un *chunk Heartbeat Ack* (c.f. Figure 2.23) en indiquant l'adresse IP secondaire qui sera active dans l'association en cours.

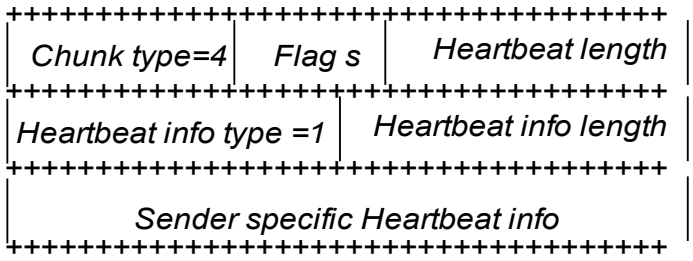



**Figure 2.22.** Format du *chunk Heartbeat Request* [RFC2960].

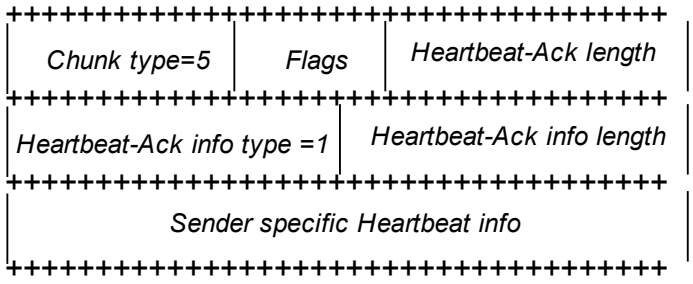



**Figure 2.23.** Format du *chunk Heartbeat-Ack* [RFC2960].

Un *chunk heartbeat* est envoyé périodiquement vers les adresses de transport de destination en veille [RQX02] (peut être active ou inactive) afin de mettre à jour leurs états d'accessibilité. La période est donnée par Hi (cf équation 1). La période d'émission d'un *chunk heartbeat* est contrôlée par le paramètre *HB.Interval* (30 seconde) [RFC2960].

$$Hi = RTOi + HB.Interval(1+\delta) \qquad (1)$$

où, $RTO_i$ est le RTO sur le chemin secondaire «i» calculé sur les *chunks heartbeat*, et δ est une valeur choisie aléatoirement dans [-0.5,0.5] à l'initialisation de l'association.

Lors de transfert de données au sein d'une association *multihomed* des considérations doivent être prises en compte :

- Quand l'émetteur est *multihomed,* le récepteur ne doit pas nécessairement envoyer un SACK vers l'adresse de transport primaire de l'émetteur.
- Lorsque le récepteur est *multihomed* et que l'émetteur a besoin de retransmettre un ou plusieurs *chunks* de données, l'émetteur peut considérer la retransmission vers une adresse secondaire de l'association [RFC2960].

Il faut noter qu'en cas d'erreur sur la liaison primaire, l'émetteur utilisera un des chemins secondaires pour émettre les données vers le récepteur. Pour une association un seul chemin est considéré comme primaire les autres sont des chemins de secours. Seulement des messages d'écoute *(Heartbeat*) circulent sur les chemins secondaires dits aussi alternatifs.

### 2.6.3 Transfert de données dans une association avec *multihoming*

Comme nous l'avons signalé ci-dessus, SCTP supporte la fonctionnalité de *multihoming* afin de permettre à des sessions, ou des associations dans la terminologie SCTP, de se maintenir même lorsque une adresse IP d'un hôte n'est plus accessible. SCTP a un système intégré de détection et de rétablissement d'erreurs, qui permet aux associations d'envoyer dynamiquement le trafic vers une adresse IP alternative de destination si cela s'avère nécessaire.

Comme indiqué dans la RFC 2960 [RFC2960], l'utilisation du *multihoming* en SCTP ajoute au protocole les fonctions de base suivantes :

- Dans une association, un chemin unique est considéré primaire. Ceci signifie qu'une des adresses IP affectées au récepteur de l'association est choisie pour être l'adresse primaire. Le chemin primaire correspond au chemin réseau qui mène à l'adresse primaire du point terminal qui lui est associé. Sauf contre indication par l'utilisateur SCTP, un point terminal devrait toujours transmettre sur le chemin primaire
- Lors de l'acquittement des *chunks* reçus, les *chunks* SACK doivent emprunter le même chemin qui a été emprunté par les *chunks* reçus.
- Dans le cas de retransmission de *chunk* vers un point terminal *multihomed*, le récepteur doit choisir une adresse de destination autre que celle à laquelle le *chunk* de données original a été envoyé. Aussi bien, quand un point terminal reçoit un *chunk* de données dupliqué, il peut changer l'adresse de destination et ne pas utiliser l'adresse source de ce *chunk* de données dupliqué pour envoyer un acquittement. En fait, il faut considérer toutes les alternatives d'adresses de transport (source/destination) pour sélectionner l'adresse source-destination la plus adéquate pour la retransmission. Il n'y a aucune stratégie adoptée pour le choix de l'adresse définie par la norme (e.g. une approche RR: *Round Robin*).
- Une autre spécificité de SCTP, est utilisée pour contribuer à l'exécution de *multihoming* : le mécanisme *heartbeating*. Ce mécanisme détecte les problèmes sur les chemins en veille et les points terminaux. Il peut donc détecter si une adresse destination est active ou inactive. Les *heartbeat chunks* sont envoyés périodiquement à toutes les destinations en veille (quelles soient actives ou inactives) [RQX02], et un compteur (Ei) (c.f figure 2.24) calcule le nombre de *chunks heartbeat* envoyés vers une destination inactive sans le retour du *chunk heartbeat*-Ack. Quand ce compteur excède une valeur maximale (*Path Maximum Retransmission*), cette adresse de destination est déclarée comme inactive. De même en absence d'une réponse après un certain temps (RTO : *T3-rtx expire*) l'émetteur peut considérer que l'adresse de transport de destination en cours comme inaccessible, et incrémente le compteur E (*error counter*) de cette adresse IP. Ainsi, si le besoin s'impose une autre adresse IP, différente de l'adresse IP primaire, est utilisée afin d'atteindre un point terminal *multihomed*, SCTP connaît laquelle des adresses est active et peut ainsi éviter d'utiliser un autre chemin défectueux.

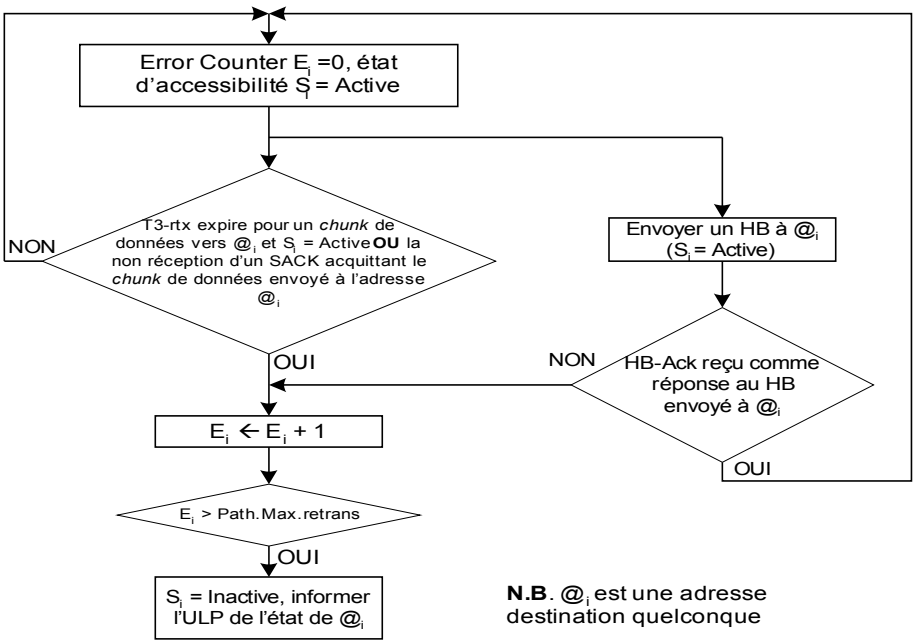



**Figure 2.24.** Procédure de transfert de données relativement à un noeud SCTP *multihomed*.

Ainsi le mécanisme *multihoming*, supporté par des machines et des équipements réseau, est une solution techniquement faisable et de plus en plus économique [IYE04]. Un hôte est *multihomed* s'il peut être adressé par des adresses IP multiples, comme c'est le cas quand le hôte a plusieurs interfaces réseau. Par conséquent, des équipements radio peuvent être connectés simultanément à plusieurs technologies d'accès (cf. figure 2.25). Ainsi, les machines pourront avoir des connexions filaires et radio. Les différentes interfaces actives suggèrent également l'existence simultanée de plusieurs chemins entre les hôtes multi-adressés (*multihomed*).

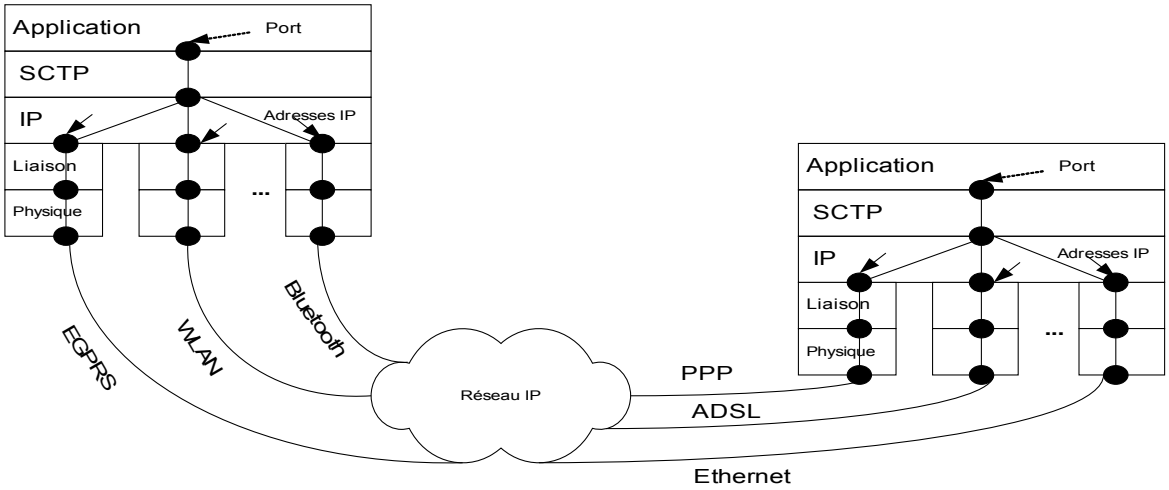



**Figure 2.25.** Exemple de noeud SCTP *Multihomed* connecté à plusieurs technologies d'accès.

## 2.7. *Multihoming* et mobilité

Pour trouver une solution qui permet le déplacement inter-réseaux IP, sans interruption de l'association en cours, deux standards sont en cours de validation. Il s'agit des *drafts* IETF suivants : «*Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration*» [STE05], et «*Mobile SCTP*»[RIE04]. Ceux ci sont une extension à la fonctionnalité *mobile IP* disponible dans IPv4 [RFC3344] et dans IPv6 [STE05].

### 2.7.1 Extension de SCTP : Adressage dynamique

Cette fonctionnalité introduit une extension au SCTP [STE05] qui lui offre la possibilité de :

1) Reconfigurer des adresses IP au cours de l'association en cours
2) Changer la route primaire (i.e. Routage vers une nouvelle adresse primaire)
3) Échanger des informations d'adaptation de couche durant l'établissement d'association [STE05]. Ce qui assure un basculement de lien sans pertes de données.

Cette extension consiste à la création de deux nouveaux types de *chunk* : *Address Configuration Change Chunk* ASCONF (cf. figure 2.26) et *Address Configuration Ackowledgement* ASCONF-ACK (cf. figure 2.27). Ces deux *chunks* contribueront à l'ajout et la suppression dynamiques des adresses IP à une association ainsi qu'à la génération d'une demande de changement d'adresse primaire au cours de la même association.

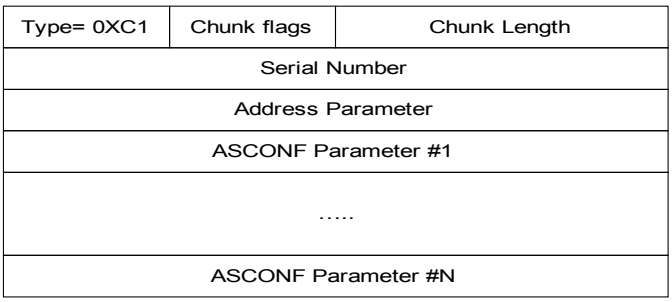



**Figure 2.26.** Format du *chunk* ASCONF [STE05].

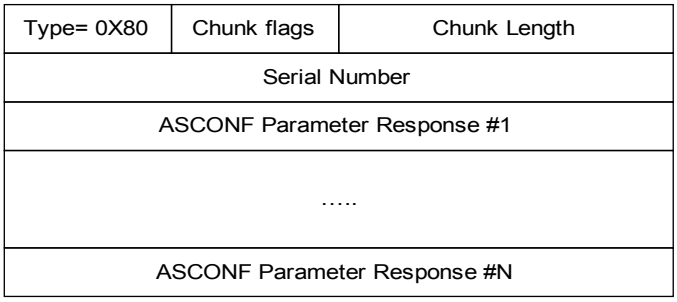



**Figure 2.27.** Format du *chunk* ASCONF-ACK [STE05].

Le *chunk* ASCONF est utilisé pour communiquer au point terminal distant une des demandes de changement de configuration. Ces demandes sont spécifiées par l'un des nouveaux paramètres introduits par cette extension. Ces demandes doivent être acquittées au moyen du *chunk* ASCONF-Ack comprenant les paramètres spécifiques pour assurer la réponse à une demande reçue (échec ou succès de l'opération). Ces *chunks* sont transmis de façon authentifiée.

De nouveaux paramètres sont introduits pour décrire la nature de l'opération exigée par l'un des deux points terminaux d'une association afin de changer sa configuration. Les deux types de *chunks* associés aux six paramètres nouvellement introduits par cette extension servent pour ajouter et supprimer dynamiquement des adresses IP d'une association et donc de changer d'adresse primaire (changer la configuration d'une association). Les six nouveaux paramètres introduits par cette extension sont :

- *Add IP Address* (ASCONF) : permet d'ajouter une nouvelle adresse IP à l'association en cours.
- *Delete IP Address* (ASCONF) : permet de supprimer une adresse IP de l'association en cours.
- *Error Cause Indication* (ASCONF-ACK) : c'est un paramètre de réponse, qu'est utilisé pour contourner une ou plusieurs causes d'erreur usuelles rencontrées en SCTP.
- *Set primary IP Address* : ce paramètre peut être intégré dans les *chunks* ASCONF, INIT ou INIT-Ack. L'intégration de ce paramètre dans INIT ou INIT-Ack peut être utilisée pour indiquer une préférence d'une adresse primaire.
- S*uccess Indication* (ASCONF-ACK) : ce paramètre est utilisé pour indiquer le succès de réception des données contenues dans le *chunk* ASCONF.
- *Adaptation Layer Indication* : ce paramètre peut apparaître dans les *chunks* INIT et INIT-Ack et devrait être remonté aux protocoles de couche supérieure du récepteur. Ce paramètre ne doit pas apparaître dans le *chunk* ASCONF. Il est envisagé qu'il soit utilisé pour le contrôle de flux et pour l'adaptation à d'autres couches qui exigent une indication qui soit contenue dans les *chunks* INIT et INIT-Ack.

### 2.7.2 Procédure ASCONF de SCTP [STE 05]

Une des conditions nécessaire de transfert du *chunk* ASCONF est que ces transferts ne doivent pas causer de congestion dans le réseau. Quand un point terminal veut transmettre un *ASCONF signaled change* à un point terminal SCTP distant il doit procéder comme suit :

1) Créer un *chunk* ASCONF. Ce *chunk* devrait contenir toutes les informations TLV (*Type Length Value*) nécessaires à être émises au point terminal SCTP distant, et une relation d'identités (*Correlation Ids*) unique relativement à chaque demande.
2) Un *serial number* doit être attribué au *chunk*. Ce paramètre doit être un numéro croissant. Le *serial Number* doit être initialisé, à l'établissement de l'association. Il a la même valeur que l'*Initial TSN* et chaque fois qu'un nouveau *chunk* ASCONF est créé il est incrémenté de 1 après avoir attribué le *Serial Number* au nouveau *chunk* créé.
3) Si aucun *chunk* ASCONF n'est en cours de transmission (non acquitté) au point terminal SCTP associé, alors un *chunk* ASCONF lui est transmis.

4) Déclencher le temporisateur T-4 RTO, en utilisant la valeur RTO de l'adresse destination sélectionnée (normalement c'est la primaire).
5) Quand un *chunk* ASCONF-Ack, acquittant le *Serial Number* précédemment envoyé, est reçu, le temporisateur T-4 RTO est arrêté et l'association relative est réinitialisée ainsi que les compteurs des erreurs de destination.
6) Traiter tous les TLVs (les données) contenues le *chunk* ASCONF-Ack afin de trouver des informations sur des états particuliers en réponse aux différentes demandes qui ont été envoyées. Utilisation des *Correlation IDs* afin de lier la demande aux réponses correspondantes.
7) Si une réponse d'erreur est reçue pour un paramètre TLV, alors tous les TLVs sans réponse sont considérées comme réussies s'ils ne sont pas remontées. Tous les TLVs après l'échec de réponse sont considérées comme non réussies à moins qu'une indication spécifique de succès soit présente pour le paramètre.
8) S'il n'y a pas de réponse(s) à des paramètres TLVs spécifiques, et que pas d'échecs indiqués, alors toutes les demandes sont considérées comme réussies.
9) Si le récepteur SCTP répond à un *chunk* ASCONF avec un *chunk* d'erreur rapportant qu'il n'a pas reconnu le type de *chunk* ASCONF. L'émetteur du *chunk* ASCONF ne doit pas envoyer des *chunks* ASCONF supplémentaires et doit arrêter son temporisateur T-4.

Si le temporisateur *T-4 RTO* expire, le point terminal SCTP devrait procéder selon les étapes suivantes [STE05] :

1) Incrémenter les compteurs d'erreurs et exécuter la détection d'échec de route concernant l'adresse destination appropriée.
2) Incrémenter les compteurs d'erreurs au cours de l'association et exécuter l'échec de détection au niveau du point terminal dans l'association.
3) Modifier la valeur RTO de l'adresse destination vers laquelle le *chunk* ASCONF a été envoyé en doublant la valeur du temporisateur RTO.
4) Retransmettre le *chunk* ASCONF précédemment transmis, si c'est possible, sur une adresse destination alternative. Un point terminal SCTP ne doit pas ajouter de nouveaux paramètres à ce *chunk*, il doit conserver le même *chunk* ASCONF précédemment transmis.
5) Réinitialiser le temporisateur *T-4 RTO*.

Notons que si une adresse destination différente est sélectionnée, alors le RTO utilisé sera celui de la nouvelle adresse destination.

### 2.7.3 Règles générales de gestion des adresses

Suite à une demande d'ajout d'une nouvelle adresse IP à une association en cours, cette adresse IP n'est pas totalement exploitée avant que l'ajout ne soit acquitté. Dans ce cas l'émetteur ne doit pas utiliser la nouvelle adresse IP comme une source pour n'importe quel paquet SCTP, sauf celui

comportant le *chunk* ASCONF. Alors que le récepteur, de la demande d'ajout d'adresse IP, peut utiliser immédiatement cette adresse comme destination. Après la réception d'un *chunk* ASCONF-Ack d'un *IP-address add*, le point terminal SCTP peut commencer à utiliser l'adresse IP ajoutée comme adresse source pour n'importe quel type de *chunk* SCTP. D'autre part, la suppression d'une adresse IP d'une association, cette adresse IP doit être considérée comme adresse destination valide pour la réception de paquets SCTP jusqu'à l'arrivée de ASCONF-Ack. Alors qu'elle ne doit pas être utilisée comme adresse source pour aucun des paquets envoyés ultérieurement. C'est à dire qu'aucun des datagrammes qui arrivent avant ASCONF-Ack, destinés à l'adresse IP à supprimer, n'est ignoré par le récepteur. Tandis que les *chunks* ABORT arrivant à destination de l'adresse IP à supprimer doivent être ignorés.

### 2.7.4 *Mobile* SCTP (mSCTP)

Le *draft* IETF «*Mobile SCTP»* [RIE04], décrit le *Mobile* SCTP comme étant une extension du protocole de transport SCTP par l'introduction de l'aspect d'adresse IP dynamique pour le *multihoming* ce qui favorise la mobilité.

La fonctionnalité principale permettant de maintenir une association active lors de changement de réseau IP est la fonction ADDIP[STE05]. Cette dernière permet de modifier/supprimer/ajouter une adresse IP faisant partie de l'association en cours, sans provoquer son interruption.

*Mobile* SCTP (mSCTP) est conçu pour une architecture de type client/serveur, avec un client mobile qui initie l'association avec le serveur fixe. mSCTP doit être utilisé avec des algorithmes de gestion de mobilité tels que *Mobile IP* ou *Dynamic DNS*.

Nous décrivons dans la suite mSCTP en proposant un exemple d'étude. Nous considérons, alors, un client mobile dans une zone comprenant deux accès radio distincts (cf. figure 2.28). Un client mobile (c_mobile) se connecte à Internet en passant par certaines technologies radio. Il lui sera attribué une adresse IP à partir de l'espace d'adressage de la localisation A par exemple (IPc_mobile_LocA). Ceci est accomplit par une des techniques d'attribution dynamique d'adresse comme DHCP (*Dynamic Host Configuration Protocol*). Le client mobile se déplace de la zone de localisation A vers la zone de localisation B et il est informé qu'il a atteint la zone de couverture d'un nouveaux réseau à partir des informations fournies par la couche physique de son NIC (*Network Interface Card*).

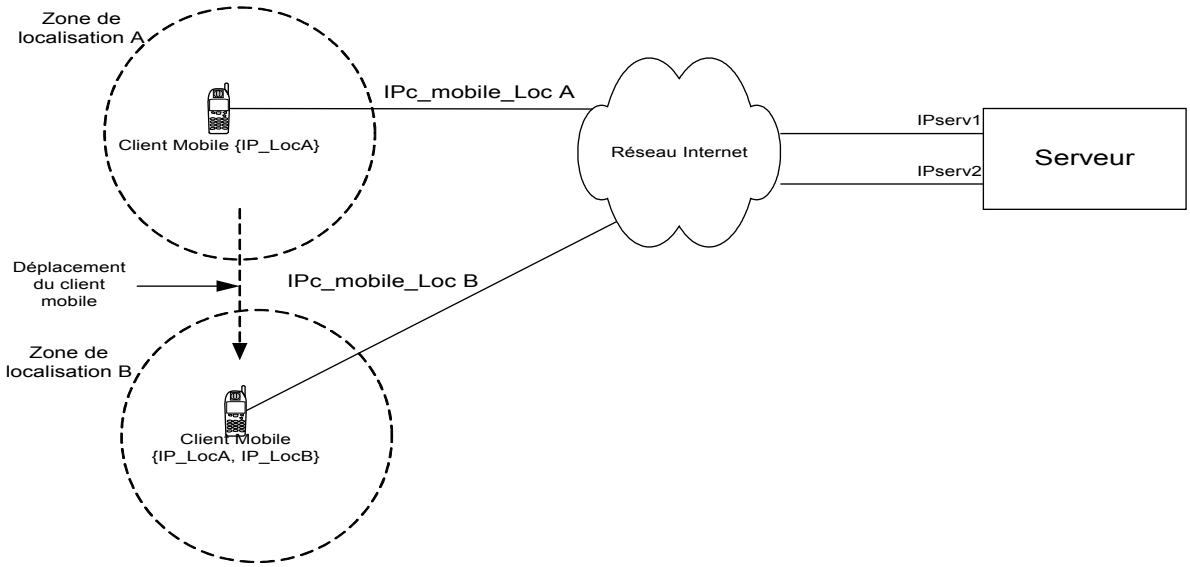



**Figure 2.28.** *Mobile* SCTP

En plus de son ancien lien (avec la zone de localisation A), le client mobile établit un nouveau lien avec le réseau de la zone de localisation B et une nouvelle adresse IP (Ipc_mobile_LocB) lui est affecté sur sa seconde interface. Le client mobile est donc *multihomed* et il est accessible par deux réseaux différents. Dans ce cas le mobile ajoute la nouvelle adresse IP, qui lui a été attribué par identification de la connexion au serveur. Afin d'avoir une distinction facile des deux liens au niveau du client mobile, plusieurs adresses IP doivent être affectées aux interfaces réseau du serveur, ce qui permet de représenter différents liens par des entrées différentes de la table de routage du client mobile.

En accédant à la zone de localisation B, le client mobile peut quitter la zone de couverture du point d'accès dont la zone de localisation est A et peut perdre le lien correspondant à sa première adresse IP. Le flux de données entre serveur et mobile est interrompu et le comportement fiable du protocole de transport assure que toutes les données sont envoyées sur le second lien dans le cas d'échec permanent sur le premier lien. Dans ce cas le client mobile informe son destinataire qu'il n'est plus accessible par la première adresse IP et demande de supprimer cette adresse de l'association. Si le client mobile a accès aux informations sur la puissance du signal radio, le *handover* vers le second lien sera initié avant que la perte de paquets se produise au niveau serveur.

En résumé, avec mSCTP, un client mobile peut avoir au moins deux adresses IP au cours de l'association existante, et durant le mode *handover* si deux points d'accès sont simultanément disponibles. En effet, un client mobile se trouvant dans la zone de localisation A, obtient une seule adresse IPc_mobile_LocA (l'association SCTP établie a une seule adresse). En se déplaçant vers la zone de localisation B, dans la zone d'intersection des deux zones de couverture (ou localisation A et B), le client mobile obtient une nouvelle adresse IPc_mobile_LocB, et ceci au moyen du mécanisme DHCP par exemple. L'ajout d'une nouvelle adresse IP se fait donc au moyen de DHCP par l'envoie d'un *chunk* ADD-IP (cf. figure 2.29). Après la réception d'un *chunk* ADDIP-Ack à partir de son point terminal SCTP qui lui est associé, la nouvelle adresse est utilisée selon le mécanisme de gestion de route (*path management mechanism*) [XIN02] qui est responsable de la

détection d'adresses IP non disponibles et le basculement sur les adresses secondaires.

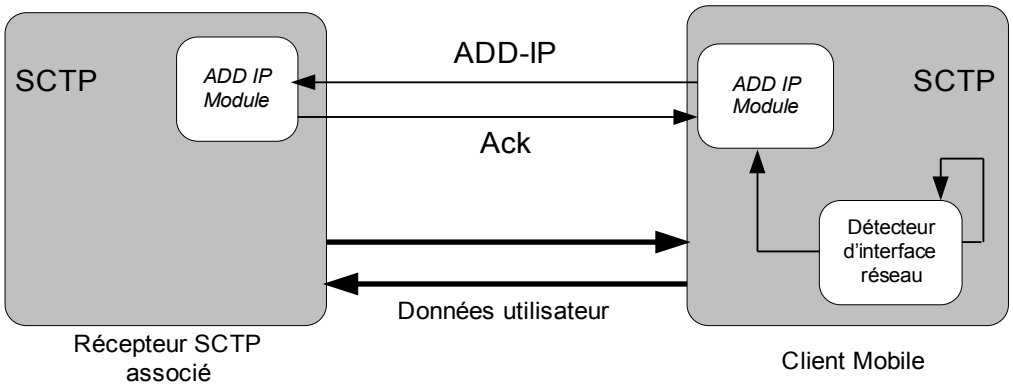



**Figure 2.29.** Un prototype *Mobile* SCTP

### 2.7.5 Combinaison de la mobilité au niveau couche liaison avec celle au niveau transport

Certaines technologies d'accès radio comme IEEE802.11 fournissent des fonctionnalités de gestion de mobilité au niveau couche liaison. Un *Handover* couche 2 est la plupart du temps restreint à la micro mobilité mais peut être avantageux en le combinant avec la gestion de mobilité niveau transport.

Si le *mobile SCTP* est utilisé dans un environnement IEEE802.11, un client mobile doit prendre 4 décisions [RIE04] :

1) En fonction des paramètres physiques le client mobile doit décider quand est ce qu'il s'associe à une station de base. Pour prendre cette décision il peut prendre en compte le niveau du signal.
2) Après l'établissement du lien radio couche 2 une méthode de configuration IP doit être activée (si le lien radio est stable).
3) Après avoir terminé la configuration du nouveau lien, une décision doit être prise lorsque cette nouvelle route est remontée au noeud destinataire associé en utilisant le mécanisme ADDIP.
4) La dernière décision concerne l'instant où le récepteur doit demander le changement de route primaire.

### 2.7.6 Insuffisances de mSCTP

SCTP, avec l'extension ADDIP (mSCTP), offre une solution pour la gestion de *handover* et de localisation dans les réseaux de radiocommunication. mSCTP a fait l'objet de plusieurs travaux [XIN02, JUN04, SEO04]. En effet, mSCTP peut être utilisé pour permettre un *handover* sans pertes

lors des sessions mobiles qui sont déclenchées par le serveur vers le client mobile et inversement. La principale hypothèse d'avoir un *handover* sans pertes est qu'un MN (*Mobile Node* ) est capable d'obtenir une nouvelle adresse IP à partir d'une nouvelle localisation. Ceci est implémenté en utilisant DHCP pour les réseaux IPv4 ou d'autres mécanismes similaires [RIE04]. Malgré ses avantages, mSCTP présente des insuffisances.

D'une part, le *mobile SCTP* ne maintient pas un *handover* simultané pour les 2 points terminaux SCTP (formant l'association). Si les deux points terminaux exécutent un HO en même temps, l'association SCTP sera perdue. Autrement dit, l'activation de deux adresses primaires simultanées n'est pas faisable avec mSCTP. Une seule adresse primaire est active à la fois, ce qui peut entraîner un échec du *handover*.

D'autre part, *mobile* SCTP, tel qu'il est décrit par [RIE04], présente des insuffisances de point de vue gestion des liens radio lors du basculement. En effet, en mSCTP les paquets de données sont émis vers l'ancienne adresse IP avant que le MN considère la nouvelle adresse IP comme adresse primaire pour l'association. Ce qui engendre des pertes de paquets lors de chaque *handover* du côté du *Correspondent Node* avant que celui ci décide finalement de changer de route primaire. De plus, mSCTP ne permet pas de gérer pratiquement la mobilité c'est qu'il ne décrit pas comment changer la route primaire lors des *handovers*. Le mSCTP actuel ne supporte pas le roaming, c'est que s'il n'y a pas d'entité pour la gestion de localisation, le *chunk* INIT pour une nouvelle association avec le *MN's home address* ne peut pas être correctement routé vers la nouvelle localisation du MN après qu'il se déplace vers un nouveau réseau. Pour remédier à ces insuffisances, les auteurs dans [AYD03] proposent un nouveau protocole basé sur le *mobile* SCTP, il s'agit du *cellular* SCTP (cSCTP).

### 2.7.7 *Cellular* SCTP : cSCTP

Cette alternative du *mobile SCTP*, consiste en une modification des *chunks* ASCONF et ASCONF-Ack afin d'informer le *Correspondant Node* (CN) du déclenchement d'un mode *handoff* au niveau du MN (*Mobile Node*). Cette modification consiste à l'utilisation d'un bit H des *chunks flags* pour indiquer le déclenchement du mode *Handover* (cf. figure 2.30). En fait, le CN à la réception d'un Add-IP, ajoute la nouvelle adresse à l'association. Si de plus le bit H=1 (mode *handover*), alors les deux adresses (l'ancienne adresse primaire et la nouvelle adresse ajoutée) seront considérées comme primaires pour le MN (en même temps). Dans ce cas le CN envoie des paquets dupliqués sur les deux adresses primaires vers le MN. Et la valeur de *cwnd* pour chacune des deux adresses est égale à la moitié de celle de l'ancienne adresse primaire. Contrairement au mSCTP où les paquets de données sont émis vers l'ancienne adresse IP avant que le MN considère la nouvelle adresse IP comme une adresse primaire pour l'association en cours. La suppression de l'ancienne adresse IP de l'association se fait lorsque cSCTP du MN décide qu'une adresse est inactive. Le MN quitte le mode HO (*handoff_mode = false*), supprime l'ancienne adresse IP de l'association et envoie au CN un *chunk* DELETE-IP ASCONF. D'où la procédure de *handover* proposée par le cSCTP fonctionne de la manière suivante :

1- Détection et obtention d'une nouvelle adresse IP : le HA (*Host Agent*) envoie des messages ROUTER SOLICITATION vers les ARs (*Access Router*) et ceux ci répondent par des messages ROUTER ADVERTISEMENT au moyen du DHCP ou *Stateless Address Auto*

*Configuration.*

2- Ajout d'une nouvelle adresse dans l'association. Après l'obtention d'une nouvelle adresse IP au niveau du point d'attachement, le *Host Agent* informe le composant cSCTP du nœud mobile de sa nouvelle adresse IP.

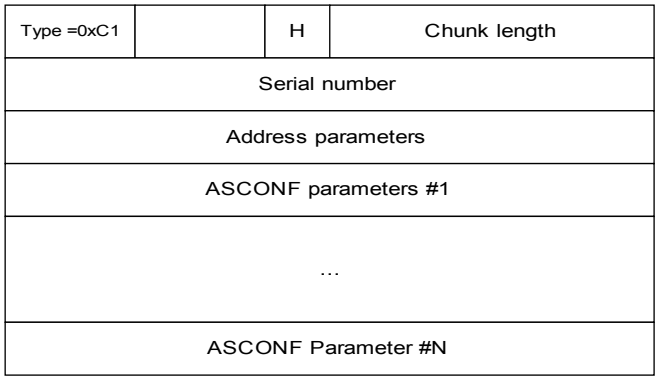



**Figure 2.30.** Modification du *chunk* ASCONF pour activer le mode *handover* [AYD03].

## 2.7.8 Mécanisme de gestion de mobilité au niveau transport (basé sur mSCTP)

Une autre alternative de mSCTP a été proposée dans [CHA04]. Elle consiste à proposer un mécanisme qui détermine les conditions d'ajout et de suppression des adresses IP, en exploitant la puissance du signal radio au niveau couche liaison dans le but d'améliorer les performances du mSCTP.

Il s'agit d'une amélioration du protocole mSCTP. Étant donné que *mobile SCTP,* tel qu'il est décrit par [RIE04], ne permet pas de gérer pratiquement la mobilité. De plus, mSCTP ne décrit pas comment changer la route primaire lors des *handovers*. Ainsi, si le changement de route primaire est appliqué tel qu'il est décrit par mSCTP, le CN (C*orrespondant Node*) subira plusieurs pertes de paquets lors de chaque *handover* avant qu'il décide finalement de changer de route primaire.

Cette amélioration est basée sur des mesures de qualité de signal au niveau couche liaison. En effet, mSCTP au niveau MN (*Mobile Node*) exécutera ADDIP chaque fois que le niveau de puissance du signal reçu à partir du nouveau *Access Router* dépasse la valeur seuil du niveau de signal autorisant des communications. Cette fonctionnalité liée à la gestion des adresses IP (ajout/suppression) est implémentée dans un bloc logique appelé AMM (*Address Management Module*) (cf. figure 2.31). Le AMM détermine quand est ce qu'il déclenche ADDIP et DELETE IP, ainsi que le changement de la route primaire et informe le mSCTP de l'action à exécuter en fonction des signaux reçus à partir de la couche liaison et du module *IP Address Acquisition Module.* Le AMM maintient des informations telles que l'interface correspondante à la route primaire courante et l'interface offrant la puissance maximale du signal au niveau couche liaison.

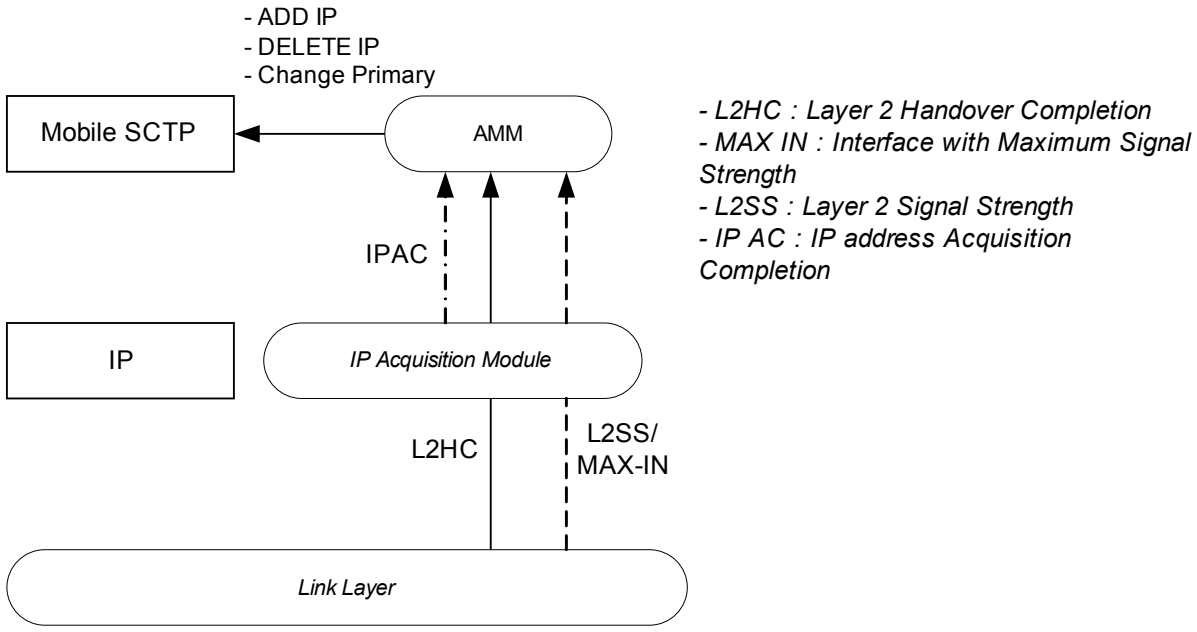



**Figure 2.31.** Amélioration proposée du *mobile SCTP*.

## 2.8. Conclusion

Dans ce chapitre nous nous sommes intéressés à l'étude du protocole SCTP en présentant certains de ses aspects fonctionnels. La suite du présent mémoire abordera de façon plus précise les caractéristiques fondamentales de SCTP et leurs apports par rapport à TCP, à savoir le *multistreaming* et le *multihoming*. Dans nos travaux de recherche nous visons l'implémentation de ce protocole dans un environnement de communications radio pour la transmission de données (EGPRS/EDGE : *Enhanced General Paquet Radio Service/Enhanced Data rates for Global Evolution*).

En tenant compte de la possibilité qu'un hôte peut communiquer sur plusieurs technologies d'accès et de l'importance de la mobilité dans les réseaux actuels, l'intégration du *multihoming* s'avère une bonne solution à adopter. Afin de permettre de se déplacer d'un réseau IP vers un autre, sans interruption de l'association en cours. Le *multihoming* tel qu'il est spécifié dans la RFC [RFC2960], peut être sujet à des modifications dans le but d'assurer la continuité des communications surtout à la suite d'un *handover* inter différentes technologies d'accès radio. Ainsi, la RFC donne une grande flexibilité pour l'implémentation de fonctionnalité de *multihoming* par l'utilisation des termes comme «*should*» ou *«may»* lors de la description de son comportement. Ceux-ci correspondent aux mots-clés d'IETF pour les parties facultatives des spécifications. Cependant, une première alternative du SCTP, exploitant le *multihoming*, était proposée dans un *draft* IETF [RIE04], il s'agit du *Mobile* SCTP (mSCTP), comme nous l'avons déjà vu dans ce chapitre. Ainsi que d'autres alternatives, basées sur le mSCTP, ont été proposées afin d'améliorer le *handover* inter-systèmes dans un environnement de radiocommunication. C'est sur cet aspect que nous contributions, présentées dans les prochains chapitres, se basent, elles consistent en des modifications apportées au protocole SCTP en couplant l'aspect *multihoming* avec la mobilité.

Les chapitres suivants se focalisent, en particulier, sur l'aspect de *multihoming* et de mobilité. SCTP tel qu'il est spécifié par la RFC 2960 est adapté aux réseaux filaires. En fait, l'idée derrière le contrôle de congestion pratiqué que se soit en SCTP qu'en TCP est relativement simple. Elle consiste à ajuster les débits de transmission des sources émettrices en fonction de la charge du réseau, en d'autres termes, si une source détecte une certaine perte de paquets, cette perte est interprétée comme étant due à un état de congestion dans le réseau. Ainsi la source réagit en diminuant le débit de transmission afin de ne pas aggraver la situation en fournissant plus de trafic au réseau congestionné. Ce mécanisme n'est pas tellement adapté aux liens à taux d'erreur élevés, notamment des liens radios GSM, GPRS, EDGE ou WLAN. C'est dans cette optique que nous présentons, ultérieurement, des propositions de modifications au mécanisme de contrôle de congestion en exploitant le *multihoming* pour la gestion de *handover* intra et inter RAT (*Radio Access Technology*) ainsi qu'une modélisation *cross-layer* entre SCTP et la couche liaison de données en EGPRS.

Le prochain chapitre se focalise sur l'exploitation de la fonctionnalité de *multistreaming,* introduite avec SCTP, et son apport comparé à TCP dans un contexte de transmission de données interactives sur un réseau EDGE.

# Chapitre 3 : Évaluation de performance de SCTP sur l'interface radio EGPRS pour un trafic HTTP

## 3.1. Introduction

Dans ce chapitre nous orientons notre étude plus particulièrement vers le mécanisme *multistreaming* en SCTP. Cette fonctionnalité telle qu'elle est conçue dans [RFC2960] permet de réduire les problèmes engendrés par le *head of Line blocking*. Ce problème qui a causé en TCP, depuis sa conception, un handicap que se soit en environnement fixe que mobile. Des études comme [IVA02, SUK03], ont prouvé l'avantage accordé par l'introduction du *multistreaming* dans les réseaux filaires afin d'éviter le problème de HOL *blocking*. Dans cette thèse, étant intéressés à l'interaction entre le protocole SCTP et les mécanismes RLC/MAC en EGPRS, nous prouvons l'avantage d'introduire le *mutistreaming* plutôt dans un environnement radio à savoir EDGE.

Dans une première étape nous décrivons les problèmes rencontrés par les protocoles de transport usuels (TCP) dans un contexte de communication radio. Ensuite nous étudions le contrôle de flux au niveau SCTP en liaison avec un mécanisme ARQ (*Automatic Repeat reQuest*) dans un réseau EGPRS. Une description de la fonctionnalité *multistreaming* est présentée par référence aux solutions équivalentes adoptées avec les protocoles de transport usuels. Enfin, des simulations réalisées sous une plateforme NS2 sont décrites et analysées afin de vérifier l'avantage du *multistreaming* de SCTP dans un contexte EGPRS par comparaison avec le protocole TCP pour un service interactif. Cette dernière partie, met l'accent également sur les limitations de SCTP dans de tels environnements.

## 3.2. Application du protocole SCTP dans un contexte radio mobile

### 3.2.1 Les problèmes rencontrés lors de l'implémentation de TCP dans un environnement radio mobile

Dans notre étude nous nous intéressons au fonctionnement de SCTP sur un réseau mobile 2.5G voire 2.75G, pour le faire nous récapitulons les problèmes rencontrés dans de tels environnements avec l'utilisation du TCP en nous basant sur l'étude faite en [RFC3481]. Étant donné que les deux protocoles ont le même mécanisme de contrôle de congestion, SCTP affrontera les mêmes problèmes.

1) Le problème majeur que TCP rencontre dans un environnement radio est qu'il n'est pas capable de détecter les pertes de paquets causées par des erreurs de transmission au niveau interface radio. En effet, TCP, conçu pour qu'il soit utilisé dans un environnement à faible taux d'erreur (réseaux filaires), interprète ces pertes comme étant des problèmes de congestion ce qui peut engendrer un déclenchement de *slow start* alors que le réseau n'est pas réellement congestionné.
2) Quand les données n'arrivent pas en séquence, ce qui mène TCP à déclencher l'algorithme *Fast Retransmission.* Alors que le ré-ordonnancement des paquets est causé par une retransmission au niveau liaison, donc il s'avère inutile d'enclencher une retransmission au niveau TCP. En fait, il a fallu que le récepteur prolonge la durée de son temporisateur pour qu'il reçoive le paquet.
3) Les réseaux mobiles comme UMTS et EGPRS peuvent être caractérisés comme étant des réseaux «faibles», du fait qu'ils ont une largeur de bande modérée en combinaison avec des retards importants. Le BDP (*Bandwidth Delay Product*) est une manière de caractériser la topologie en terme de quantités de données qui peuvent être tenues et des retards qui peuvent se produire. Un BDP peut causer, à certain moment, l'accumulation des informations dans le réseau durant le transfert de données. Ceci affectera la communication entre l'émetteur et le récepteur puisque la signalisation est retardée.
4) Pour TCP, un RTT élevé coûte énormément en retransmissions, puisque on perd beaucoup de temps durant les phases de *Slow Start* et de *congestion avoidance*.
5) Le rythme d'émission de segments TCP est donné par la fréquence d'arrivée des acquittements au niveau de l'émetteur («*self clocking*» du TCP). L'asymétrie des liens radio fait qu'il peut y avoir perte de *self clocking* ce qui engendre une baisse de performances de TCP.
6) Les utilisateurs mobiles qui sont entrain d'utiliser les interfaces d'accès radio à certains réseaux sont confrontés non pas uniquement à une dégradation du débit, mais aussi à une déconnexion. Malgré que la déconnexion n'est pas souhaitable, il n'est pas possible de l'éviter et elle est préférable si le temps hors connexion n'introduit pas de problèmes de congestion pour TCP.

Dans la suite nous essayons d'assurer un comportement optimal de SCTP dans un environnement radio. D'abord nous allons justifier notre choix du protocole SCTP au dépend de TCP. La caractéristique apportée par SCTP, exploitée dans ce chapitre, est le *multistreaming*. C'est grâce à cette fonctionnalité, comme s'est décrit dans le première chapitre, que nous pouvons réduire les délais engendrés par le problème de HOL *blocking* causé par TCP lors de transmission des données. Ensuite nous nous intéressons aux améliorations que SCTP peut apporter pour une application de type HTTP1.0.

## 3.3. Le *Multistreaming* en SCTP

Le SCTP permet la transmission de plusieurs flots (*stream*) de données au sein d'une même association (multiplexage de flots). Le fait de considérer des *stream*s indépendants pour transmettre les datagrammes (seulement l'ordre des messages qui doit être maintenu) dans une seule association, permet de réduire le *Head of Line Blocking* [CIS03] au niveau applicatif du récepteur SCTP.

De manière générale, la transmission de différents flux de données est basée sur l'une des trois approches présentées par la figure 3.1. Dans chacune des trois situations, le hôte A envoie trois types de données (notés de 1 à 3) au hôte B. Lors de la première approche, l'émetteur A ouvre une connexion par type de flux de données à transmettre vers le récepteur B. Bien que cette approche présente une séparation logique des données basée sur leurs types, ce genre de connexions multiples affecte le mécanisme de contrôle de congestion de TCP et attribue à une application une bande passante plus large et ce au détriment des autres applications et d'autres flux de données dans le réseau.

Alors que la deuxième approche, permet le hôte A de multiplexer et démultiplexer les trois types de données à travers une seule connexion. Les applications exploitant cette approche n'affectent pas le mécanisme de contrôle de congestion de TCP. Cependant, cette approche augmente la complexité pour le programmeur de l'application, puisque cette dernière doit, elle-même, assurer efficacement la procédure de transmission de données ainsi que sa bonne gestion.

Tant qu'à la troisième approche, elle utilise le protocole UDP. Cette approche est similaire à la deuxième. Cependant, les programmeurs d'application doivent assurer leur propre service de fiabilité aussi bien que leur propre multiplexage/démultiplexage. Ceci est dû à l'incertitude introduite par UDP et au fait qu'il offre un service sans connexion.

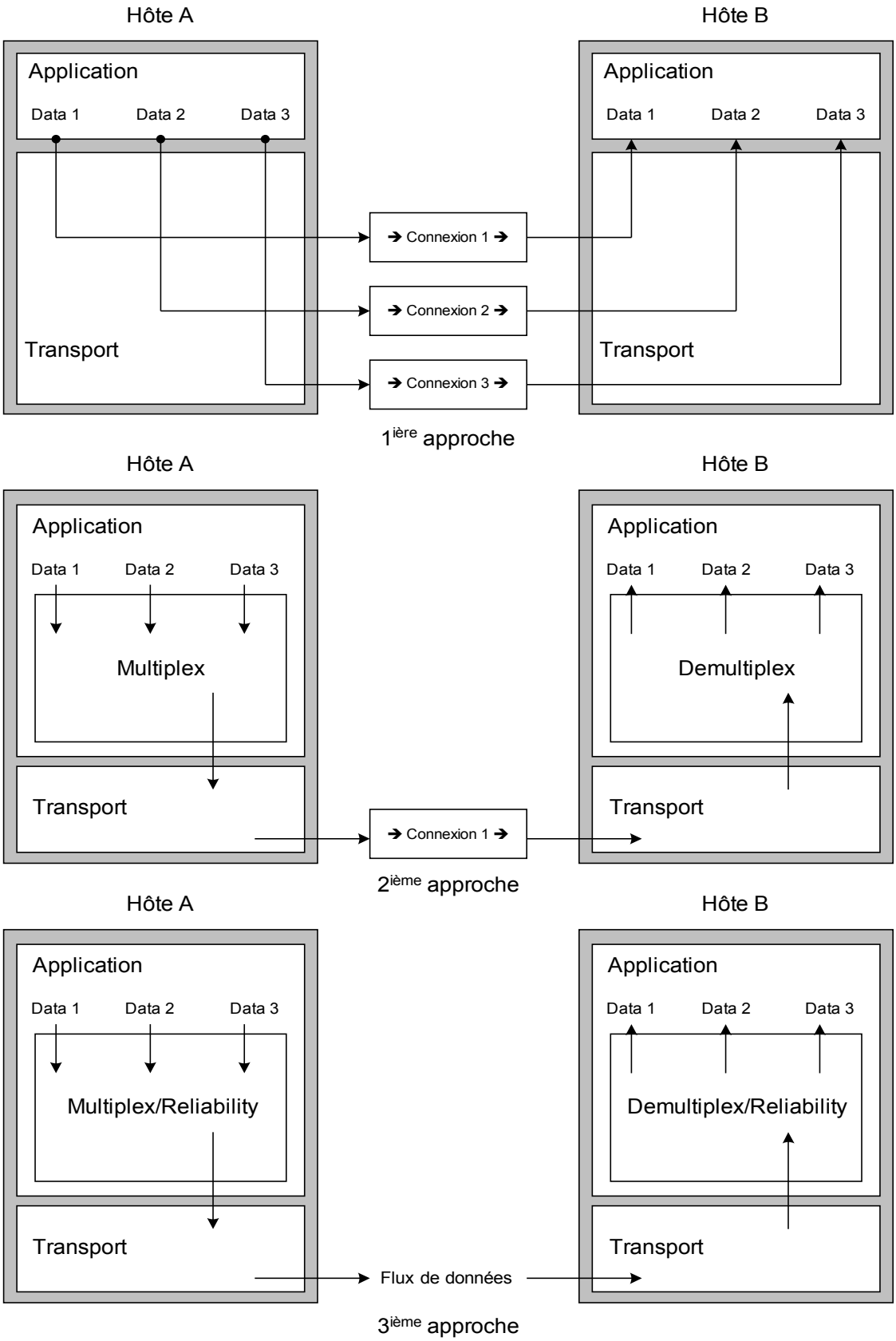



**Figure 3.1.** Approches traditionnelles de transmission parallèles de données.

L'introduction du protocole SCTP et du concept de *stream*, offre aux applications une transmission de données de différents types lors d'une même association. Cette nouvelle approche combine les avantages de connexions multiples de bout en bout et le multiplexage/démultiplexage des applications [GER03]. Un *stream* SCTP est une séparation logique propre à une association SCTP. C'est un lien logique et continu de transmission qui est établi au sein d'une association entre deux points terminaux SCTP. Un point terminal (client ou serveur) SCTP peut, en effet, activer plusieurs *stream*s sortants (ou entrants) lors de l'établissement d'une association (cf. figure 3.2). Chaque *stream* est muni d'une mémoire de stockage qui lui est indépendante des autres pour envoyer et recevoir des données. Les *stream*s sont actifs tout au long de la durée de vie de l'association SCTP.

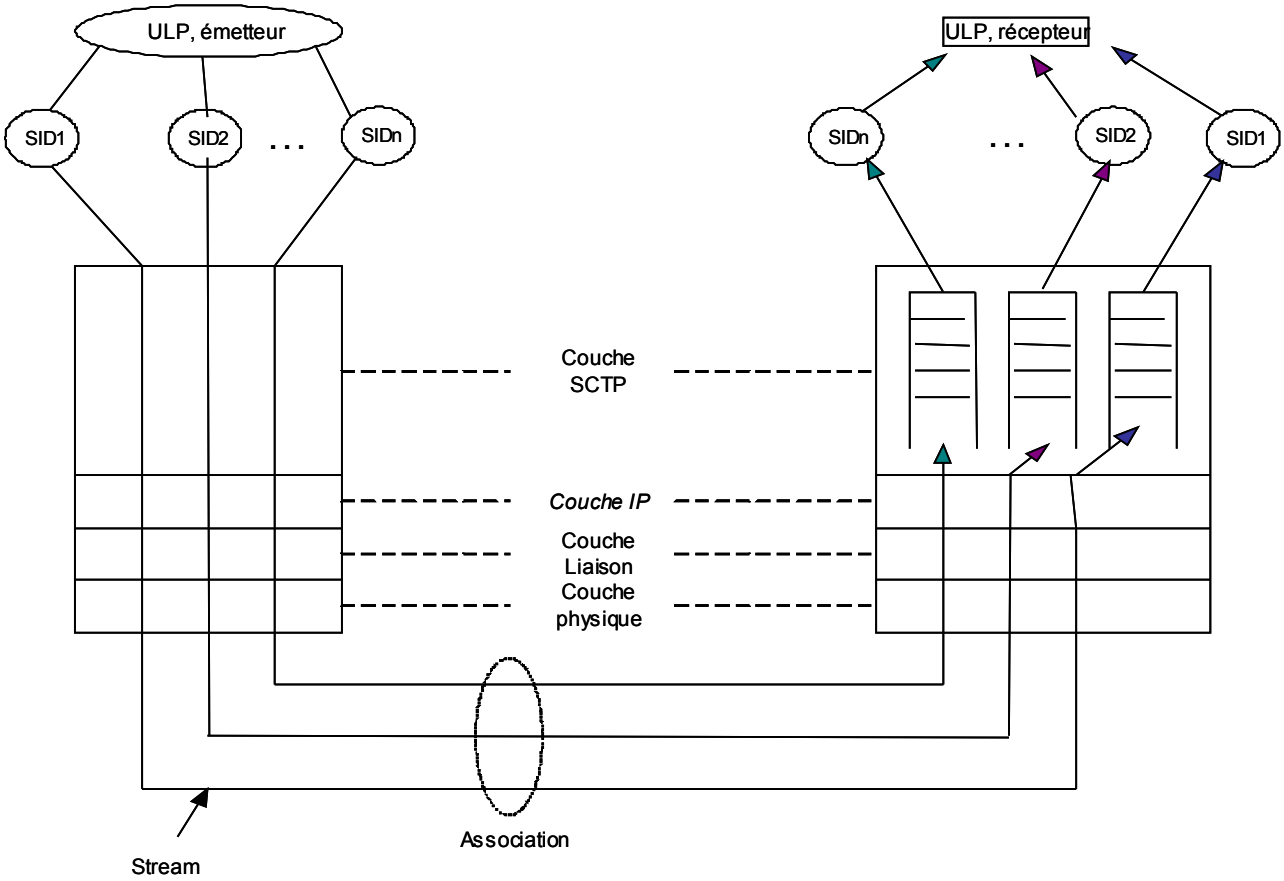



**Figure 3.2.** Exemple d'une association SCTP (*multistreaming* activé).

Nous traitons dans ce qui suit un exemple d'activation de *multistreaming* nous permettons ainsi de mieux comprendre cet aspect. Nous supposons établir une association SCTP entre un client A et un serveur B (cf. figure 3.3). Le client A demande l'ouverture de trois *stream*s avec le serveur B (les *stream*s sont numérotés de 0 à 2), alors que celui ci a un seul type de données à transmettre vers le client A. Ainsi le serveur B maintient un seul *stream* sortant vers le client A (numéroté *stream* 0). Cependant, la configuration de SCTP du côté du client A gère bien cet aspect de *multistreaming* pour la transmission de données à travers les *stream*s. Les concepteurs du protocole SCTP [RAN03] ont précisé qu'une implémentation de SCTP doit concevoir un algorithme de gestion de l'allocation de *stream*s et de leurs utilisations. Ainsi, la livraison des données relatives à un *stream* SCTP

donné, doit respecter les délais de transmission. Deux algorithmes ont été considérés dans [RAN03]: le *round-robin* (RR) et le *first-come first-serve* (FCFS).

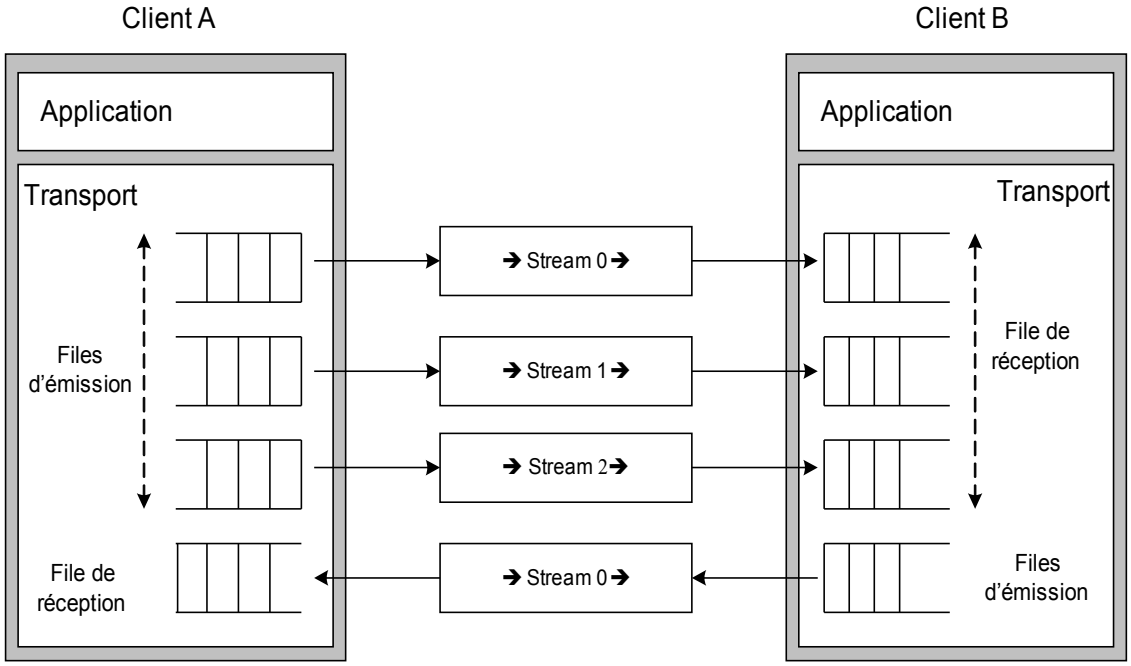



**Figure 3.3.** Une association SCTP avec *Multistreaming* du côté du client A.

Selon une politique RR, le client A choisit un *chunk* de données à partir de la file d'attente d'émission du *stream* 0. Si le client A peut envoyer encore des données, tel le permettent la taille de la fenêtre de réception du serveur B et la taille de la fenêtre de congestion du client A, il envoie un *chunk* de données à partir de la file d'attente d'émission du *stream* 1. Un *chunk* de données sera envoyé par la suite, à partir de la file d'attente d'émission du *stream* 2. Ensuite, le client A émettra un *chunk* de données sur le *stream* 0. Ce processus de *round-robin* continue tout au long de la durée de vie de l'association entre le client A et le serveur B.

Alors qu'avec une politique FCFS, le client A transmet chaque *chunk* dans l'ordre de sa réception à partir de sa couche application indépendamment de la file d'attente d'émission du *chunk*.

Les améliorations de l'aspect *multistreaming* se poursuivent au sein de l'IETF avec l'introduction de PR-SCTP (*Partial-Reliability*) par le *draft*[7] [RAN03a]. Le PR-SCTP a permet aux *stream*s SCTP de transporter efficacement des données en tolérant quelques pertes. Ainsi, les *stream*s SCTP fournissent des aspects fonctionnels puissants aux applications sans augmenter leurs complexités.

Ainsi le *multistreaming* prouve un avantage remarquable de point de vue résolution des problèmes de pertes et livraison de données apparus avec le TCP. En effet, lors de transfert des données au sein d'une association SCTP où le *multistreaming* est activé deux niveaux de numérotation sont mis en évidence. Une numérotation au niveau association avec le TSN et une autre au niveau du *stream* avec le SSN. Si une perte se produit au niveau d'un *stream* donné, elle n'affecte pas la livraison des *chunks* de données contenus dans les autres *streams*. Ce qui permet de

7 Un *draft* internet est un travail en cours d'amélioration.

réduire les retards de transmission causés par les problèmes de HOL *Blocking*. Dans la suite de ce chapitre nous nous focalisons à prouver par simulation l'apport du *multistreaming* introduit avec SCTP par rapport au protocole TCP, en utilisant une couche applicative de type HTTP1.0.

## 3.4. HTTP : protocole de couche application dans un réseau EGPRS

Dans le cas d'un réseau EGPRS, le SGSN (*Serving GPRS Support Node*) est responsable de la livraison des paquets aux stations mobiles dans sa zone de service à travers les stations de base.

EGPRS définit deux mécanismes de retransmission afin d'assurer un niveau de fiabilité de transmission de données acceptable. Un mécanisme de retransmission est utilisé au niveau de la couche RLC/MAC (*Radio Link Control/Medium Access Control*) entre le MS (*Mobile Station*) et le BSS (*Base Station Subsystem*) et le second opère au niveau de la couche LLC (*Logical Link Control)* entre le SGSN et MS (cf. figure3.4).

Dans notre contexte d'étude de performances de SCTP sur un réseau mobile nous considérons comme application un service interactif de type HTTP, mais non pas sur TCP comme c'est le cas dans [ERI02]. Dans ce cas nous aurons trois niveaux de retransmissions de ce type d'application sur EGPRS :

1) De bout en bout entre MS et le serveur (couche SCTP), [*]
2) Entre MS et SGSN : *Serving GPRS Support Node*, [**]
3) Entre MS et BSS/PCU[8], [***]

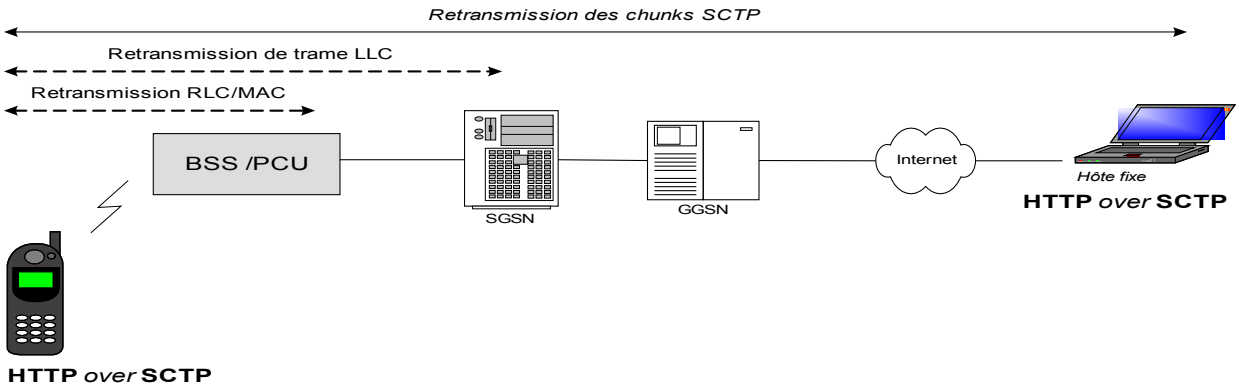



**Figure 3.4.** Architecture étudiée.

**(*)** : SCTP, offrant la possibilité du *multistreaming*, paraît mieux adapté au HTTP dans le sens de l'indépendance entre les différents *Streams* ouverts. En fait, une erreur de réception sur un *stream* exige seulement la retransmission sur ce dernier. En plus, un mécanisme d'acquittement sélectif est activé donc seuls les *chunks* de données perdus ou erronés seront retransmis. D'autres

8 *Base Station Subsystem / Packet control Unit* . Le PCU est un équipement logiquement associé au contrôleur de station de base chargé des aspects radio introduits avec le GPRS.

part, dans le but de mieux fiabiliser la réception d'une page web nous considérons dans nos simulations un ordre de priorité de sorte à privilégier les *streams* contenant les données les moins volumineux telles que le texte par rapport aux images, afin de gagner sur le temps d'attente de téléchargement d'une page entière.

**(**)** : les retransmissions LLC sont effectuées entre SGSN et MS au niveau de la couche LLC pour confirmer le transfert d'informations. Le mécanisme de retransmission LLC utilise un *Timer* de retransmission pour déclencher les retransmissions, et une fenêtre pour contrôler le nombre maximale de trames séquentiellement envoyées à un instant donné (non acquittées).

**(***)** : quant aux retransmissions RLC, EGPRS utilise un protocole de couche liaison RLC fiable, où les blocs RLC sont utilisés comme étant l'unité de retransmission entre BSS et MS. Sur le lien descendant (resp. montant), une trame LLC est envoyée vers (resp de) la station mobile sous forme de blocs RLC, qui passent par le canal radio (canal bruité). Le transfert de blocs RLC est contrôlé par un mécanisme ARQ (*Automatic Repeat reQuest*) sélectif. Après la réception des blocs RLC le récepteur (MS ou BSS) retourne des messages ACK/NACK (*Acknowledgment/Negative Acknowledgement*), périodiquement. L'émetteur retransmet uniquement les blocs RLC erronés ou perdus.

## 3.5. SCTP *over* ARQ

Afin d'assurer une transmission fiable de bout en bout dans un environnement radio de 2.75G (EDGE), nous étudions le comportement d'un tel réseau lorsque nous utilisons un protocole de transport fiable orienté message : le SCTP. Ce protocole sera présent donc au niveau du terminal mobile et sur la partie fixe du réseau. Sur la partie radio, caractérisée par la présence de différents phénomènes engendrant des pertes de données (tels que *shadowing, path loss,* interférences…), un protocole ARQ est implémenté au niveau couche 2 afin de détecter et de retransmettre les trames erronées. Notre objectif est de justifier l'apport de SCTP par rapport à TCP dans le cadre d'étude de l'interaction entre les protocoles de couches transport et le mécanisme ARQ dans un environnement radio. En nous référons aux études faites sur ce plan, nous pouvons prouver l'utilité d'introduction d'un nouveau protocole de transport.

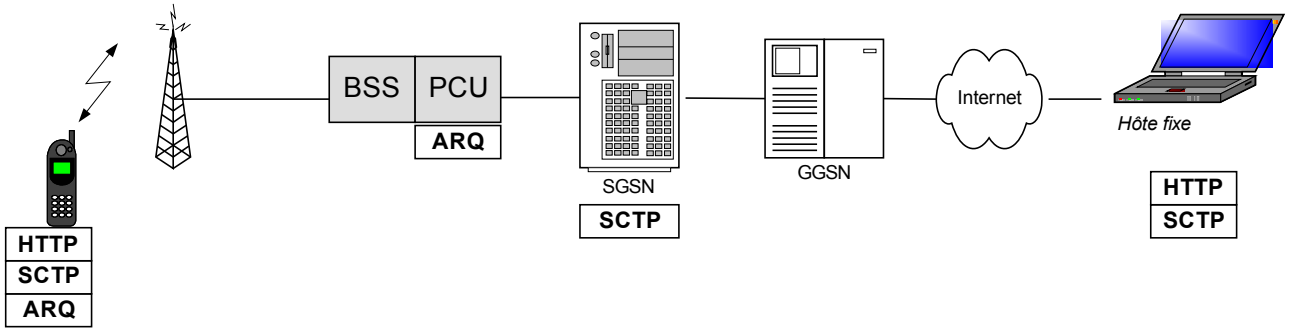



**Figure 3.5.** Configuration de réseau.

Nous étudions la configuration donnée par la figure 3.5. Nous aurons un émetteur ARQ au

niveau du terminal mobile (MS), et un récepteur ARQ au niveau du BSS. Supposant que la couche liaison utilise un mécanisme ARQ sélectif dont l'unité de trame est le SDU (*Service Data Unit*). Un SDU correspond à un paquet IP supportant un paquet SCTP.

Nous aurons alors : $SDU = N \cdot PDU$ ,

Les PDUs (*Packet Data Unit*) sont transmis lorsque la couche MAC indique qu'il y a des ressources de transmission disponibles. A la réception d'un PDU, après décodage, la couche physique exécute la détection d'erreur, et transmet le résultat avec les données vers la couche liaison. Lorsque tous les $N_{PDU}$ de PDUs constituant un SDU sont reçus correctement, le SDU est rassemblé et livré aux couches supérieures tout en gardant le séquencement d'origine du SDU. A chaque SDU, l'émetteur demande au récepteur un rapport d'état. Selon le *selective repeat ARQ*, le récepteur envoie un PDU contenant un rapport d'état (*Status PDU*) indiquant les PDUs correctement reçus et ceux à retransmettre. A la réception d'un *status PDU*, les PDUs stockés dans une mémoire tampon de retransmission de l'émetteur sont supprimés ou retransmis selon le *Status PDU*. Chaque PDU peut être transmis au plus $M_t$ transmissions. Quand on atteint $M_t$, l'entité émettrice ignore ce PDU ainsi que tous les PDUs appartenant au même SDU.

L'aspect fonctionnel du *status PDU* au niveau RLC/MAC s'approche de celui du *chunk* SACK au niveau SCTP. En effet, à la réception d'un SACK (*Selective Acknowledgement*) on a l'acquittement des *chunks* de données (~PDU de données au niveau SCTP) jusqu'au numéro du *Cumulative TSN Ack Point* et de ceux dont le TSN est donné par les *Gap Ack.* Dans ce cas l'émetteur doit retransmettre les *chunks* de données dont les TSNs ne sont pas indiqués dans le SACK. Le nombre de retransmissions est un paramètre configurable selon le taux de persistance de l'ARQ et le BDP du réseau, tout en tenant compte de la réception des SACKs dupliqués car au bout de la réception du 4$^{ième}$ SACK indiquant les mêmes vides de TSN, le protocole SCTP déclenche l'algorithme de *fast retransmit*. La persistance de l'ARQ doit être modérée afin d'éviter les gigues qui peuvent se produire.

## 3.6. Introduction d'un trafic de type HTTP

Le protocole HTTP (*HyperText Transfet Protocol*) est l'un des protocoles les plus utilisés sur le WEB. Typiquement, les clients demandent des documents du serveur web et les affichent à l'utilisateur une fois que les documents ont été cherchés. HTTP est un protocole orienté message, où chaque message est conforme aux spécifications RFC1945 [RFC1954]. Lorsque plusieurs fichiers incorporés (gifs/jpgs…) sont transférés en utilisant HTTP, on désire que chaque fichier soit transféré correctement. Cependant, la livraison en séquence de ces fichiers n'est pas exigée. En fait, les fichiers incorporés doivent être affichés au bout du temps le plus court possible. La plupart des utilisateurs web essaye d'atteindre cet objectif en ouvrant plusieurs connexions TCP au niveau du serveur et en divisant les demandes GET pour ces fichiers incorporés sur ces connexions. Cela permet aux navigateurs de rendre autant de fichiers incorporés qui peuvent être cherchés sur les connexions TCP multiples, en même temps.

Comme nous l'avons vu dans le premier chapitre, SCTP sépare la notion de transmission de *streams* de données de celle d'une association. Ce qui permet d'avoir plusieurs *streams* au sein d'une association, le transfert de données sur ces *streams* est totalement ordonné, alors que les données appartenant à différents *streams* sont partiellement ordonnées au sein d'une association et

ceci peut réduire le retard causé par les problèmes de *head-of-line blocking*.

Dans [RAJ02] l'auteur prouve que SCTP est mieux adapté au protocole HTTP que TCP dans le sens qu'il réduit le temps de latence[9] perçu par l'utilisateur, et améliore aussi le *Throughput*[10]. Il a considéré deux architectures de serveur, à savoir :

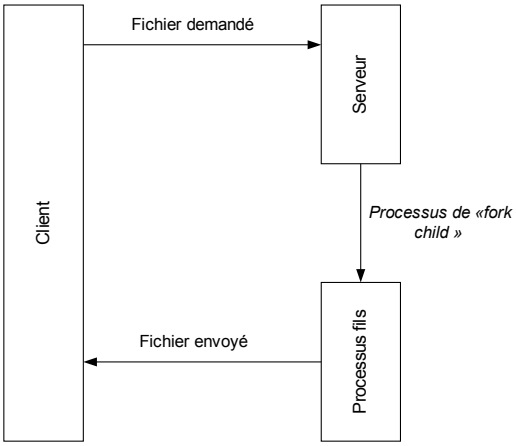
1) Dans le cas de transfert d'un seul fichier (le protocole FTP : *File Transfert Protocol*), une architecture similaire est considérée pour les deux protocoles TCP et SCTP (figure 3.4 (a)). Le client établit une connexion avec le serveur et envoie une demande HTTP GET. Le serveur reçoit la demande et crée un processus fils pour le traitement de la demande. Ce dernier traite la demande et si elle peut être satisfaite, le fils envoie le fichier demandé, autrement il émet un message d'erreur approprié.

2) Dans le cas de transfert de plusieurs fichiers (c'est à dire des fichiers incorporés) chacun des deux protocoles a une architecture qui lui est spécifique. Concernant TCP l'architecture est donnée par la figure 3.4 (b). D'abord, le client TCP établit une connexion avec le serveur TCP. Ensuite envoie au serveur TCP la demande du premier fichier (file 0). Le serveur TCP crée un processus fils qui à son tour émet soit un fichier demandé soit un message d'erreur. À la réception du premier fichier, le client TCP prépare des demandes pour un nombre spécifique de fichiers (c'est à dire les fichiers incorporés). Les demandes pour tous les fichiers sont envoyées par le client TCP d'une façon continue (*pipelined*). Le processus fils du côté serveur TCP reçoit ces demandes et envoie les fichiers demandés l'un après l'autre. L'alignement des demandes par le client TCP réduit la latence, puisque le serveur TCP n'attend pas la réception d'une nouvelle demande après l'émission d'un des fichiers incorporés, la demande arrive tant que le premier transfert est en cours. Le transfert de fichier est réalisé en utilisant une connexion persistante, c'est à dire que tous les fichiers consécutifs sont transférés sur une seule connexion TCP.

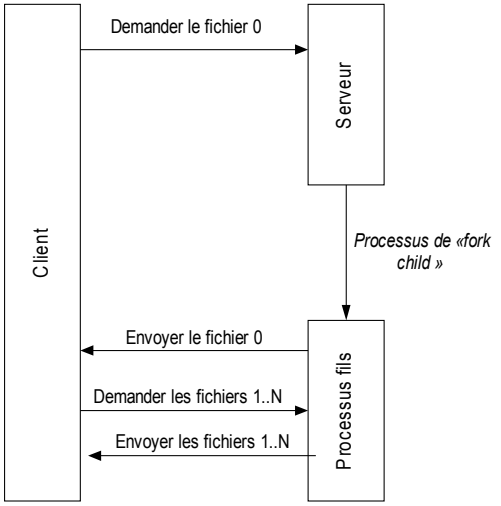
3) Toujours dans le cas de transfert de plusieurs fichiers, à la différence que c'est le protocole SCTP qui est mis en évidence. L'architecture du serveur avec SCTP est différente de celle avec TCP, et est donnée par la figure 3.4 (c). Le client SCTP établit une connexion avec le serveur SCTP, envoie une demande pour le premier fichier et le serveur SCTP crée un processus fils pour traiter la demande. Ce dernier envoie le fichier demandé sur le premier *stream* (*stream 0*). Après la réception du premier fichier, le client SCTP procède à l'émission des demandes de fichiers consécutives. Le processus fils du côté serveur SCTP reçoit ces demandes et au lieu de les émettre tous l'un après l'autre comme le cas du TCP, il les envoie en parallèle dans différents *streams*. Par conséquent, tous les fichiers sont transférés simultanément et la perte de paquets pour un *stream* particulier n'affecterait pas les autres *stream* ce qui permet d'éviter les problèmes de HOL *blocking*. Le nombre de *stream,* qui peut être établi au sein d'une seule association, est négocié entre le client SCTP et le serveur SCTP [RFC2960] à l'instant de l'établissement de l'association. Si le nombre de *streams* est inférieur au nombre de fichiers à émettre, certains ou tous les *streams* auront besoins d'être utilisés plus qu'une fois.

9 *Latency* : la latence est définie comme étant le temps écoulé entre l'instant où un document particulier est demandé et l'instant où il est entièrement reçu.

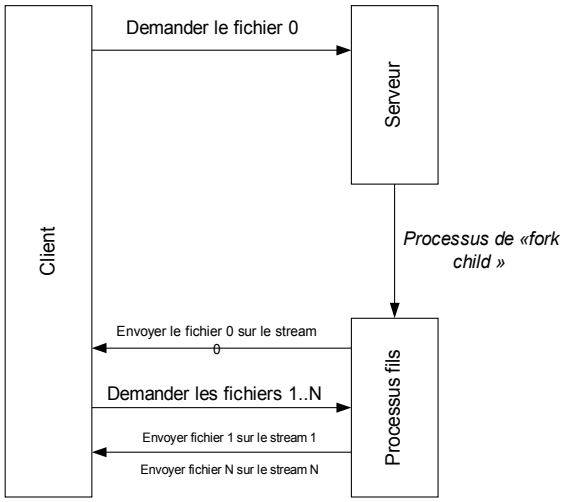
10 *Throughput* : *est défini* comme étant le temps écoulé entre l'instant où un document est demandé et l'instant où le document et tous les fichiers incorporés qui lui sont attribués sont disponibles au récepteur.




(a) Transfert d'un seul fichier – TCP et SCTP




(b) Transfert de fichiers incorporés TCP




(c) Transfert de fichiers incorporés SCTP

**Figure 3.6.** Architecture Client/serveur pour le transfert de HTTP sur TCP/SCTP.

En fait, un *stream* est vu comme une sous connexion dans une association SCTP. Pour une association SCTP à plusieurs *streams* nous gagnons en terme de ressources et de retard que nous pouvons réserver lors de l'établissement des connexions TCP. C'est que tous les *streams* d'une même association partagent les mêmes mécanismes de *congestion Avoidance*, alors que les différentes connexions TCP ont différents paramètres de *congestion avoidance*, ce qui amènera le client TCP à un partage excessif de la bande passante, ceci est dû au fait qu'une connexion TCP donnée ignore l'existence des autres connexions.

## 3.7. Implémentation de SCTP sous NS2

En nous basons sur la description RFC3481 [RFC3481] qui décrit une implémentation du TCP sur des réseaux mobiles 2.5G/3G, celle du SCTP doit inclure les aspects suivants :

- SCTP appliqué à un réseau 2.5G/3G doit supporter des tailles de fenêtres appropriée en se basant sur le BDP de bout en bout. En fait, les spécifications TCP limitent la taille de fenêtre du récepteur à 64Kbps. Si le BDP de bout en bout s'attend d'être plus important que 64 Kbps, l'option *Window Scale* peut être utilisée pour surmonter les limitations. Du fait que plusieurs systèmes d'exploitation utilisent par défaut de petit récepteur TCP et émettent des tampons de l'ordre de 16Kbps donc, même pour un $BDP < 64\text{Kbps}$ , la taille du tampon fixée par défaut doit être augmentée au niveau émetteur et au niveau récepteur afin d'avoir une fenêtre plus large.
- TCP appliqué à un réseau 2.5G/3G doit initialiser sa fenêtre de congestion à $CWND = min(4 \cdot MSS, max(2 \cdot MSS, 4380\,Bytes))$ , ce qui permet d'augmenter la fenêtre de congestion initiale d'un paquet à 2 voire 4 paquets (pour ne pas dépasser approximativement 4Kb), alors que la notion de MSS (*Maximum Segment Size*) n'existe pas en SCTP, selon la [RFC2960], le CWND initiale doit être initialisé à 2*MTU.
- Les implémentations SCTP sur 2.5G/3G doivent laisser la liberté aux constructeurs de choisir les valeurs de MTU variant des plus petites valeurs (576 octet) aux plus grandes qui sont supportées par le type de lien en usage (comme 1500 octet pour les paquets IP sur Ethernet). Étant donné que la taille de fenêtre est comptée en unité de paquets, un **MTU important** permet au SCTP d'augmenter plus rapidement sa fenêtre de congestion. Puisque les constructeurs sont généralement encouragés de choisir des valeurs importantes, ceci peut dépasser les valeurs par défaut de IP MTU de 576 octet pour Ipv4 [RFC1191] et de 1280 octet pour Ipv6. Alors que cette recommandation est applicable aux réseaux 3G, les opérations sur les réseaux 2.5G doivent prendre des précautions [RFC3150].
- **SCTP *Timestamps Option* (émetteur/récepteur)** : une liaison 2.5G/3G est dédiée à un seul hôte. Elle exécute donc, uniquement un faible degré de multiplexage statique entre différents flux. De même, la transmission de paquet et les retards d'attente d'un lien 2.5G/3G dépassent le RTT du chemin. Ceci engendre d'importantes variations de RTT, du fait que les paquets stockés dans la file de l'émetteur SCTP explorent plus de bande passante, ou du fait que des paquets quittent la file alors que l'émetteur SCTP réduit sa charge en réponse à une perte de paquet. De plus, le retard engendré sur un lien 2.5G/3G peut dépasser même le RTT de bout en bout. Ce qui résulte d'importante variations sur le RTT du chemin et peut causer aussi des hors temps défectueux. L'agent *Timestamp SCTP* introduit des horodatages dans chaque paquet, permettant ainsi à l'émetteur de distinguer les transmissions initiales des retransmissions. En enlevant l'ambiguïté de retransmission, l'algorithme de *Karn* peut être éliminé [KEV03], et des retransmissions fiables sur le chemin supplémentaire peuvent être utilisées afin de mettre à jour le RTT estimé et garder la valeur la plus précise du RTO. Avec l'horodatage, l'émetteur a plus d'échantillons pour mettre à jour le RTT estimé pour le(s) destination(s) supplémentaire(s).

De plus le SCTP offre un acquittement sélectif ce qui permet de réduire le nombre des retransmissions ainsi que les expirations du hors temps.

Nous développons nos simulations sous NS2 (*Network Simulator*) [KEV03]. Le simulateur NS-2 est particulièrement adapté à l'étude de réseau à commutation de paquets et à la réalisation de simulations. Dans nos simulations nous utilisons un agent SCTP développé par le laboratoire

*Protocol Engineering Lab* à *University of Delaware* [ARM03] sous NS-2. Nous exploitons ce simulateur auquel, et selon les besoins de nos simulations, nous ajoutons d'autres fonctionnalités. L'agent SCTP de base, tel qu'il développé par [ARM03], supporte les caractéristiques suivantes :

- Établissement d'une association
- Transmission des *Data Chunks*
- Acquittement des *Data Chunks* reçus
- Gestion du *Timer* de retransmission
- Des points terminaux SCTP à multi-accès (*multihomed)*
- SID et SSN
- Livraison en/hors séquence
- Rapport des trous sur les TSNs des données reçues
- SCTP *slow start congestion Avoidance*
- Détection d'échec au niveau nœud SCTP *(Endpoint Failure Detection)*
- *Path Failure Detection*
- *Path Heartbeat* (sans le contrôle de couche supérieure)

La figure 3.7 présente la structure fondamentale modélisant le SCTP tel qu'il est implémenté sous NS 2. Nous réservons une annexe 2 , dans le présent mémoire, pour valider le simulateur SCTP.

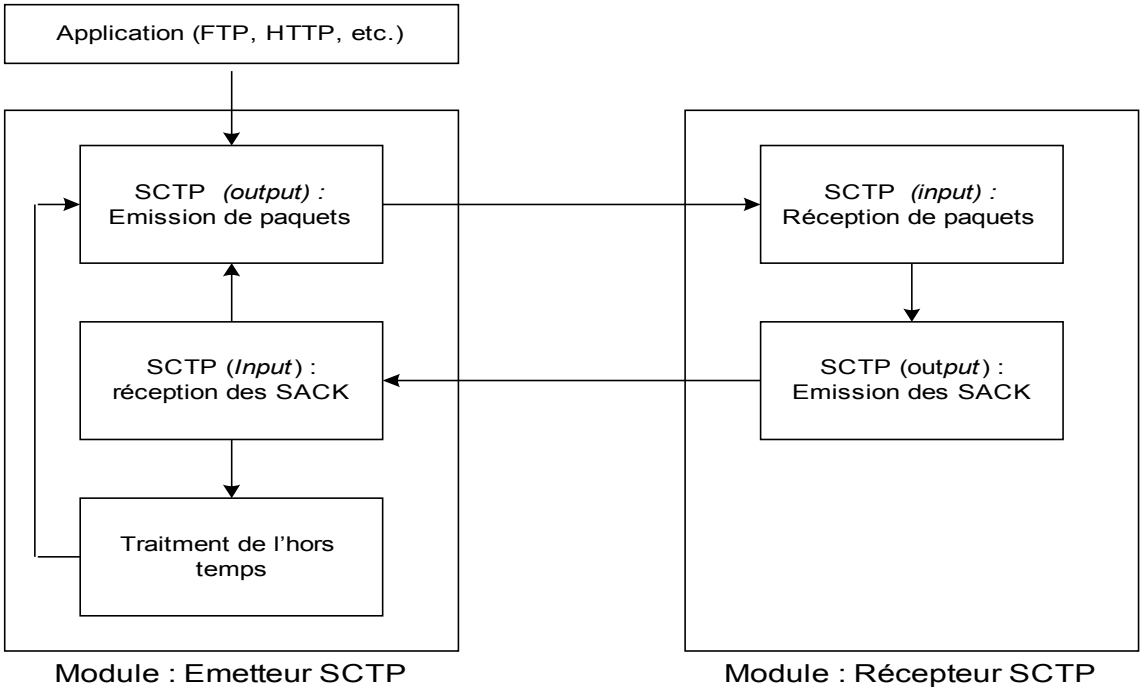



**Figure 3.7.** Structure de la modélisation SCTP sous NS 2.

En ce qui concerne un nœud à multi-accès, l'architecture actuelle de NS ne le permet pas. Par

conséquent pour implémenter cette fonctionnalité, une solution a été proposée dans [KEV03] qui consiste à voir un tel nœud comme l'ensemble d'un *Core Node* et d'un *Interface Node* pour chaque interface simulée (c'est à dire par adresse). Autrement dit chaque adresse ajoutée à un noeud *multihomed* est équivalente à un *interface Node* (c.f. la figure 3.8).

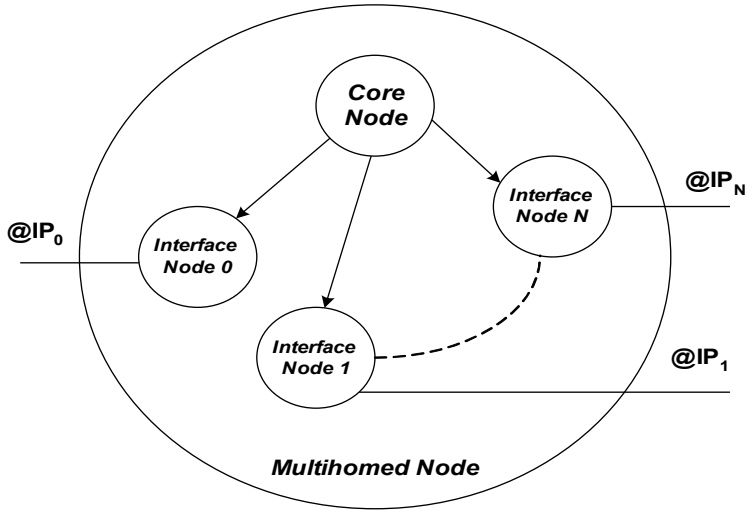



**Figure 3.8.** Nœud SCTP *multihomed* sous NS2**.**

Le trafic est écoulé uniquement à travers (de/vers) les nœuds d'interface, le nœud cœur est utilisé pour le routage et est connecté à chaque nœud d'interface par un lien uni-directionnel vers ces nœuds. Par contre ces liens sont utilisés pour déterminer dynamiquement quel nœud d'interface à utiliser pour émettre vers une destination particulière. Les numéros de TSN, dans l'implémentation, commencent à partir de 1 et un TSN non défini prend la valeur de (-1) afin de contrôler le séquencement des *Chunks* .

Dans notre étude nous considérons une communication entre un hôte fixe et un hôte mobile (dans une première phase). Pour étudier l'interaction entre le protocole de transport SCTP et le Protocole ARQ au niveau couche liaison d'un réseau de type EGPRS nous utilisons un module RLC/MAC qui a été développé à l'ENST [DIA03]. Les simulations menées, sur une durée de 1000sec, dans le présent chapitre ne tiennent pas compte de la fonctionnalité *multihoming*, que nous la désactivons vu que l'objectif visé, à ce niveau, est de montrer l'utilité d'introduire un multiplexage de flot (*multistreaming*) au niveau transport afin de justifier notre choix de SCTP dans nos travaux de recherches.

Le but de nos simulations est de comparer les performances de SCTP avec sa caractéristique de *Multistreaming* sur RLC/MAC EGPRS avec celles du TCP classique sur RLC/MAC EGPRS. Pour le faire nous considérons un service de type interactif au niveau application, c'est le protocole HTTP1.0. Nous choisissons de travailler avec la version HTTP1.0 non pas la version HTTP1.1, vu que cette dernière crée une connexion TCP persistante alors qu'avec HTTP1.0, la connexion était temporaire et qu'autant de session TCP sont créées qu'il y a d'objets à charger dans la page HTML. Ce qui approche l'aspect d'ouverture de plusieurs flots de données, par conséquent la transmission d'un objet par *stream,* l'idée que nous allons la concrétiser par simulations. Nous simulons la

configuration de réseau donnée par la figure 3.9.

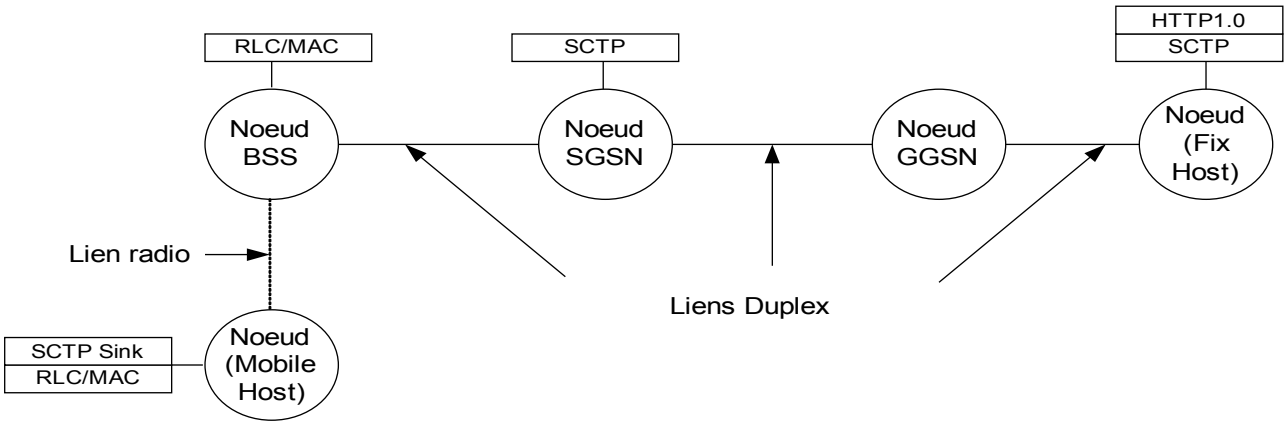



**Figure 3.9.**Configuration du réseau à simuler

## 3.8. Les paramètres de simulations

### 3.8.1 Configuration de la couche RLC/MAC et les paramètres de lien

Comme nous l'avons indiqué ci-dessus nous considérons la technologie EDGE. Les blocs de données RLC sont transmis en utilisant le schéma de codage MCS 4 (*Modulation Coding Scheme N°4*). La taille de la fenêtre RLC est de 384 blocs RLC/MAC. Nous supposons qu'il y a quatre intervalles de temps (*timeslots*) disponibles sur le lien descendant et un intervalle de temps sur le lien montant (4+1). Sur l'interface radio nous considérons un modèle d'erreur blocs uniformément distribué. L'interface Gb est modélisée par un lien duplex dont le débit de transmission est de 64Kbps et le retard de propagation est de 1ms. En ce qui concerne l'interface Gi, est modélisée également par un lien duplex, de débit de transmission relatif de 5Mbps et de retard de propagation de 10ms. Nous introduisons au niveau de cette interface un modèle d'erreur paquets uniformément distribué de taux d'erreur de l'ordre de 1%. Un lien duplex de débit de transmission de l'ordre de 5Mbps et de retard de transmission de 10ms est considéré entre le GGSN et le serveur.

### 3.8.2 Les paramètres couche transport

Au niveau transport nous utilisons deux types de protocoles comme c'est indiqué ci-dessus (TCP et SCTP) afin de pouvoir comparer l'apport d'un nouveau protocole de transport par rapport à TCP. Les paramètres utilisés sont donnés par les tableaux 3.1 et 3.2.

| | **Mobile** | **IP server** |
|---|---|---|
| MTU | 1500 bit | 1500bit |
| DataChunkSize | 1468 bit | **1468bit** |
| OutStreams | 3 | **3** |
| Initial Cwnd | 2 | **2** |
| Initial Rwnd | **64Kbit** | 64 Kbit |
| Initial RTO | 1sec | 1sec |

Tableau 3.1. Les paramètres de l'agent SCTP

| | **Mobile** | **IP server** |
|---|---|---|
| MTU | 1500 bit | 1500bit |
| Packet size | 1460 | **1460bit** |
| Initial Cwnd | 2 | **2** |
| Initial Rwnd | **64Kbit** | 64 Kbit |
| Initial RTO | 1sec | 1sec |

Tableau 3.2Les paramètres de l'agent TCP

### 3.8.3 Validation du simulateur SCTP

Avant de simuler une couche applicative de type HTTP 1.0, nous avons fait des simulations avec le protocole FTP, dans les mêmes conditions décrites ci-dessus. L'idée est que dans le cas de transmission d'un flux continu de données, SCTP et TCP opèrent de façons similaires puisque ils ont des mécanismes de contrôle de congestion et de contrôle de flux proches si non pas identiques. Nous représentons dans la figure 3.10, l'évolution de la taille de fenêtre de congestion au cours du temps pour un trafic FTP en environnement EGPRS.

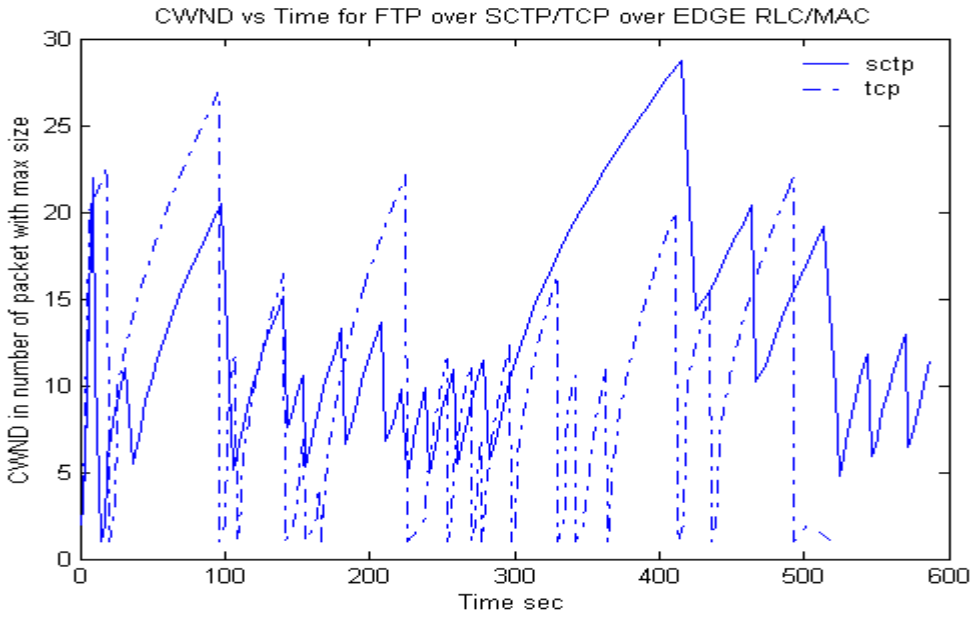



**Figure 3.10.** Évolution de CWND au cours du temps pour une application FTP sur SCTP resp TCP sur une couche RLC/MAC EGPRS.

Nous remarquons, comme c'est prévu, que les deux protocoles de transport SCTP et TCP ont un comportement comparable pour la transmission d'un flux de données continu de type FTP particulièrement dans un environnement radio EGPRS. Ce résultat est conforme à celui présenté par les auteurs dans [RAJ02], où ils ont montré que les deux protocoles ont un comportement comparable pour la transmission d'un flux FTP sur un réseau filaire (fixe). Après avoir validé le bon fonctionnement du simulateur nous poursuivons nos simulations mais maintenant en activant une session web, dans laquelle nous supposons l'existence d'objets incorporés. Dans le but de prouver l'utilité de considérer un nouveau protocole autre que le TCP, nous mettons l'accent sur l'aspect *Multistreaming*.

### 3.8.4 Les paramètres couche application

Les transactions HTTP1.0 générées lors des simulations sont de type client (MS) / serveur (*IP server*) selon le modèle développé dans [FHP01]. Le schéma de séquencement de ces transactions est donné par la figure 3.11.

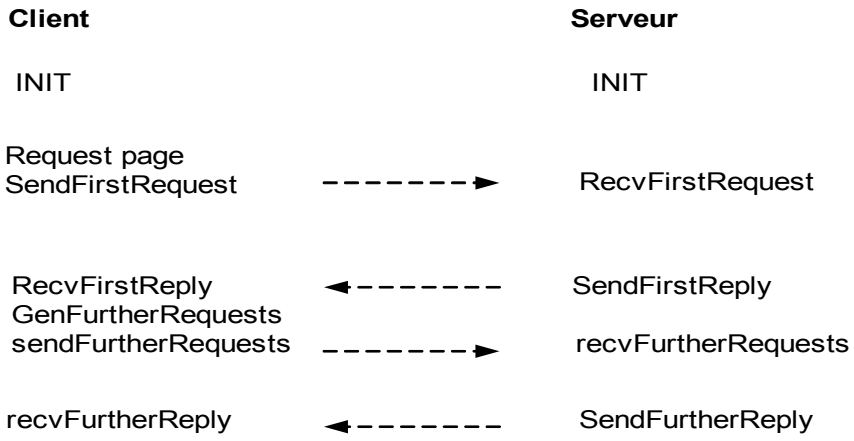



**Figure 3.11.** Les interactions Client/Serveur pour télécharger une page Web

Une page Web générée par ce modèle est composée des éléments suivants [BRU97] :

- La taille d'une page Web est une variable aléatoire de distribution de Pareto, de taille moyenne de 10Koctet et pour $\alpha = 1.2$
- Une page contient des objets incorporés dont le nombre et la taille sont des variables aléatoires de distribution de Pareto. La taille moyenne des objets est de 4Koctets et $\alpha = 1.1$. Le nombre moyen des objets, que nous considérons dans nos simulations, est 3 et $\alpha = 1.5$.

## 3.9. Résultats obtenus et commentaires

Dans une première étape nous montrons l'apport de SCTP par rapport à TCP pour transporter un

trafic de type HTTP1.0, dans le cas où un seul mobile est en communication dans une cellule EGPRS. Pour le faire nous simulons la configuration du réseau donnée par la figure 3.9 pour un seul mobile et avec un seul *stream* ouvert au sein de l'association SCTP établie entre le client mobile et le hôte fixe (serveur). Les résultats obtenus avec SCTP sont comparés à ceux obtenus avec TCP auquel nous activons l'option SACK afin de se situer dans les mêmes conditions, au niveau couche transport, qu'avec le SCTP. La figure 3.12 représente les variations du temps de réponse en fonction du taux d'erreur blocs (BLER : *Bloc Error Rate*) sur l'interface radio pour un seul client mobile. Le temps de réponse considéré correspond à la différence entre l'instant d'émission de la demande de téléchargement et l'instant de réception de la première réponse. Nous constatons que les deux protocoles de transport ont un comportement comparable pour de faibles taux d'erreurs blocs. Ceci s'explique par le fait que les deux protocoles ont le même mécanisme de contrôle de congestion. Et comme pour SCTP nous ne considérons pas le *multihoming* et qu'un seul *stream* est activé alors dans ce cas une simple association SCTP est équivalente à une connexion TCP.

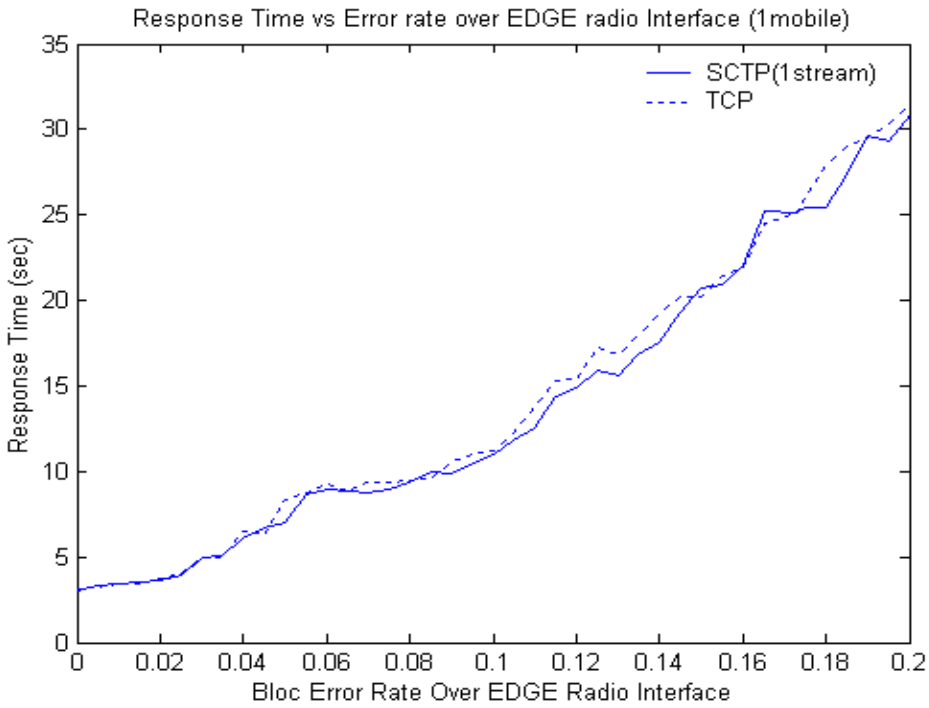



**Figure 3.12.** Temps de réponse en fonction du taux d'erreurs bloc pour HTTP1.0 au dessus de TCP et SCTP (*1stream*) au dessus de RLC/MAC EDGE.

La figure 3.13 illustre l'amélioration de performance apportée par le *multistreaming* au niveau d'une association SCTP par rapport au TCP. En fait, nous considérons une association SCTP au cours de laquelle il a eu lieu d'ouvrir 3 *streams*. Les résultats obtenus montrent bien l'amélioration de performance introduite par ce mécanisme au niveau transport, ce qui est dû au fait que le *multistreaming* permet d'éviter les problèmes de *head of line blocking* donc accélération du temps de réponse.

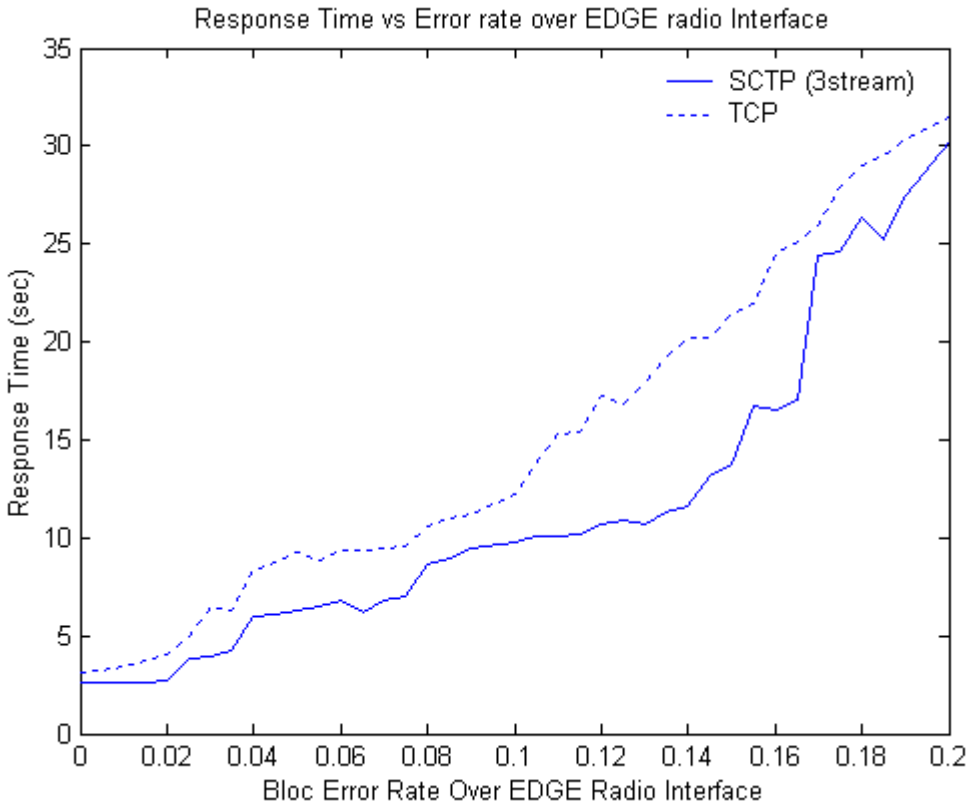



**Figure 3.13.** Temps de réponse en fonction du taux d'erreur blocs pour HTTP1.0 au dessus de TCP et SCTP (*3streams*) au dessus de RLC/MAC EDGE.

Notons que pour de faibles taux d'erreur bloc, la différence entre les deux protocoles de transport est négligeable. Par exemple pour un taux d'erreur bloc égal à 0.001, nous avons une différence entre les temps de réponse, relevés relativement à chacun des deux protocoles, est de moins de 0.4 sec. Tandis que nous apercevons une différence remarquable entre les temps de réponses pour chacun des deux protocoles avec des taux d'erreur plus élevés. En fait, SCTP performe mieux que le TCP pour des taux d'erreurs bloc élevés. Prenons par exemple un taux d'erreur égal à 0.14, le temps de réponse perçu par une couche transport SCTP est inférieur de plus de 5 sec à celui perçu par une couche TCP. Ceci est dû à l'activation du *multistreaming,* au sein de l'association SCTP, qui accélère le temps mis pour le téléchargement des pages web, suite à la résolution du problème de *head of line blocking*. Cette amélioration, apportée par SCTP, sera plus claire si nous augmentons la charge de la cellule EGPRS considérée.

Dans ce qui suit nous traitons le transfert de trafic HTTP1.0 dans une cellule EGPRS chargée. Nous supposons que nous avons quatre mobiles dans une cellule EGPRS et nous comparons les résultats obtenus entre TCP et SCTP pour lequel 3 *streams* sont activés. La figure 3.14 représente les variations des temps de réponse en fonction du taux d'erreurs bloc sur l'interface radio pour chacun des quatre mobiles desservis par la cellule EGPRS.

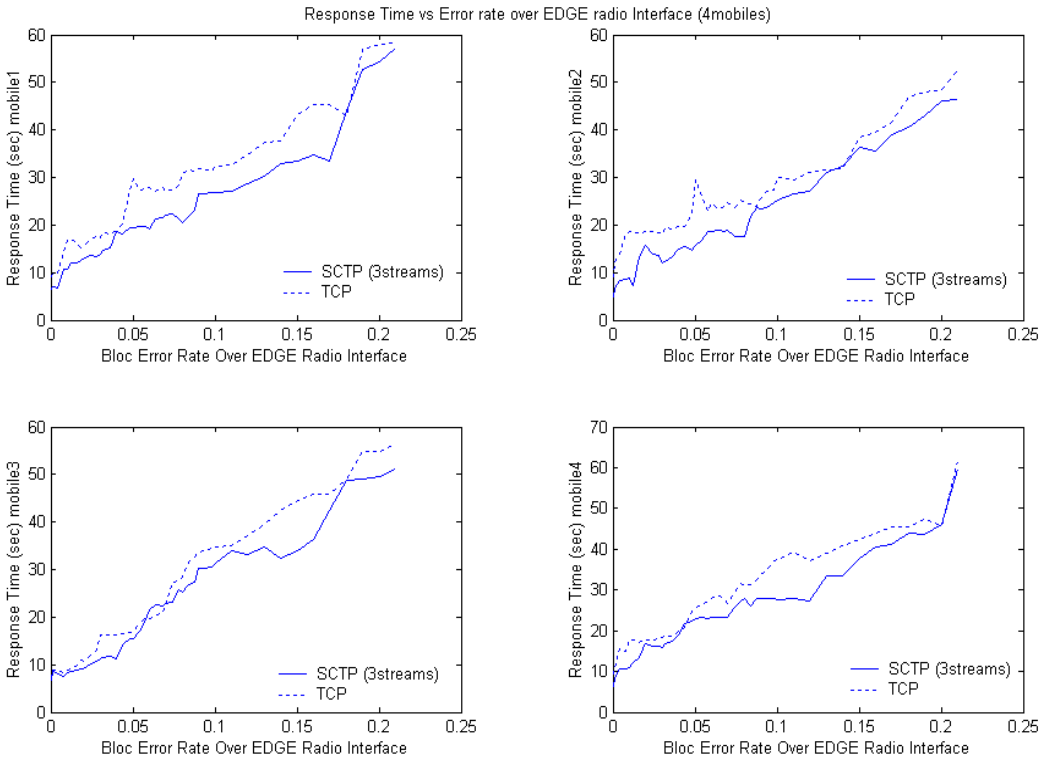



**Figure 3.14.** Temps de réponse en fonction du taux d'erreur blocs pour HTTP1.0 au dessus de TCP et SCTP (*3streams*) au dessus de RLC/MAC EDGE (pour 4 mobiles).

Les résultats obtenus montrent bien que SCTP offre une meilleure performance que TCP. Ceci est dû au problème de *head of line blocking* qui dégrade les performances de TCP avec la production des erreurs de transmission sur l'interface radio. En effet, la transmission est bloquée au niveau de l'émetteur TCP jusqu'à la retransmission des segments erronés (cf. figure 3.15). Malgré que les PDU n°5, 6 et 7 sont bien reçus par le récepteur TCP, ne seront remontés à l'ULP que lorsque le PDU n°4 sera retransmis. Par conséquent, le PDU n°4 bloque la livraison des données aux couches supérieures ce qui engendre un blocage HOL. Tandis que dans le cas de SCTP lors de production d'une erreur la transmission n'est bloquée qu'au sein du *stream* concerné si l'ordre est exigé (cf. figure 3.16), et l'émetteur peut continuer à transmettre des paquets HTTP1.0. Dans l'exemple présenté par la figure 3.16 seulement les *chunks* contenus dans le *stream* 1 qui ne sont pas remontés aux couches supérieures suite à la perte du *chunk* n°2 du *stream* 1. En fait, un *stream* comme nous l'avons décrit antérieurement dans ce mémoire, est un canal logique unidirectionnel établi au sein d'une association entre deux points terminaux SCTP. Dans un *stream* les messages utilisateurs sont remontés en séquence sauf ceux qui sont soumis pour un service de livraison hors séquence (U=1).

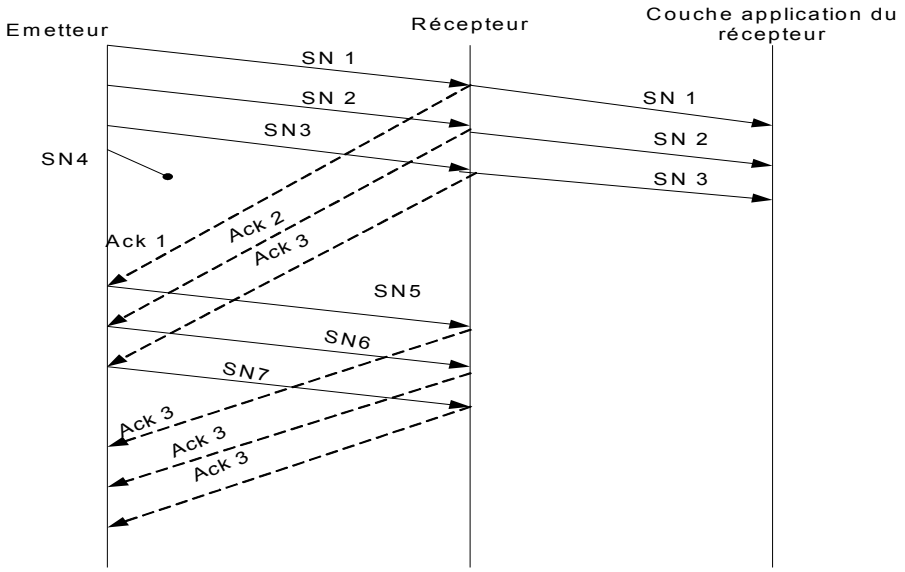



**Figure 3.15.** *Head of Line Blocking* en TCP : Le PDU 4 bloque la livraison des données à la couche supérieure.

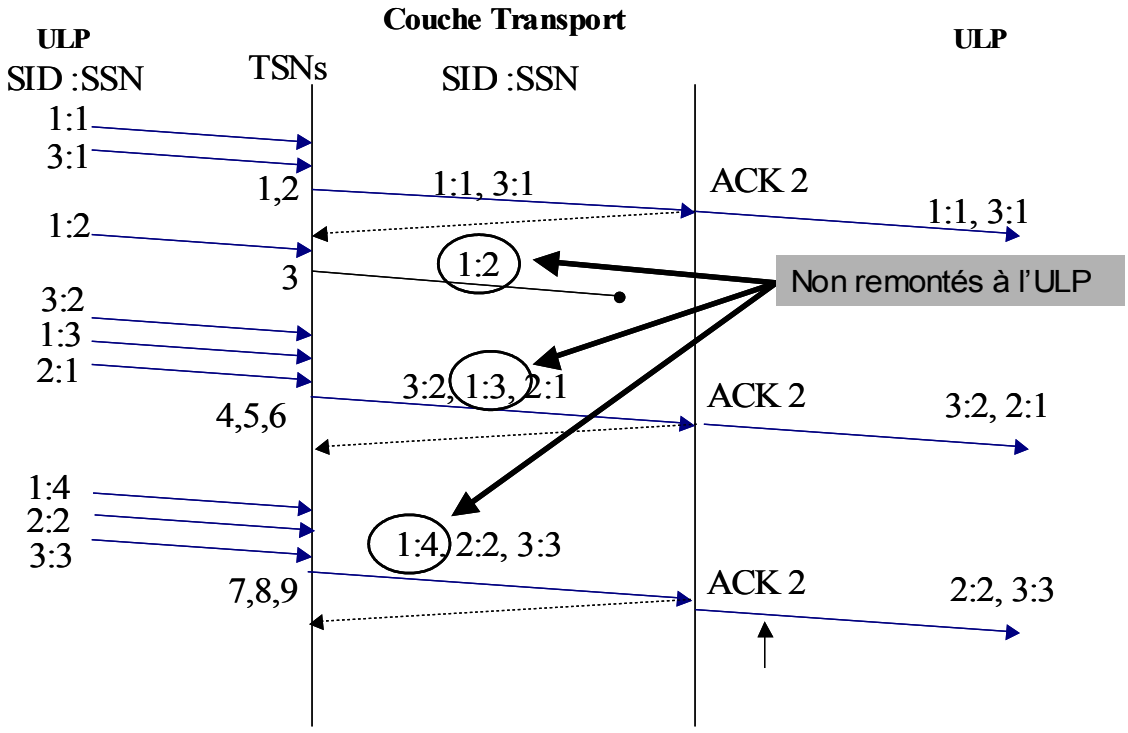
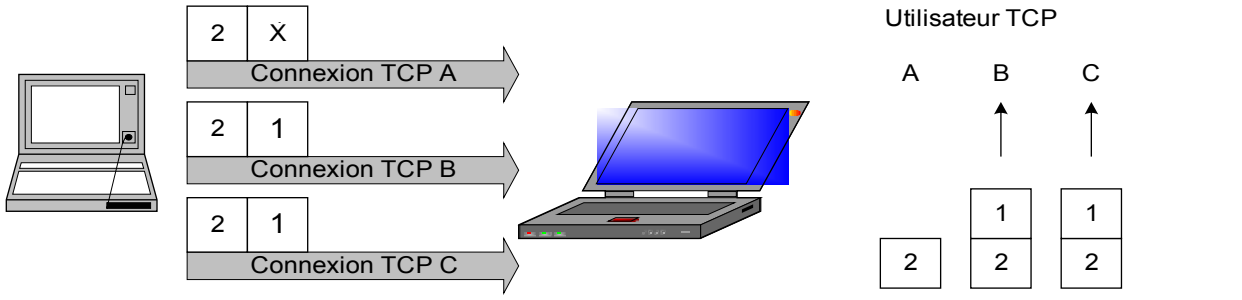
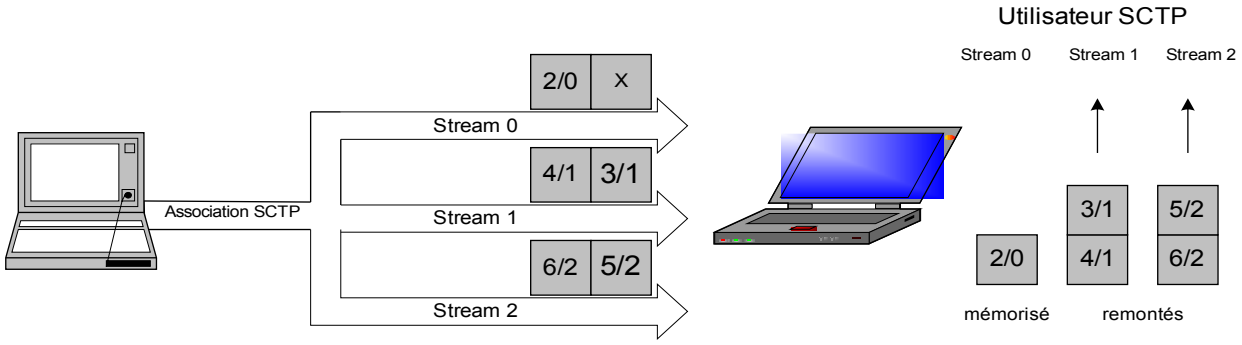



**Figure 3.16.** Illustration du *multistreaming* en SCTP (ici l'ordre est exigé U=0).

Normalement lors de transfert de données avec le protocole HTTP, le client ouvre différentes connexions TCP, une par fichier, qui sera fermée une fois que le fichier est complètement transféré (cf. figure 3.17 (a)). Cette approche permet d'éviter le problème de HOL *blocking* étant donné que les fichiers sont livrés à l'utilisateur final dès leurs arrivées (dans l'exemple donné par la figure 3.17 (a) le premier fichier ne sera livré qu'après la bonne réception du premier segment). Cependant, cette solution augmente les délais de transmission engendrés par l'ouverture et la fermeture des connexions TCP, ce qui provoque le gaspillage de ressources par le fait d'avoir plusieurs connexions TCP ouvertes en même temps entre les mêmes points terminaux. Comme les serveurs TCP ont une limitation du nombre de connexions TCP [IVA02] qu'ils peuvent ouvrir en même temps, dont la majorité est exploitée avec le même client, ce qui diminue le nombre de clients qui peuvent être servis simultanément. Ce problème est résolut par SCTP en utilisant une seule association avec différents *streams* pour différents fichiers (cf. figure 3.17 (b)). Avec cette solution le serveur peut économiser des ressources, étant donné qu'il ne peut avoir qu'une association par client. En plus du fait que nous économisons des ressources et nous évitons les retards dus aux ouvertures des connexions TCP, tous les *streams* dérivant de la même association, partagent les mêmes mécanismes d'évitement de congestion. Alors que chaque connexion TCP a ses propres paramètres d'évitement de congestion, ce qui implique un partage excessif de la bande passante.




(a) Une connexion TCP par fichier




*(b)* une association SCTP avec plusieurs *streams*

**Figure 3.17.** Problème de *Head of Line blocking*.

Nous notons également que lorsque nous augmentons la charge sur l'interface radio, les deux protocoles convergent vers un comportement similaire. Nous constatons alors que l'avantage de SCTP est susceptible de disparaître quand le réseau commence à devenir congestionné.

Pour bien visualiser l'avantage d'introduire SCTP dans un environnement radio au lieu de TCP, il s'avère judicieux de voir le comportement au niveau de la couche RLC/MAC. Des traces sur l'évolution du numéro de séquence des blocs RLC en fonction du temps sont données par les figures 3.18 et 3.19. La figure 3.18 est obtenue pour le protocole TCP et la figure 3.19 est relative au protocole SCTP . Elles confirment les résultats observés sur les figures 3.13 et 3.14. En effet, pour SCTP nous pouvons constater que le débit au niveau RLC est plus important qu'en TCP. Ceci est dû au fait que le mécanisme de *multistreaming* est activé et sert pour compenser les erreurs de transmission sur l'interface radio et par suite résoudre les limitations du TCP. Nous remarquons également, que ces améliorations de performances se restreignent tant que le réseau n'est pas congestionné. En fait, le contrôle de congestion en SCTP est déclenché par adresse de transport et non pas pour chaque *stream*, donc pour améliorer la performance du SCTP il fallait introduire le mécanisme de *multihoming*.

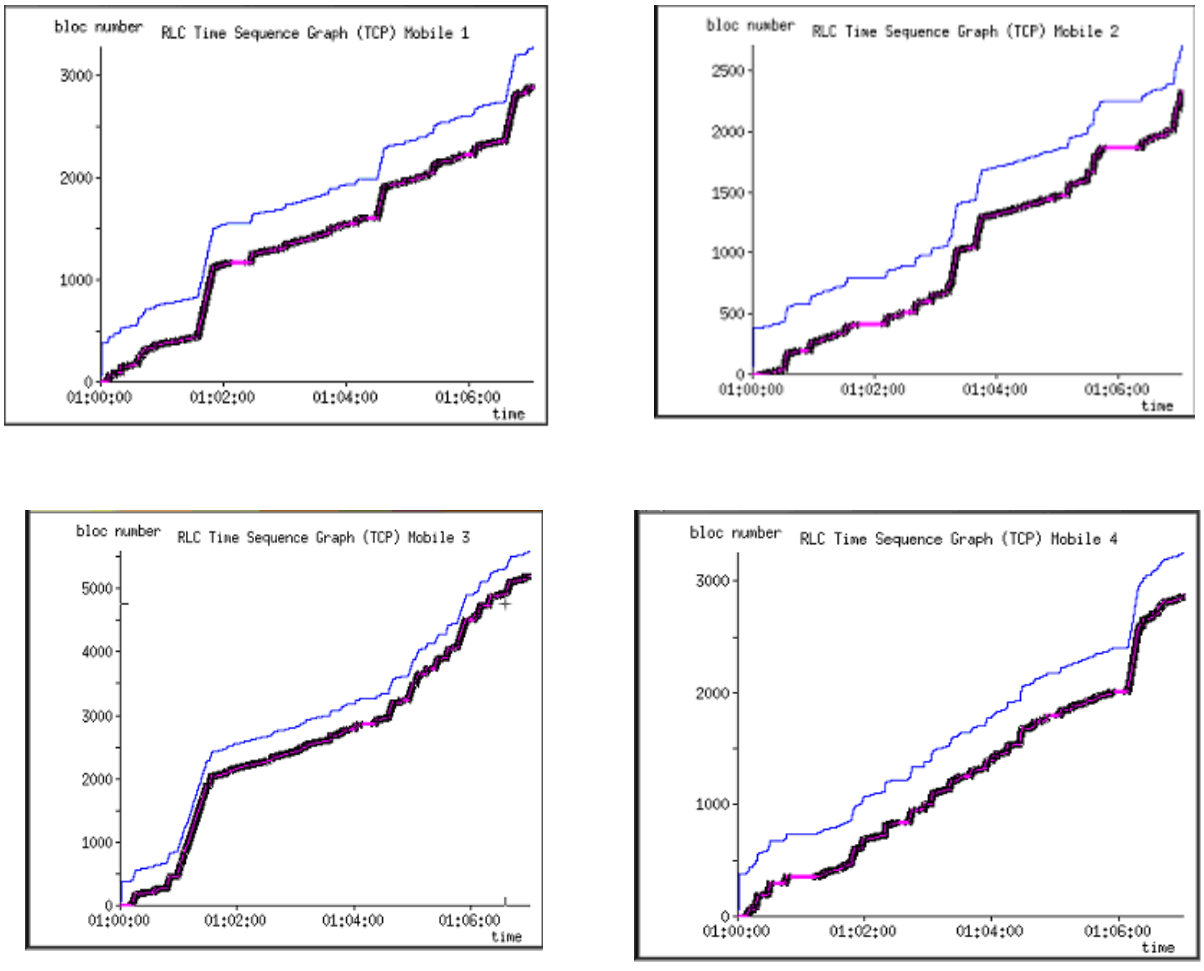



**Figure 3.18.** Numéro de séquence des blocs RLC pour HTTP1.0 au dessus de TCP au dessus de RLC/MAC EDGE

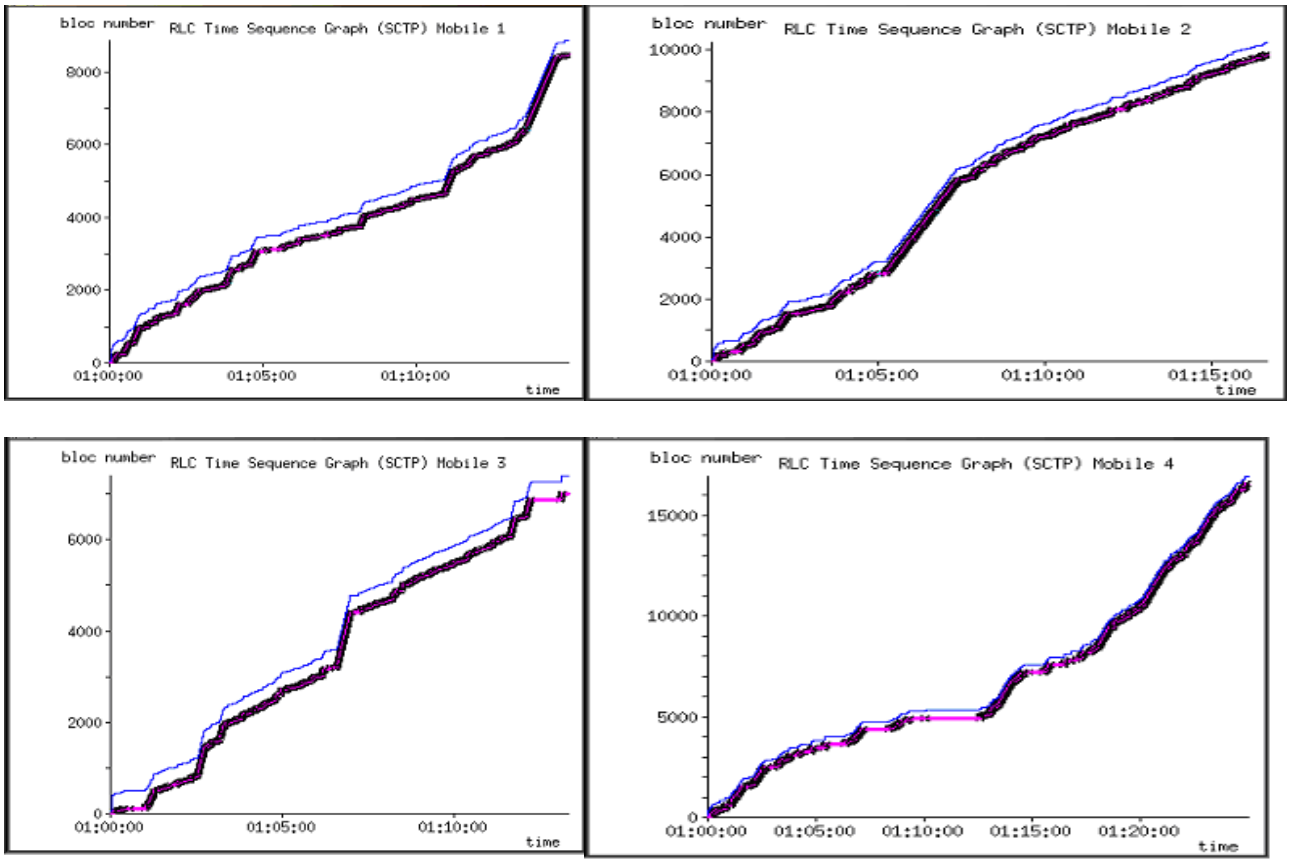



**Figure 3.19.** Numéro de séquence des blocs RLC pour HTTP1.0 au dessus de SCTP au dessus de RLC/MAC EDGE

## 3.10. Conclusion

Au sein d'une association SCTP l'ouverture de plusieurs *streams* permet d'éviter le problème de HOL *blocking*. Cet aspect a été démontré par plusieurs études antérieures dans un contexte filaire [RAJ02]. Ce chapitre s'est focalisé à prouver l'avantage du *multistreaming* dans un réseau EGPRS pour une application interactive de type HTTP1.0. Les simulations que nous avons menées ont montré que SCTP offre un service plus fiable que le TCP. En fait, les évolutions du temps de réponse, visualisées par l'application utilisateur, en fonction du taux d'erreurs bloc sur l'interface radio sont meilleures dans le cas du SCTP que dans le cas de TCP. Mais cet avantage reste valable dans le cas d'un réseau non congestionné. C'est que d'après les résultats obtenus, en augmentant la charge du réseau, SCTP avec sa caractéristique de *multistreaming* présente un comportement comparable à TCP. À partir de là deux orientations d'études sont possibles.

La première repose sur une modification de la fonctionnalité de *multistreaming*. Il s'agit d'introduire un aspect de gestion de priorité inter-*stream* et essayer de configurer les *streams* sur les flux RLC/MAC en EGPRS afin de différentier la qualité de service et les priorités relatives aux différents flux de données sur l'interface radio. Une solution a été proposée dans [GER04] qui consiste à ajouter un bit de priorité au *stream*. Dans le but de privilégier un *stream* prioritaire par rapport à un autre *stream* non prioritaire. Cependant, cette approche, gérée du côté émetteur SCTP, s'avère restrictive dans la mesure où deux niveaux de priorité ne suffisent pas notamment dans les réseaux multi-services de nouvelles générations. Cette alternative de priorité inter-*streams,* malgré son optimalité sur le plan d'exploitation de bande passante reste insuffisante, vu qu'elle ne résout pas le problème de contrôle de congestion déclenché suite à des erreurs sur l'interface radio.

La deuxième approche, qui constitue la suite de nos travaux de recherches au sein de cette thèse, vise une gestion de la mobilité en fonction des fonctionnalités de la couche SCTP. Nous nous intéressons dans la suite de ce mémoire à l'exploitation de la caractéristique *multihoming*. Nous proposons une modification du protocole de contrôle de congestion en SCTP de façon à lier les réactivités des couches supérieures aux évènements des couches basses (RLC/MAC en EGPRS). Cette modification servira principalement à la gestion de *handover* inter et intra RAT (*Radio Access Technology*), ce qui constitue l'objectif du prochain chapitre.

# Chapitre 4 : *Multihoming* et *Handover Data* intra et inter-RAT

## 4.1. Introduction

Le *multihoming*, une des caractéristiques de SCTP, réfère à une situation où un hôte terminal ait plusieurs chemins de communication qu'il peut emprunter [GUN04]. En SCTP, le *multihoming* fournit une redondance entre deux noeuds distants. Chaque noeud peut être accessible par plusieurs adresses IP négociées lors de l'établissement de l'association. C'est grâce à cette caractéristique que les performances de transmission de données sur les réseaux de radiocommunications, se sont améliorées [SUK03].

Plusieurs extensions ont été définies pour adapter SCTP aux environnement radio, le *mobile SCTP* (mSCTP) par exemple. Cependant, les approches proposées dans la littérature des protocoles de transport, ne tiennent pas compte de l'information radio dans la gestion des resélections de cellules et de *handover*. En nous inspirons des études antérieures dans le domaine de gestion de *handover* intra et inter-RAT et de l'aspect *multihoming* de SCTP couplé avec la mobilité, nous proposons dans le présent chapitre une nouvelle extension de SCTP qui vise l'amélioration des performances de la couche transport dans un contexte de transmission radio. Nous définissons dans ce but un nouveau *chunk* de contrôle SCTP : le *Qos_Measurement_Chunk*. Ce *chunk* permet la transmission de l'information du client mobile vers son point terminal SCTP associé. Cette information est utilisée par la couche SCTP pour adapter le débit de transmission aux conditions de transmission sur l'interface radio (particulièrement utiles dans une situation de *Handover* ou de resélection de cellule).

Ce chapitre se focalise sur la présentation de notre proposition qui consiste à introduire des modifications au protocole SCTP. Il s'agit du *QoS_Measurement_Chunk,* cette extension de SCTP est basée d'une part sur un mécanisme *cross-layer* entre SCTP et la couche liaison de données, d'autre part sur la création d'un nouveau *chunk* de contrôle SCTP. L'utilisation combinée de *Qos_Measurement_Chunk* et du mécanisme de *multihoming,* assure une amélioration des performances de gestion de *handover* de données. D'abord nous situons notre approche par rapport aux solutions et alternatives proposées qui visent le couplage du *multihoming* et de la mobilité pour une meilleure gestion de *handover* de données dans un environnement présentant une hétérogénéité de technologies d'accès radio. Ensuite, nous décrivons les scénarii simulés basés sur la gestion de

*handover* inter et intra RAT. Enfin, nous montrons à l'aide des résultats obtenus l'amélioration de performances introduite avec le mécanisme de *QoS_Measurment_Chunk* par rapport au SCTP standard avec *multihoming*.

## 4.2. Formalisation de la modification proposée de SCTP

Dans les réseaux actuels, la mobilité d'un réseau vers un autre constitue un sujet de recherche vers lequel plusieurs études se sont orientées. Assurer la continuité d'une communication en basculant le lien d'un certain type de réseau d'accès vers un autre, n'est pas toujours évident. *Mobile* SCTP offre un moyen pour assurer cette continuité. Cependant, mSCTP, pour assurer la gestion de mobilité doit être couplé à *Mobile* IP. En effet, la gestion de mobilité se compose de deux phases. La première phase, assurée par *Mobile IP*, concerne la gestion de localisation afin d'identifier la localisation d'un MN (*Mobile Node*). Tant que la deuxième phase est à la charge de mSCTP, elle gère le changement de lien avec le *multihoming* et l'adressage dynamique.

Malgré ses insuffisances, mSCTP couplé au *Mobile* IP présente la solution de base la plus adoptée pour gérer un *handover* inter RAT (*Radio Access Technology*). Des études comme [LUO03, PED04, YLI03, MIH02, AAH03a, AAH03] proposent des mécanismes qui coopèrent avec les protocoles de mobilité à différents niveaux : au niveau transport on utilise le *Mobile SCTP* et au niveau réseau le *Mobile IP*. Un gestionnaire de mobilité est introduit afin de configurer dynamiquement les interfaces pour différentes RAT au niveau du MN (*Mobile Node*). D'autres travaux, comme nous l'avons signalé dans le second chapitre, se basent sur mSCTP. Le *cellular* SCTP, par exemple, introduit une amélioration pour palier aux insuffisances de mSCTP, mais reste restrictif c'est qu'il permet la transmission dupliquée des paquets de données, sur deux adresses primaires activées en même temps, ce qui engendre une redondance au niveau du récepteur SCTP et donc gaspillage de la bande passante. Aussi l'idée de conditionner le changement d'adresse IP par la puissance du signal radio permet d'améliorer les performances de mSCTP mais ne met pas en question le mécanisme de contrôle de congestion au niveau couche transport.

D'autres auteurs, cherchent la solution de réalisation de *handover* inter-RAT sans pertes et optimale à partir des couches basses. Une approche a été proposée par J.Sachs [SAC03], il s'agit de concevoir une couche GLL (*Generic Link Layer*). L'intégration de ce concept dans les futurs réseaux d'accès radio, permet d'avoir des *handovers* inter-systèmes fiables et sans pertes. L'une des raisons d'introduction de la couche GLL [BEM04, SAC04] est de favoriser un traitement générique de données pour différentes technologies d'accès radio. En fait, le concept GLL permet l'intégration de différentes RATs avec des caractéristiques et des exigences hétérogènes [SAC04a]. En effet, la *Generic Link Layer* est une entité qui fournit une couche de liaison de données fonctionnant pour différentes technologies d'accès radio et peut être identifiée comme un bloc de fonctions couche liaison qui peut être adapté facilement aux caractéristiques des nouvelles technologies d'accès radio. La couche GLL constitue une interface unifiée aux couches supérieures (cf. figure 4.1) agissant comme une couche de convergence de plusieurs RATs, et cachant l'hétérogénéité des environnements fondamentaux de plusieurs RAT. Elle contrôle et améliore les fonctionnalités de la couche RLC/MAC de différentes RAT supportées, dans le but d'augmenter les performances de la couche application. La GLL fournit une architecture modulaire qui couvre facilement l'intégration et la co-opération de différentes types de RATs [MAG04].

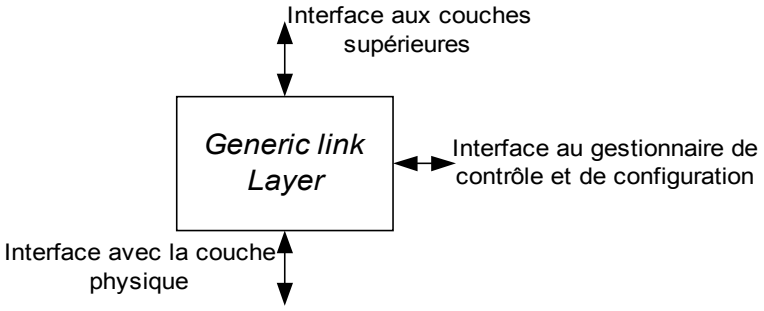



**Figure 4.1.** Fonctions et interfaces GLL [SAC03].

Une entité GLL, dans son apparence permet d'assurer un *handover* inter-RAT fiable et sans pertes. Ce concept reste incomplet, c'est qu'une GLL, pour accomplir sa mission, nécessite une fonctionnalité de gestion des ressources radio dans un réseau à multi-accès radio. En effet, une entité GLL, localisée entre le plan de contrôle et le plan utilisateur doit avoir la possibilité d'un changement dynamique du chemin des flux de données des couches supérieures entre différentes RATs. De plus, tel qu'elle est conçue, une GLL ne met pas en cause le contrôle de congestion au niveau des couches supérieures. En fait, les évènements au niveau couches RLC/MAC et physique restent ambigus aux couches supérieures. Dans ce cas les erreurs sur l'interface radio, pour n'importe quelle technologie d'accès radio, sont toujours interprétées comme étant des problèmes de congestion au niveau transport. Par conséquent concevoir un mécanisme qui assure l'interaction entre les couches transport et liaison est indispensable pour assurer la fiabilité des *handover* inter-RATs.

En résumé, une proposition de gestion de *handover* inter-RAT doit assurer deux fonctions principales :

- Configuration dynamique des interfaces pour différentes RATs,
- Traitement générique de données pour différentes technologies d'accès radio.

Nous présentons dans la suite notre solution de gestion de *handover* inter-RAT, basée sur le protocole SCTP. Notre approche consiste à introduire des modifications sur le mécanisme de contrôle de congestion au niveau SCTP afin de décorréler les problèmes de congestion des dégradations du support de transmission au niveau RLC/MAC. Pour le faire nous aurons besoin de remonter des mesures de qualité de services (QoS), relevées au niveau de l'interface radio, à la couche SCTP, en exploitant les caractéristiques fondamentales de SCTP à savoir le *multihoming* et la génération des *chunks* de contrôle. En nous inspirons des alternatives proposées avec mSCTP, cSCTP ou GLL, une solution de bonne gestion de *handover* inter-RAT doit garantir la convergence des mécanismes de gestion de mobilité, de contrôle de congestion et de différents critères de QoS sur les RATs (cf. figure 4.2).

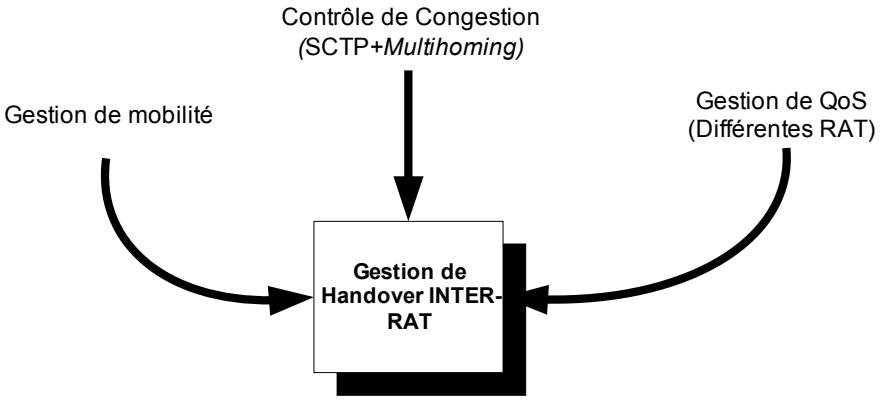



**Figure 4.2.** Solution proposée pour la gestion de *handover* inter-RAT.

## 4.3. Contexte générale des simulations

Nous représentons dans la figure 4.3 la configuration de réseau avec laquelle nous allons simuler notre proposition de modification du protocole SCTP. Les simulations consistent à étudier différents scénarii de *handover* dans un réseau EGPRS puis vers un réseau WLAN (*Wireless Local Area Network*), en variant les paramètres de la couche transport et en les adaptant aux conditions radio. La décision de *handover* est à la charge du réseau. C'est le comportement sur le lien descendant que nous envisageons dans notre étude. Au niveau transport nous considérons non seulement le SCTP standard et notre modification mais également nous considérons le TCP avec l'une de ses propositions d'aménagement le Eifel, qui seront présentés dans l'annexe 3. Le fait, de simuler différents protocoles de transport nous permet de bien justifier l'apport de notre modification introduite au protocole SCTP standard

Nous considérons une association SCTP (resp une connexion TCP) de bout en bout entre le mobile et son serveur IP (le Hôte fixe).

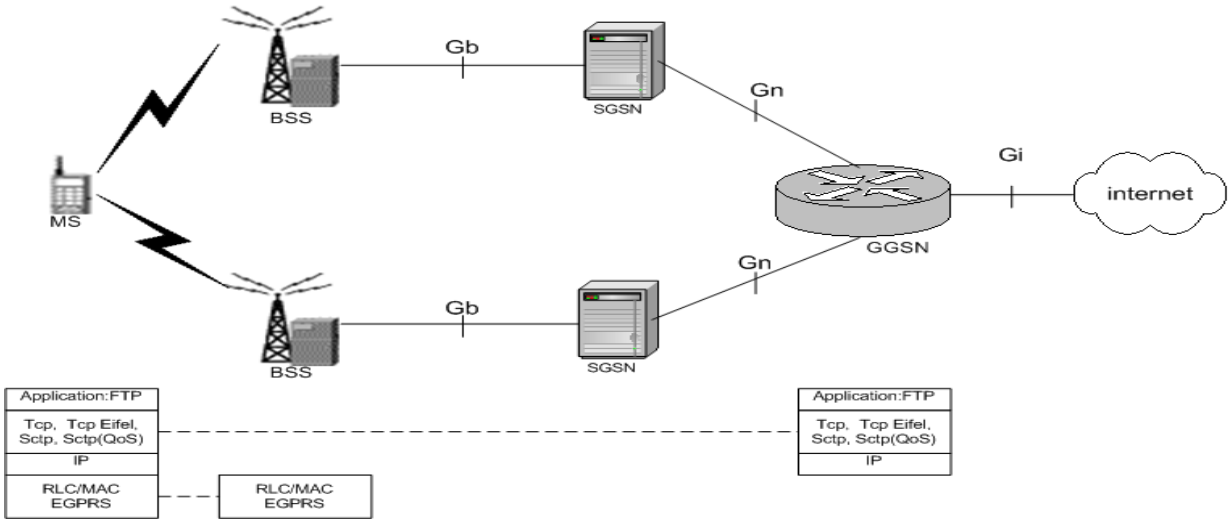



**Figure 4.3.** Architecture simulée.

Lors de l'établissement d'un contexte PDP (*Packet Datat Protocol context*) entre le mobile et le GGSN, une association SCTP (resp une connexion TCP) est également établie entre ces deux équipements. L'association SCTP est «multihomée» du côté serveur IP (GGSN). Par contre, le mobile ne dispose que d'une seule adresse IP (invariante pendant toute la durée de vie du PDP contexte). L'adresse primaire de l'association SCTP correspond à une adresse IP située sur le même sous réseau que le SGSN auprès duquel le mobile a établi le PDP contexte. Cette adresse peut être amenée à changer si le mobile changera de SGSN. Dans ce cas le mobile changera d'adresse primaire de destination (parmi la liste d'adresses secondaires attribuées au moment de l'ouverture de l'association).

### 4.3.1 Paramètres des simulations

Nos simulations sont développées sous une plateforme NS2.27 (*Network Simulator 2.27*) [KEV03]. Nous cherchons par ces simulations à étudier le comportement de la couche transport conjointement au comportement au niveau RLC/MAC dans un réseau EGPRS. Les modifications que nous souhaitons introduire viseront le protocole SCTP. Au niveau couche liaison nous exploitons un simulateur RLC/MAC GPRS/EGPRS. Il s'agit d'une évolution du GPRS RLC/MAC [3GPP0460]. Cette technologie permet la transmission de données à haut débit sur l'interface radio. EGPRS présente plusieurs modifications, telles que l'augmentation de la taille de fenêtre et l'utilisation de la redondance incrémentale (IR *Incremental Redandancy*). Il utilise une nouvelle modulation (8-PSK) et donc de nouveaux schémas de codage qui augmentent la capacité par intervalle de temps (*timeslot*). Les tailles de fenêtre peuvent être choisies par TBF (*Temproray Bloc Flow*) entre 64 et 1024 blocs avec un pas de 64. Elles dépendent également du nombre d'intervalles de temps attribués au TBF. Afin de tester le *handover* inter-RAT nous utilisons le simulateur MAC 802.11 développé sous NS2.27 .

Sur l'interface radio nous utilisons un schéma de codage de type cs3 (C*oding Scheme n°3*) pour transmettre les blocs RLC, avec une taille de fenêtre de 384 blocs RLC/MAC. A un instant donné, un TBF ouvert utilise 3 PDCH (*Packet Data CHannel*) en *downlink* et 1 PDCH en *uplink*.

L'interface Gb est modélisée par un lien duplex de débit de transmission de l'ordre de 64kbps avec un *delay* de propagation de 100ms. L'interface Gi est modélisée par un lien duplex de débit de transmission de l'ordre de 5Mbps avec un *delay* de propagation de 100ms. Enfin le lien entre GGSN et le hôte fixe est un lien duplex de 5Mbps et de retard de propagation de l'ordre de 100ms relativement à chacune des deux interfaces du serveur IP.

Le modèle d'erreur sur la partie fixe, interface Gi, est uniformément distribué avec un taux d'erreur paquets moyen de 1%. Sur l'interface radio nous utilisons un modèle d'erreur uniformément distribué avec un taux d'erreur blocs (blocs RLC/MAC) moyen variable au cours des simulations.

## 4.4. Modification de SCTP : *QoS-Measurement-chunk*

Dans cette partie nous simulons le scénario donné par la figure 4.3 et nous utilisons deux versions de SCTP à savoir le SCTP standard et SCTP avec l'extension que nous introduisons par le *QoS_Measurement_Chunk*. Ce scénario se présente de la manière suivante :

- Le mobile est initialement desservi par la partie du réseau composée du : serveur IP-GGSN-SGSN1-BSS1, avec les paramètres de liens donnés par le paragraphe 4.3.1.
- Au niveau SCTP du serveur IP nous forçons l'adresse source du serveur liée à la route d'accès (GGSN – SGSN1- BSS1). Ceci est assuré avec *force-source* au niveau NS, ce qui permet d'avoir une seule adresse active à un instant donné.
- Ensuite une dégradation de la qualité du lien radio est déclenchée et qui consiste à augmenter le taux d'erreur blocs RLC/MAC.
- Après une durée moyenne de 20 ms nous déclenchons le *handover* vers la seconde adresse du serveur IP. Le GGSN, en fonction des données qui lui sont transmises par le mobile, décide de changer de route à destination du mobile. Le MS sera alors desservi par la partie du réseau définie par : Serveur IP- GGSN-SGSN2-BSS2. Ce changement de route d'accès réseau est assuré en considérant la transmission sur la seconde adresse au niveau serveur IP.

Nous considérons un RTT sur l'interface radio de 100ms. Nous supposons que la transmission de données est initialement déclenchée avec le CS3 puis nous dégradons le débit par le changement du schéma de codage à CS1. Après une certaine durée (de moyenne 20ms) nous avons fermeture des TBFs avec la première cellule et établissement de la liaison avec la seconde cellule avec un CS 3.

### 4.4.1 Extension de SCTP : *QoS-Measurement-chunk*

La solution, que nous proposons dans cette étude, considère qu'une erreur de transmission se produisant lors de transmission des datagrammes ne déclenchera pas les mécanismes de contrôle de congestion du côté de l'émetteur. Au niveau du récepteur, l'idée revient à masquer la transmission au protocole de transport en la reprenant au niveau de la couche 2 par l'intermédiaire d'un mécanisme ARQ fiable (qui n'est pas toujours possible à cause des retards dus aux retransmissions). Les solutions existantes comme l'algorithme Eifel [RFC3522] exigent aux deux extrémités (émetteur et récepteur) de supporter l'option *timestamp* de TCP. Dans notre approche nous considérons le protocole SCTP pour sa fiabilité et ses caractéristiques innovantes de transmission. En fait, nous proposons une modification du protocole SCTP de façon à transmettre au réseau n'importe quel type d'informations venant de la couche liaison voire même de la couche physique. Elle consiste à créer une alternative *Cross-Layer* basée sur l'aspect des *chunks* de contrôle en SCTP. Nous créons un nouveau *chunk* de contrôle appelé *QoS_Measurement_chunk.* Ce *chunk* est utilisé pour remonter les informations liées aux conditions de transmission radio à la couche transport afin d'adapter les paramètres de contrôle de congestion aux variations sur l'interface radio. Cette information déterminera quand déclencher le changement d'adresse en SCTP.

Le *QoS_Measurement_chunk* permet au mobile de remonter au GGSN des informations sur la qualité du lien radio (taux d'erreurs trames PDCP, débit maximum supporté par le lien radio,...). Ce *chunk* peut également servir pour informer le GGSN de toute modification du débit disponible sur le lien radio (dégradation des conditions radio, changement de schéma de codage, ...). Ces informations peuvent être utilisées par le GGSN pour mettre à jour la fenêtre d'anticipation dans le sens descendant en fonction des conditions de transmission sur l'interface radio.

A l'ouverture de l'association, le mobile transmet au GGSN (au moyen de la procédure d'activation de PDP contexte) les attributs du support radio utilisé par le terminal mobile

(notamment le débit maximale qui peut servir à fixer le seuil ssthreshold). Le mobile transmet au GGSN les schémas de codage utilisés pour la retransmission sur les interfaces radio sur le DL (*Down Link)* et le UL (*Up Link*). Le GGSN est en courant de chaque changement du schéma de codage utilisé lors de l'ouverture de TBF. Ceci est typiquement lié à la procédure d'adaptation de lien ou de *handover*. Toutes ces informations sont transmises dans le *QoS_Measurement_Chunk* dont le format est donné par la figure 4.4.

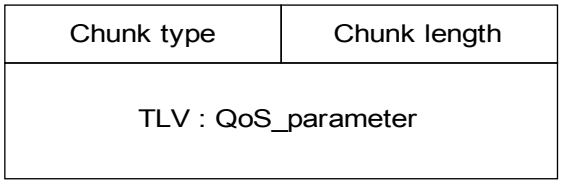



**Figure 4.4.**Format du chunk *QoS_Measurement_Chunk*

Les champs du *QoS_Mesearment_Chunk* sont :

- *Chunk Type :* 16bits (*unsigned integer*)
- *Chunk Length* : 16 bits (*unsigned integer*)
- *QoS_Parameter* : ce champ devrait inclure des informations sur les valeurs courantes des paramètres de QoS de transmission mesurées par l'émetteur (station mobile) sur l'interface radio (i.e. BLER, débit maximal…).

Dans une première étape nous essayons de remonter les informations du schéma de codage utilisé dans un réseau EGPRS, en supposant une variation dynamique du schéma de codage au cours d'une communication. Le but de remonter une telle information est de varier la fenêtre de congestion sur la partie fixe du réseau en fonction du débit maximale toléré par le schéma de codage activé. Ce qui permet d'adapter le flux de transmission de données au niveau transport selon le comportement de l'interface radio c'est à dire de la couche liaison voire même physique.

La réception d'un *QoS_Measurement_Chunk* contenant l'information sur le schéma de codage sur le DL juste après le *handover*, entraîne une mise à jour de *cwnd* dans la nouvelle cellule selon la formule suivante :

$$cwnd_{newCell} = cwnd_{oldCell} - C(CS_{oldCell}) \cdot RTT + C(CS_{newCell}) \cdot RTT$$

Où, $C(CS_{oldCell})$ est le débit maximal assuré par le CS activé sur la première cellule avant le *handover*, et $C(CS_{newCell})$ est le débit maximal assuré par le CS activé sur la seconde cellule après le *handover*.

Une autre modification est introduite pour le calcul du *ssthresh* dans la nouvelle cellule, c'est que nous gardons la dernière valeur de *ssthresh* sur la première cellule. Si le mobile est en phase de *congestion avoidance* sur la première cellule il reste dans cette phase sur la nouvelle cellule avec une mise à jour de *cwnd* (juste après le *handover* et si aucune erreur de transmission ne se produit). Si le mobile est en phase de *slow start* sur l'ancienne cellule, il garde cette phase sur la nouvelle cellule avec une valeur de cwnd = cwnd (newCell) non pas à 1 ou 2 MTU, comme s'est spécifié par

la norme [RFC2960], et par suite la phase de *slow start* ne se déclenche pas systématiquement lors de changement d'adresse comme c'est le cas en SCTP standard.

La figure 4.5, illustre l'évolution du RTT au niveau RLC/MAC, le RTT présente un pic lors des fermetures des TBFs de l'ordre de 1.6sec, pour SCTP avec *QoS_Measurement_Chunk* qui est inférieure à celui dans le cas du SCTP standard qui est de 1.85sec (cf. figure 4.6).

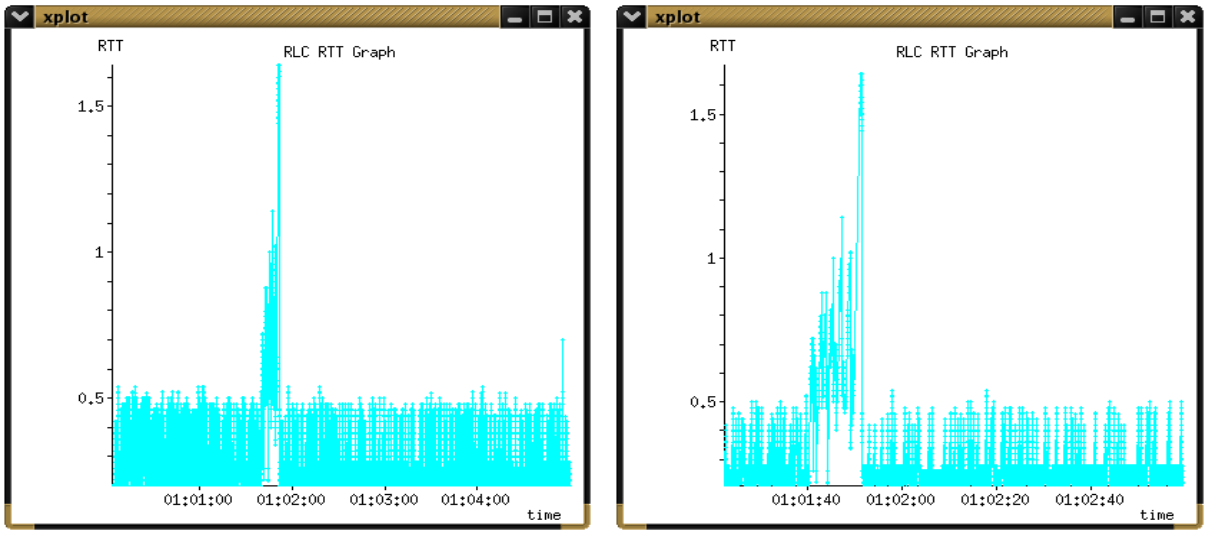



**Figure 4.5.** RTT au niveau RLC/MAC au cours de la communication en utilisant le SCTP avec l'extension du *Qos_Measurement_Chunk* (+zoom).

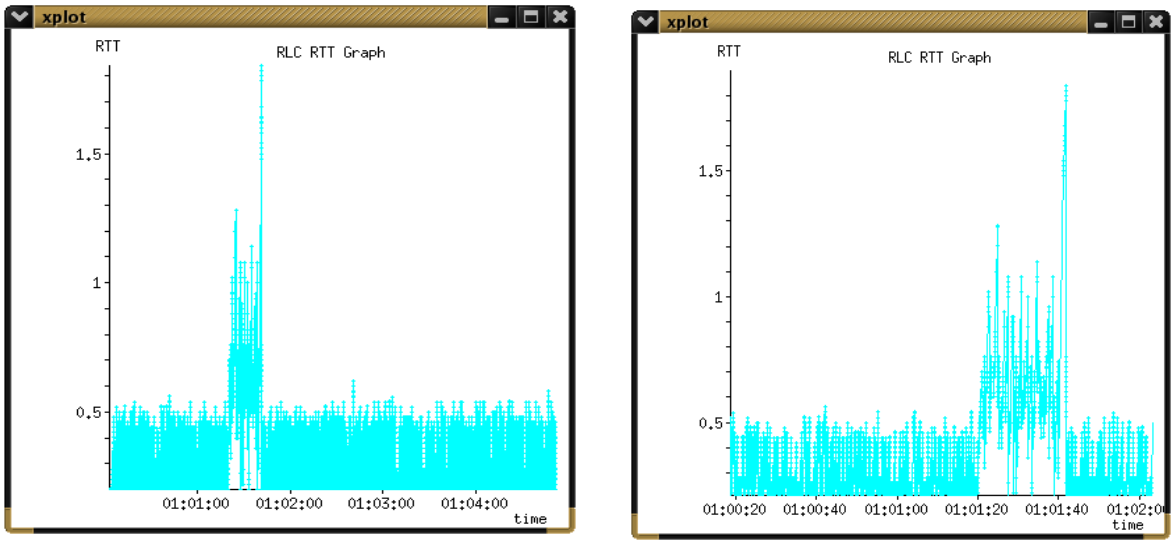
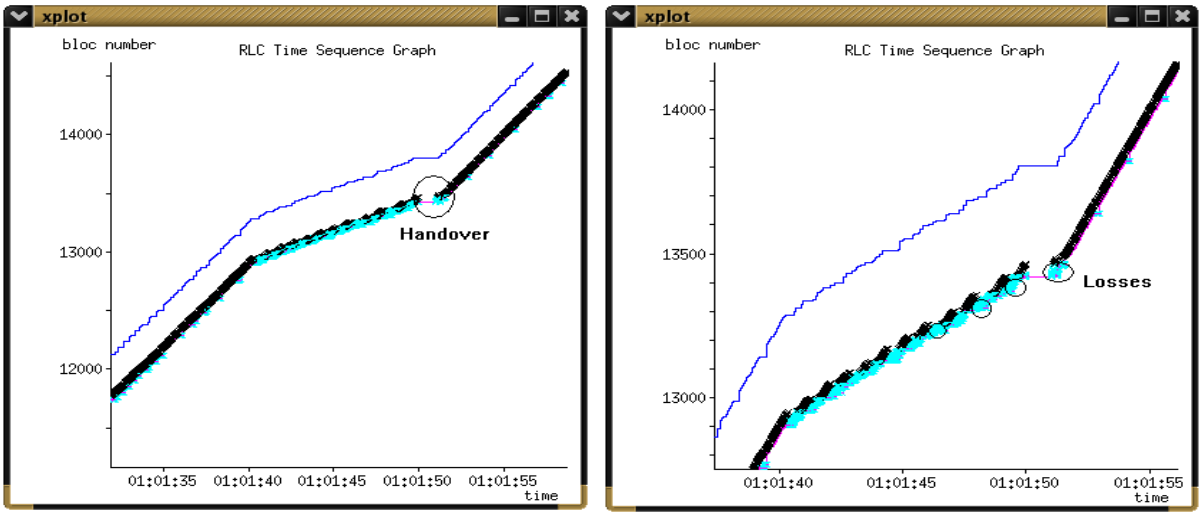



**Figure 4.6.** RTT au niveau RLC/MAC au cours de la communication en utilisant le SCTP standard(+zoom).

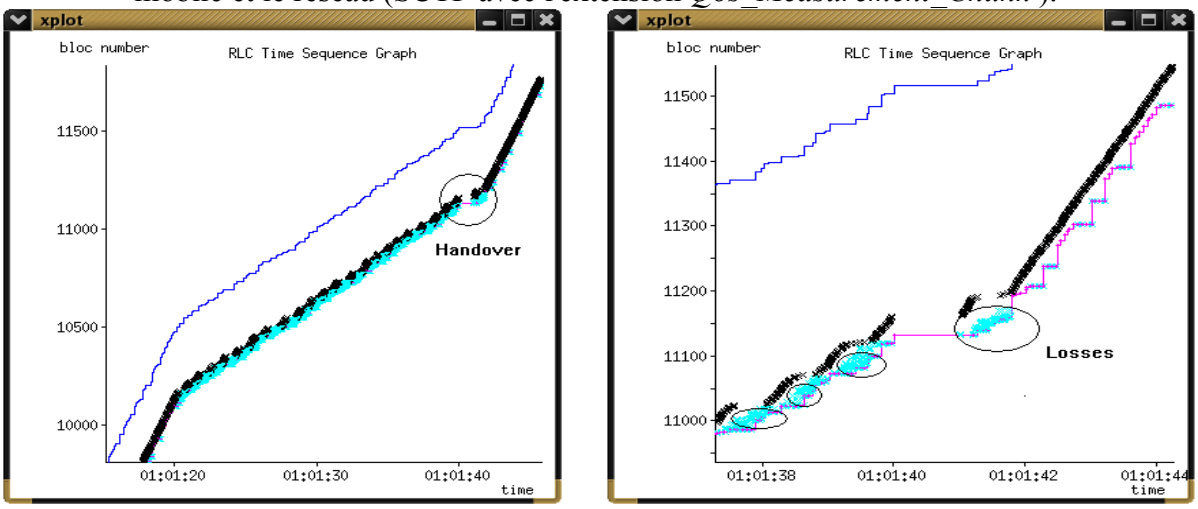
L'évaluation de performance du mécanisme *Qos_Measurement_Chunk* est étudiée au niveau SCTP et au niveau RLC/MAC. L'évolution du numéro de séquence au niveau RLC/MAC est donnée par la figure 4.7 pour SCTP avec l'extension *Qos_Measurement_Chunk* et par la figure 4.8 pour SCTP Standard. Ces deux figures illustrent les pertes dues à la dégradation de la qualité du lien introduite par le modèle d'erreur (nous considérons un modèle d'erreur uniformément distribué de

moyenne $10^{-2}$) et suivie d'une coupure de la connexion avec la première cellule serveuse à cause du *handover*, et de l'établissement de connexion avec la nouvelle cellule. Par comparaison avec le SCTP standard avec *multihoming* nous remarquons que le débit de transmission de l'extension proposée est plus important au niveau RLC/MAC ce qui est prouvé par l'évolution du TSN donnée par les figures 4.10 et 4.11.




**(a)** **(b) : zoom (a)**

**Figure 4.7.** Exemple de variation du numéro de séquence RLC durant la communication entre le mobile et le réseau (SCTP avec l'extension *Qos_Measurement_Chunk* ).




**(a)** **(b) : zoom (a)**

**Figure 4.8.** Exemple de variation du numéro de séquence RLC durant la communication entre le mobile et le réseau (SCTP standard).

SCTP standard initialise *cwnd* à 1*MTU pour la phase de *slow start* succédant le changement de lien alternatif (cf. figure 4.9 à t=100s). La valeur de *cwnd* dans l'extension proposée ne change pas de valeur et croît régulièrement après le *handover*. C'est une modification que nous l'avons introduite au niveau du code du mécanisme de contrôle de congestion. Ce qui explique le fait que

l'évolution de *cwnd* est meilleure pour l'extension proposée et par suite une augmentation continue du flux de transmission (cf. figure 4.9 et 4.10). Ceci peut provoquer des problèmes de congestion sur la partie fixe du réseau, même avec les résultats de simulation NS ce n'est pas le cas, comme c'est illustré par la figure (zoom 4.11), car le lien radio de type EGPRS a un flux de transmission moins important que le réseau coeur.

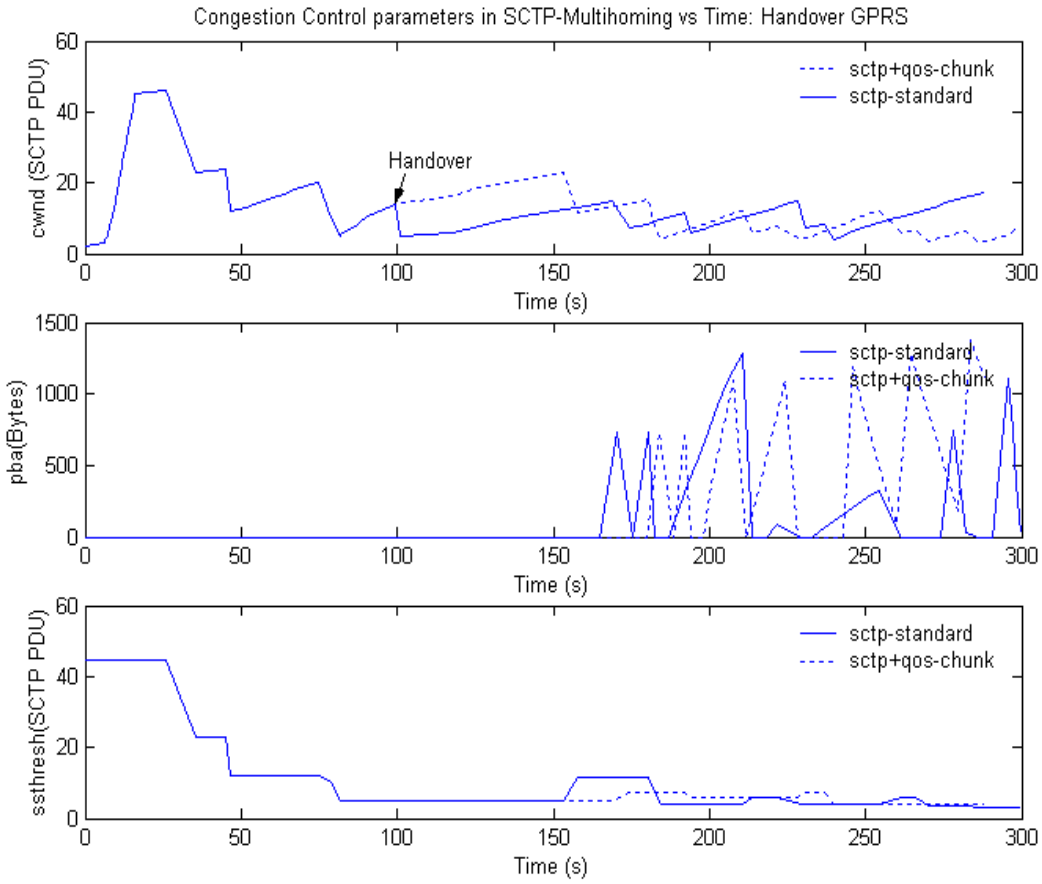



**Figure 4.9.** Exemple de variation des paramètres de contrôle de congestion en fonction du temps dans le cas SCTP avec extension par comparaison avec le SCTP standard.

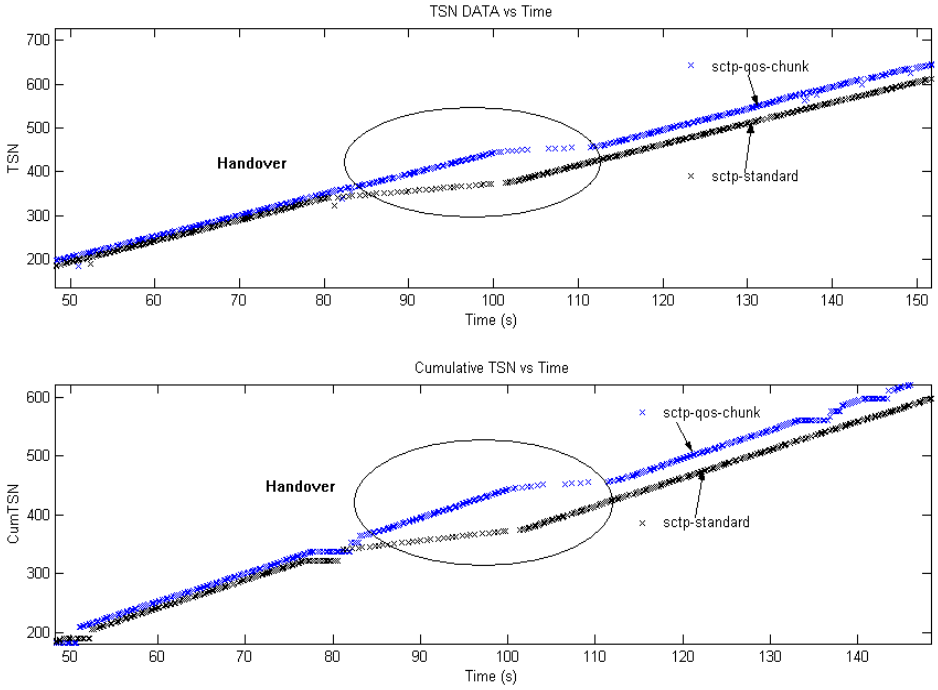



**Figure 4.10.** Évolution du TSN au niveau SCTP au cours de l'association.

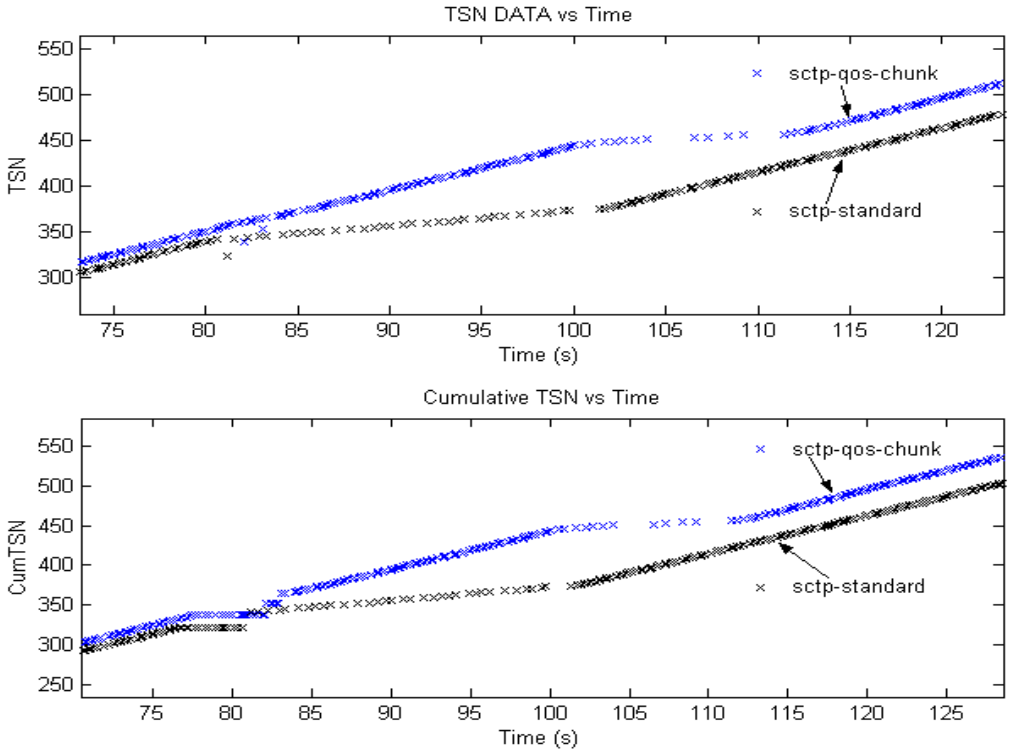



**Figure 4.11.** Évolution du TSN au niveau SCTP au cours de l'association (zoom de figure 4.10).

La modification que nous suggérons dans nos travaux de recherches au protocole SCTP, est utile pour une meilleure adaptation du protocole de transport aux environnements radio. SCTP avec son mécanisme de *multihoming* assure une meilleure gestion de *handover* de données. L'introduction d'un nouveau type de *chunk* (*QoS_Measurement_chunk*) offre la possibilité d'adapter les variations de *cwnd* aux conditions de transmission sur l'interface radio. Une association SCTP n'exécute pas le *slow start* comme c'est le cas de SCTP standard qui ralenti le débit d'émission. Avec cette extension de SCTP, un *handover* de données est mieux exécuté que dans le cas des protocoles de transport usuels, ce que nous venons de prouver dans le cas de technologie d'accès radio homogène (EGPRS/EGPRS). Dans le paragraphe suivant nous étudions l'évaluation de performances de notre modification, apportée à SCTP, dans le cas où des changements dynamiques des conditions de transmissions sur l'interface radio (dans notre cas nous considérons des variations des schémas de codage au niveau RLC/MAC) se produisent.

### 4.4.2 *QoS_Measurement_Chunk* : Comportement pour un schéma de codage variable au cours d'une communication

La procédure de *QoS_Mesearment_Chunk* ne modifie pas de façon directe l'algorithme de contrôle de congestion du SCTP standard, c'est plutôt une adaptation aux conditions radio. En effet, le problème mis en évidence est le changement des conditions radio et son impact sur le contrôle de flux au niveau transport. Les pertes sur l'interface radio sont généralement interprétées comme étant des problèmes de congestion sur la partie fixe du réseau. Le *QoS_Measurement_Chunk* est une proposition qui permet de remonter des informations sur les paramètres de l'interface radio au niveau transport. Ceci permet d'adapter le flux de transmission de données par une mise à jour des paramètres de contrôle de congestion principalement la taille de la fenêtre de congestion (*cwnd*) et le seuil *slow start* (*ssthresh*) (cf. figure 4.12).

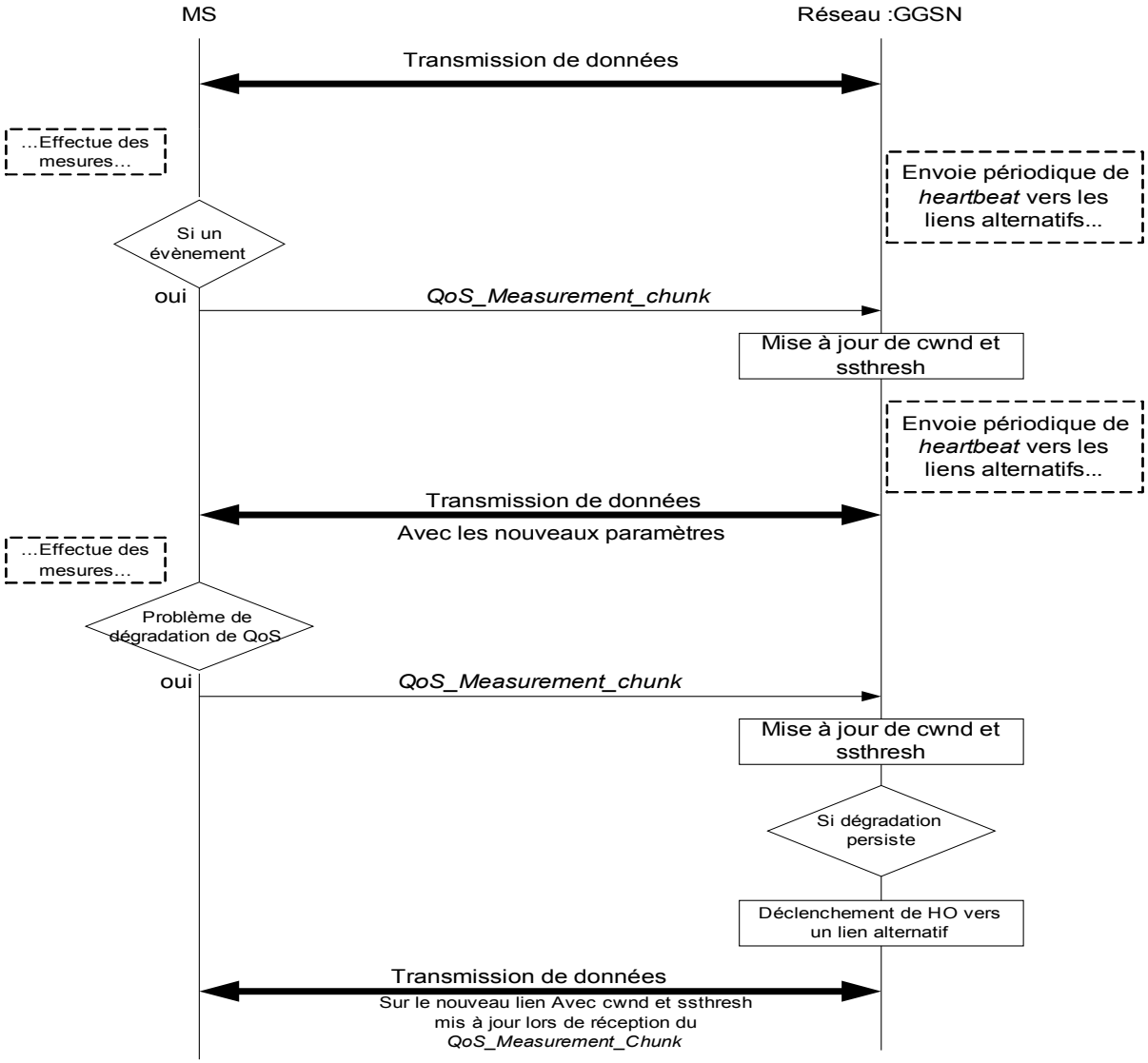



**Figure 4.12.** Signalisation de l'approche du *QoS_Measurement_Chunk*.

La mise à jour de *cwnd* et *ssthresh* se fait en fonction de leurs dernières valeurs respectives et du BDP du lien radio comme c'est décrit dans le troisième chapitre du présent rapport. La décision de *handover* est à l'initiative du réseau et se produit suite à une comparaison aux paramètres de qualité minimale requise échangés entre le MS et le réseau à l'établissement du PDP contexte (autrement dit à l'ouverture de l'association SCTP).

Le *QoS_Measurement_Chunk* est un *chunk* SCTP de contrôle indépendant du *Heartbeat chunk*. Son émission ne peut pas être périodique comme c'est le cas pour les *chunks Hearbeat* (plutôt aléatoire). Ce *chunk* permet un contrôle instantané du flux de transmission. Dans ce qui suit nous étudions le comportement de cette extension de SCTP d'abord dans le cas d'un changement déterministe des paramètres de transmission ensuite dans le cas d'un changement probabiliste.

### 4.4.3 Comportement du *QoS_Measurement_Chunk* : variations déterministes du schéma de codage

Dans cette partie nous considérons les mêmes paramètres de simulation décrits ci-dessus, et nous faisons varier le schéma de codage de façon déterministe ainsi que le taux d'erreur bloc (BLER). Le changement du schéma de codage est considéré de sorte à dégrader la qualité de la transmission sur l'interface radio et à activer le *handover* radio vers la seconde cellule. Le tableau 4.1 décrit le changement du schéma de codage au cours d'une communication.

Tableau 4.1. Variations déterministes du schéma de codage.

| ***Instant*** | ***Schéma de codage*** | ***BLER*** |
|---|---|---|
| 0sec | CS4 | 0.01 |
| 200sec | CS3 | 0.015 |
| 400sec | CS2 | 0.1 |
| 600sec | CS1 | 0.15 |
| 700sec | Déclenchement du HO | |
| CS3 sur la nouvelle cellule | | 0.015 |

La figure 4.13 représente les variations du RTT au niveau RLC/MAC au cours d'une communication en activant les variations des schémas de codage décrites par le tableau 4.1 dans le cas de SCTP avec l'extension *QoS_Measurement_Chunk*.

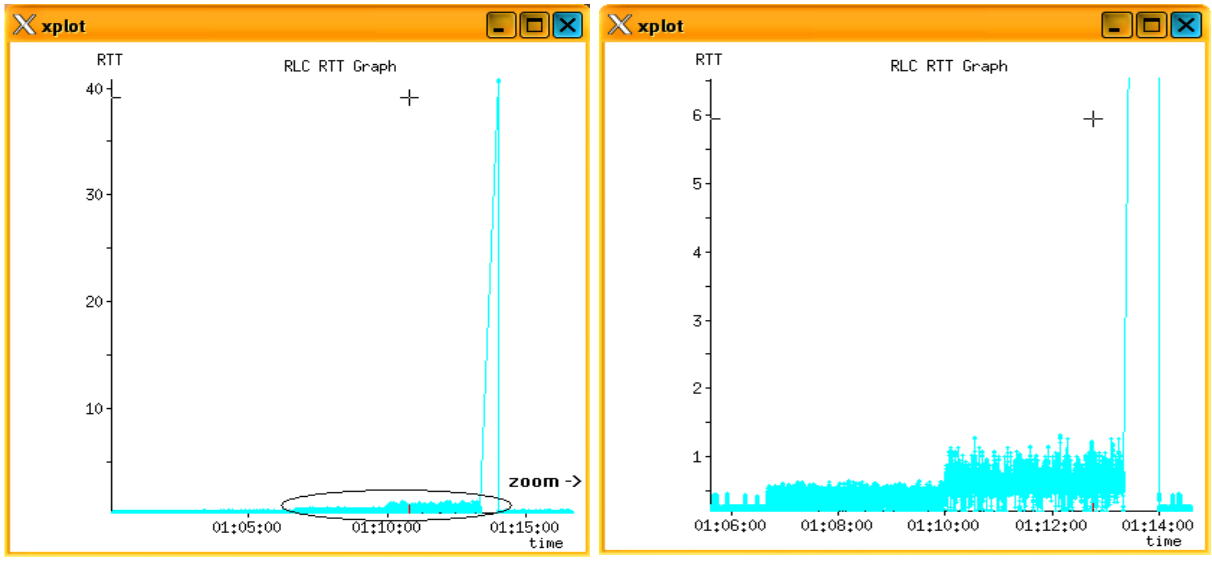



**Figure 4.13.** Variations du RTT au niveau RLC/MAC pour SCTP avec l'extension *QoS_Measurement_Chunk*.

En utilisant la même configuration avec SCTP standard, nous constatons deux pics de RTT un

lors de changement du schéma de codage de CS2 à CS1 et un autre lors de la phase de *handover* (cf. figure 4.14).

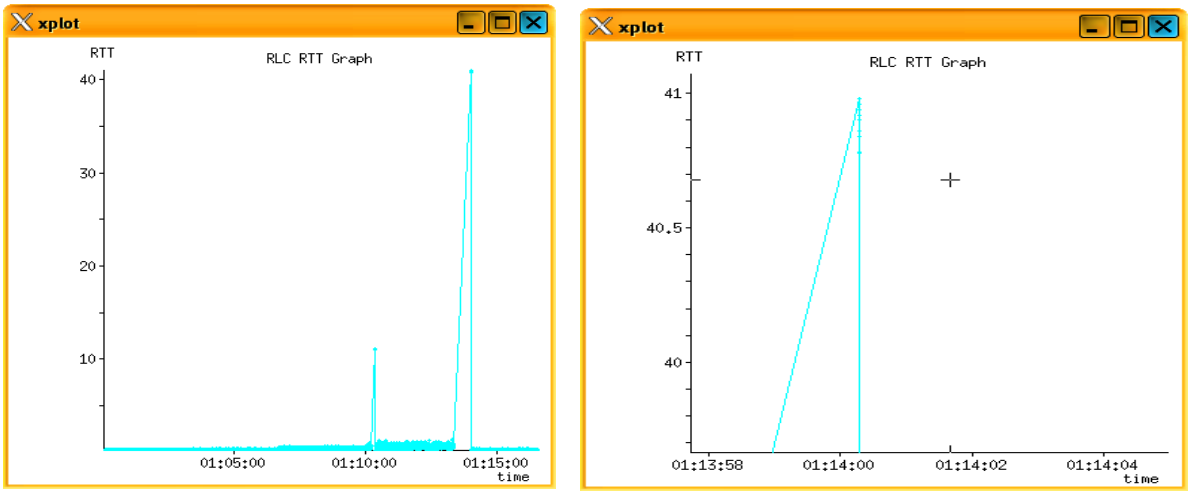



**Figure 4.14.** Variations du RTT au niveau RLC/MAC pour SCTP standard (zoom autour du deuxième pic).

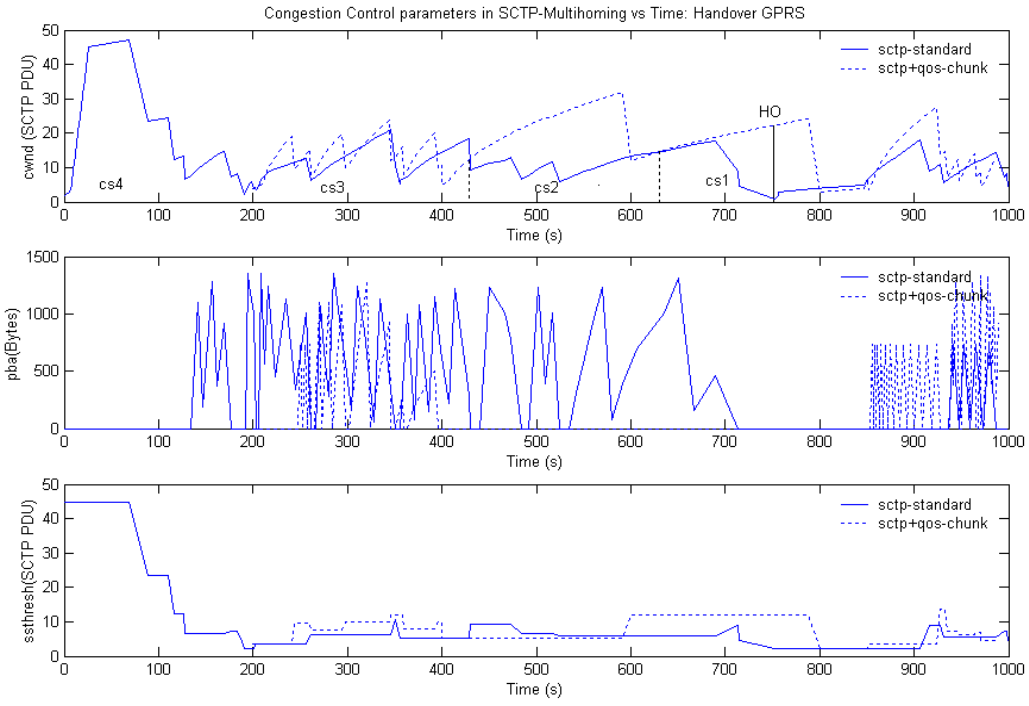



**Figure 4.15.** Variation des paramètres de contrôle de congestion en fonction du temps dans le cas de SCTP avec extension par comparaison avec le SCTP standard pour une variation déterministe du schéma de codage.

Nous remarquons que l'extension proposée permet d'éviter la production des hors temps

(*timeout*) à cause de dégradation de qualité comme pour le cas du SCTP standard. La figure 4.15 montre l'évolution des paramètres de contrôle de congestion en fonction du temps.

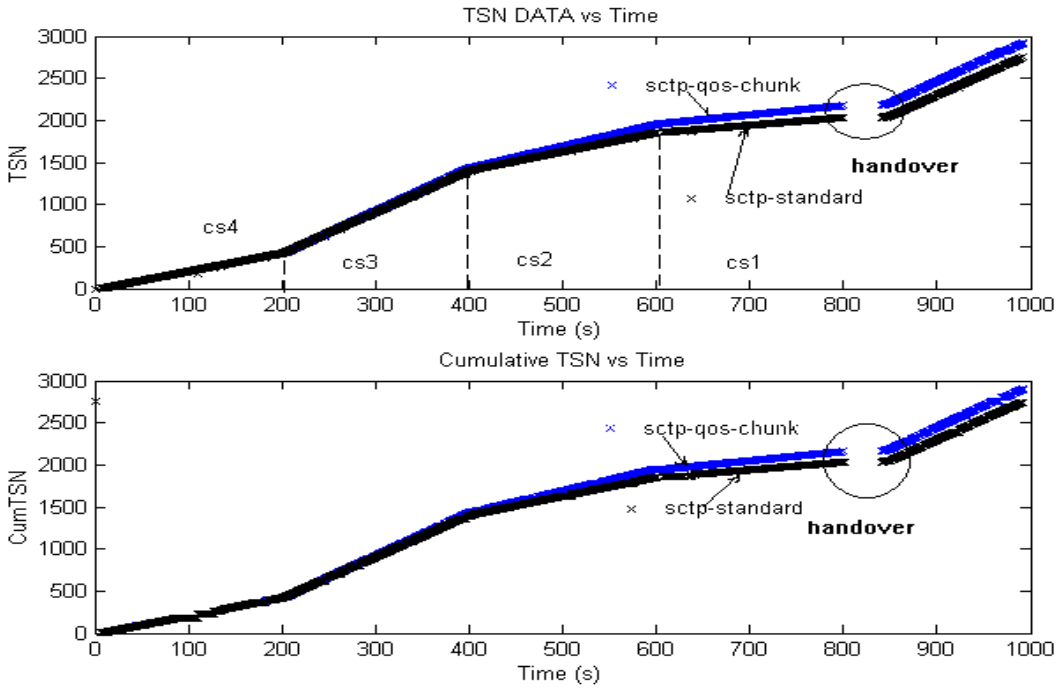



**Figure 4.16.** Évolution du TSN au niveau SCTP au cours de l'association.

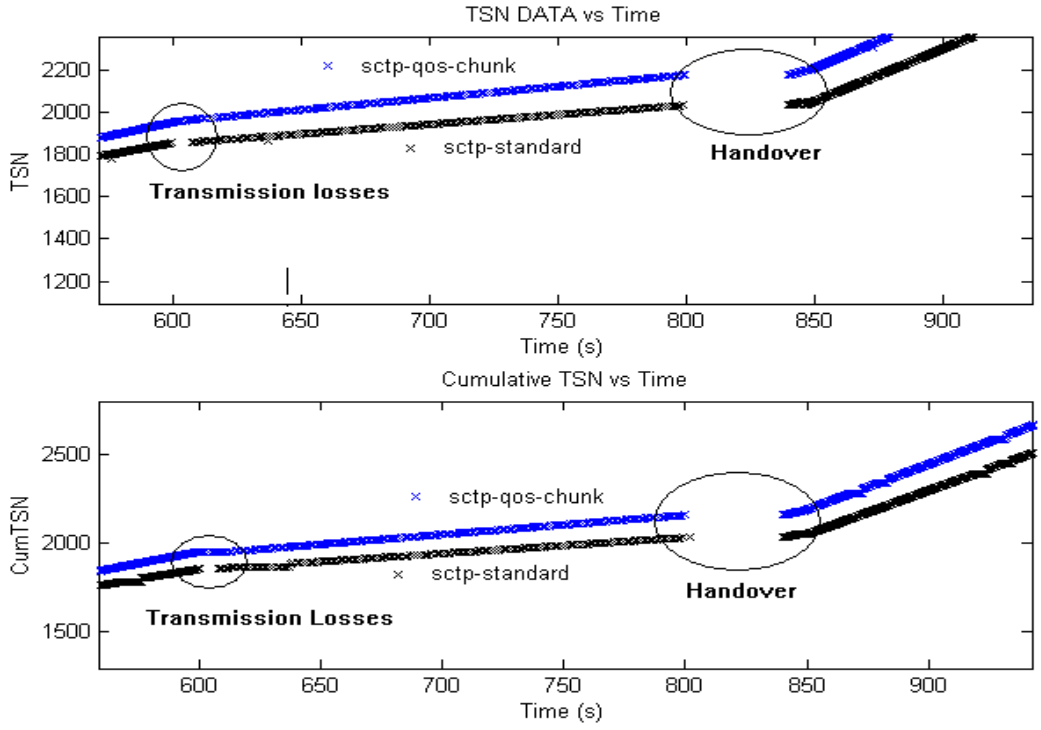



**Figure 4.17.** Évolution du TSN au niveau SCTP au cours de l'association (zoom de la figure 4.16).

Les variations du numéro de séquence au niveau RLC/MAC sont données par la figure 4.18 pour SCTP avec l'extension *QoS_Measurement_Chunk* qui sont comparées aux variations du numéro de séquence au niveau RLC/MAC avec SCTP standard données par la figure 4.19.

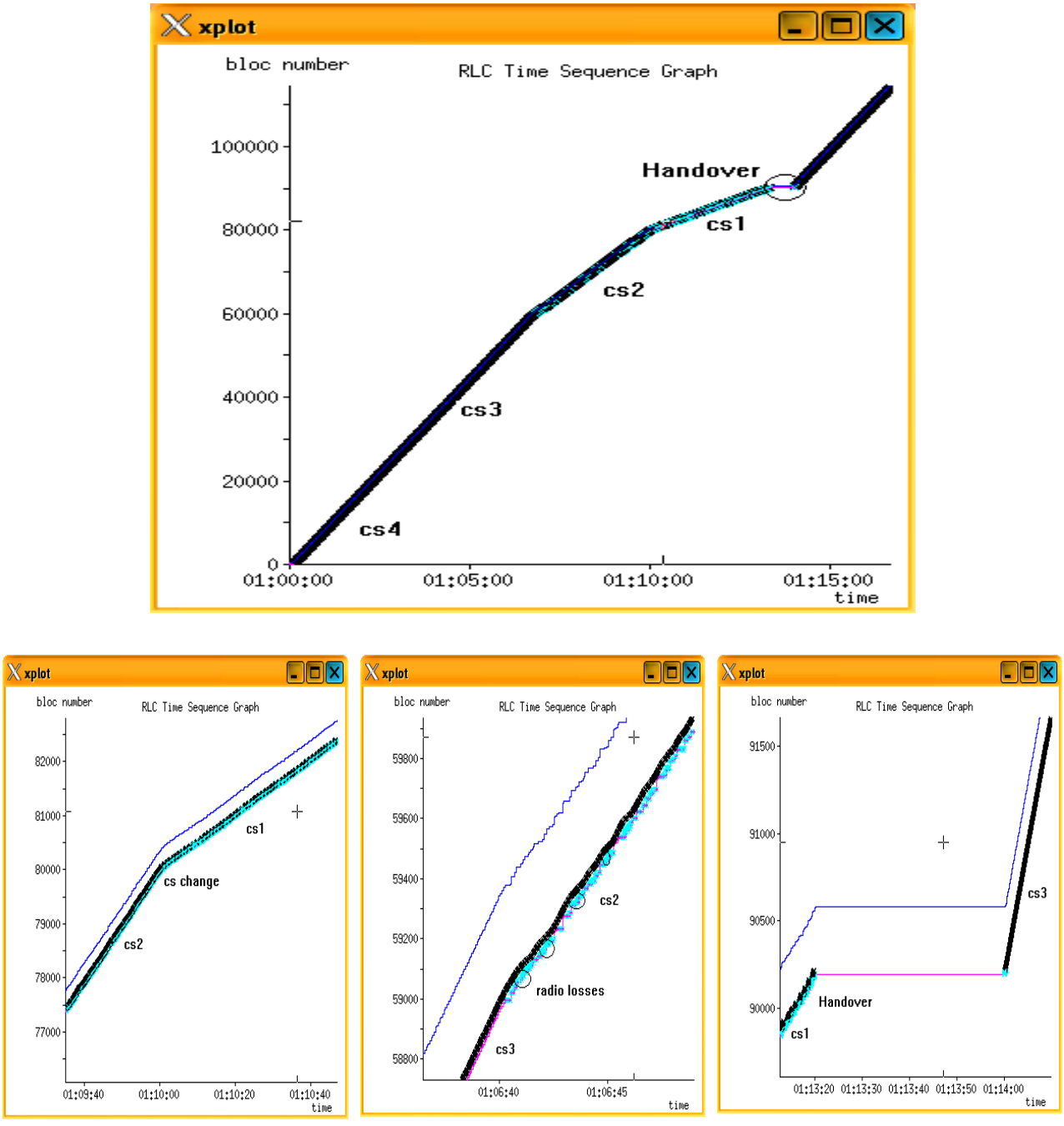



**Figure 4.18.** Variations du numéro de séquence au niveau RLC/MAC avec SCTP avec l'extension *QoS_Measurement_Chunk*.

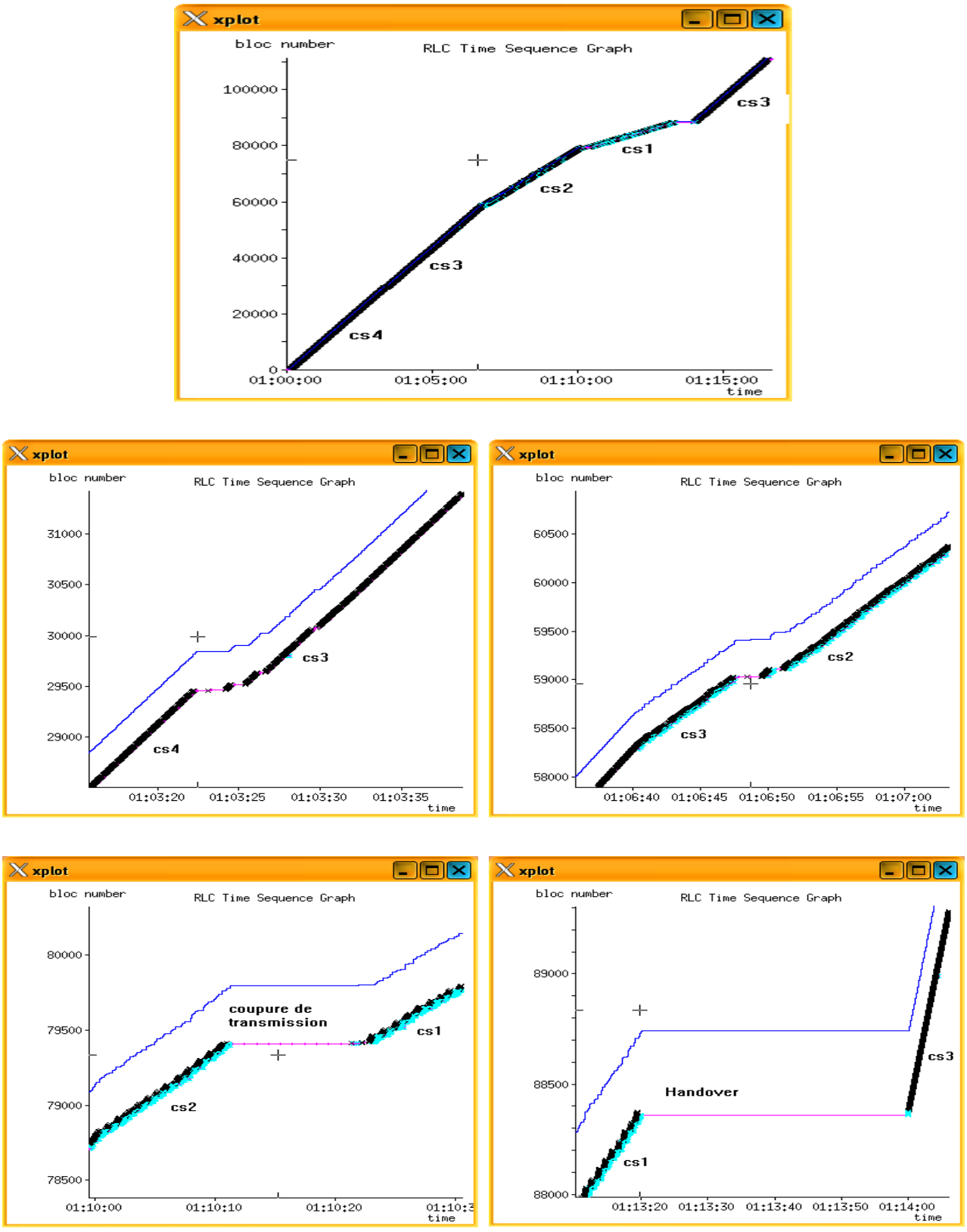



**Figure 4.19.** Variations du numéro de séquence au niveau RLC/MAC avec SCTP standard.

Le fait d'adapter la taille de la fenêtre de congestion en tenant compte des dégradations de la qualité de transmission sur l'interface radio permet d'avoir un contrôle de flux plus conservateur et par suite une augmentation du débit de transmission par comparaison à SCTP standard. Nous notons des pertes de données lors de dégradation de qualité de CS2 à CS1 dans le cas de SCTP standard par contre avec l'extension de *QoS_Measurement_Chunk* nous n'avons pas de pertes c'est plutôt une évolution continue des transmissions (voir figure 4.16 et 4.17).

Nous avons considéré un changement déterministe de la paire (schéma de codage, BLER) afin de tester le comportement de la modification suggérée en changeant les paramètres de transmission. La procédure de *QoS_Measurement_Chunk,* utilisée conjointement avec le *multihoming,* a montré de meilleures performances, particulièrement une fois utilisée dans une situation de *handover*. Dans ce qui suit et avant d'exécuter notre approche dans un contexte de *handover* inter-RAT, nous comptons tester le comportement du *QoS_Measurement_Chunk* dans le cas d'un changement aléatoire des conditions de transmission sur l'interface radio.

### 4.4.4 Comportement du *QoS_Measurement_Chunk* : variations aléatoires du schéma de codage

Les résultats des simulations présentés à ce niveau considèrent un changement aléatoire du schéma de codage et le BLER correspondant selon le modèle donné par la figure 4.20.

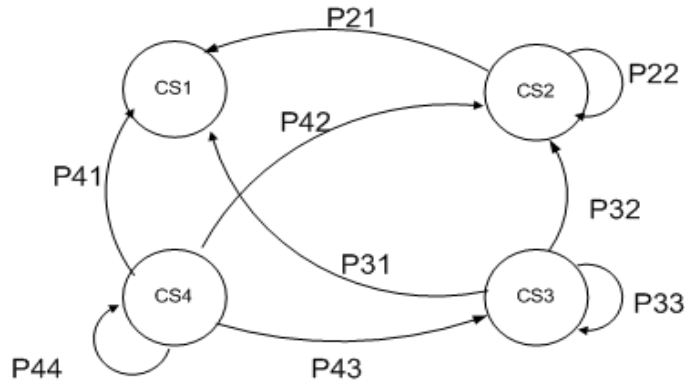



**Figure 4.20.** Modèle aléatoire de changement du schéma de codage.

Avec :

Pij sont les probabilités de transition d'état (i) vers (j).

Un état de cette chaîne est défini par le schéma de codage et le BLER correspondant.

P21=0.4, P22=0.6, P31=0.2, P32=0.5, P33=0.3, P41= 0.1, P42=0.3, P43=0.4 et P44=0.2.

L'état initial est CS4, le temps mis dans chaque état est un temps aléatoire de distribution exponentielle, de moyennes 200sec, 400sec, 600sec et 800sec respectivement à cs4, cs3, cs2 et cs1. L'état CS1 est choisi comme état absorbant afin de pouvoir déclencher un *handover* à partir de cet état. Nous avons considéré uniquement les transitions qui visent la dégradation de qualité afin de

pouvoir bien visualiser le comportement de l'extension proposée surtout à l'instant de déclenchement de *handover* radio suite à une dégradation persistante de la qualité de transmission sur l'interface radio.

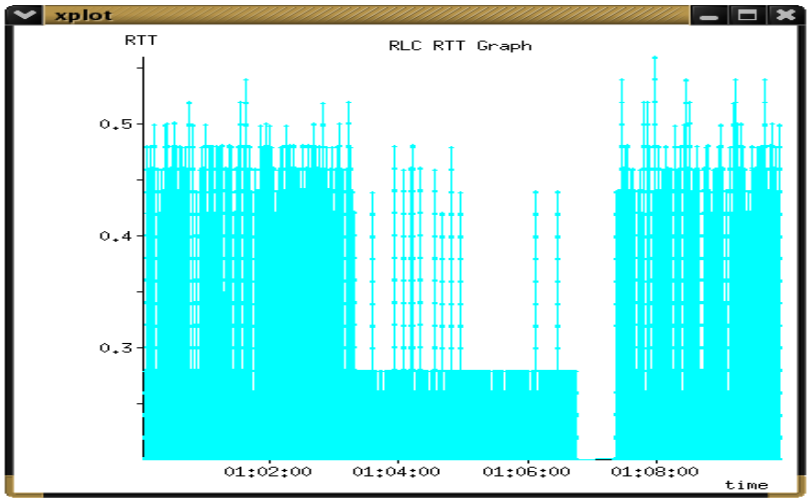



**Figure 4.21.** Exemple de variations du RTT au niveau RLC/MAC pour SCTP avec l'extension *QoS_Measurement_Chunk*.

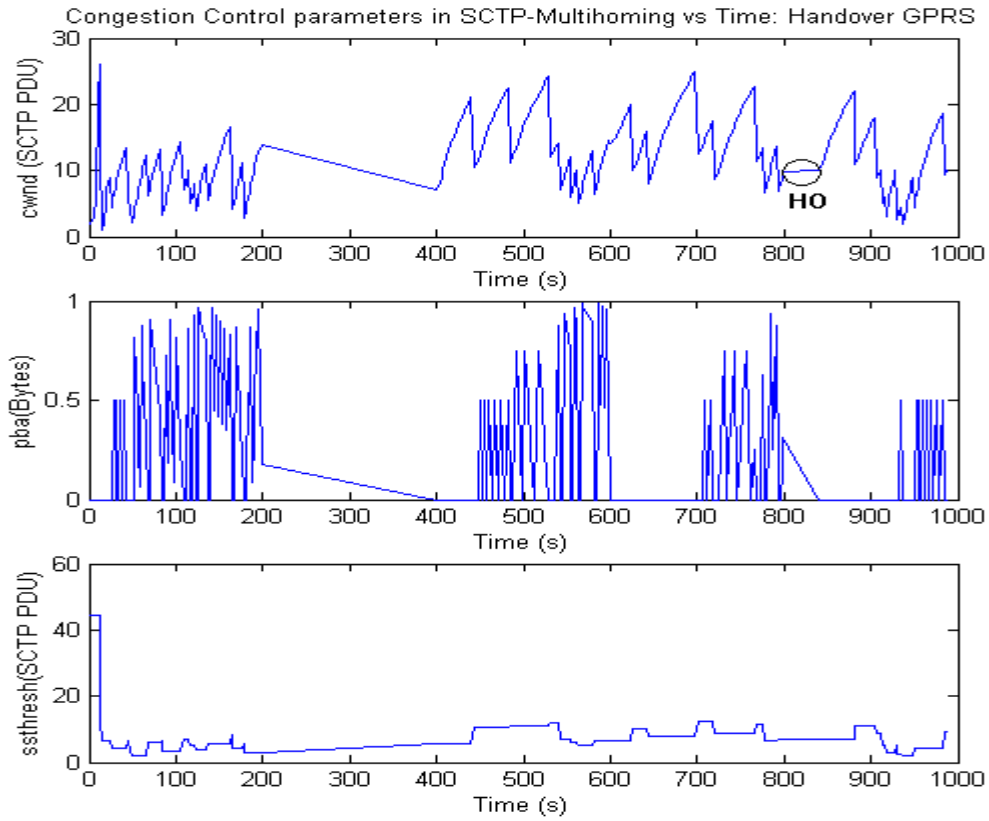



**Figure 4.22.** Exemple de variation des paramètres de contrôle de congestion en fonction du temps dans le cas SCTP avec extension  pour une variation aléatoire du schéma de codage.

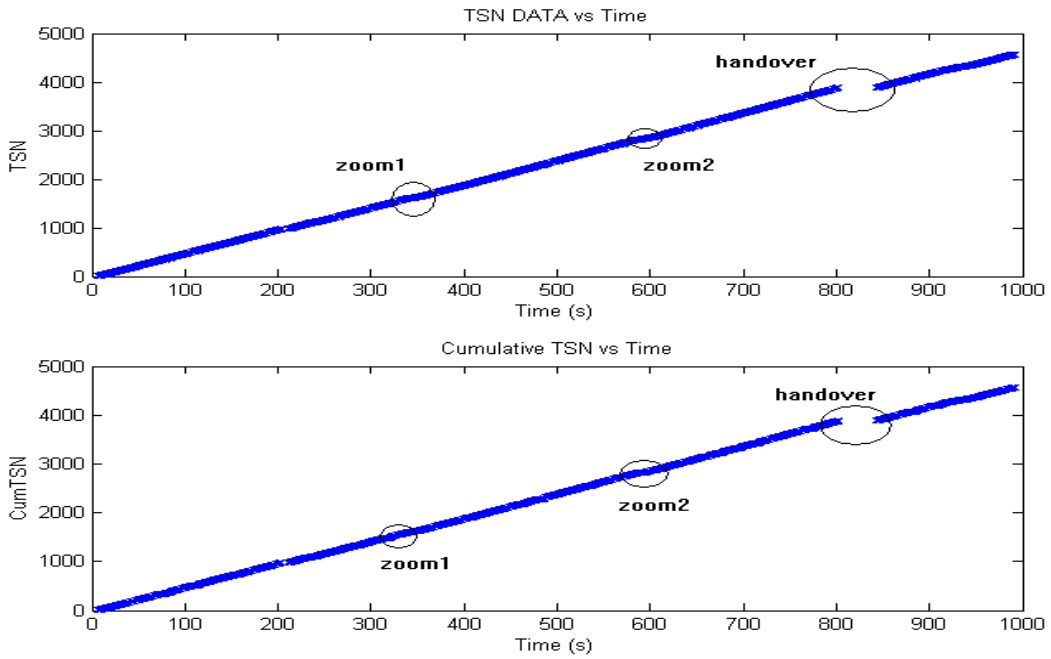
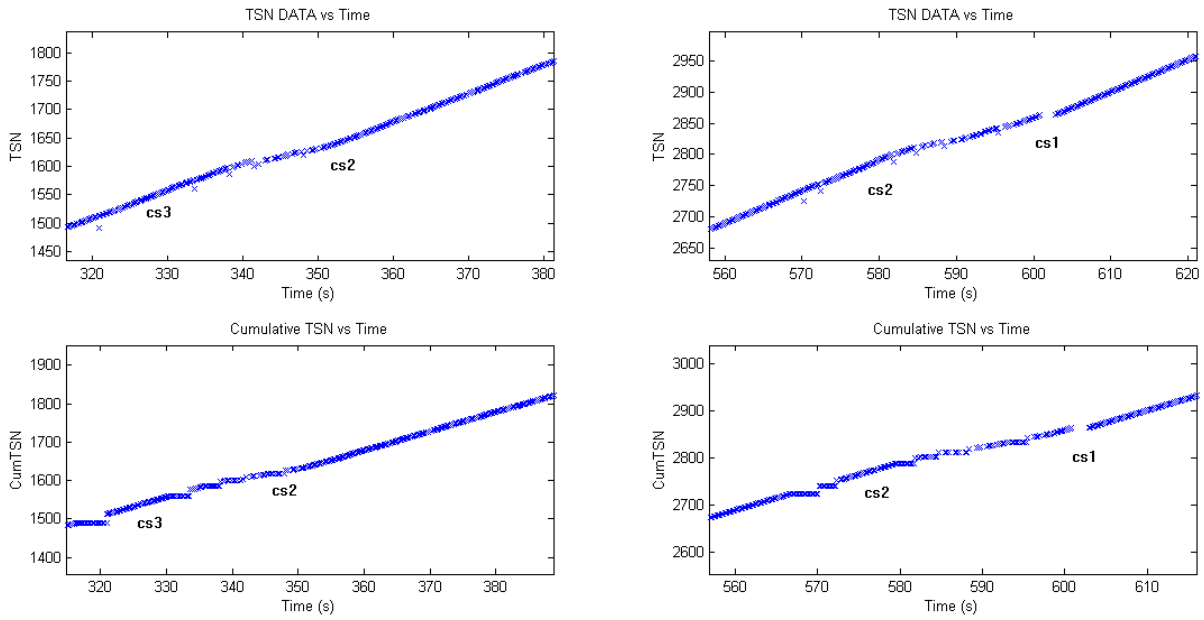



**Figure 4.23.** Évolution du TSN au niveau SCTP avec l'extension *QoS_Measurement_Chunk* au cours de l'association.




zoom1 zoom2

**Figure 4.24.** Évolution du TSN au niveau SCTP avec l'extension *QoS_Measurement_Chunk* au cours de l'association (zoom figure 4.23).

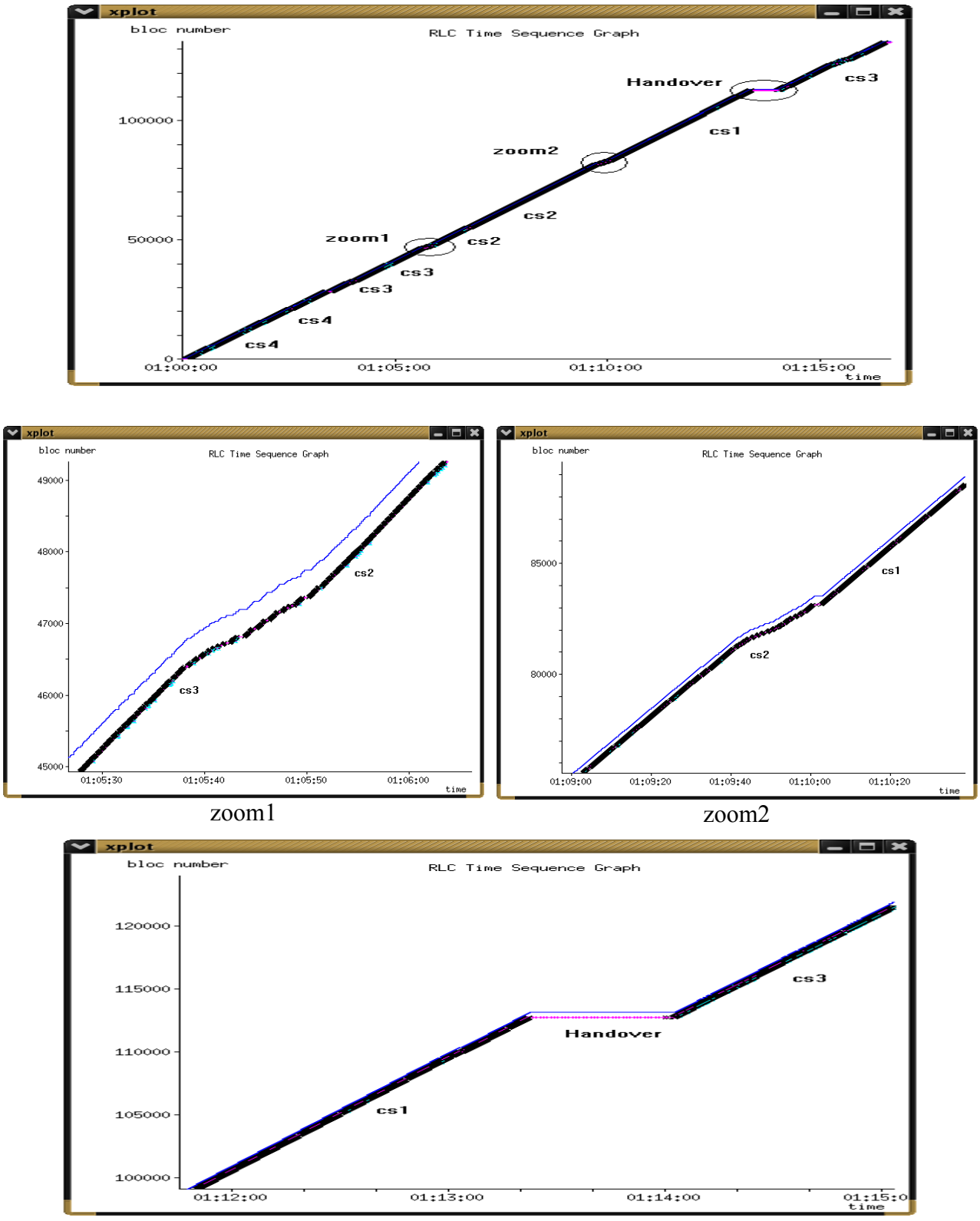



**Figure 4.25.** Exemple de variations du numéro de séquence au niveau RLC/MAC avec SCTP avec l'extension *QoS_Measurement_Chunk*.

Nous remarquons que la variation aléatoire des conditions de transmission sur l'interface radio (CS, BLER) remontée au niveau transport, n'affecte pas le flux de transmission de données de point de vue pertes de données. En effet, que se soit au niveau SCTP (cf. figures 4.23 et 4.24) ou au niveau RLC/MAC (cf. figure 4.25) nous avons des numéros de séquence qui varient de façon continue en fonction du temps c'est à dire nous n'avons pas un freinage du rythme de transmission à cause des problèmes de pertes. Les suppositions de notre extension liées au contrôle de congestion sont bien vérifiées ce qui est illustré par la figure 4.22 représentant les variations de la fenêtre de congestion et du *ssthresh* en fonction du temps. Comme le changement du schéma de codage et du BLER correspondant, ainsi que le temps de variation sont aléatoires donc il est un peu difficile d'identifier les instants de changement, mais le *handover* est repérable et nous vérifions bien que *cwnd* après le changement de lien obéit à la formule donnée dans le paragraphe 4.4.1 (cf. figure 4.22). Le but de lancer des simulations tenant compte des variations aléatoires des conditions radio est de valider l'extension de *QoS_Measurement_Chunk* du SCTP que nous proposons, ce qui est prouvé par les résultats obtenus.

### 4.4.5 Cas de deux mobiles : Comportement du *QoS_Measurement_Chunk* avec variations déterministes du schéma de codage

Dans cette partie nous considérons la même configuration des simulations que précédemment, sauf que nous comptons tester le comportement du *QoS_Measurement_Chunk* dans le cas d'un réseau chargé (ici nous considérons deux mobiles). Nous considérons le cas d'un schéma de codage variable de façon déterministe avec l'extension du *QoS_Measurement_Chunk* dans le cas de deux mobiles en communications qui subissent les mêmes conditions de transmission. Pour gérer ces deux mobiles nous considérons un ordonnancement de type RR (*Round Robin*). Nous représentons les variations des paramètres de contrôle de congestion relativement à chaque mobile afin de pouvoir interpréter le comportement de notre modification dans le cas d'un réseau chargé.

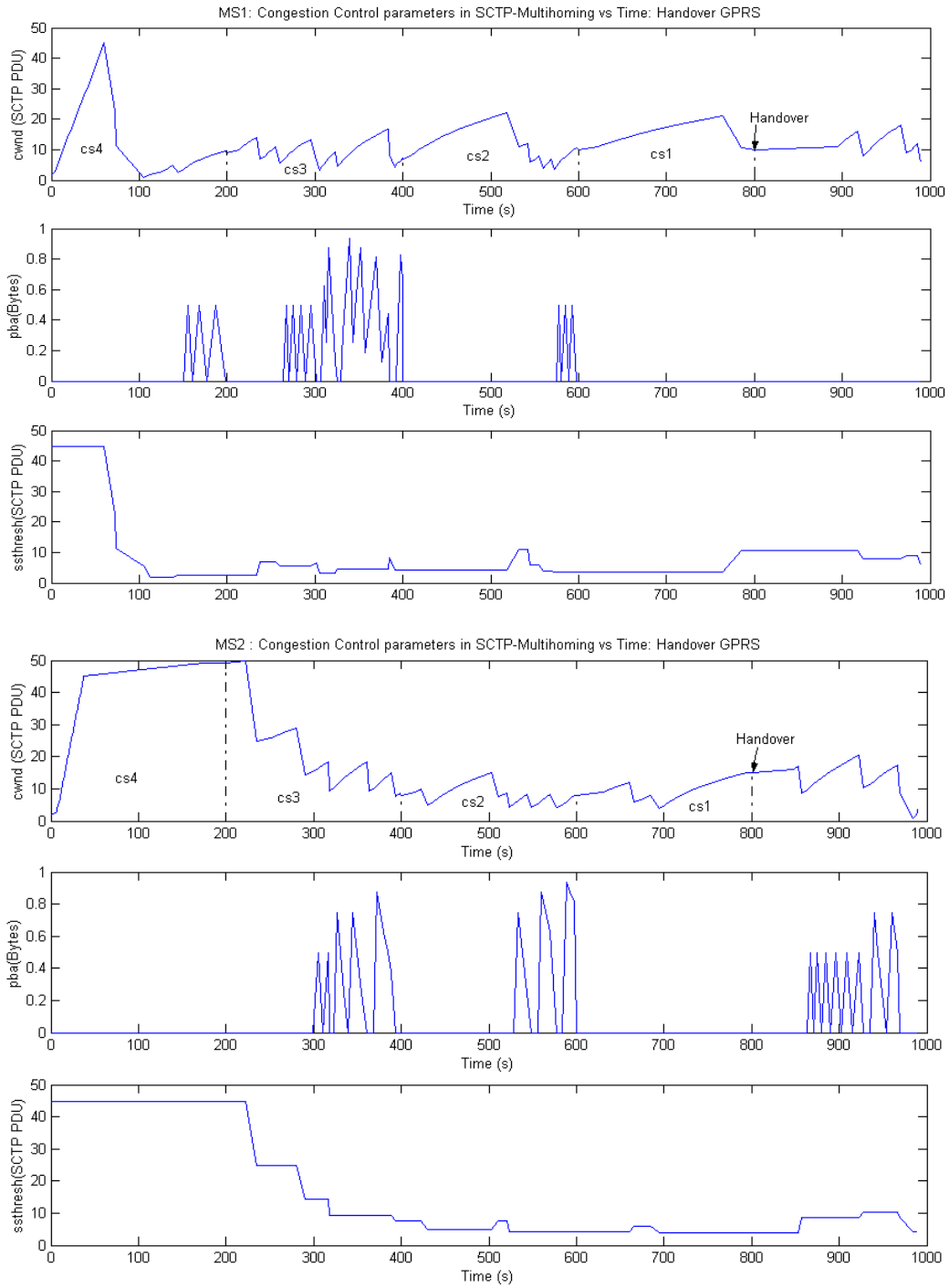



**Figure 4.26.** Exemple de variation des paramètres de contrôle de congestion en fonction du temps dans le cas de SCTP avec extension pour une variation déterministe du schéma de codage (2 mobiles : MS1 en haut, MS2 en bas).

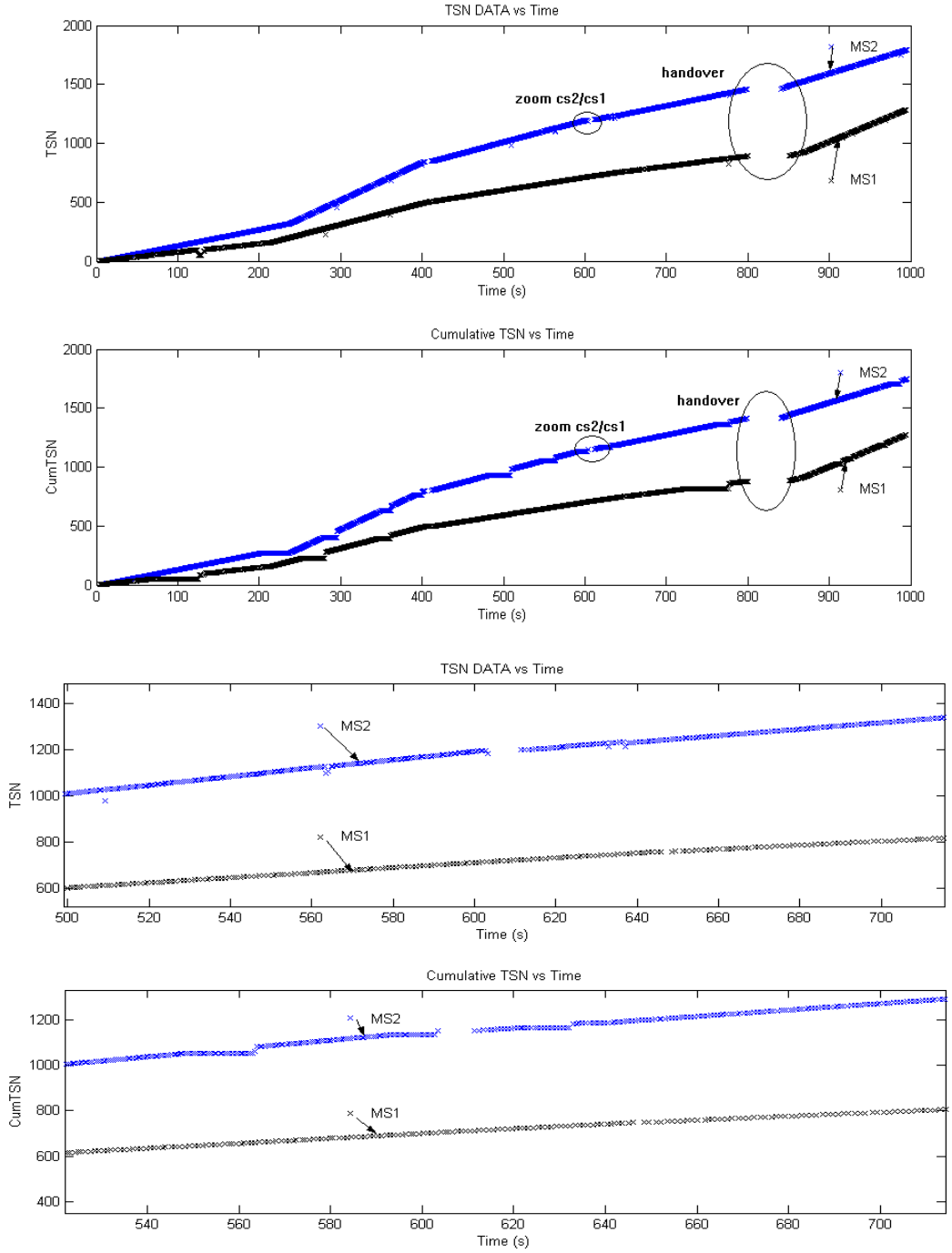



**Figure 4.27.** Évolution du TSN au niveau SCTP avec l'extension *QoS_Measurement_Chunk* au cours de l'association (2 mobiles) (+ zoom sur le changement de cs2 vers cs1).

Les deux mobiles commencent à transmettre à différence de temps très faible de l'ordre 1ms, nous constatons que le mobile 2 a un flux de transmission plus important que celui du mobile1 (c.f.

figures 4.26 et 4.27). Lors du changement de cs2 à cs1 le mobile 2 subit un hors temps (*timeout*) sans pouvoir influencer sa communication. En ce qui concerne les variations des paramètres de congestion, la figure 4.26 montre que l'évolution de *cwnd* et de *ssthresh* en fonction du temps suit bien notre supposition liée à l'extension mise en oeuvre par cette étude. Lors de la phase de *handover*, pour les deux mobiles, *cwnd* recommence sur la nouvelle cellule avec $cwnd_{new}$, que nous avons donné dans le paragraphe 4.4.1, pas avec un *slow start*.

À ce niveau, toutes les simulations, auxquelles nous avons fait recours, ont servi pour la validation de notre approche dans différents contextes de dégradation de la qualité de transmission sur l'interface radio. Nous nous sommes intéressés particulièrement au *handover* inter SGSN dans le cas de réseaux de type GPRS ou EGPRS. Les résultats obtenus nous ont montrés une bonne performance de l'approche proposée, à savoir le *QoS_Measurment_Chunk*. Que se soit comparé au SCTP standard, à TCP ou à TCP Eifel, le mécanisme introduit par l'usage de *QoS_Measurment_Chunk* offre une amélioration de la qualité de communication dans le sens d'assurance de la continuité de celle ci suite à un *handover*. Maintenant, et pour valoriser notre approche nous allons la tester dans le cas d'un *handover* inter-système ou plutôt inter-RAT et plus particulièrement un *handover* d'une cellule EGPRS vers un point d'accès WLAN.

### 4.4.6 Comportement du *QoS_Measurement_Chunk* : Application dans un contexte de *handover data* Inter-RAT (EGPRS/WLAN)

Le *QoS_Measurement_Chunk* contient principalement le débit utile maximale pour un contexte PDP établi au sein d'une association SCTP. Ce *chunk* est échangé à l'établissement de l'association pour informer le réseau de la capacité maximale fournie par le réseau d'accès à l'utilisateur final. Le mobile envoie un *QoS_Measurement_chunk* chaque fois que ses paramètres d'accès changent. Cette information peut être particulièrement utile pour un *handover* de données, et particulièrement dans un contexte multi-RAT. Ce paragraphe aborde le *handover* de données entre un réseau d'accès de type EGPRS et un réseau d'accès de type WLAN.

Nous comptons simuler un *handover* inter-RAT : il s'agit d'un *handover* de données d'une cellule EGPRS vers un point d'accès WLAN. Le scénario simulé est donné par la figure 4.28, donnant les différents éléments des deux réseaux mis en évidence, et par la figure 4.29 décrivant l'architecture protocolaire considérée. Les paramètres de simulations relatifs à la cellule EGPRS sont identiques à ceux utilisés dans les scénarii précédemment étudiés. Nous supposons que la dégradation de qualité dans la cellule EGPRS est commandée par le BLER. Sur l'interface radio WLAN nous considérons un modèle de propagation de type «*Two ray ground*» qui considère le chemin direct et un chemin réfléchi. Pour la gestion de *handover* EGPRS/WLAN nous considérons une approche d'accouplement avec pertes comme nous n'avons pas une connexion directe entre la passerelle WLAN et les équipements réseau EGPRS. Le trafic WLAN ne passerait pas par le réseau de coeur EGPRS. Nous supposons, également, que le hôte fixe (le serveur) est *multihomed* où chaque interface correspond à une technologie d'accès radio différente. Le hôte mobile n'est pas *multihomed*. De là l'adresse primaire de l'association est celle relative au lien EGPRS (interface 1, cf. figure 4.28) et une adresse alternative correspond au lien WLAN (interface 2, cf. figure 4.28).

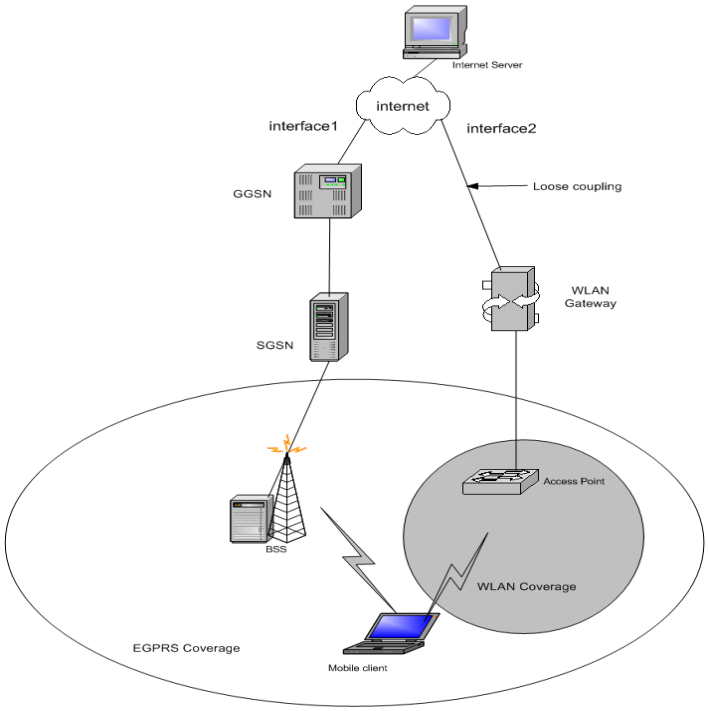



**Figure 4.28.** Architecture simulée pour un *handover* EGPRS/WLAN.

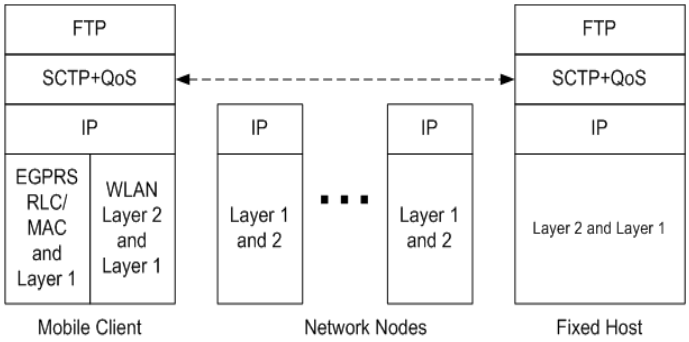



**Figure 4.29.** Architecture protocolaire.

La norme IEEE 802.11 WLAN peut offrir un débit utile élevé pour l'accès d'utilisateur [LEH03], et est envisagée pour coopérer avec le réseau cellulaire EGPRS afin de fournir de meilleurs services de données aux clients mobiles. Ici nous présentons les résultats simulés du *handover* de données entre EGPRS et WLAN avec l'extension proposée comparés au SCTP standard. Nous considérons SCTP avec l'extension *QoS_Measurement_Chunk* aussi bien au niveau du client mobile qu'au niveau de l'hôte fixe. Le client mobile supporte les deux technologies d'accès, EGPRS et WLAN, aux niveau des couches RLC/MAC et physique. Nous supposons, comme nous l'avons spécifié, que l'hôte fixe est un noeud SCTP *multihomed* dont chaque interface correspond à un type de technologie d'accès radio. Le mobile dispose d'une adresse unique pendant toute la durée de la communication, il fournit une seule adresse IP pour la gestion de *handover* que nous supposons déclenché à l'initiative du réseau. La figure 4.30 représente l'évolution au cours du temps des

paramètres de contrôle de congestion relative au protocole SCTP avec l'extension du *QoS_Measurment_Chunk* comparée au SCTP standard. Les résultats prouvent que l'extension proposée présente de meilleures performances au niveau transport. En effet, après l'exécution du *handover*, la fenêtre de congestion (*cwnd*) augmente sans interruption et sans le déclenchement de la phase de *slow start* comme c'est le cas pour le SCTP standard qui exécute l'évolution normale de la communication et de l'amélioration de la qualité de réception.

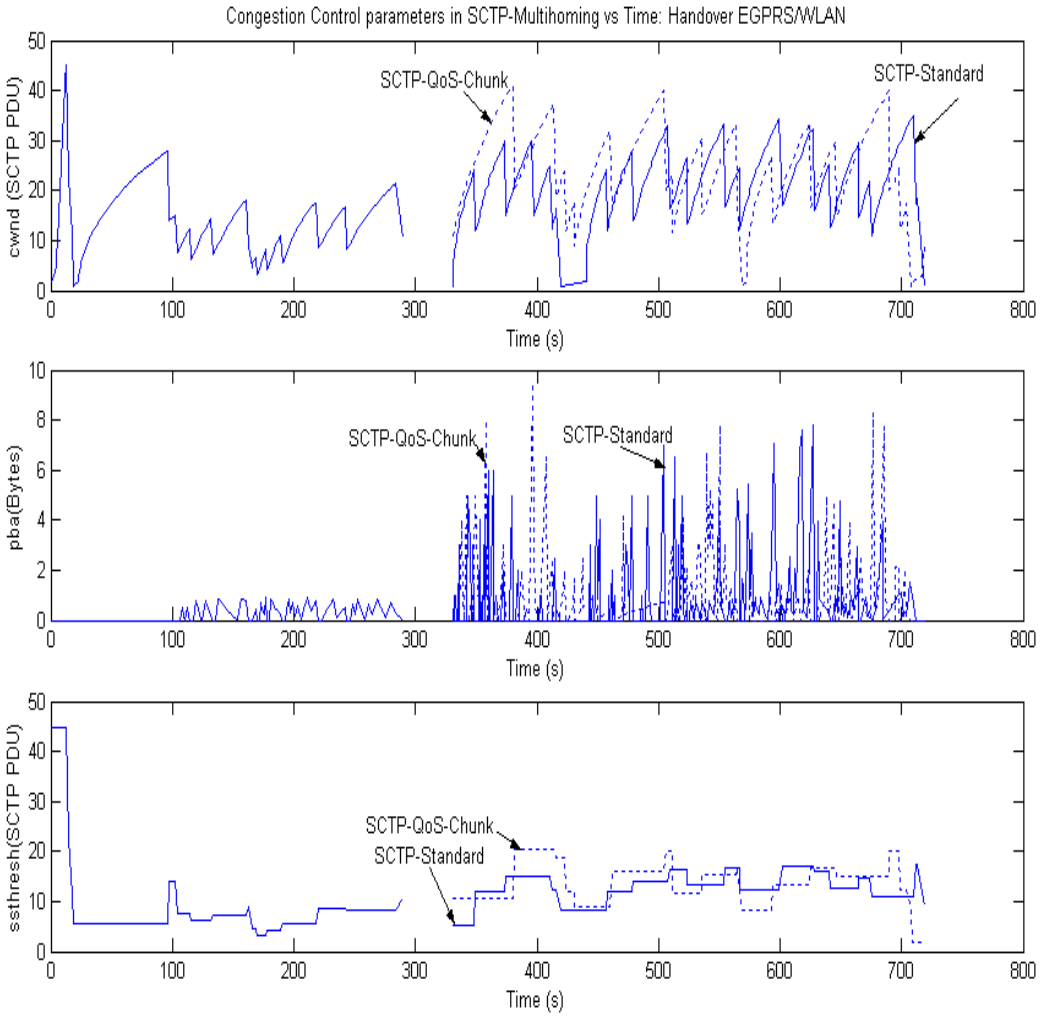



**Figure 4.30.** Variations des paramètres de contrôle de congestion dans un contexte de *handover* EGPRS/WLAN.

Concernant l'extension *QoS_Measurment_Chunk,* l'augmentation du débit est plus rapide que dans le cas de SCTP standard. La figure 4.31 montre l'évolution du numéro de séquence (TSN) des *chunks* de données transmis. Nous remarquons que SCTP avec *QoS_Measurement_Chunk* présente un comportement meilleur que celui du SCTP standard. En fait, il fournit un débit plus élevé. Juste

après le basculement d'interface le SCTP standard déclenche des retransmissions importantes par comparaison à ce qui se produit avec SCTP avec *QoS_Measurement_Chunk*. Ceci se produit dans le but de reconstruire les pertes causées par des dégradations de qualité de service sur le lien EGPRS et le déclenchement d'une phase *slow start* (cf. figure 4.32). La courbe de *cumulative* TSN montre la réception de TSNs dupliqués ce qui provoque une expiration de temporisateur et donc le déclenchement d'une phase *Slow start* (cf. figure 4.30 à 415sec). Alors que dans le cas de SCTP avec *QoS_Measurement_Chunk*, nous n'avons aucune expiration de temporisateur parce que l'émetteur transmet sans interruption sans pertes et ainsi les pertes sur le lien radio sont décorrélées de la congestion sur la liaison fixe (filaire) du réseau.

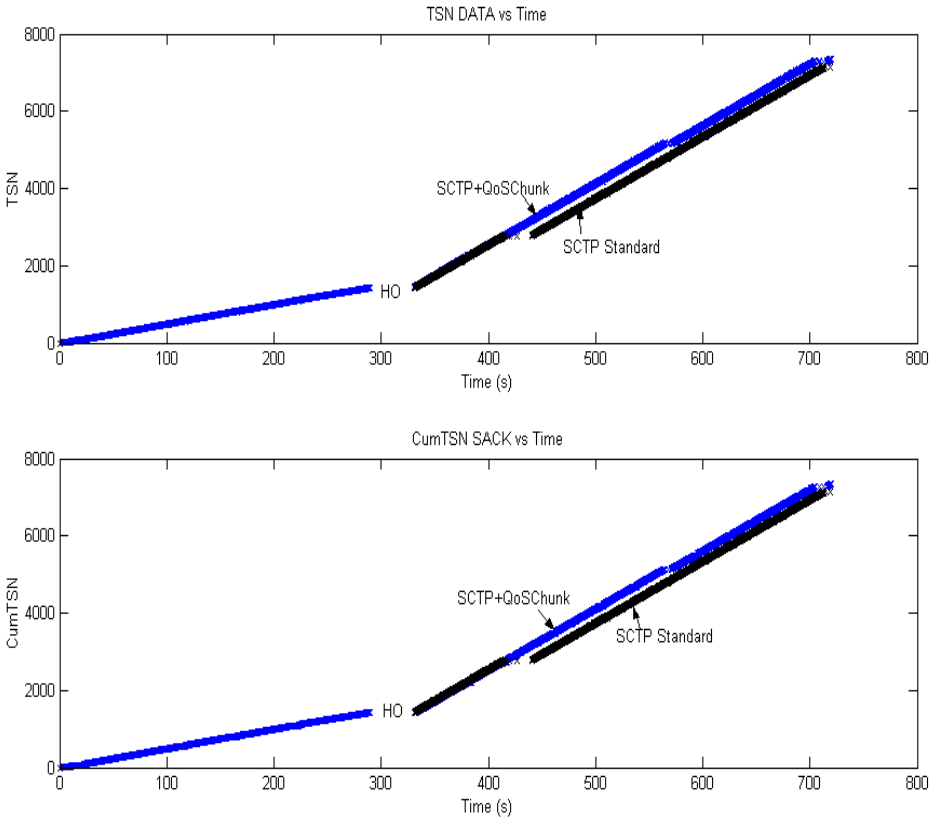



**Figure 4.31.** Variations du TSN au niveau SCTP.

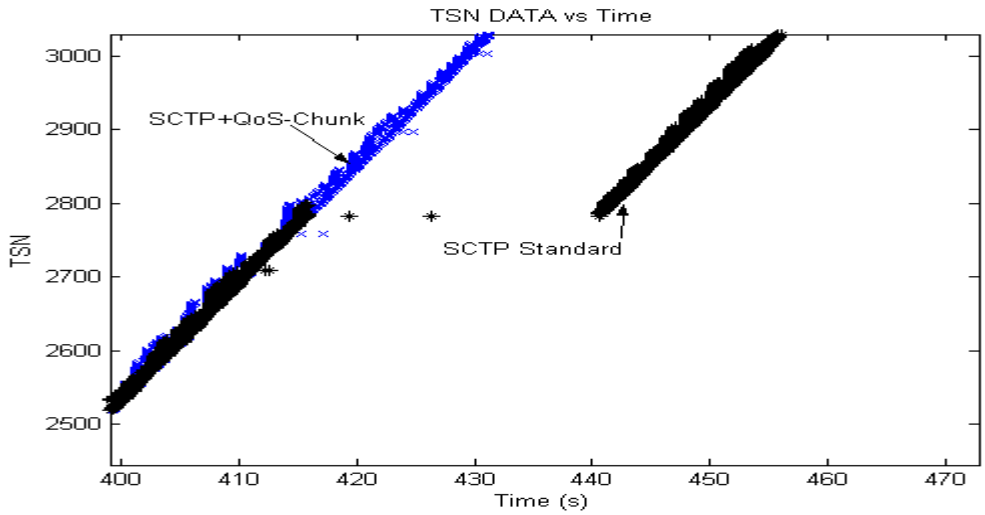



**Figure 4.32.** Variations de TSN data après *Handover* (sur l'interface WLAN).

## 4.5. Conclusion

Dans ce chapitre nous avons présenté une nouvelle extension de SCTP qui se base sur un couplage du *multihoming* et de mobilité. Le *QoS_Measurement_Chunk* est une approche qui a pour objectif l'amélioration des performances de la couche transport en environnement radio. Cette solution est inspirée des alternatives proposées avec le mSCTP, cSCTP ou encore avec l'idée d'une couche GLL.

Nous proposons des modifications des algorithmes de contrôle de congestion en SCTP en fonction de l'information remontée par le mobile au réseau (GGSN) par l'intermédiaire du *chunk* de contrôle *QoS_Measurement_Chunk*. Ce nouveau mécanisme est utile pour exécuter le *handover* de données inter RAT. L'exemple que nous avons étudié est le *handover* de données d'une cellule EGPRS vers un point d'accès WLAN.

Les résultats de simulation prouvent que SCTP avec *QoS_Measurement_Chunk* est plus performant que le SCTP standard. La modification que nous avons proposée, est utile pour une adaptation du protocole de transport à un contexte radio. SCTP avec sa caractéristique de *multihoming* auquel s'ajoute le *QoS_Measurement_Chunk,* assure le *handover* de données inter et intra-RAT avec un minimum de pertes. En fait, les variations de *cwnd* sont adaptées aux conditions de transmissions sur l'interface radio. L'association SCTP n'exécute pas suite à un *handover* le *slow start* comme dans le cas de SCTP standard qui ralentit le débit de transmission. Notre approche permet une meilleure gestion de *handover* de données qu'avec les protocoles de transport usuels aussi bien dans le cas d'une technologie d'accès radio homogène, que dans le cas des technologies d'accès hétérogènes mettant en cause différents paramètres de qualité de service.

Dans la modification développée dans ce chapitre, nous avons considéré une simple relation de variations des paramètres de contrôle de congestion suite à une variation des conditions de

transmission sur l'interface radio (que se soit sur le lien primaire, avec la cellule serveuse, ou suite à un *handover*). Dans le chapitre suivant nous proposons d'établir des relations plus concrètes qui permettent de lier le débit utile au niveau couche transport (SCTP) au débit utile au niveau couche liaison de données (EGPRS). Ce qui nous permet de lier les variations de la taille de fenêtre de congestion (*cwnd*) au débit utile au niveau RLC/MAC par des équations mathématiques paramétrables.

# Chapitre 5 : Aspect *Cross-Layer* entre le mécanisme RLC/MAC EDGE/EGPRS et SCTP

## 5.1. Introduction

Dans les chapitres précédents nous avons exploité les deux caractéristiques fondamentales de SCTP, liées à la procédure de transfert de données, et leurs apports surtout en environnements radiocommunication. Nous avons également proposé une modification du protocole SCTP. Elle consiste en un couplage du *multihoming*, de la mobilité et de l'aspect de *chunk* de contrôle en SCTP, pour une meilleure gestion de *handover* de données dans un environnement présentant une homogénéité ou une hétérogénéité des technologies d'accès radio. Notre solution, basée sur un mécanisme *cross-layer* entre SCTP et la couche liaison de données en EGPRS/EDGE, propose une modification des algorithmes de contrôle de congestion afin d'assurer une adaptation du flux de transmission au niveau transport aux variations des conditions de transmissions sur l'interface radio.

L'objectif du présent chapitre est de formuler cet aspect *cross-layer* par des relations mathématiques qui dépendent, bien évidemment, des conditions de propagation et de transmission sur l'interface radio. Ces équations consistent à exprimer le débit utile au niveau transport en fonction du débit utile au niveau RLC/MAC.

Pour le faire, nous présentons les caractéristiques fondamentales introduites par la technologie EDGE/EGPRS et exploitées dans nos simulations. Ensuite, nous nous intéressons aux différents aspects *cross_layer* et leurs réalisations. Par la suite, nous développons des simulations qui nous permettent de représenter le débit utile au niveau SCTP en fonction du débit utile au niveau RLC/MAC EGPRS/EDGE. Enfin, nous approximons les variations mesurées par des variations analytiques théoriques dont le degré d'approximation est testé par des outils de statistiques (Khi2).

## 5.2. Caractéristiques de la technologie EDGE/EGPRS

### 5.2.1 Modulation et codage

Il s'agit d'un nouveau mode de modulation qui permet d'augmenter notablement le débit efficace sur l'interface radio GSM. Il améliore ainsi l'efficacité spectrale et la capacité du réseau. A la limite, les modifications sur le réseau ne portent que sur les émetteurs-récepteurs des stations de base. Le

déploiement d'une couverture radio EDGE est plus complexe que pour le GPRS. En effet, le changement de la modulation radio, outre qu'il impose des évolutions matérielles des stations de base GSM, entraîne une modification sensible de la portée, en fonction du débit et de la qualité requise.

La modulation utilisée dans le GSM est la modulation GMSK (*Gaussian Minimum Shift Keying)*, qui associe à chaque bit un état. Pour atteindre des débits élevés par intervalle de temps, l'EDGE utilise la modulation 8-PSK. Avec cette modulation, on a huit états (voir figure 5.1). Ainsi, le nombre de symboles transmis dans une certaine période est le même que pour le GPRS mais cette fois, chaque symbole transmis contient 3 bits donc le débit est augmenté. Cependant, la contrepartie est que la distance entre symbole est moindre qu'avec le GPRS. Le risque d'interférence inter-symbole s'en trouve accru. Si les conditions de réception sont bonnes, cela ne pose pas de problèmes mais dans le cas contraire, il y aura des erreurs. Des bits supplémentaires seront utilisés pour ajouter plus de codes de corrections d'erreurs afin de reconstruire les données.

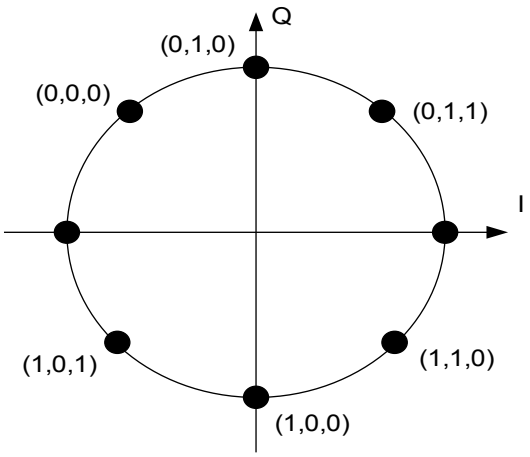



**Figure 5.1.** Constellation de modulation 8-PSK.

La R98 définit quatre schémas de codage (CS : *Coding Scheme*) différents : CS-1, CS-2, CS-3 et CS-4, pour les blocs radio transmis dans des blocs de données RLC. Dans la R99, la structure des blocs radio pour le transfert de données utilisateur est différente pour GPRS et EGPRS. Pour EGPRS, un bloc radio (quatre *bursts* normaux, 20ms) pour le transfert de données se compose d'un en-tête RLC/MAC et d'un ou deux blocs de données RLC. Neuf modulations et schémas de codages (MCS : *Modulation and Coding Schemes*) différents, de MCS-1 à MCS-9, sont définis pour les blocs radio EGPRS transportant des blocs de données RLC. Le détail des schémas de codages EGPRS est donné dans le tableau 5.1. La transmission et la réception des flux de données sont les mêmes pour GPRS et EGPRS, excepté pour EGPRS MCS-9, 8 et 7, où quatre *bursts* sont utilisés pour supporter deux blocs RLC.

Les MCSs sont répartis en quatre familles [EMM03] : A, A', B et C. Chaque famille inclut plusieurs MCSs. Une unité de données de base avec une taille fixe est associée à chaque famille. Un bloc radio codé avec un MCS peut supporter une ou plusieurs unités de données de base de sa famille (voir figure 5.2). Les tailles de l'unité de données de base pour les familles A, A', B et C sont respectivement 37, 34, 28, 22 octets.

Tableau 5.1. Modulation et schémas de codage en EDGE [STU03].

| Schéma de codage | Modulation | CR *(Code Rate)* | Données par bloc radio (bit) | Famille | Débit Maximal (kbps) |
|---|---|---|---|---|---|
| MCS-9 | 8-PSK | 1.0 | 2x592 | A | 59.2 |
| MCS-8 | | 0.92 | **2x544** | A' | 54.4 |
| MCS-7 | | 0.76 | 2x448 | B | 44.8 |
| MCS-6 | | 0.49 | 592 **(544+48)** | A/A' | 29.6 (27.2) |
| MCS-5 | | 0.37 | 448 | B | 22.4 |
| MCS-4 | GMSK | 1.0 | 352 | C | 17.6 |
| MCS-3 | | 0.80 | 296 **(272+24)** | A/A' | 14.8 (13.6) |
| MCS-2 | | 0.66 | 224 | B | 11.2 |
| MCS-1 | | 0.53 | 176 | C | 8.8 |

Dans le tableau **5.1**, la troisième colonne donne le taux de codage des données. La quantité de données brute transmise durant un bloc radio est présentée en quatrième colonne. Les valeurs en gras indiquent l'existence de bits de remplissage (*padding*).

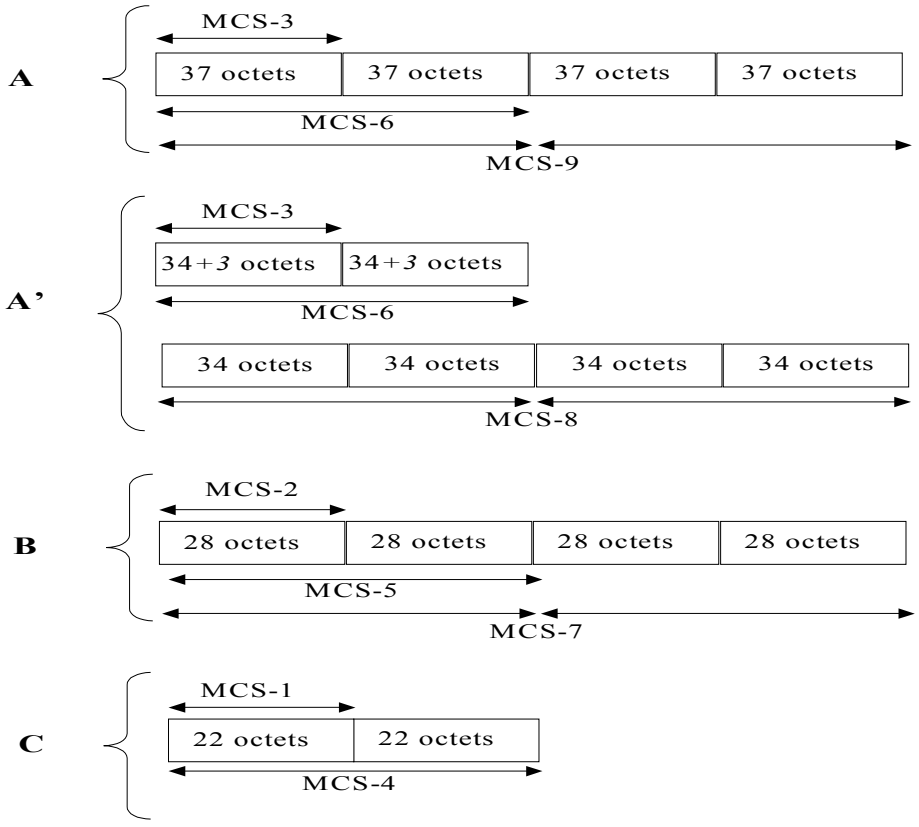
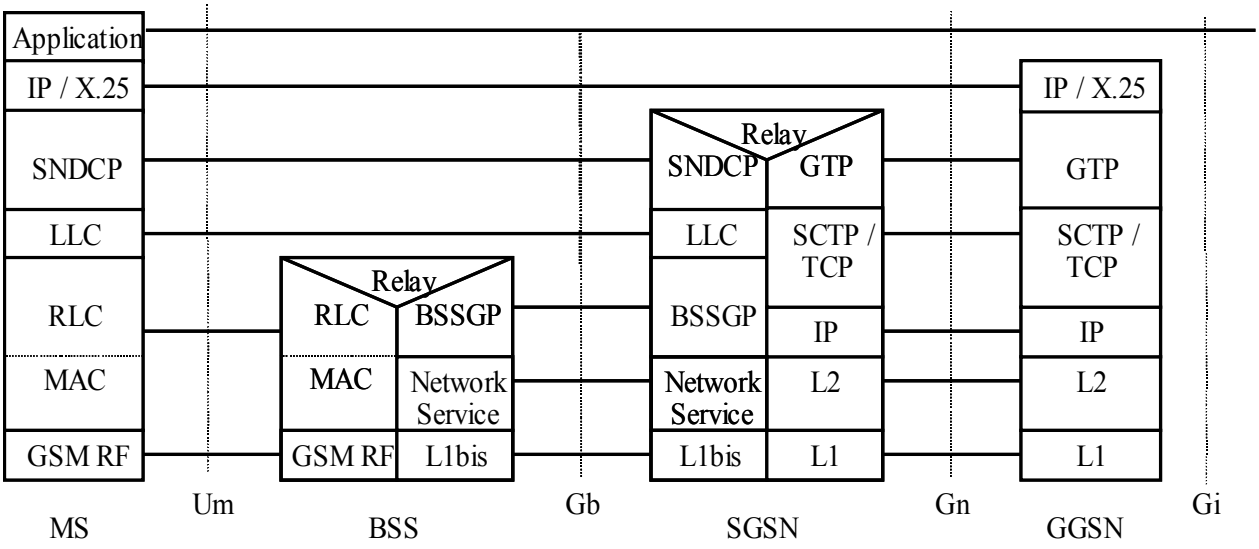



**Figure 5.2.** Différentes familles de MCS et leurs unités de données utiles[EMM03] .

### 5.2.2 Pile protocolaire

En EGPRS, un mobile gère des piles de protocoles situées sur deux plans différents : le plan de signalisation et le plan de transmission. Le plan de transmission sert à transférer toutes les données utilisateurs. Le plan de signalisation sert à assurer la gestion de la mobilité (MM : *Mobility Management*), mais aussi la transmission de messages courts [SAM02]. Sur le plan signalisation on trouve au sommet la couche GMM (*GPRS Mobility Management*) surmontée des couches SM (*Session Management*) et GSMS (*GPRS SMS*). Sur le plan de transmission, on trouve au sommet de la pile le protocole SNDCP (*Sub-Network Dependent Convergence protocol*) surmonté de différents PDP (*Packet Data Protocol*) (voir figure 5.3).




où : **GTP** : *GPRS Tunneling Protocol ,* **SNDCP :** *Sub-network Dependent Convergence Protocol*,

**BSSGP** : *Base Station System GPRS Protocol ,***LLC** : *Logical Link Control* , **RLC** : *Radio Link Control*

**Figure 5.3.** Pile protocolaire sur le plan transmission.

### 5.2.3 Évaluation de performances et contrôle de flux en EDGE/EGPRS

Les modifications introduites avec le standard EDGE concernent principalement la couche RLC/MAC et la couche physique. La caractéristique fondamentale de EDGE [STU03] est le mécanisme de contrôle de qualité de lien LQC *(Link Quality Control)* qui permet l'adaptation de la modulation et des schémas de codage aux changements de la qualité du lien radio (MCSs). Bien que l'adaptation de lien (LA : *Link Adaptation*) soit déjà définie dans les standards GPRS.

Le contrôle de qualité de lien est un terme employé pour des techniques adaptant le codage canal du lien radio à la variation de la qualité de canal. Différents schémas de codage et modulation sont optimaux pendant différentes situations, selon la qualité de lien. Le LQC utilisé pour EDGE est

exécuté par les techniques de :
- Adaptation de lien (LA),
- IR *Incremental Redundancy*.

#### 5.2.3.1 Contrôle de flux

Dans le but de supporter des débits élevés d'une manière efficace la procédure de contrôle de flux du protocole RLC/MAC, telle qu'elle est décrite en GPRS, doit être modifiée. La taille de fenêtre (WS : *Window Size*) [STU03], est augmentée de 64 en GPRS à l'ordre de 64 à 1024 dans EGPRS. Par conséquent l'espace du numéro de séquence (SNS : *Sequence Number Space*) est adapté de 128 à 2048 et le numéro de séquence du bloc (BSN : *Bloc Sequence Number*) est augmenté de 7 bit à 11 bit.

La modification au niveau couche MAC (*Medium Access Control*) introduite en EGPRS est liée à l'établissement de TBF (*Temporay Block Flow*) sur le lien descendant. Cette procédure est identique à celle en GPRS sauf que certains messages d'affectation de canaux de trafic ou de contrôle (exp : IMMEDIATE ASSIGNMENT ou PACKET DOWLINK ASSIGNMENT) [EMM03] contiennent des paramètres supplémentaires qui sont dédiés au TBF en mode EGPRS. Ces paramètres sont :

- La taille de fenêtre en EGPRS, qui donne la taille de fenêtre utilisée par le protocole RLC pour le transfert de blocs de données RLC sur le lien descendant.
- Le mode de mesure de qualité de lien, qui indique aussi bien la procédure adoptée par le mobile pour remonter les mesures de qualité par intervalle de temps ou par TBF, que la méthode de remontée des mesures d'interférence.
- Le transfert de blocs de données RLC en EGPRS utilise exactement les mêmes concepts qu'en GPRS [EMM03]. Les blocs RLC sont transmis en séquence et le contrôle est assuré grâce au mécanisme de fenêtre glissante.

Le réseau choisit la taille de la fenêtre durant la phase d'établissement de TBF. La taille de fenêtre RLC dépend du nombre de TS (*Time Slot*) alloué au mobile. Une taille maximale de la taille de fenêtre a été définie relativement à chaque nombre de TS attribué (voir tableau 5.2).

Tableau 5.2 Taille de la fenêtre RLC en fonction du nombre de TSs alloués[EMM03]

| **Nombre de TSs alloués** | ***1*** | ***2*** | ***3*** | ***4*** | ***5*** | ***6*** | ***7*** | ***8*** |
|---|---|---|---|---|---|---|---|---|
| **Taille minimale de fenêtre RLC** | 64 | 64 | 64 | 64 | 64 | 64 | 64 | 64 |
| **Taille maximale de fenêtre RLC** | 192 | 256 | 384 | 512 | 640 | 768 | 896 | 1024 |

#### 5.2.3.2 Retransmission des données

En GPRS, lors de la réception d'une trame erronée, la retransmission de cette trame se fait selon le type de codage d'origine, avec lequel la trame est envoyée la première fois. Alors avec EDGE, la retransmission se fait avec le type de codage adéquat. Ceci est à cause de la fonction de réassemblage des paquets introduite avec la technologie EDGE.

Le taux de transfert de données, en EDGE, dépend non seulement de la modulation et du schéma de codage mais également de la qualité du lien et du temps de propagation. La technique de mesure d'un réseau EDGE est d'analyser chaque bloc de transmission composé de quatre séquences et d'en estimer la probabilité d'erreur. En cas de problème, une adaptation automatique de la modulation et du schéma de codage (donc du débit) est effectuée.

## 5.3. Principe d'une optimisation *Cross-Layer*

Le sujet d'une amélioration introduite avec une conception *cross-layer* [SAN03], a été développé dans le groupe d'étude sur les réseaux d'accès mobiles à large bande (MBWA : *Mobile Broadband Wireless Access networks*) au sein de l'IEEE, dans le but d'améliorer le débit utile et réduire la latence sur les liens montant et descendant.

La motivation d'une amélioration de performances des réseaux a augmenté l'intérêt pour les protocoles qui se fondent sur des interactions entre différents niveaux. Récemment, les systèmes *cross-layer* sont apparus, particulièrement pour les réseaux radio-mobiles. La recherche, spécifique pour des systèmes *cross-layer*, vise une optimisation globale des performances de la pile protocolaire d'un réseau plutôt qu'une optimisation locale de chaque couche. La technique consiste à créer un système de plus en plus sensible aux particularités des réseaux mobiles tout en tenant compte en même temps de l'information fournie par les différentes couches [AND04].

Le concept d'une modélisation *cross-layer* est basé sur l'architecture protocolaire de sorte que les couches du modèle OSI (*Open System Interconnection*) peuvent échanger des informations afin d'améliorer les performances globales de réseau. Le modèle en couche a été principalement conçu pour les réseaux filaires [LIL06]. En environnement radio (par exemple WLANs, EDGE...), où les utilisateurs partagent le même support de transmission (l'interface aire), une sous-couche MAC est introduite au modèle OSI au niveau 2 qui redéfinit légèrement la fonctionnalité de la couche réseau. L'inconvénient majeur du modèle en couches [LIL06] est qu'il est fortement rigide et ne montre aucune flexibilité dans les environnements à conditions de transmission variables. Quoique le modèle OSI a servi les concepteurs de réseau durant des années, avec des principes adoptés par diverses implémentations et applications, il ne pourrait pas suivre la croissance des demandes des applications et la pénétration des technologies radio.

L'optimisation *Cross-Layer* définit un concept général de communication entre les couches, étant donné certaines interactions entre elles, et ayant pour résultat des améliorations de performances du réseau. Elle vise, en couplant les fonctionnalités des couches du modèle OSI, l'amélioration des performances du système. L'idée d'une optimisation *cross-layer* pratique est de trouver une abstraction appropriée de chaque couche [RAI04] et des mécanismes adéquats d'interactions inter-couches. Les interactions *cross-layer* possibles, comme elles sont définies par [LIL06], sont

données par la figure 5.4, nous représentons en couleur foncée les interactions considérées dans nos travaux.

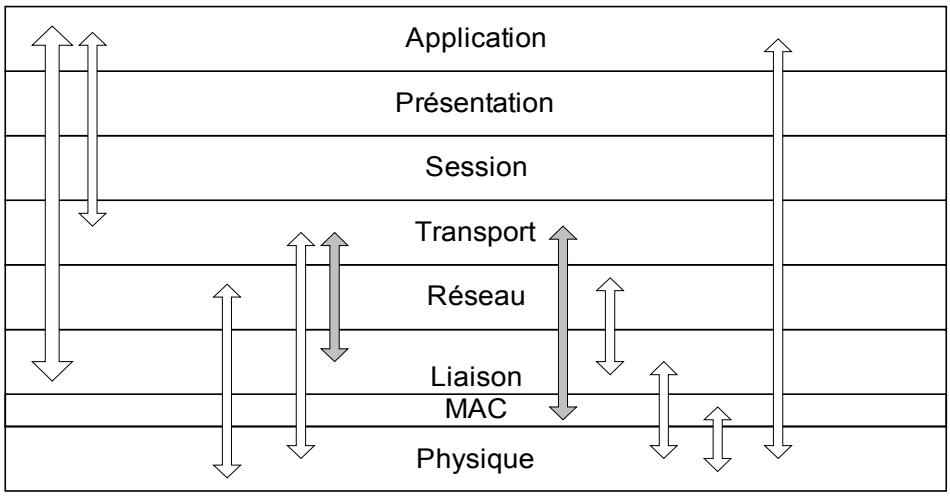



**Figure 5.4.** Les interactions *Cross-Layer* possibles [LIL06].

Les paramètres *cross-layer* appropriés, qui pourraient être échangés entre les couches, dépendent des fonctionnalités considérées pour l'interaction *cross-layer*, des contraintes et des objectifs spécifiés par les applications de l'utilisateur final. L'information potentiellement utile *cross-layer* se différencie comme c'est décrit en [RAI04] par :

- L'information de l'état du canal (exp : estimation de la réponse de canal en temps et en fréquence, l'information de localisation, la puissance du signal, le niveau d'interférence...),
- Les paramètres liés à la QoS (exp : le débit en sortie, le retard de transmission, le taux d'erreur binaire ou bloc...),
- L'information liée aux ressources (disponibilité, possibilité de réception multi-utilisateurs...),
- Le modèle de trafic (les flux de trafic, la connaissance du débit, la fragmentation de données, la taille de paquet et les informations sur les tailles de file d'attente),
- etc.

Dans ce qui suit nous nous intéressons à l'étude des interactions *cross-layer* entre la couche liaison et la couche transport.

## 5.4. Modélisation *Cross-Layer* : Transport/Liaison

### 5.4.1 Problématique

Dans les réseaux mobiles, la capacité d'un lien n'est pas une quantité fixe et dépend du protocole MAC (*Medium Access Control*) utilisé, et des conditions de propagation. Les protocoles MAC sont conçus [XIN05] pour réduire les collisions, afin d'assurer un débit utile élevé, et une distribution

équitable de la bande passante disponible aux différents noeuds concurrents. Puisque le débit utile dans un réseau mobile dépend du protocole et des paramètres de la couche MAC, la question d'optimisation du débit de bout en bout doit être mise en valeur dans un contexte *cross-Layer,* autrement dit la stratégie de contrôle de flux doit être implémentée aussi bien au niveau liaison qu'au niveau transport.

Plusieurs améliorations ont été apportées à la couche liaison des systèmes radio de 2.75 G (EDGE) et 3G (UMTS). Cependant, d'autres améliorations liées au contrôle d'admission des paquets et à l'ordonnancement sont nécessaires pour maximiser la capacité et la satisfaction des utilisateurs. De nombreuses études se sont intéressées au développement des algorithmes d'ordonnancement au niveau MAC telles que [PAN99, KER03, XIN05]. Dans notre étude nous focalisons nos intérêts à la recherche d'une relation permettant de lier la réactivité au niveau couche transport aux changements des conditions de propagation et aux adaptations de la couche liaison.

Nous abordons la question d'interactions *cross-layer*, lorsque les données de la couche physique et de la couche MAC sur le support radio sont partagées avec la couche transport afin de fournir des méthodes efficaces d'allocation de ressources et d'applications réseau. Les réseaux futurs selon [SAN03], devront fournir un lien *«impedance matching»* entre les conditions instantanées du canal radio et les besoins en terme de capacité liés au trafic et aux conditions de congestion rencontrées dans le monde de transmission de paquets. L'adoption rapide de la technologie radio continue, couplée à la croissance explosive de l'Internet, augmentera la demande des services sur les réseaux mobiles. L'hypothèse de notre étude est les gains qui peuvent être accrus par les techniques *cross-layer* qui éliminent la séparation qui existe actuellement entre les couches physique et MAC, et la couche transport de la pile protocolaire.

## 5.4.2 Modélisation

### 5.4.2.1 Solutions existantes dans la littérature

Les mécanismes *cross-layer* devraient permettre la possibilité de consulter l'information disponible en temps réel sur chaque couche du réseau, de nos jours ceci n'est pas possible d'après [AND04]. Les auteurs dans [MAR04] affirment que quelques fonctions de réseau, telles que la gestion d'énergie, la sécurité et la coopération sont des aspects *cross-layer* de nature. Les auteurs proposent une architecture *cross-layer* présentant un plan d'états de réseau qui répartit chaque fonctionnalité partagée de réseau entre les différentes couches.

Différents aspects *cross-layer* sont décrits par les auteurs dans [AND04]. Dans cette article, les auteurs définissent les interactions possibles qui peuvent avoir lieu entre les couches du modèle OSI, ainsi que les informations qui peuvent être échangées entre elles. Le tableau 5.3 récapitule les données qui influent les performances du réseau au niveau de chaque couche.

Tableau 5.3. Les informations influant les performances du réseau [AND04].

| ***Couche*** | ***Données*** |
|---|---|
| Application | Algorithme de contrôle de topologie, localisation du serveur, le plan du réseau, |
| Transport | Fenêtre de congestion, les instants du hors temps (*timeout*), le taux de perte de paquets |
| Réseau | Affinité de routage, durée de vie de routage, routage multiple |
| Liaison/MAC | Bande passante, qualité du lien, retard de transmission d'un paquet MAC |
| Physique | Localisation du mobile, modèle de mobilité, information du SNR, les conditions de transmissions radio |

Certaines solutions *cross-layer* proposent l'intégration d'un nouveau module au niveau du noeud mobile. Il s'agit d'une fonctionnalité qui assure la communication inter-couches. Dans [DZM05], par exemple, les auteurs définissent la méthode de *cross-layer* par l'introduction d'un module supplémentaire dans la pile protocolaire du noeud mobile. Ce module, appelé *Congestion Control Module* (CCM), est capable d'ajuster le flux de données sortant en fonction des mesures de capacité. Il est inséré au dessous de la couche transport (cf. figure 5.5). Ce module coopère avec la couche liaison pour fournir à la couche 4 l'information de contrôle de congestion en utilisant une technique de collaboration *cross-layer*.

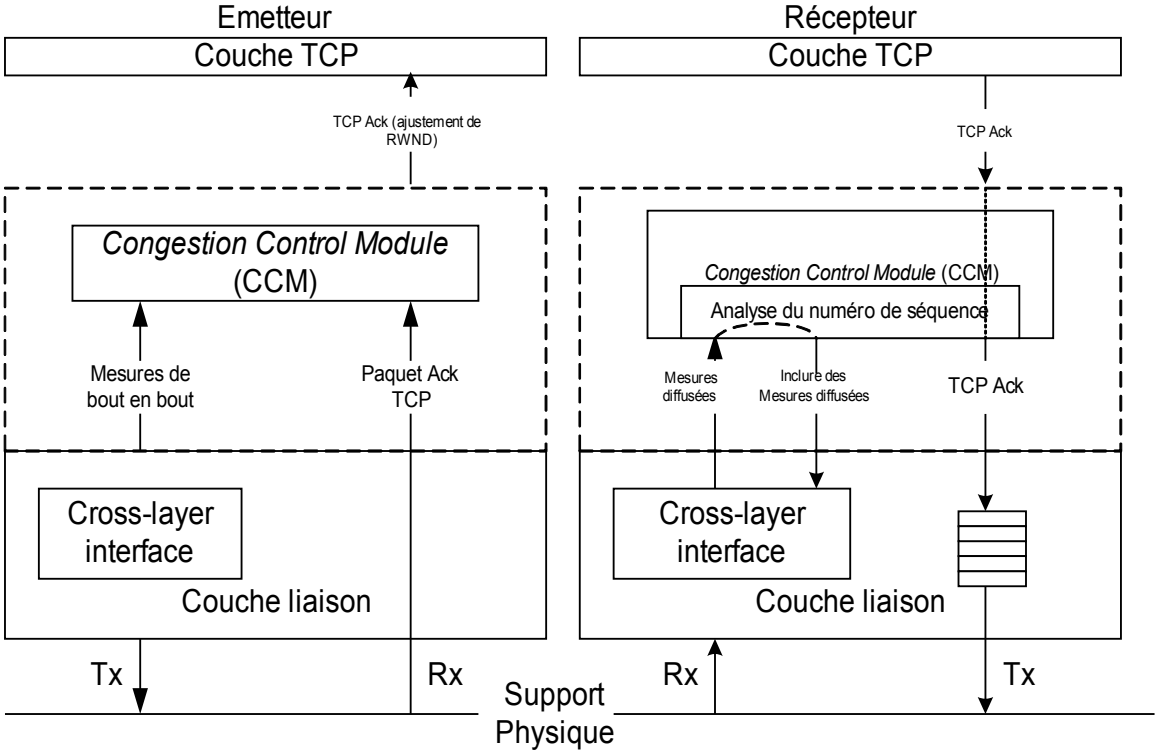



**Figure 5.5.** Architecture du protocole C$^3$TCP (*Cross-layer Congestion Control*)[DZM05] .

### 5.4.3 Solution proposée

Nous visons le problème de congestion réseau dans les réseaux mobiles comme raison de dégradation potentielle de performance. La congestion se produit quand la quantité de données transmises sur le réseau dépasse la capacité disponible, ce qui engendre une perte des données transmises. Une telle situation diminue la fiabilité de réseau. Les protocoles de transport fiables améliorent leurs performances par l'implémentation de différents mécanismes de rétablissement d'erreurs. La solution proposée pour éviter la congestion est de contrôler et optimiser la quantité de trafic transmise sur le réseau, tout en tenant compte de la disponibilité limitée des ressources du réseau.

Nous modélisons un aspect *cross-layer* entre la couche transport et la couche liaison dans un contexte de réseau cellulaire de type EDGE/EGPRS. Au niveau 4 nous considérons le protocole SCTP. Le but de cette modélisation est d'étudier l'adaptation du mécanisme de transmission au niveau transport à celui au niveau liaison dans le cas de EDGE. Notre objectif est d'avoir un lien entre la réactivité au niveau couches supérieures et les changements des conditions de transmission ainsi qu'avec les adaptations de la couche liaison.

Nous partons de l'idée proposée dans [THI04] qui consiste à concevoir un algorithme d'ordonnancement inter-utilisateurs qui améliore les performances au niveau TCP dans le cas des réseaux mobiles. L'algorithme développé se base sur une conception *cross-layer* qui suppose une relation de proportionnalité entre la taille de la fenêtre de congestion (*cwnd*) et le débit utile au niveau MAC. En effet, lier *cwnd* au débit utile de transmission au niveau liaison permet de limiter les problèmes de congestion au niveau 4. Les erreurs de transmission au niveau 2 et 1 sont interprétées par la couche transport comme des problèmes de congestion.

Pour cela nous simulons le comportement du débit utile au niveau 4 ainsi que celui au niveau 2, et estimons la relation entre les deux paramètres. Le débit utile peut être étroitement approximé par le rapport de la taille de fenêtre de congestion sur le RTT (*Round Trip Time*) moyen. La fenêtre de congestion est proportionnelle au débit utile au niveau couche MAC [THI04]. Cette approximation est justifiée par le fait que plus le débit au niveau MAC est important plus le nombre de paquets envoyés et des acquittements reçus par l'émetteur est important ce qui engendre une augmentation remarquable de la fenêtre de congestion au niveau transport. Le facteur de proportionnalité est supposé être le même pour tous les utilisateurs et donc n'influence pas les métriques d'ordonnancement. D'autre part, le RTT calculé au niveau transport est la somme de plusieurs composantes, y compris les temps d'attente, d'ordonnancement et de transmission du paquet sur l'interface radio, les retards de traitement et de stockage au niveau réseau et au niveau de l'émetteur ainsi que les retards dus à la transmission sur la partie fixe du réseau.

Au niveau SCTP, comme décrit dans le deuxième chapitre de ce mémoire, quand le seuil s*sthreshold* (*Slow Start threshold*) est atteint, le SCTP entre en phase de *congestion avoidance* au cours de laquelle la taille de la fenêtre de congestion augmente linéairement par un paquet pour chaque ACK reçu. La progression de fenêtre de congestion au cours de cette phase est limitée à une taille maximale négociée entre l'émetteur et le récepteur pendant l'établissement d'association SCTP et mise à jour pendant le processus de communication. Le SCTP a été à l'origine conçu pour les réseaux filaires où les pertes de paquet se produisent la plupart du temps en raison de congestion.

Pour cette raison le mécanisme de *congestion avoidance/recovery* est la seule réaction suite aux pertes SCTP.

Pour faire la modélisation notons *B* le débit utile pour la transmission d'une certaine quantité de données. Le débit *B* peut être obtenu en fonction de la taille des données *D* et du temps *T* mis pour la transmission de ces données sur un lien spécifique par :

$$B = \frac{D}{T}$$

Le débit utile *B* que nous calculons lors des simulations inclut les paramètres suivants :
- le délai d'attente,
- le retard associé à la transmission d'en-tête des couches physique et liaison.

Ceci signifie que les dépassements dus à l'ajout des en-têtes aux différentes couches (cf. figure 5.6) se produisant avant la transmission de données réellement sur le support physique, sont considérés comme un facteur réduisant le débit utile disponible sur le lien.

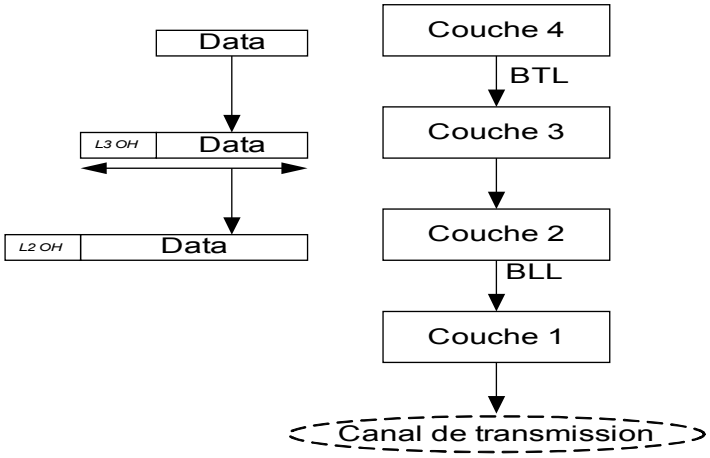



**Figure 5.6.** Modélisation du débit utile au niveau des couches protocolaires.

Soit $B_{TL}$ : le débit utile relevé au niveau 4 (TL: *Transport Layer*), $B_{LL}$ : le débit utile relevé au niveau 2 (LL: *Link Layer*). Ces deux grandeurs sont données par les équations suivantes :

$$B_{TL} = \frac{D_{TL}}{T_{TL}} \quad \text{et} \quad B_{LL} = \frac{D_{LL}}{T_{LL}}$$

Avec,

$$T_i = T_{i_{Ack}} - T_{i_E} \quad , \; i \in \{TL, LL\}$$

où : $T_{iack}$ est l'instant de réception d'un acquittement relativement à un paquet données et $T_{iE}$ est l'instant d'émission de ce même paquet de données.

L'objectif est de trouver une relation entre $B_{TL}$ et $B_{LL}$ afin d'adapter l'évolution de la taille de fenêtre de congestion aux conditions de propagation et de transmission radio (cf. figure 5.7), soit :

$$B_{TL} = f(B_{LL})$$

où *f* est une fonction du temps qui dépend à la fois des conditions de transmission aux niveaux 2.

Nous déterminons cette fonction au moyen des simulations et relativement à chaque schéma de codage et à chaque valeur du taux d'erreur bloc (BLER).

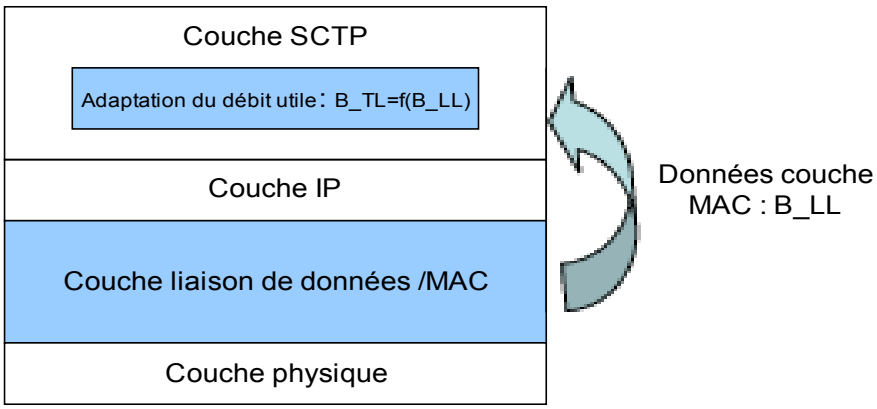



**Figure 5.7.** Principe *cross-layer* développé.

### 5.4.3.1 Description des simulations

Pour modéliser analytiquement les réactivités au niveau 4 en fonction des données MAC nous devons nous situer en régime permanent. Un état stationnaire du réseau correspond à une phase de *congestion avoidance*. L'identification de cette phase se fait en SCTP au moyen du paramètre *pba* (*partial-bit-ack*) qui prend la valeur zéro hors de la phase de *congestion avoidance*.

Nous étudions le lien descendant d'une liaison radio, au sein de laquelle le client mobile fait des mesures radio et les envoie au serveur. Lors des simulations nous considérons un seul mobile. Nous considérons également un lien duplex entre le GGSN et le SGSN sur lequel nous supposons avoir un taux de perte paquet de l'ordre de $10^{-5}$. Sur l'interface radio nous supposons avoir des conditions radio variables pour cette raison nous présentons dans le tableau 5.4 les valeurs de BLER considérées relativement à chaque schéma de codage.

Tableau 5.4. Les valeurs des BLER simulées relativement à chaque schéma de codage

| ***Schéma de codage*** | ***BLER*** |
|---|---|
| MCS1 | $10^{-4}$ |
| MCS2 | $10^{-3}$ |
| MCS3 | $10^{-2}$ |
| MCS4 | $10^{-1}$ |
| MCS5 | $10^{-5}$ |
| MCS6 | $10^{-4}$ |
| MCS7 | $10^{-3}$ |
| MCS8 | $10^{-2}$ |
| MCS9 | $10^{-1}$ |

Nous regardons la connexion client mobile/serveur de bout en bout. Nous considérons également un flux de données continu de type FTP (*File Transfert Protocol*).

### 5.4.3.2 Modifications proposées sur le contrôle de congestion

Le client mobile est un appareil de mesure. Nous exploitons cet aspect pour faire des mesures sur l'interface radio principalement les mesures de bande passante ou de débit utile disponible.

Le mobile effectue des mesures et envoie la valeur de la bande passante, calculée au niveau MAC, au serveur. Ce rapport de mesures est remonté au moyen du SACK *chunk* du SCTP. Le serveur en recevant la nouvelle valeur de $B_{LL}$, calcule la valeur correspondante de $B_{TL}$ et met à jour la valeur de la fenêtre de congestion ($cwnd_{computed}$) (cf. figure 5.8). Étant donné que le serveur est en phase de *Slow Start*, la nouvelle valeur de la fenêtre de congestion (*cwnd*) sera égale au minimum de la valeur donnée par le *slow start* et de celle obtenue en fonction de $B_{LL}$.

Soit donc ;

$$cwnd_{computed} = RTT \cdot B_{TL} = RTT \cdot f(B_{LL}) \quad ;$$

$cwnd_{computed}$ est la valeur de la taille de fenêtre de congestion calculée à partir de la valeur du débit utile au niveau MAC. RTT est le *Round Trip Time* au niveau SCTP.

$$cwnd = min(cwnd_{computed}, cwnd_{ss}) \quad ;$$

où $cwnd_{ss}$ est la valeur de la taille de fenêtre de congestion calculée au cours de la phase *slow start*.

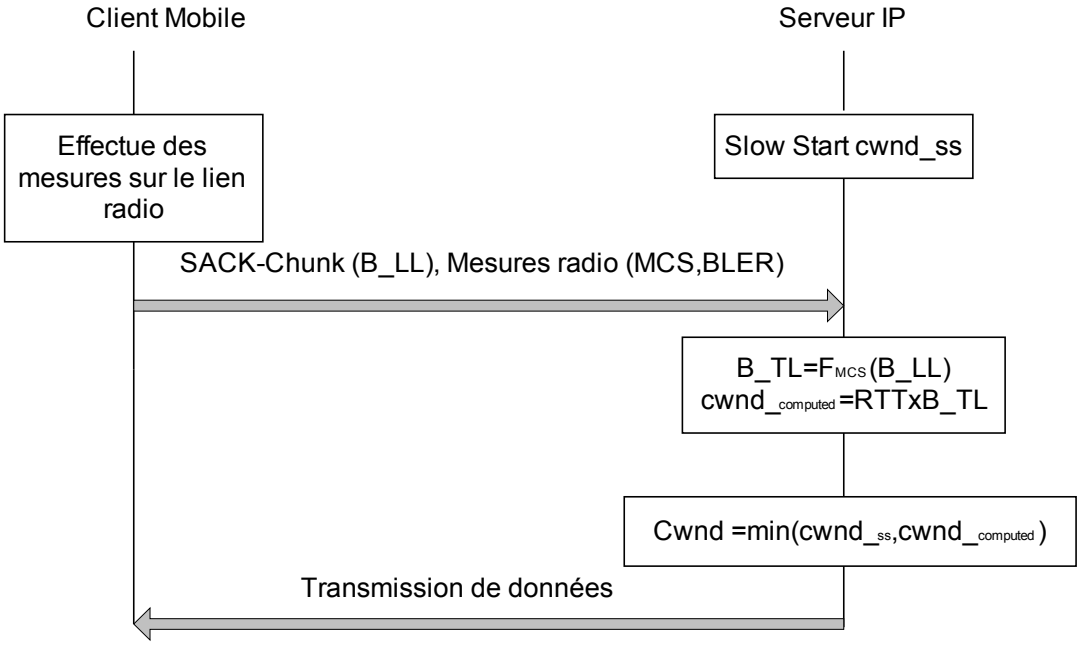



**Figure 5.8.** Mise à jour de la taille de fenêtre de congestion en fonction de la valeur du débit utile au niveau MAC.

$F_{mcs}$ est une fonction paramétrable qui dépend des conditions de transmission radio, pour cette raison nous aurons une fonction par schéma de codage.

Dans notre approche le mécanisme de contrôle de congestion se restreint à la phase de *Slow Start*. Autrement dit, au cours de son évolution, théoriquement le système n'atteint pas le régime permanent. De plus il n'y a pas de problème de congestion sauf dans des cas extrêmes d'absence de signal radio c'est-à-dire de dégradations importantes du lien. De là les exécutions usuelles de l'algorithme de contrôle de congestion se trouvent ignorées. De plus la phase de *slow start* ne consiste plus en une simple incrémentation de la fenêtre de congestion en fonction de la réception des acquittements. Mais elle est plutôt conditionnée de plus par les conditions de transmission sur le lien radio à savoir le débit utile disponible au niveau MAC. Ce qui nous permet d'avoir une "harmonie" entre le contrôle de flux au niveau MAC et le contrôle de flux au niveau SCTP.

### 5.4.3.3 Résultats des simulations

Nos simulations sont développées sur une plateforme NS2. Nous considérons, comme s'est spécifié antérieurement dans ce chapitre, une couche transport SCTP, et une couche RLC/MAC EGPRS. Le client mobile se présente comme s'est décrit par la figure 5.9.

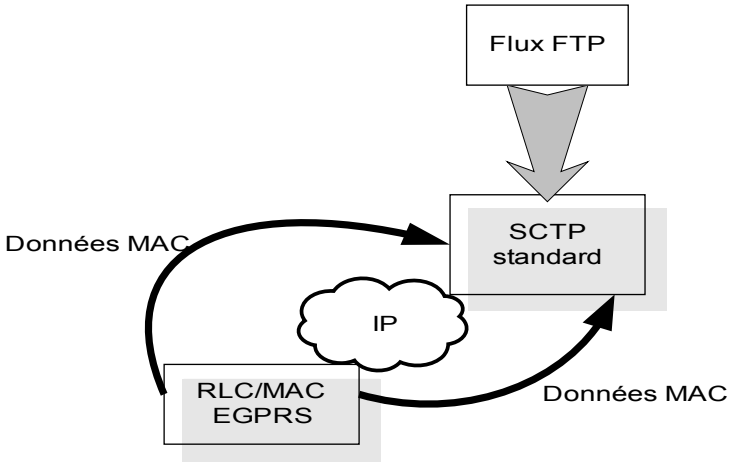



**Figure 5.9.** Architecture protocolaire du client Mobile dans un contexte *Cross_Layer.*

Les résultats obtenus représentent des statistiques relevées sur la liaison client Mobile/serveur de bout en bout, et nous ne nous intéressons qu'au lien descendant.

Nous représentons dans ce qui suit des exemples de simulations obtenues. Notre hypothèse est d'approximer l'évolution du débit utile au niveau 4 en fonction du débit utile au niveau 2 par une fonction paramétrable qui soit un polynôme linéaire ou quadratique. Les résultats dont nous tenons compte portent sur les neufs schémas de codage de la technologie EDGE. Comme chaque schéma de codage présente des conditions de transmission différentes des autres, nous avons obtenu une fonction par condition de transmission radio liée au couple {(MCSi, BLER) où $i \in \{1,2,3,4,5,6,7,8,9\}$ }. Pour les figures nous donnons un exemple de schéma de codage par

famille.

Les figures 5.10 et 5.11, représentent l'évolution de $B_{TL}$ (le débit utile au niveau SCTP) en fonction du $B_{LL}$ (le débit utile au niveau EGPRS) ainsi qu'une approximation respective quadratique et linéaire pour un exemple de schémas de codage de la famille A à savoir MCS9.

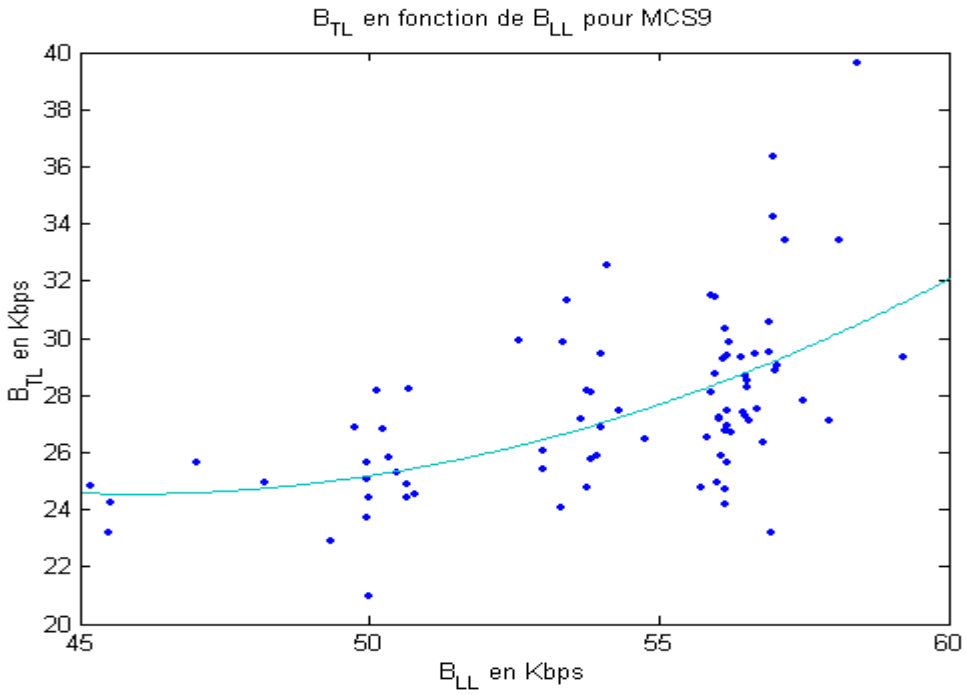



**Figure 5.10.** Variation du $B_{TL}$ en fonction du $B_{LL}$ pour MCS9 avec une approximation quadratique.

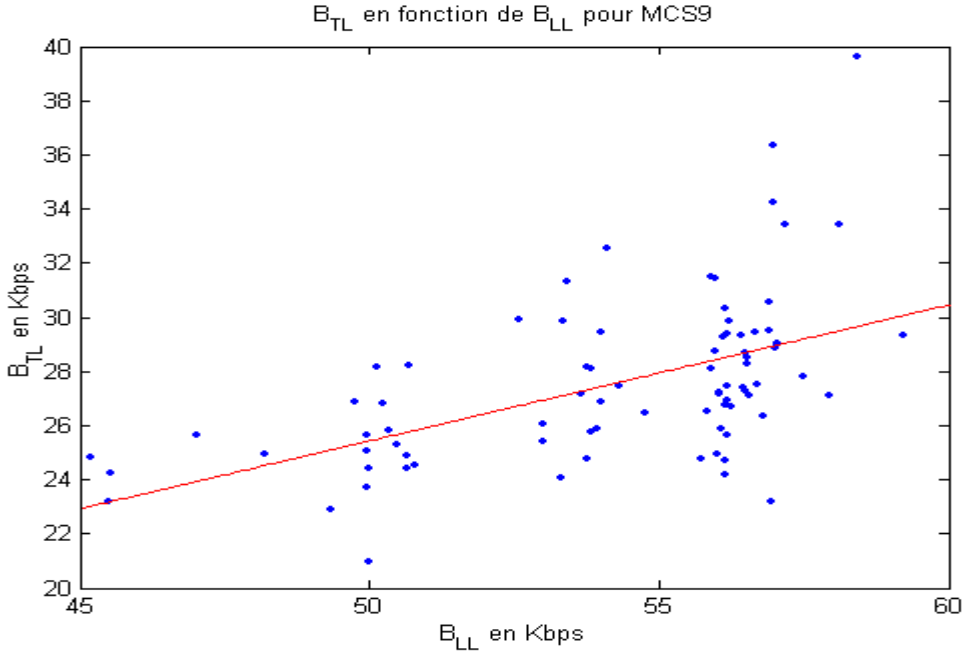



**Figure 5.11.** Variation du $B_{TL}$ en fonction du $B_{LL}$ pour MCS9 avec une approximation linéaire.

Les figures 5.12 et 5.13, représentent l'évolution de $B_{TL}$ (le débit utile au niveau SCTP) en fonction du $B_{LL}$ (le débit utile au niveau EGPRS) ainsi qu'une approximation respective quadratique et linéaire pour un exemple de schémas de codage de la famille B à savoir MCS7.

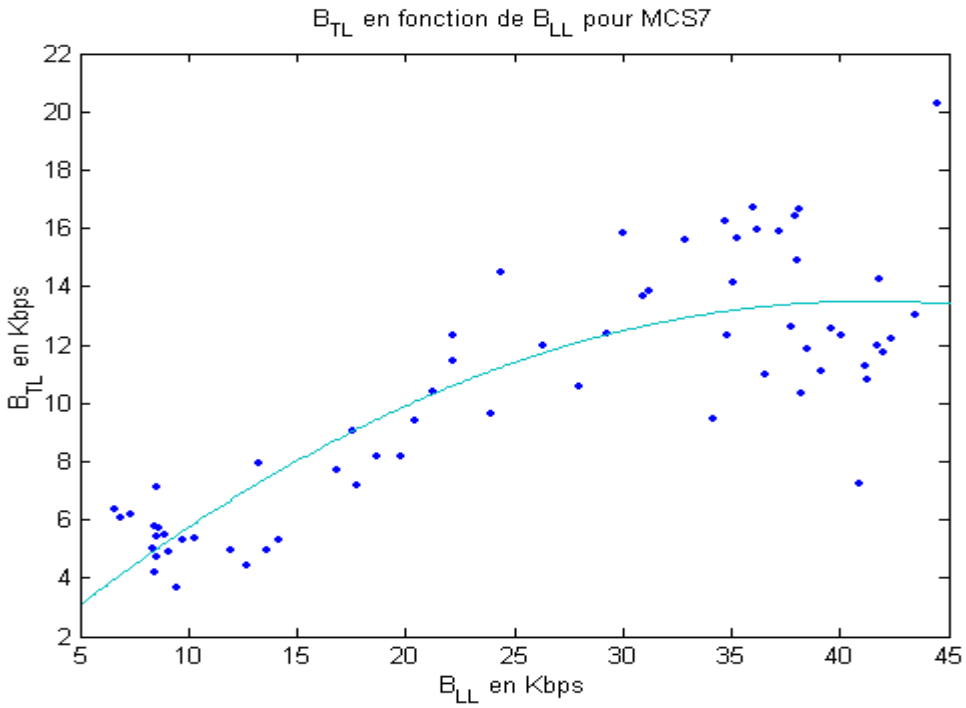



**Figure 5.12.** Variation du $B_{TL}$ en fonction du $B_{LL}$ pour MCS7 avec une approximation quadratique.

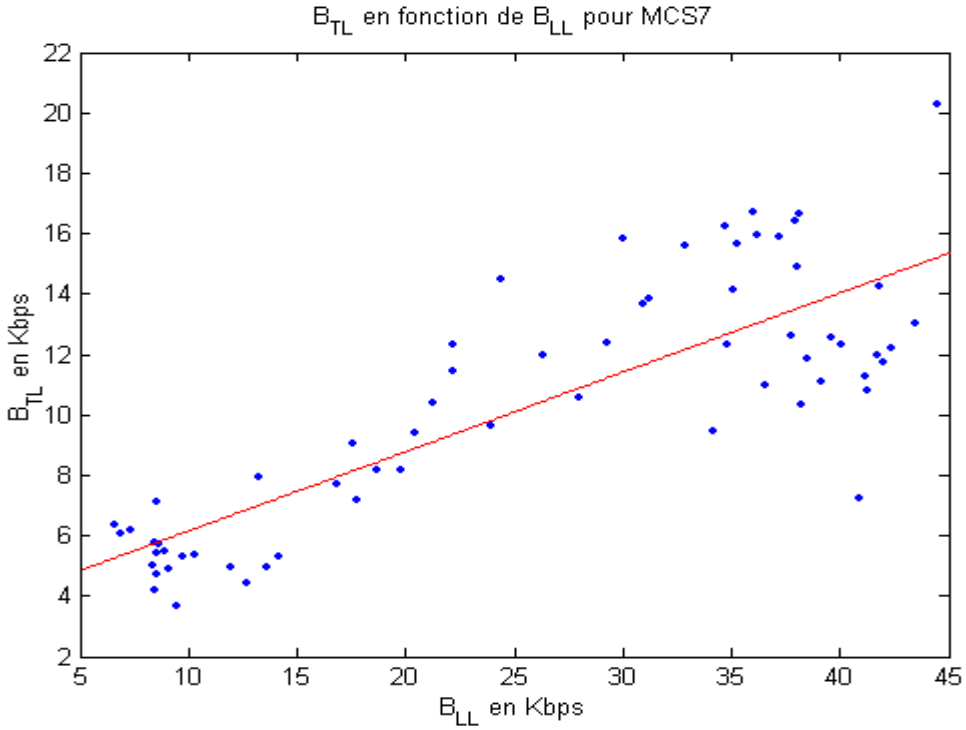



**Figure 5.13.** Variation du $B_{TL}$ en fonction du $B_{LL}$ pour MCS7 avec une approximation linéaire.

Les figures 5.14 et 5.15, représentent l'évolution de $B_{TL}$(le débit utile au niveau SCTP) en fonction du $B_{LL}$(le débit utile au niveau EGPRS) ainsi qu'une approximation respective quadratique et linéaire pour un exemple de schémas de codage de la famille C à savoir MCS1.

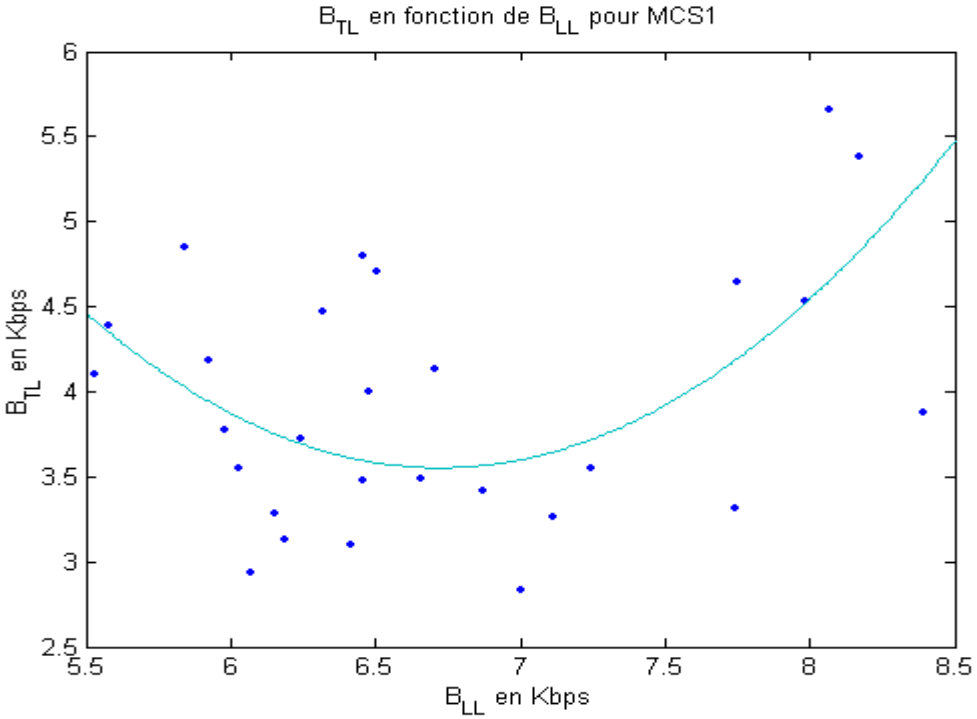



**Figure 5.14.** Variation du $B_{TL}$ en fonction du $B_{LL}$ pour MCS1 avec une approximation quadratique.

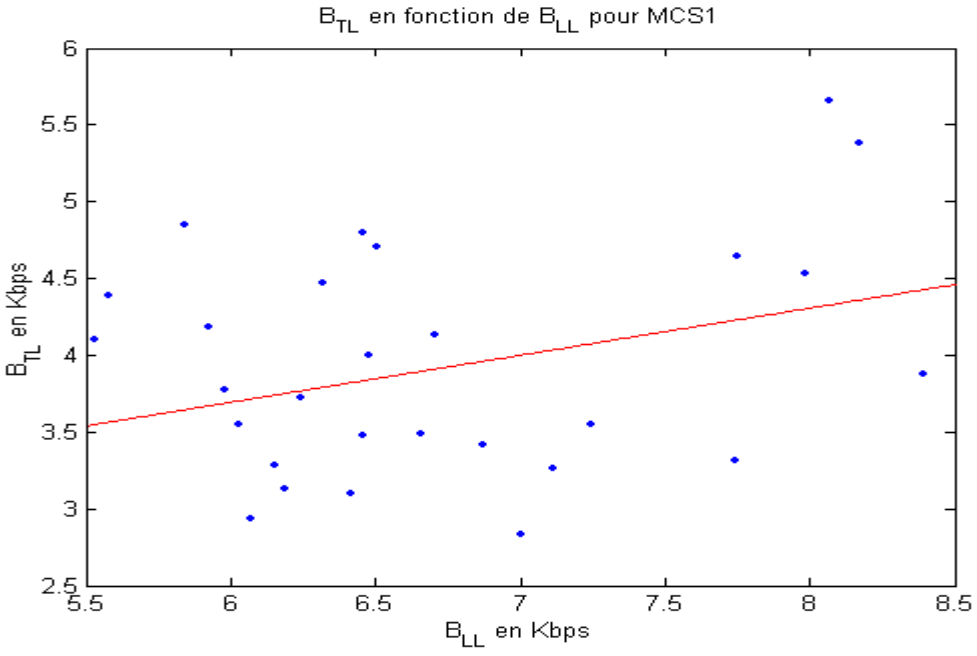



**Figure 5.15.** Variation du $B_{TL}$ en fonction du $B_{LL}$ pour MCS1 avec une approximation linéaire.

Parfois nous rencontrons sur les courbes des points plus ou moins condensés vu que nous considérons en plus des données transmises et acquittées les retransmitions qui peuvent se produire sans échec.

### 5.4.3.4 Validation des résultats obtenus

Pour pouvoir décider les formes des fonctions théoriques à en prendre en compte, par schéma de codage, nous avons eu recours à des tests statistiques de type Khi2. Notre problème est de savoir si nous pouvons approximer les variations observées de $B_{TL}$ en fonction $B_{LL}$ par les variations théoriques que nous avons obtenues. Le test Khi-deux est un outil statistique qui, d'après [KHI2], ne permet pas de répondre exactement à notre question mais permet de prendre une décision pratique. Ce test consiste à calculer la quantité :

$$\chi^2 = \sum_{j=1}^{k} \frac{(n_j - np_j)^2}{np_j}$$

où, $(n_1, n_2, \ldots, n_k)$ est la répartition observée (simulée),

et $(np_1, np_2, \ldots, np_k)$ est la répartition théorique.

La quantité $\chi^2$ à $\nu = (k-1)$ degrés de liberté, est une variable aléatoire dont la loi a été étudiée par K.Pearson et qui est tabulée.

Pour pouvoir rejeter l'hypothèse d'approximation, au risque $\alpha$ choisi, il faut que la quantité $\chi^2$ calculée soit supérieure à la valeur donnée par la table en fonction de $\alpha$ et du nombre $\nu$ de degrés de liberté. En pratique $\alpha$ est choisi aux environ de 5% [KHI2]. Autrement dit grâce à des tables nous pouvons déterminer le seuil $\chi^2_{se}$ tel que $P(\chi^2 > \chi^2_{se}) = \alpha$ .

Donc si $\chi^2 < \chi^2_{se}$ nous acceptons l'hypothèse d'approximer les variations observées de $B_{TL}$ en fonction de $B_{LL}$ par les variations théoriques données par les tableaux 5.5 et 5.6 au risque de nous tromper de $\alpha$%.

Les tableaux 5.5 et 5.6 regroupent l'ensemble des tests Khi2 que nous avons effectués sur les hypothèses d'approximation quadratique respectivement linéaire pour chaque schéma de codage.

Tableau 5.5. Approximation des variations observées de $B_{TL}$ en fonction $B_{LL}$ par une fonction théorique quadratique.

| ***Schéma de codage*** | ***Fonction théorique*** | ***Degré de liberté*** | ***Seuil de confiance à 5% ( $\chi^2_{se}$ )*** | $\chi^2$ ***Variable de K.Pearson*** | ***Décision*** $\chi^2 < \chi^2_{se}$ |
|---|---|---|---|---|---|
| **MCS1** | $B_{TL}=0.61\cdot B_{LL}^2-8.2\cdot B_{LL}+31$ | 31 | 44.9853 | 3.4553 | OUI |
| **MCS2** | $B_{TL}=0.081\cdot B_{LL}^2-1.5\cdot B_{LL}+12$ | 24 | 36.4150 | 1.1786 | OUI |
| **MCS3** | $B_{TL}=0.73\cdot B_{LL}^2-18\cdot B_{LL}+120$ | 28 | 41.3371 | 5.8378 | OUI |
| **MCS4** | $B_{TL}=-0.039\cdot B_{LL}^2+0.89\cdot B_{LL}+1.4$ | 23 | 35.1724 | 4.9755 | OUI |
| **MCS5** | $B_{TL}=-0.03\cdot B_{LL}^2+1.3\cdot B_{LL}-2.6$ | 27 | 40.1132 | 30.2260 | OUI |
| **MCS6** | $B_{TL}=0.37\cdot B_{LL}^2-0.86\cdot B_{LL}+9.3$ | 28 | 41.3371 | 6.1888 | OUI |
| **MCS7** | $B_{TL}=-0.0079\cdot B_{LL}^2+0.65\cdot B_{LL}-0.0011$ | 63 | 82.5287 | 28.4499 | OUI |
| **MCS8** | $B_{TL}=-0.0026\cdot B_{LL}^2+0.65\cdot B_{LL}-0.4$ | 34 | 48.6023 | 11.9833 | OUI |
| **MCS9** | $B_{TL}=0.038\cdot B_{LL}^2-3.5\cdot B_{LL}+100$ | 80 | 101.8794 | 122.3585 | **NON** |

Tableau 5.6. Approximation des variations observées de $B_{TL}$ en fonction $B_{LL}$ par une fonction théorique linéaire.

| ***Schéma de codage*** | ***Fonction théorique*** | ***Degré de liberté*** | ***Seuil de confiance à 5% ( $\chi^2_{se}$ )*** | $\chi^2$ ***Variable de K.Pearson*** | ***Décision*** $\chi^2 < \chi^2_{se}$ |
|---|---|---|---|---|---|
| **MCS1** | $B_{TL}=0.31\cdot B_{LL}+1.9$ | 31 | 44.9853 | 4.3737 | OUI |
| **MCS2** | $B_{TL}=0.002\cdot B_{LL}+5.5$ | 24 | 36.4150 | 0.6454 | OUI |
| **MCS3** | $B_{TL}=0.26\cdot B_{LL}+2.7$ | 28 | 41.3371 | 9.8951 | OUI |
| **MCS4** | $B_{TL}=0.09\cdot B_{LL}+4.6$ | 23 | 35.1724 | 7.2233 | OUI |
| **MCS5** | $B_{TL}=0.41\cdot B_{LL}+2.5$ | 27 | 40.1132 | 31.1021 | OUI |
| **MCS6** | $B_{TL}=0.37\cdot B_{LL}+0.65$ | 28 | 41.3371 | 15.4388 | OUI |
| **MCS7** | $B_{TL}=0.26\cdot B_{LL}+3.5$ | 63 | 82.5287 | 30.4152 | OUI |
| **MCS8** | $B_{TL}=0.47 B_{LL}+1.7$ | 34 | 48.6023 | 12.2607 | OUI |
| **MCS9** | $B_{TL}=0.5\cdot B_{LL}+0.33$ | 80 | 101.8794 | 19.3156 | OUI |

Concernant le schéma de codage MCS9 nous avons fait les calculs jusqu'à un seuil de confiance à 0.1% qui est égale, d'après la table, à 124.8392 pour un degré de liberté égale à 70, ce qui est bien

supérieur à la variable de K.Pearson calculée pour une approximation quadratique. De là les variations de $B_{TL}$ en fonction de $B_{LL}$, pour MCS9, peuvent être approximées par la fonction théorique $B_{TL} = 0.038 \cdot B_{LL}^2 - 3.5 \cdot B_{LL} + 100$ avec un risque de se tromper de 0.1%.

Nous constatons qu'avec la méthode de KHI2, que se soit l'approximation par une fonction théorique linéaire ou par une fonction théorique quadratique, les deux alternatives fournissent une bonne approximation des variations observées de $B_{TL}$ en fonction de $B_{LL}$ avec un risque de se tromper de 5% (dans la majorités des MCSs). Pour pouvoir trancher entre les deux types de fonctions nous faisons recours aux calculs des statistiques de type écart type, variance et coefficient de variation (cf tableau 5.7). En effet, pour comparer deux séries statistiques il est parfois judicieux de comparer l'écart type ($\sigma_X$) et la moyenne en faisant le quotient, nous obtenons alors l'écart type relatif appelé aussi le coefficient de variation (cv), donné par l'équation suivante :

$$cv = \frac{\sigma_X}{E[X]} = \frac{\sqrt{E[X^2] - E[X]^2}}{E[X]}$$

Tableau 5.7. Calcul des statistiques relatives au $B_{TL}$ simulée et aux approximations linéaire et quadratique.

| ***Schéma de codage*** | ***Statistiques considérées*** | ***$B_{TL}$ simulée*** | ***$B_{TL}$ approximation linéaire*** | ***$B_{TL}$ approximation quadratique*** |
|---|---|---|---|---|
| **MCS1** | **Écart type** | 0.7922 | 0.2479 | 0.4519 |
| | **Variance** | 0.6276 | 0.0615 | 0.2042 |
| | **cv** | **0.2015** | 0.0620 | **0.1182** |
| **MCS2** | **Écart type** | 0.3850 | 0.0024 | 0.0901 |
| | **Variance** | 0.1483 | $5.6727\ 10^{-6}$ | 0.0081 |
| | **cv** | **0.0699** | $4.315\ 10^{-4}$ | **0.0174** |
| **MCS3** | **Écart type** | 1.5013 | 0.4203 | 1.1397 |
| | **Variance** | 2.254 | 0.1767 | 1.2990 |
| | **cv** | **0.2552** | 0.0715 | **0.1046** |
| **MCS4** | **Écart type** | 1.3817 | 0.4018 | 0.8179 |
| | **Variance** | 1.9092 | 0.1615 | 0.6690 |
| | **cv** | **0.2428** | 0.0710 | **0.1428** |
| **MCS5** | **Écart type** | 4.281 | 2.5141 | 2.7343 |
| | **Variance** | 18.3269 | 6.32069 | 7.476396 |
| | **cv** | **0.4829** | 0.2837 | **0.2952** |
| **MCS6** | **Écart type** | 3.0496 | 2.3655 | 2.7658 |

| ***Schéma de codage*** | ***Statistiques considérées*** | ***$B_{TL}$ simulée*** | ***$B_{TL}$ approximation linéaire*** | ***$B_{TL}$ approximation quadratique*** |
|---|---|---|---|---|
| | **Variance** | 9.3003 | 5.5956 | 7.6497 |
| | **cv** | **0.4615** | 0.3585 | **0.4257** |
| **MCS7** | **Écart type** | 4.1066 | 3.3424 | 3.4372 |
| | **Variance** | 16.8644 | 11.1715 | 11.8141 |
| | **cv** | **0.3988** | 0.3278 | **0.3365** |
| **MCS8** | **Écart type** | 7.8993 | 7.2987 | 7.4049 |
| | **Variance** | 62.3989 | 53.2708 | 54.8330 |
| | **cv** | **0.3644** | 0.3395 | **0.3404** |
| **MCS9** | **Écart type** | 3.0727 | 1.6260 | 1.6625 |
| | **Variance** | 9.4417 | 2.6437 | 2.7638 |
| | **cv** | **0.1116** | 0.0593 | **0.0745** |

Nous mettons en gras les statistiques considérées dans nos conclusions. En effet, d'après les valeurs indiquées dans le tableau 5.7 relatives au coefficient de variation (cv), nous constatons qu'une approximation quadratique de variation de $B_{TL}$ en fonction de $B_{LL}$ est meilleure qu'une approximation linéaire pour les schémas de codage MCS1, MCS2, MCS3, MCS4, MCS5, MCS6, MCS7, MCS8 et MCS9.

## 5.5. Conclusion

La création d'un mécanisme *cross-layer,* liant la réactivité au niveau transport aux variations des conditions de propagation et de transmission au niveau des couches liaison de données et physique, permet d'adapter le contrôle de flux au niveau 4 à celui au niveau 2. Ceci dans le but d'éviter le déclenchement des phases de ralentissement de transmission inutiles et gênantes sur le plan de QoS offerte. Cependant, un tel mécanisme contribue à l'augmentation de la fiabilité du réseau.

Dans notre cas, en complément de la modification proposée dans le chapitre précédent, nous avons développé dans le présent chapitre une modélisation mathématique du mécanisme *cross_layer*. Cette modélisation a permis de lier les variations du débit utile au niveau SCTP à celles du débit utile au niveau RLC/MAC EDGE par des polynômes de second degré pour tous les MCS 1,2,3,4,5,6,7,8 et 9. Notre approche vise une modélisation relative à une liaison de bout en bout. Les relations proposées sont paramétrables dépendantes du schéma de codage activé et du BLER relatif. Autrement dit, les coefficients des équations varient en fonction des conditions de propagation et de transmission sur l'interface radio. Cette proposition sera accomplie par des expériences pratiques des équations proposées toutefois une plateforme SCTP appliquée à un environnement EDGE/EGPRS sera à la disposition de l'équipe.

# Chapitre 6 : Conclusions générales et perspectives

Nous avons examiné dans ce rapport plusieurs mécanismes d'interactions inter-couches et de communication adaptative pour les réseaux de type EDGE/EGPRS et pour des situations de *handover data* inter-systèmes. Les idées présentées dans ce mémoire peuvent améliorer de façon significative la performance du protocole SCTP en environnement mobile et par conséquent amélioration de la qualité de service offert.

## 6.1. Contributions

L'introduction générale de ce mémoire décrit le cadre de nos travaux de recherches ainsi qu'une présentation détaillée des différents chapitres. Le deuxième chapitre décrit les fonctions de base de SCTP. Nous avons focalisé cette présentation sur le *multihoming* et les extensions de SCTP proposées pour la gestion de *handover data*. Cette étude critique, nous a permis de mettre en évidence les insuffisances des solutions proposées dans la littérature, ce qui nous a aidé à formuler notre approche de modification de SCTP basée sur une association du *multihoming* à la mobilité. Toujours en environnement mobile, plus précisément, nous avons considéré un contexte EDGE/EGPRS pour montrer l'avantage du *multistreaming* en SCTP dans l'amélioration de la qualité de service offert. C'est dans cette optique que le troisième chapitre a été développé. Nous nous y sommes intéressés au mécanisme de *multistreaming* et le rôle qu'il joue dans le contournement des problèmes de HOL *blocking* apparus avec le TCP. Comparé, aux solutions existantes telle que l'ouverture de plusieurs connexions TCP par objet d'une session HTTP, SCTP avec *multistreaming* offre une solution optimale et plus performante. Nous avons montré au moyen des simulations développées sous une plateforme NS2, que le temps de réponse d'une session HTTP1.0 formée de pages à 3 objets en moyenne chacune, est moins élevé (variant en fonction du taux d'erreur bloc sur l'interface radio) avec SCTP *Multistreaming* qu'avec le TCP. Nous avons choisi HTTP1.0, afin d'avoir des conditions comparables, au niveau applicatif, pour les deux protocoles TCP et SCTP. Comme HTTP1.0 exige l'ouverture d'une connexion TCP par objet. Les résultats obtenus ont prouvé une meilleure performance de SCTP avec *multistreaming* (3 *streams* activés) comparée à TCP, grâce à l'évitement de HOL *blocking*, c'est que la congestion sur un *stream* ne bloque pas la transmission sur les autres *streams*.

Dans le chapitre 4, nous avons défini la procédure de *QoS_Measurement_Chunk*. C'est un nouveau *chunk* de contrôle que nous avons proposé pour remonter les mesures de critères de qualité relevées sur l'interface radio (MS) à la couche SCTP (SGSN et GGSN). En fonction de ces mesures (MCS, BLER) SCTP adapte sa fenêtre de congestion. D'où une modification du mécanisme de contrôle de congestion en fonction des conditions de transmission radio. Cette procédure, basée sur une corrélation du *multihoming* et de la mobilité, montre son apport particulièrement lors de l'exécution de *handover data* inter-systèmes. Pour valider et prouver le bon fonctionnement de notre proposition nous avons simulé différents scénarii dépendant de différents schémas de variations des conditions de transmission sur l'interface radio.

D'abord nous avons considéré une variation déterministe du schéma de codage et du BLER relatif sur le lien primaire. Ce dernier correspond à une association SCTP établie entre le terminal mobile et le serveur via une cellule serveuse EDGE/EGPRS. Nous supposons une dégradation persistante de la qualité de service afin de déclencher un *handover data* vers une autre cellule EDGE/EGPRS liée à un SGSN différent. La procédure de *QoS_Measurment_Chunk* de SCTP assure ainsi une meilleure performance que le SCTP standard avec *multihoming* aussi bien au niveau transport qu'au niveau RLC/MAC, ainsi que dans le cas d'un réseau desservant un mobile ou plus.

Ensuite, nous avons considéré le même scénario de simulation que précédemment pour un schéma de variations aléatoires des conditions de transmission sur l'interface radio (MCS, BLER). Enfin, pour mieux exploiter notre proposition nous avons considéré un *handover data* de EDGE/EGPRS vers un réseau WLAN. Nous avons analysé les performances de SCTP avec *QoS_Measurment_Chunk* aussi bien au niveau transport en nous intéressant aux :

- variations de la fenêtre de congestion permettant de vérifier les changements introduits au mécanisme de contrôle de congestion,
- variations du numéro de séquence au moyen desquelles nous visualisons les pertes de données et la quantité de données transmises.

Au niveau RLC/MAC nous avons tracé les variations du numéro de séquence des blocs RLC et les résultats, relatifs à chacun des plans d'observation, ont montré une amélioration des performances de SCTP avec *QoS_Measurment_Chunk*.

Dans le cinquième chapitre, nous nous sommes intéressés à une résolution mathématique de l'idée proposée dans le chapitre 4. L'objectif était de lier la taille de la fenêtre de congestion au niveau SCTP au débit utile sur l'interface radio. Il s'agit d'une fonction par (MCS, BLER) permettant de mettre à jour la *cwnd* selon les mesures remontées par le terminal mobile au réseau dans le cas d'un réseau EDGE/EGPRS. Au moyen de tests statistiques nous avons constaté que des approximations quadratiques sont meilleures que celles linéaires pour MCS1-2-3-4-5-6-7-8 et 9. C'est grâce à cette modélisation que le mécanisme de contrôle de congestion se restreindra uniquement à une phase de *slow start* différente de celle usuellement adoptée.

## 6.2. Perspectives

SCTP, étant conçu initialement pour le transport de la signalisation, ce protocole grâce à ses caractéristiques innovantes s'avère un bon support pour des applications multimédia. Plusieurs équipes de recherche en réseaux s'intéressent de plus en plus à ce protocole, plus particulièrement à son exploitation pour la résolution des problèmes de *handover data* inter-RAT. D'autre part, les mécanismes d'interaction inter-couches dans les réseaux sans fil font l'objet d'une activité importante de la part des chercheurs en réseaux. C'est sur ces deux principaux points que se concentrent les travaux de recherche présentés dans cette thèse. Nous décrivons à présent les directions futures possibles autour des domaines de recherche abordés dans cette thèse.

Dans un premier temps, une exploitation du mécanisme de *multistreaming* pourrait être développée plus profondément. C'est que ce mécanisme pourrait jouer un rôle important dans la gestion des priorités entre les différentes classes de service. Cette approche engendrerait un gain en terme d'efficacité d'utilisation des ressources disponibles. Une idée qui pourrait être développée dans les futurs travaux est relative à la configuration d'un *stream* par PFC (*Packet Flow Context*, c.f. Annexe 4). Ce qui permettrait d'avoir un contrôle de flux commun entre celui du niveau transport et celui de la couche RLC/MAC engendrant ainsi une optimisation d'allocation des ressources disponibles.

D'un autre côté, la modélisation inter-couches proposées dans le quatrième chapitre, relative à l'extension *QoS_Measurment_Chunk*, pourrait proposer une politique de contrôle de congestion plus compliquée. Cela pourrait consister à intégrer des modèles de canaux radio plus compliqués, pour évaluer le délai des paquets et modéliser plus finement les liens montant et descendant. En relation avec ceci, une combinaison du *multihoming* et du *multistreaming* avec priorité pourrait améliorer les procédures de gestion de ressources et de contrôle d'admission dans les réseaux sans fils. Il s'agit d'activer deux adresses en même temps, une primaire et une secondaire correspondant chacune à une interface avec un réseau d'accès. L'idée consisterait à envoyer les données volumineuses demandant une bande passante importante sur le réseau présentant plus de ressources disponibles.

Finalement, la modélisation analytique, développée dans le cinquième chapitre, pourrait avoir plus de valeur si elle serait accompagnée par des tests pratiques sur des réseaux réels. Une implémentation réelle de notre contribution sur une plateforme EDGE/EGPRS utilisant une implémentation logicielle de SCTP pourrait être envisagée. De nouvelles études sont nécessaires pour évaluer les fonctions théoriques proposées sur différentes configurations de réseaux mobiles mettant en avant l'idée de modélisation d'une connexion de bout en bout.

# Annexe 1 : Terminologie SCTP : *chunk* de données et *chunk* SACK

## 1. *Chunk* de données

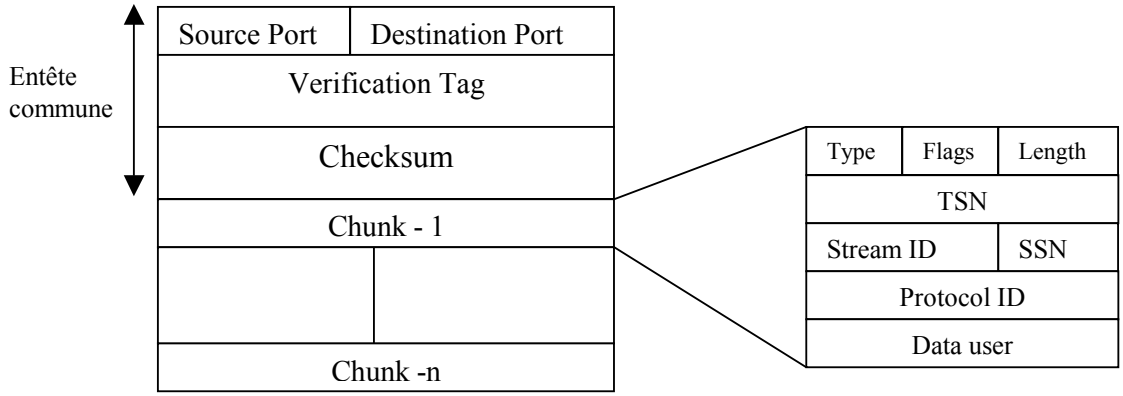



**Figure A.1.1.** Format d'un paquet SCTP.

Le tableau suivant explique la signification de chaque champ d'un paquet SCTP ainsi que les champs d'un *chunk* data.

| Champ | Représentation | Signification |
|---|---|---|
| *Source Port* | 16 bits | Numéro de port de l'émetteur |
| *Destination Port* | 16 bits | Numéro de port vers lequel le paquet est destiné |
| *Verification tag* | 32 bits | Le récepteur d'un paquet SCTP utilise la valeur du *verification tag* pour valider l'émetteur de celui ci |
| *Checksum* | 32 bits | SCTP utilise l'algorithme *Adler 32*, calculé sur le paquet SCTP entier. |
| *Type* | 8 bits | Ce champ identifie le type d'informations supportées par ce *chunk* [RFC2960]:<br>0 - Payload Data (DATA)<br>1 - Initiation (INIT)<br>2 - Initiation Acknowledgement (INIT ACK)<br>3 - Selective Acknowledgement (SACK)<br>4 - Heartbeat Request (HEARTBEAT) |

| Champ | Représentation | Signification |
|---|---|---|
| | | 5 - Heartbeat Acknowledgement (HEARTBEAT ACK)<br>6 - Abort (ABORT)<br>7 - Shutdown (SHUTDOWN)<br>8 - Shutdown Acknowledgement (SHUTDOWN ACK)<br>9 - Operation Error (ERROR)<br>10 - State Cookie (COOKIE ECHO)<br>11 - Cookie Acknowledgement (COOKIE ACK)<br>12 - Reserved for Explicit Congestion Notification Echo (ECNE)<br>13 - Reserved for Congestion Window Reduced (CWR)<br>14 - Shutdown Complete (SHUTDOWN COMPLETE)<br>15 to 62 - reserved by IETF<br>63 - IETF-defined Chunk Extensions<br>64 to 126 - reserved by IETF<br>127 - IETF-defined Chunk Extensions<br>128 to 190 - reserved by IETF<br>191 - IETF-defined Chunk Extensions<br>192 to 254 - reserved by IETF<br>255 - IETF-defined Chunk Extensions |
| *Flags* | 8 bits | 00000UBE :<br>U : *Unordered* , affecté à 1, indique qu'il ne s'agit pas de données livrées en séquence, pas de SSN attribué à ce *data chunk*.<br>B : *Begining* , prend la valeur 1, indiquant le premier fragment d'un message utilisateur.<br>E : *Ending* , prend la valeur 1, indiquant le dernier fragment d'un message utilisateur. **B E** :<br>**1 0** le premier fragment d'un message utilisateur<br>**0 0** fragment intermédiaire d'un message utilisateur<br>**0 1** le dernier fragment d'un message utilisateur<br>**1 1** Message utilisateur non fragmenté. |
| *Length* | 16 bits | Ce champ indique la taille en octets d'un *chunk* de données. Cette taille est calculée à partir du champ *type* jusqu'au *user data* sans considérer les données de bourrage (*padding data*). |
| *TSN* | 32 bits | *Transmission Sequence Number* : c'est un numéro utilisé intérieurement par SCTP. Un TSN est attaché à |

| Champ | Représentation | Signification |
|---|---|---|
| | | chaque *chunk* contenant des données utilisateur pour permettre au nœud SCTP récepteur d'accuser sa réception et détecter des livraisons dupliquées.<br>Le choix du *Initial TSN* est aléatoire et peut être utilisé, avec le *verification tag,* comme identifiant d'une association donnée.<br>L'émetteur doit envoyer des *Chunks* de données dans un paquet SCTP avec un ordre croissant de TSN. |
| *Stream ID* | 16 bits | Ce champ identifie le *stream* auquel ces données utilisateurs appartiennent. S'il n'est pas spécifié le *stream0* sera utilisé. |
| *SSN* | 16 bits | SCTP attribut un SSN (*Stream Sequence Number*) à chaque message qui lui est communiqué par l'utilisateur SCTP. De côté réception, SCTP assure que les messages sont livrés à l'utilisateur SCTP en séquence dans un *stream* donné. Cependant, si un *stream* peut être bloqué en attendant le message utilisateur en séquence suivant, la livraison d'autres *streams* peut s'exécuter.<br>Quand un message utilisateur est fragmenté par SCTP pour le transport, le même SSN doit être attribué à chacun des fragments du message.<br>En transmettant des données en séquence et hors séquence, un nœud SCTP n'incrémente pas son SSN en transmettant un *chunk* de données dont le bit U=1. |
| *Protocol ID* | 32 bits | Cette valeur représente l'identificateur de protocole indiqué par l'application (ou couche supérieure). Cet identificateur n'est pas utilisé par SCTP, mais peut être utilisé par certaines entités de réseau aussi bien que par l'application du récepteur pour identifier le type d'information supportée par le *chunk* de données.<br>La valeur 0 n'indique aucune application.<br>SCTP ne sera pas responsable de standardiser ou vérifier n'importe quels *protocol ID,* SCTP reçoit simplement l'identificateur de la couche supérieure et le transmet avec les données correspondantes. |
| *User data* | De taille variable | Il s'agit des données utiles utilisateur. L'implémentation doit compléter la fin des données jusqu'à 4 octets avec des octets nuls. N'importe quelle bourrage ne doit pas être inclus dans le champ longueur du *chunk*. |

# 2. *Chunk* SACK

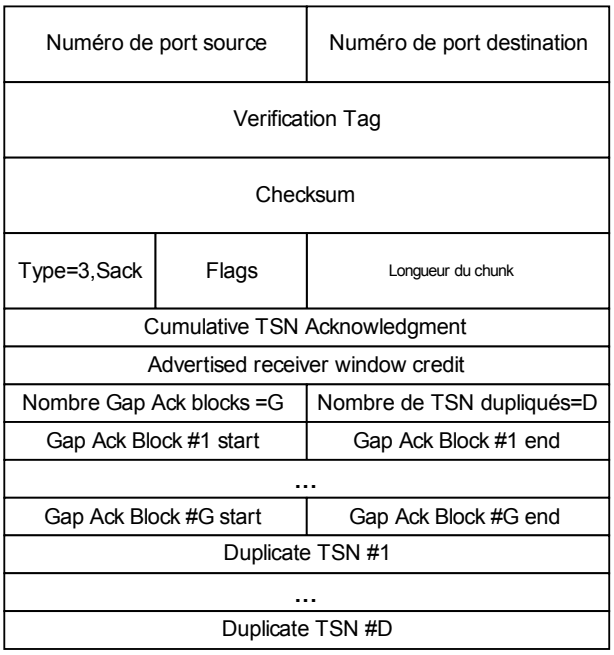



**Figure A.1.2.** Format d'un *chunk* SACK.

Le tableau ci-dessous présente la signification de chaque champ formant un *chunk* SACK.

| Champ | Représentation | Signification |
|---|---|---|
| *Cumulative TSN Acknowledgement* | 32 bits | Contient le TSN du dernier *chunk* de données reçu avant le premier *gap*. Ce paramètre acquitte tous les TSNs inférieurs ou égales à sa valeur. |
| *Advertised received Window Credit* | 32 bits | Ce paramètre indique la taille du *buffer,* en octets, de l'émetteur du SACK, mise à jour. |
| *Number of Gap Ack blocks* | 16 bits | Indique le nombre de *Gap Ack Blocks* inclus dans ce SACK. |
| *Number of duplicate TSNs* | 16 bits | Ce paramètre indique le nombre de TSNs dupliqués que le nœud (émetteur du SACK) a reçus. |
| *Gap Ack Blocks start* | 16 bits | Ce paramètre indique l'*offset* TSN de début pour ce *Gap Ack Block*. Pour calculer le TSN actuel il faut additionner le *cumulative TSN Ack* à cet *offset* TSN.<br>Ce TSN calculé indique le premier TSN reçu dans ce *Gap Ack Block.* |

| Champ | Représentation | Signification |
|---|---|---|
| | | Chaque *Gap Ack Block* acquitte des TSNs reçus en séquence après une coupure dans le séquencement de ceux ci. Par définition tous les TSNs acquittés par *Gap Ack Blocks* sont supérieurs au *Cumulative TSN Ack.* |
| *Gap Ack Blocks end* | 16 bits | Ce paramètre indique le dernier *offset* TSN pour ce *Gap Ack Block*. Pour calculer le TSN actuel il faut additionner le *cumulative TSN Ack* à cet *offset* TSN.<br>Ce TSN calculé identifie le TSN du dernier data *Chunk* reçu dans ce *Gap Ack Block* |
| *Duplicate TSN* | 32 bits | Ce paramètre indique le nombre de fois qu'un TSN a été reçu en deux exemplaires depuis que le dernier SACK a été envoyé. Chaque fois qu'un récepteur reçoit un duplicata de TSN (avant d'envoyer un SACK) il l'ajoute à la liste de duplicata. Le compteur des duplicata est re-initialisé à zéro après l'envoi de chaque SACK. |

# Annexe 2 : Validation du Simulateur SCTP sous NS2.27

## 1. Paramètres de simulations

Dans une première étape et afin de s'assurer du bon fonctionnement du protocole SCTP implémenté sous NS version 2.27, nous avons lancé des simulations en adoptant la configuration suivante :

Nous considérons deux nœuds NS simples auxquels on relie un agent SCTP, comme s'est développé par Armando L. Caro Jr Janardhan Iyengar du PEL (*Protocol Engineering Laboratory, University of Delaware)*

Le lien entre ces deux noeuds est un *duplex-link* de 0.5Mb de débit et de retard égale à 200ms

*//----------------------*

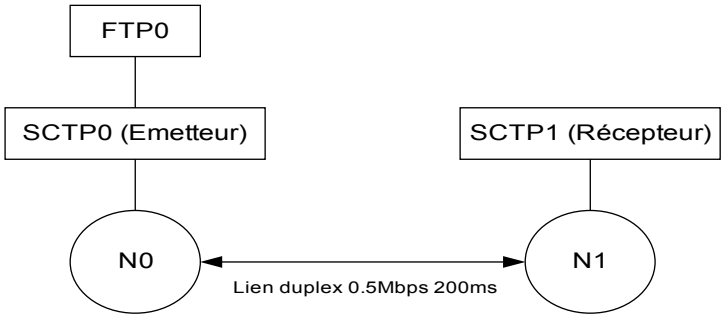



**Figure A2.1.** Architecture Simulée.

*set n0 [$ns node]*

*set n1 [$ns node]*

*$ns duplex-link $n0 $n1 .5Mb 200ms DropTail*

*//----------------------*

Le nœud émetteur est configuré de sorte que nous avons un MTU de 1500 octet, la taille de paquet SCTP est de 1468, nous supposons que cette association ne supporte qu'un seul *stream* et que la fenêtre de congestion est initialisée à 2*MTU.

*//----------------------------*

*set sctp0 [new Agent/SCTP]*

*$ns attach-agent $n0 $sctp0*

*$sctp0 set mtu_ 1500*

*$sctp0 set dataChunkSize_ 1468*

*$sctp0 set numOutStreams_ 1*

*$sctp0 set initialCwnd_ 2*

*//----------------------------------------*

Le noeud récepteur est configuré de sorte que nous avons un MTU de 1500 octet, que la taille de la fenêtre de réception est initialisée à 128 Koctet et nous supposons que nous avons un SACK pour chaque paquet entrant (c'est à dire useDelayedSacks_ 0(false) , si useDelayedSacks_ 1 nous aurons l'activation du *delayed Ack*).

*//----------------------------------------*

*set sctp1 [new Agent/SCTP]*

*$ns attach-agent $n1 $sctp1*

*$sctp1 set mtu_ 1500*

*$sctp1 set initialRwnd_ 131072*

*$sctp1 set useDelayedSacks_ $false*

*$ns connect $sctp0 $sctp1*

*$sctp1 listen*

*//----------------------------------------*

Au niveau applicatif nous considérons comme application le transfert de fichier simple FTP, dans le but d'avoir des résultats comparatifs à ceux obtenus avec TCP dans les mêmes conditions, puisque le protocole SCTP n'introduit aucun avantage par rapport à TCP dans le cas d'un flux continu de données de type FTP, par exemple.

*//-------------------------------------*

*set ftp0 [new Application/FTP]*

*$ftp0 attach-agent $sctp0*

*//-------------------------------------*

Nous avons lancé les simulations sur une durée de 100 secondes

# 2. Résultats et interprétations des simulations

Parmi les paramètres que nous jugeons indispensables pour analyser le comportement d'un protocole de transport sont les variations de la taille de fenêtre de congestion (*cwnd*) en fonction du temps, les variations du temps d'aller/retour (*RTT*) durant une simulation, les variations du numéro de séquence des paquets de données émis (TSN, Data).

**a ). RTT**

La figure A2.2 représente les variations de la valeur approximée du RTT en fonction du temps. La valeur de RTT est donnée par la formule suivante :

Soit R une mesure de RTT, sachant que nous ne disposons que des valeurs liées aux paramètres

RTO, SRTT et RTTVAR :

$R = srtt$

$R' = (srtt' - (1 - 0.125 * RTO) * srtt) / 0.125 * RTO$ ,

Où R' et srtt' sont de nouvelles mesures, respectivement, de RTT et SRTT

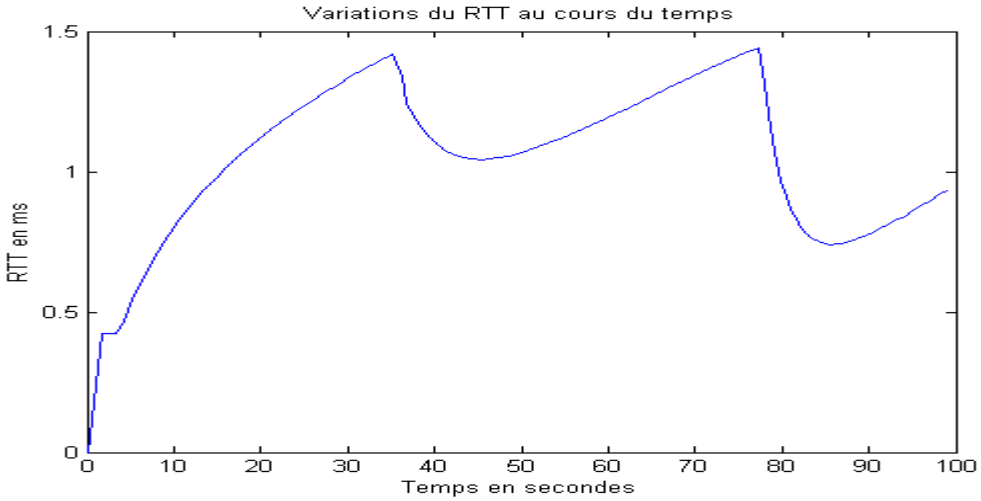



**Figure A2.2.** Variations du RTT au niveau SCTP.

**b ). CWND**

A ce niveau nous représentons les variations de la taille de fenêtre de congestion en fonction du temps, le paramètre *cwnd_* est extrait du fichier de trace SCTP fourni par l'agent SCTP développé, et exploité lors des simulations.

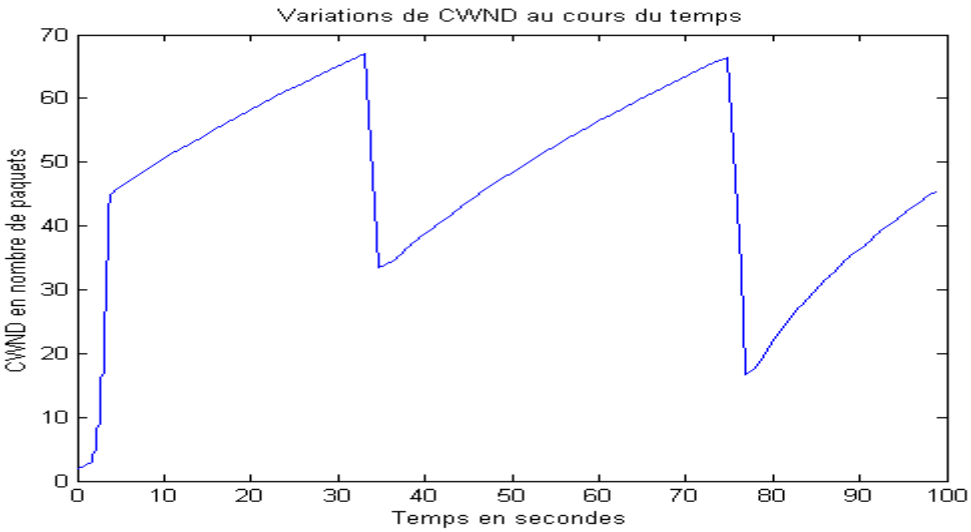



**Figure A2.3.** Variations de la taille de la fenêtre de congestion au cours du temps.

Nous remarquons une chute périodique de la fenêtre de congestion, c'est le déclenchement d'une

phase de *Congestion avoidance*. En moyenne chaque 66 messages. C'est un comportement normale vu que nous n'avons pas considéré un modèle d'erreur entre l'émetteur et le récepteur. Le réseau, après l'atteinte d'un certain seuil d'émission (*ssthreshold*), entre dans son état permanent avec le déclenchement d'une phase d'évitement de congestion.

### c ). PBA : *Partial_Byte_Acked*

Pour pouvoir expliquer la réduction de *cwnd* à sa moitié (ou encore le passage en une phase de *congestion avoidance*), nous avons pensé à représenter le *partial_bytes_acked* (pba). C'est un nouveau paramètre introduit par SCTP lors de la phase de *congestion avoidance* et qui mesure le nombre total d'octets de tous les nouveaux blocs acquittés par le SACK reçu incluant les blocs acquittés par le cumulative TSN et par les blocs *Gap-Ack*.

Nous représentons sur la même figure A2.4 le pba et le cwnd pour pouvoir identifier la coïncidence des phases de *congestion avoidance*.

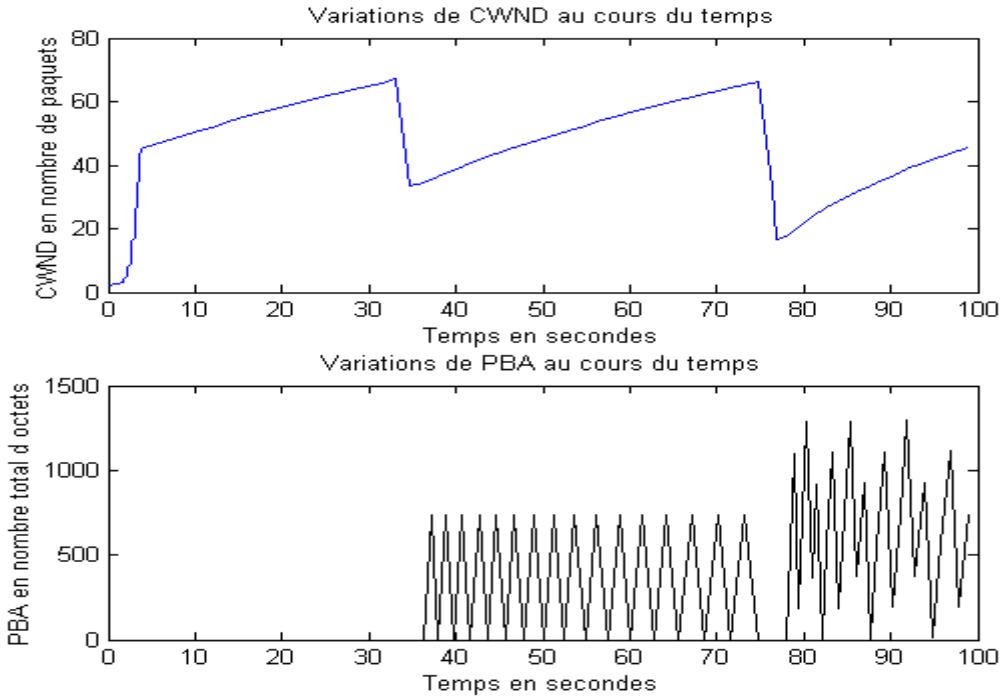



**Figure A2.4.** Variations des paramètres de contrôle de congestion (cwnd et pba).

### d ). TSN et SSN

Nous nous intéressons dans cette partie aux numéros de séquence des paquets SCTP (TSN). Nous représentons l'évolution des TSNs de tous les types confondus de paquets que se soit de données ou de contrôle, sachant que pour les SACK, il s'agit du *cumulative TSN Point* (c.f. Figure A2.5).

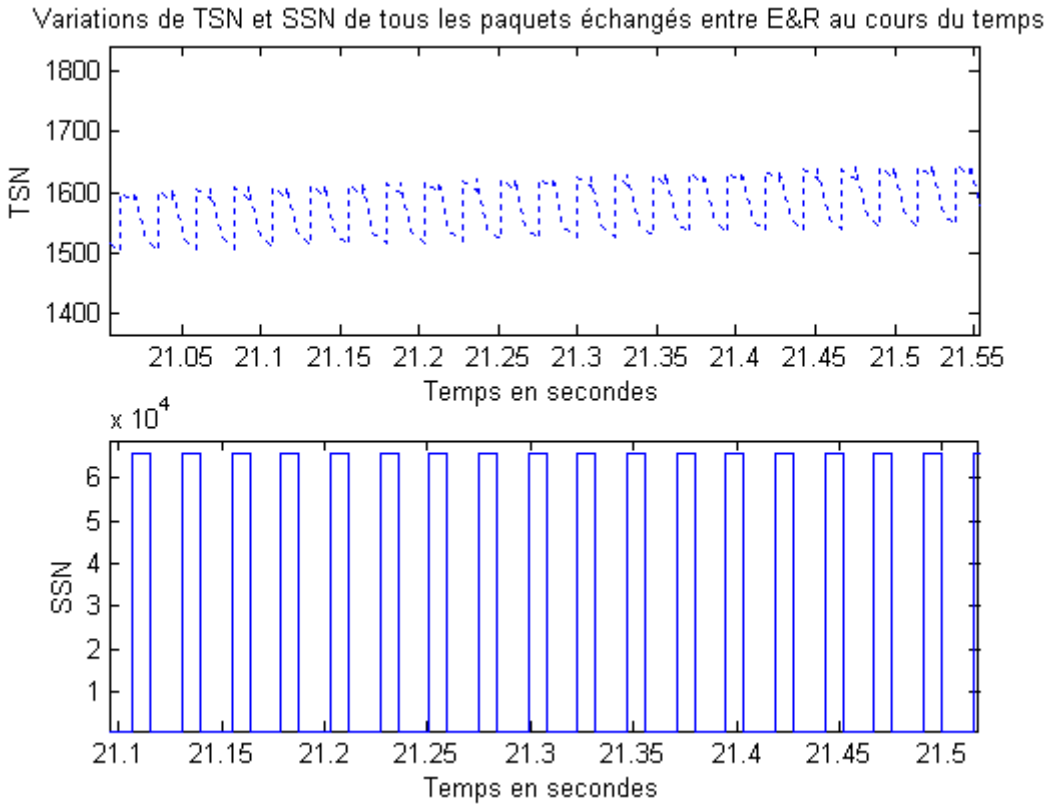



**Figure A2.5.** Variations des numéros de séquence : au niveau paquet et au niveau *stream*.

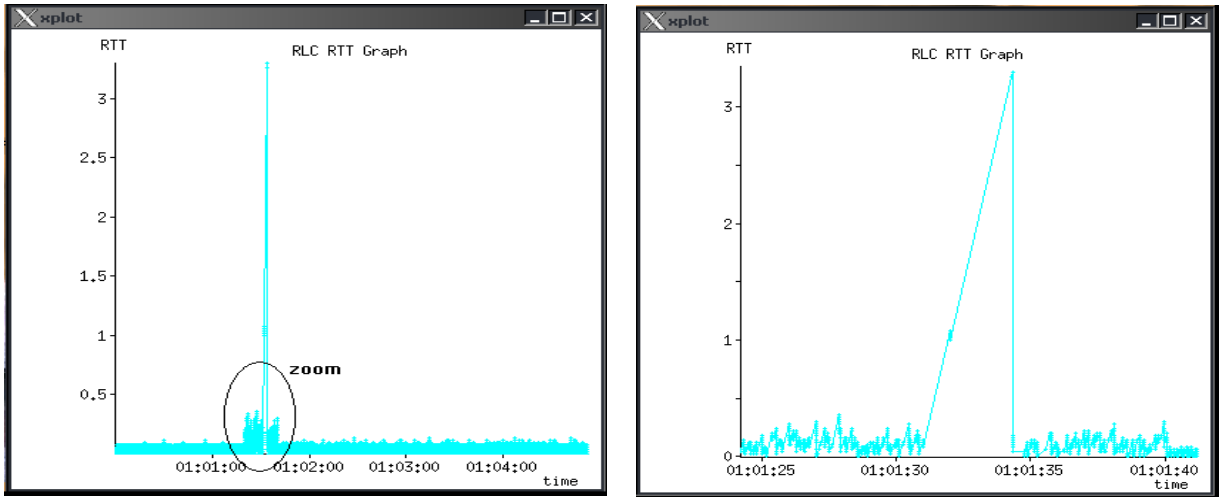
Pour les SSNs les valeurs maximales représentent celles indéfinies qui correspondent aux paquets de contrôle, ce qui est logique puisque les données de contrôle sont envoyées avec les *chunks* de données au sein du même paquet SCTP.

Les diminutions de TSN sont dues aux retransmissions des paquets non acquittés, suite à la réception d'un SACK dupliqué. Il ne s'agit pas d'une expiration du temporisateur de retransmission car si c'est le cas nous aurons une chute de la fenêtre de congestion à 1 (phase de *slow start*).

# Annexe 3 : Performances des protocoles TCP et TCP Eifel sur un lien radio en *handover*

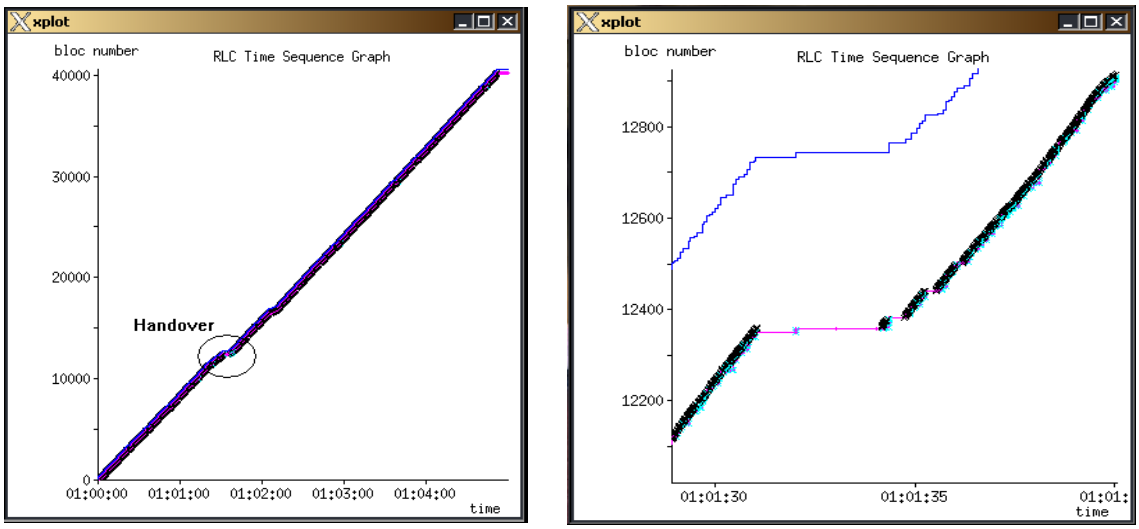
## 1. Performances de TCP

Dans ce paragraphe nous étudions le comportement de TCP standard dans un contexte de lien radio en *handover*. Lors du changement de lien radio nous avons une coupure de l'ancienne connexion TCP entre le mobile et le serveur et le rétablissement d'une autre connexion. La fermeture des TBFs entre le mobile et BSS1(*Base Station Subsystem*) correspond à la coupure de connexion TCP ce qui explique l'interruption de transmission donc des vides des numéros de séquences au niveau RLC, ce qui engendre un pic du temps de retransmission au niveau RLC/MAC (3.3s) (c.f figure A3.1). Nous avons une interruption de transmission de durée de l'ordre de 5 seconde qui apparaît au niveau de la courbe de variation du numéro de séquence des blocs RLC/MAC en fonction du temps (c.f figure A3.2).




(a) (b) : zoom (a),instant du *Handover*

**Figure A3.1.** RTT au niveau RLC/MAC sur les deux cellules.




(a) (b) : zoom(a), *Handover*

**Figure A3.2.** Exemple de variation du numéro de séquence RLC durant la communication entre le mobile et le réseau.

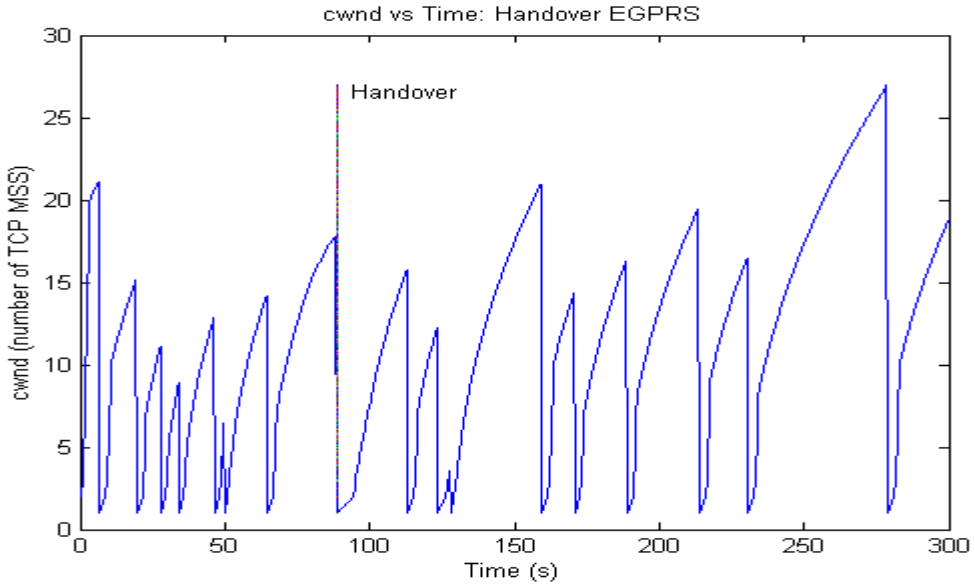



**Figure A3.3.** Évolution de la taille de fenêtre de congestion sur les deux connexions.

Le basculement de lien entre le mobile et le réseau se fait par la fermeture des TBFs ouverts avec la première cellule et l'ouverture de nouveaux TBFs avec la seconde cellule. Cette procédure est accompagnée d'une coupure de l'ancienne connexion TCP et le rétablissement d'une autre connexion ce qui engendre une ré-initialisation de la fenêtre de congestion à l'instant du rétablissement. Ceci entraîne la transmission en phase de *slow start* sur la seconde cellule (c.f figure A3.3) alors qu'en SCTP avec extension de *QoS_Measurement_Chunk* nous avons une continuité de la transmission

(pas de *slow start*)). Ce rétablissement de connexion engendre des trous dans le numéro de séquence des segments TCP (c.f figure A3.4).

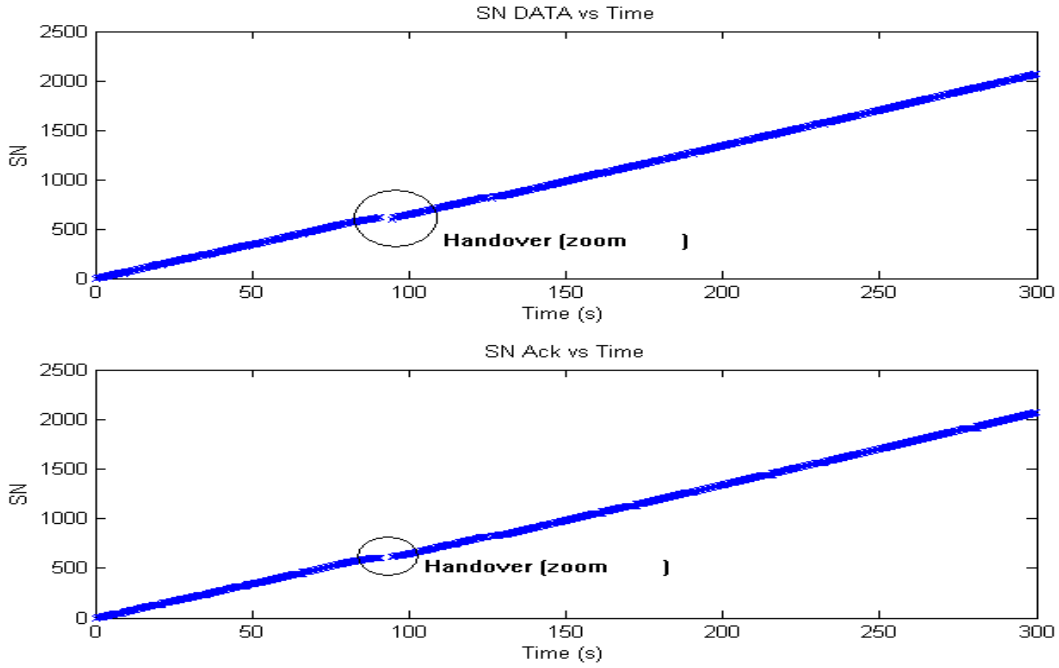



**Figure A3.4.** Exemple de variation du numéro de séquence TCP (données et acquittements relatifs).

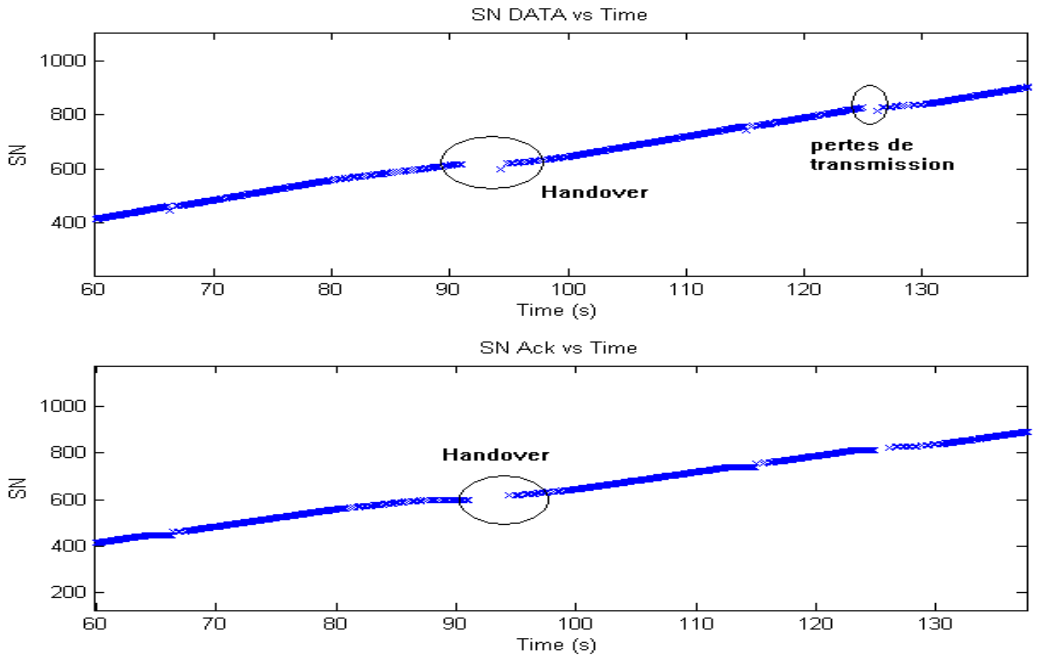


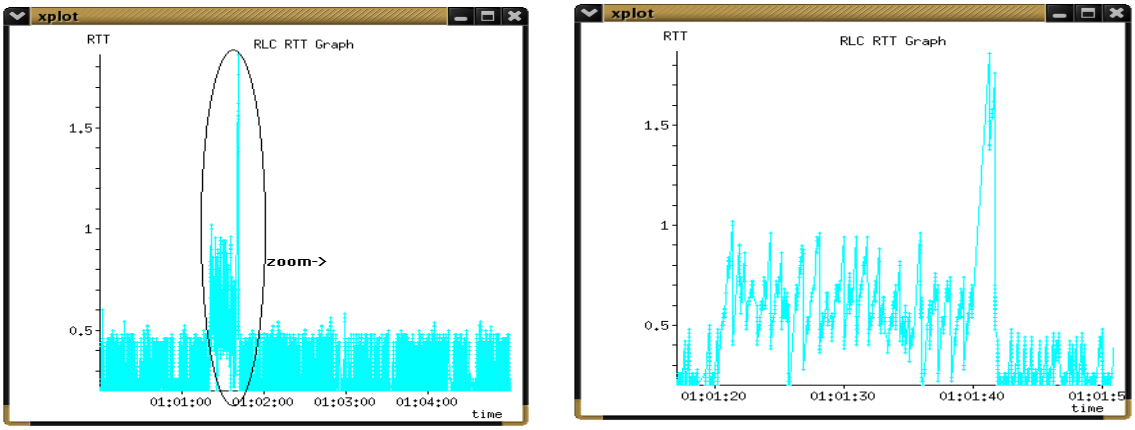

**Figure A3.5.** Exemple de variation du numéro de séquence TCP (données et acquittements relatifs) (zoom).

# 2. Performances de TCP Eifel

Eifel est une extension de TCP qui permet d'améliorer les performances de ce protocole par une suppression des retransmissions inutiles et leurs impacts sur le contrôle de flux. Le principe de cet algorithme est d'enlever l'ambiguïté de transmission en TCP en utilisant la variante *Timestamp*. Cet algorithme consiste à mémoriser l'instant (*timestamp*) de la première retransmission (*T1*) d'un segment TCP et de le comparer à la valeur du *timestamp* retournée par l'acquittement (*Tack*).

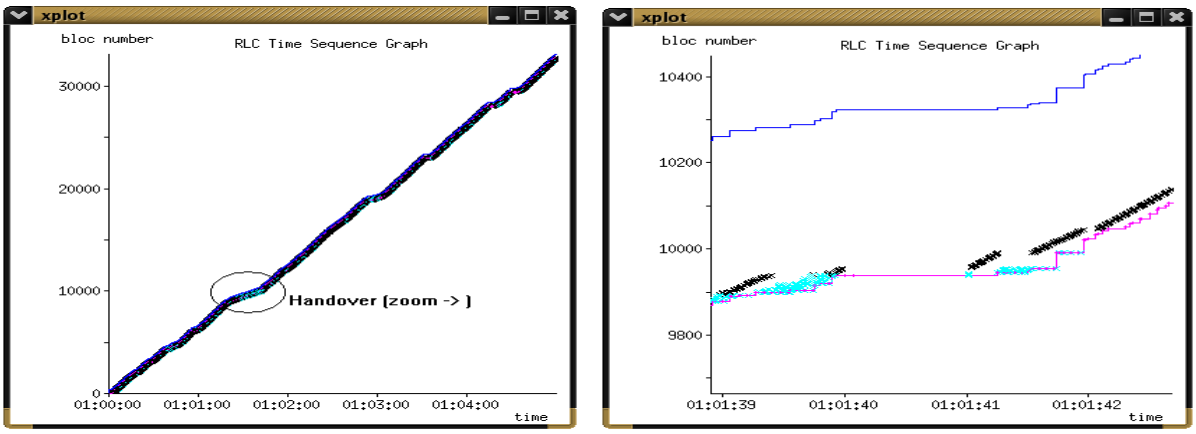
Si $Tack < T1$ , alors une retransmission supplémentaire est détectée et un algorithme de réponse est déclenché. L'algorithme de réponse consiste à ramener la fenêtre de congestion (*cwnd*) et le seuil de *slow start* (*ssthresh*) aux valeurs relatives avant la retransmission supplémentaire, et de poursuivre la transmission de données comme si aucun *timeout* ne s'est produit.

Dans ce paragraphe nous mettons en évidence l'apport de cet algorithme, dans un environnement radio, par rapport à TCP. La figure A3.6 illustre les variations du *RTT* au niveau de la couche RLC/MAC, nous notons un pic de *RTT* de 1.86sec lors de la fermeture des TBFs déclenchée par le *Handover* radio. Ceci présente une amélioration importante par rapport à TCP. De nombreuses études abordent le problème dans la littérature [LUD00].




(a) (b) : zoom(a), *Handover*

**Figure A3.6.** Exemple de variation du RTT au niveau RLC/MAC avec TCPEifel.




(a) (b) : zoom(a), *Handover*

**Figure A3.7.** Exemple de variation du SN des blocs RLC/MAC avec TCPEifel.

D'autres mesures ont été relevées au niveau de la couche transport, la figure A3.8 représente les variations de la fenêtre de congestion au cours du temps. Comme pour le TCP standard à la fermeture des TBFs et le rétablissement d'une nouvelle connexion TCP nous avons un *slow Start* qui est déclenché (*cwnd* = 1).

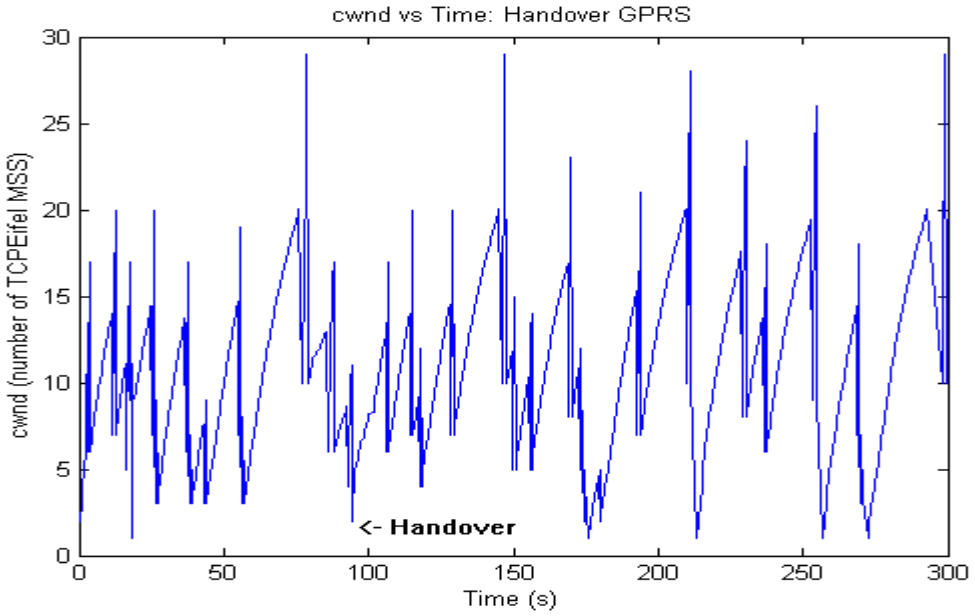



**Figure A3.8.** Évolution de la taille de fenêtre de congestion sur les deux connexions (TCPEifel).

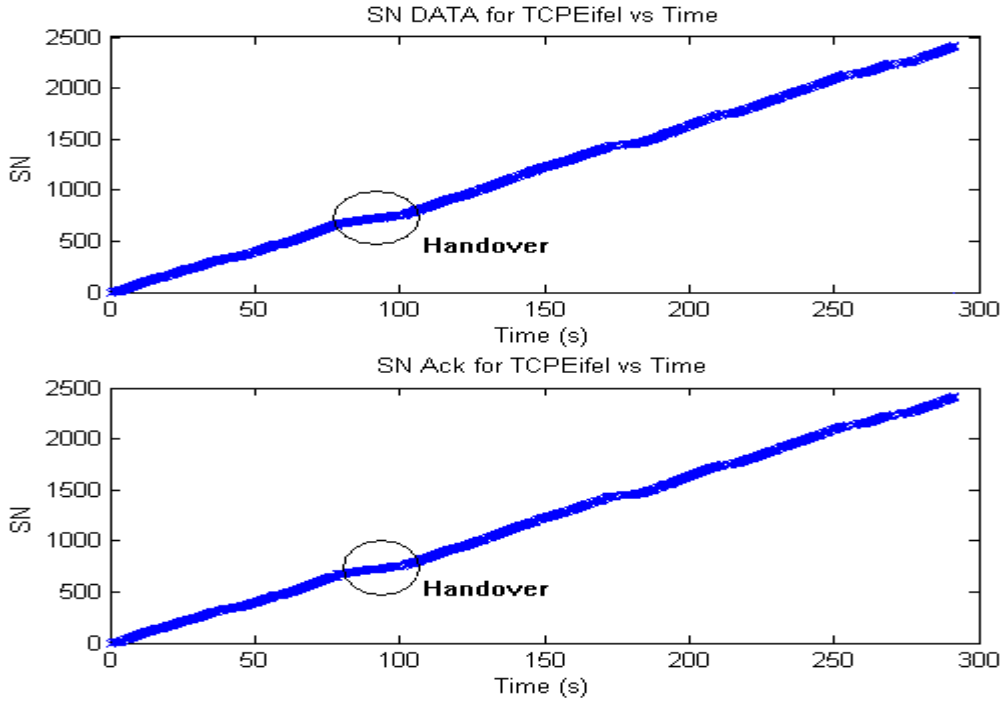



**Figure A3.9.** Exemple de variation du numéro de séquence au niveau transport (TCPEifel).

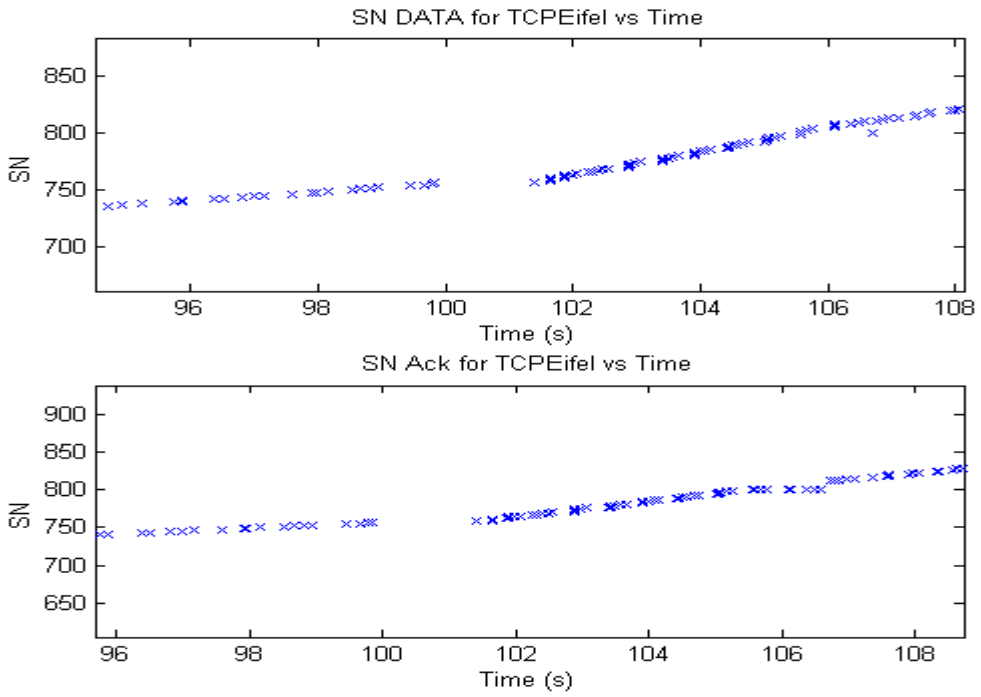



**Figure A3.10.** Exemple de variation du numéro de séquence au niveau transport (TCPEifel) (zoom sur la partie de *handover*).

# Annexe 4 : Gestion de QoS et contrôle de flux en EDGE/EGPRS

L'évolution des techniques de communications numériques offre l'augmentation des débits de transmission et l'introduction de nouveaux types de services de communications. De nouveaux standards, liés à la fois à l'amélioration des systèmes de communications cellulaires et à de nouvelles conceptions de ces systèmes comme GSM/EDGE, continuent à apparaître.

EDGE est une solution concurrente de l'UMTS dans les bandes GSM 900MHz, GSM 1800MHz et GSM 1900MHz connue aussi par EGPRS (*Enhanced General Packet Radio Service*) [HAK06]. Le but d'introduire la technologie EDGE est d'augmenter le débit de transmission de données qui peut être supporté par une porteuse de largeur 200Khz.

Selon les standards GPRS R97/98, l'exploitation simultanée de plusieurs applications avec différents critères de QoS exige l'activation de plusieurs contextes PDP ce qui engendre l'utilisation de plus qu'une adresse IP et APN (*Access Point Name*).

Dans les réseaux mobiles de nouvelles génération un mobile peut avoir de diverses applications qui prévoient différents types de services de communications. Par exemple, les services comme le courrier électronique et le téléchargement de page web peuvent avoir des critères de qualité de service pas trop critiques par comparaison aux services audio ou vidéo qui exigent des conditions de QoS importantes. Ainsi, en raison de différents types de services possibles, le support de plusieurs flux avec différents profils de QoS est indispensable. Un profil QoS est défini dans un contexte PDP. Un contexte PDP primaire peut être associé à une adresse IP du mobile. Pour supporter plusieurs flux de QoS par adresse IP un ou plusieurs contextes PDP secondaires [3GPP23.060] peuvent être créés pour un mobile. Étant donné un contexte PDP primaire activé, un mobile peut activer plusieurs contextes PDP secondaires avec différents profils de QoS selon ses besoins et la capacité du réseau.

Dans cette annexe nous présentons une étude théorique qui vise la gestion de la qualité de service dans un réseau EGPRS en nous basant sur l'activation de plusieurs contextes PDP entre un mobile et le réseau. De plus nous cherchons à travers cette étude de lier la gestion de QoS et de priorité au niveau liaison à celle au niveau transport. Pour le faire nous exploitons la procédure de contrôle de flux en EDGE dite *BSS Packet Flow Context* .

## 1. Procédure d'activation de contexte PDP secondaire

Pour pouvoir utiliser les services du réseau EGPRS, le mobile doit préalablement établir une session avec le réseau (SGSN, GGSN), c'est le contexte PDP. Cette procédure permettra de fixer

plusieurs paramètres nécessaires à l'usage du service EGPRS. Un contexte PDP est relatif à un profil de QoS bien déterminé. Si un utilisateur mobile demande différents services exigeant divers critères de QoS, alors plusieurs contextes PDPs doivent être créés entre le mobile et le réseau. Ce qui correspond à l'allocation de plusieurs adresses IP au nombre de contextes PDP actifs ainsi que plusieurs identificateurs de réseaux externes APNs. Ce qui constitue un problème de point de vue disponibilité des adresses IP par exemple. De là l'idée de supporter plusieurs profils de QoS par une unique adresse IP et un unique APN, c'est la procédure d'activation de contexte PDP secondaire. Cette procédure peut être utilisée pour activer un contexte PDP en réutilisant la même adresse IP ainsi que d'autres paramètres d'un contexte PDP déjà actif, mais pour un profil de QoS différent [3GPP23.060]. Un TI (*Transaction* Identifier) unique et un unique NSAPI (*Network layer Service Access Point Identifier*) identifient chaque contexte PDP partageant la même adresse PDP et le même APN.

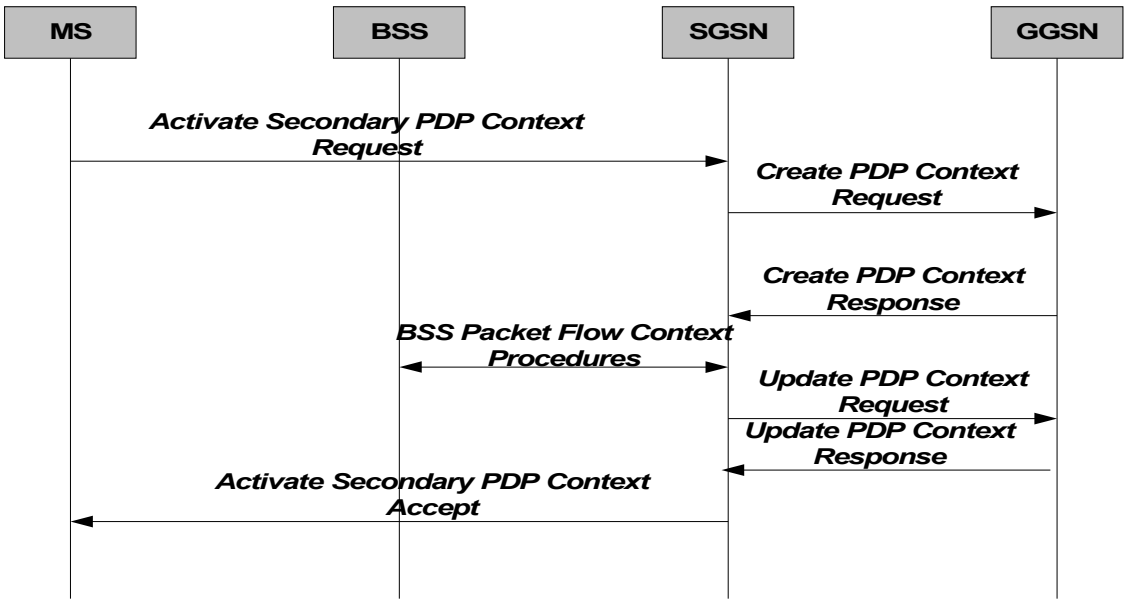



**Figure A4.1.** Création de contexte PDP secondaire [3GPP23.060].

Le mobile envoie un message *Activate Secondary PDP Context Request* contenant des informations comme : *Linked TI, NSAPI, TI, QoS Request, TFT, Protocol Configurations Options*. Où le *linked TI* indique la valeur de TI attribuée à un des contextes PDPs déjà activés pour cette adresse PDP et cet APN. Le SGSN transmet au GGSN de façon transparente le TFT[11] (T*raffic Flow Template*) afin d'activer une classification de transfert de données sur le lien descendant. Le SGSN valide la demande d'activation de contexte PDP secondaire en utilisant la valeur de TI indiquée par le paramètre *linked TI*. Le SGSN envoie au GGSN un message *create PDP Context Request* contenant des informations comme : *QoS Negotiated, TEID, NSAPI, Primary NSAPI, TFT, Protocol Configuration Options, Serving network identity,...*), où le *primary NSAPI* est la valeur du NSAPI attribuée à un des contextes PDPs déjà activés pour cette adresse PDP et cet APN. Le GGSN utilise le même réseau de transmission de données utilisé par les contextes PDPs activés pour cette adresse PDP, génère une nouvelle entrée dans sa table de contextes PDP et enregistre le TFT [3GPP23.060]. La nouvelle entrée permet au GGSN de router les PDP PDUs sur différents *tunnels* GTP entre le SGSN et le réseau de données.

11 Le TFT contrôle quel flux de trafic est configuré sur un contexte PDP secondaire.

Si le mobile et le réseau supportent les fonctionnalités *BSS Packet Flow Context*, ces procédures seront exécutées comme s'est décrit dans le paragraphe 2. Le SGSN informe le GGSN des nouveaux attribues de QoS en lui envoyant un message *Update PDP context request*. Le SGSN choisit une priorité radio et un PFI (*Packet Flow Identifier*) en se basant sur la QoS négociée et envoie au mobile un message *Activate PDP Context Accept* (*PDP Type, PDP Address, TI, QoS Negotiated, Radio Priority, Packet Flow Id, Protocol Configuration Options*). Le *QoS Negociated* doit tenir compte de l'*Aggregate BSS QoS Profile* négocié lors de la création de *packet flow context*. Pour chaque nouveau contexte PDP activé, un profile QoS et un TFT sont exigés.

# 2. Procédure BSS *Packet Flow Context* (BSS-PFC)

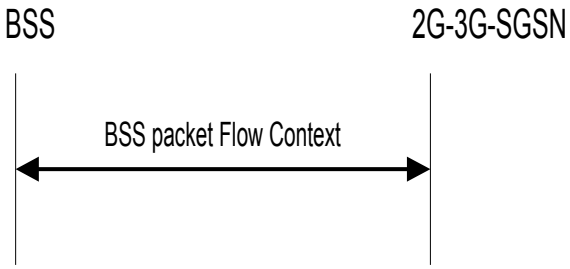



**Figure A4.2.** *BSS Packet Flow Context*.

Le SGSN fournit au BSS une information liée aux données utilisateur transmises en mode A/Gb. L'information *BSS Packet Flow Context* décrit des caractéristiques de QoS liées aux données transmises. Si les procédures *BSS Packet Flow Context* sont supportées par le réseau ceci est mentionné dans le système d'information [TS 44.060] et par le mobile ce qui est mentionné dans le *MS Network Capability* [3GPP24.008]. Tous les BSS PFC liés à un mobile sont mémorisés dans un *MS context BSS* spécifique. Le BSS peut contenir des contextes BSS pour plusieurs mobiles. Dans un contexte BSS. Les contextes BSS PFC sont identifiés par un PFI, qui est attribué par le SGSN. Un BSS PFC est partagé entre un ou plusieurs LLC SAPI's du même mobile avec des profiles de QoS identiques ou similaires.

Si le réseau supporte la procédure BSS-PFC, le MS indique une valeur PFI durant l'établissement du TBF en UL [3GPP101.349]. Un TBF est une connexion physique entre le réseau et la MS. Il assure la transmission unidirectionnelle d'un certain nombre de blocs de données sur un ou plusieurs canaux physiques. La valeur du PFI identifie le PFC initial utilisé durant le TBF [3GPP101.349]. Dans la norme[3GPP23.060] 4 flux de paquets sont prédéfinis dont les PFIs sont réservés pour les services : *Best- effort service, SMS, TOM (Tunnelling of Messages), Signalling*. Le SGSN alloue un PFI par un contexte PDP activé. Une valeur PFI non réservée a un sens pour un mobile uniquement lorsque le SGSN attribue cette valeur de PFI au BSS relativement au mobile en question.

### a ). Création de PFC

En recevant une demande de transmission sur le lien montant ou descendant des LLC-PDUs pour lesquels aucun BSS-PFC n'est activé dans le BSS, ce dernier demande au SGSN le téléchargement de BSS-PFC (c.f figure A4.3).

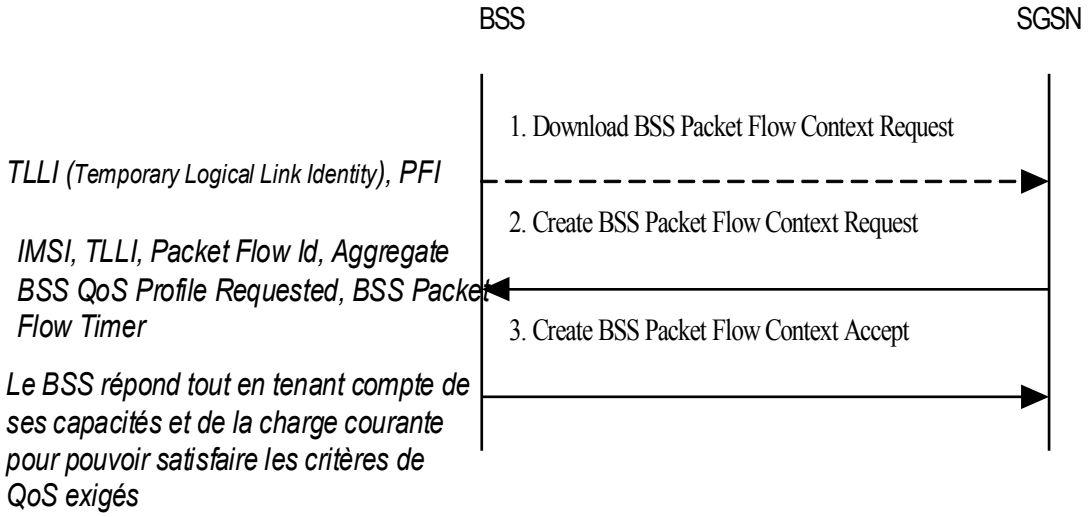



**Figure A4.3.** Procédure de création de BSS-PFC

Si une demande de création d'un BSS-PFC reçue par le BSS pour un MS en phase de *handover* de données alors le BSS ignore la demande et exécute les procédures décrites dans [3GPP48.018]. Ensuite, le SGSN transmet au BSS associé les informations comme (*IMSI, TLLI, Packet Flow Id, Aggregate BSS QoS Profile Requested, BSS Packet Flow Timer*) contenues dans le message *create BSS Packet Flow Context Request.* Le SGSN déduit l'*Aggregate BSS QoS Profile Request*[12] à partir du profile de QoS négocié lors d'activation de contextes PDP qui partagent un même flux de paquet.

Le BSS, selon les ressources qu'il dispose (eg. la capacité du *buffer*) et la charge courante [3GPP48.018], peut restreindre les exigences en QoS demandées. En fait, si le BSS a reçu l'élément d'information *Allocation/Retention Priority* lors de la demande de création de BSS-PFC, il l'utilise pour déclencher une politique d'ordonnancement liée à la création de PFC ou à l'interruption d'autres PFCs. Une fois le BSS crée un BSS-PFC, il insère les paramètres correspondants dans un contexte BSS et répond le SGSN par un message contenant les attributs suivants (*IMSI, Packet Flow Id, Aggregate BSS QoS Profile negociated*).

Le SGSN peut demander une modification du contenu d'un BSS-PFC suite à une modification, activation ou une désactivation d'un contexte PDP. Dans ce cas la procédure de création de BSS-PFC est utilisée mais le BSS ne crée pas de nouveaux BSS-PFCs, plutôt il met à jour les paramètres

12 ***Aggregate BSS QoS profile*** : Le profile de BSS QoS combiné sur tous les contextes PDPs qui partagent le même *packet flow*. Ce paramètre définit la QoS que le BSS doit assurer pour un *packet flow* donné entre le BSS et le SGSN (exp sur les interfaces Um et Gb). Ce paramètre est négocié entre le SGSN et le BSS.

des BSS-PFCs existants [3GPP23.060]. À son tour le BSS peut demander des modifications du contenu d'un BSS-PFC suite à un changement de la disponibilité des ressources du BSS.

### b ). Transfert de données dans une situation d'activation de plusieurs contextes PDP

Lorsque plusieurs contextes PDPs existent pour la même adresse PDP d'un mobile, le GGSN route les N-PDUs dans le sens descendant vers différents Tunnels GTP à la base du TFT attribué au contexte PDP [3GPP23.060]. Le mobile est responsable de la création ou de la modification de contextes PDPs et de leurs attributs de QoS. Le MS doit définir des TFTs de sorte que les PDP PDUs transmis sur le lien descendant sont routés sur un PDP qui garantit la meilleure QoS demandée par le récepteur de ce PDU. Pour chaque PDP PDU sur le lien montant, le MS doit choisir le contexte PDP qui engendre le meilleur profil de QoS demandé par l'émetteur de ce PDP PDU. La figure A4.4 illustre la procédure de transfert de données en présence de plusieurs contextes PDPs activés.

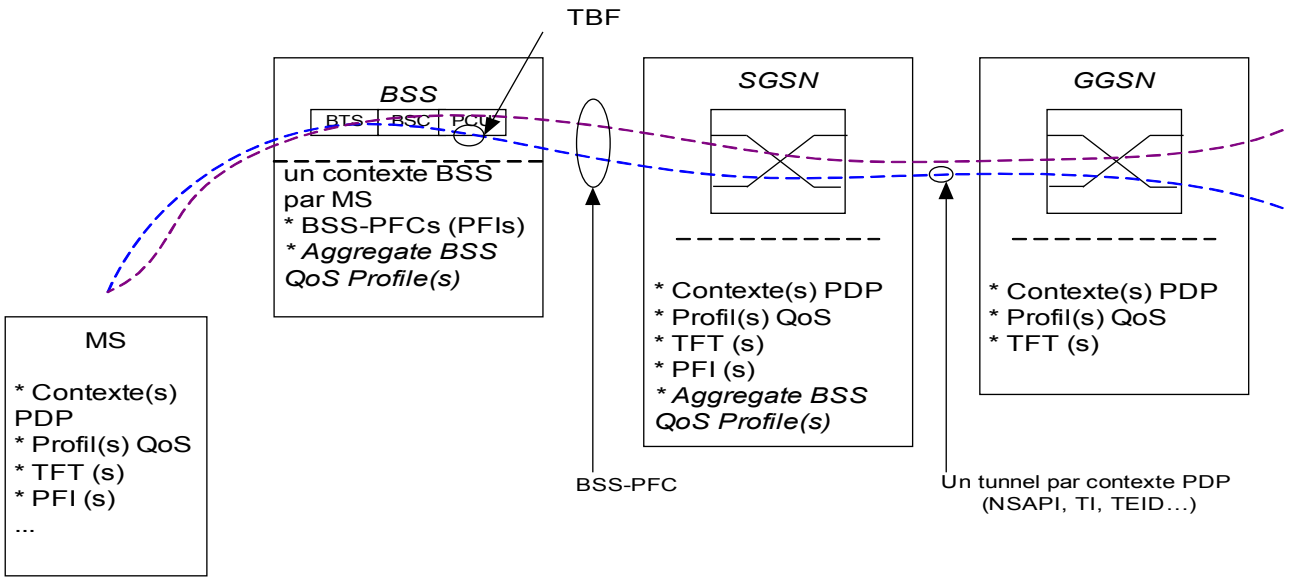



**Figure A4.4.** Transfert de données en EDGE/EGPRS.

Lors d'établissement d'une communication, plusieurs flux de données avec différentes exigences de QoS peuvent être activés. Un flux correspond à un échange de données entre deux points terminaux. Chaque flux peut avoir un profil de QoS spécifique. Lors d'une communication, un profil QoS relatif à un mobile est spécifié dans un contexte PDP par adresse IP. Par conséquent, afin de supporter plusieurs profil QoS par adresse IP, un contexte PDP primaire et un ou plusieurs contextes PDP secondaires peuvent être créés relativement à chaque profil de QoS. Quand une communication est établie, un contexte PDP primaire est activé, qui contient le profil de QoS par défaut. Si différents profils de QoS sont exigés pour différents flux (par exemple, E-mail, audio ou vidéo conférence...) au cours d'une même communication, alors des contextes PDP secondaires peuvent être activés avec différents profils de QoS.

Si un utilisateur mobile télécharge une page web. À l'établissement de la communication le

terminal ouvre un contexte PDP primaire (une adresse IP et un APN) pour un trafic de classe interactive, supposant qu'on a de plus un contexte BSS avec un PFI 1 relatif au *Best Effort.* L'utilisateur ensuite active un lien qui renvoie vers une séquence vidéo. On a alors un nouveau flux de trafic de type *Streaming*. Et le terminal active un contexte PDP secondaire pour transmettre les paquets constituants la séquence vidéo sollicitée, avec le même PFI (1). De là on a la même adresse IP et le même APN sont utilisés pour deux types de trafic, à savoir un trafic interactif et un trafic de type *streaming*. Afin d'améliorer la QoS à la réception, nous pouvons proposer d'attribuer un ordre de priorité aux paquets de données transmis. En effet, cet ordre de priorité est fixé au niveau couche transport qui permet de favoriser les données demandant moins de ressources et exigeant un temps de réponse faible. Ces paquets sont transmis sur les contextes PDP classés selon leurs ordres de création et par PFC selon le profil de QoS exigé.

# Liste des publications

1. Mériem Afif, Philippe Martins, Sami Tabbane, Philippe Godlewski, “SCTP-Cross Layer Mechanism Performance For EDGE Data Handover in Variable Radio Transmission Conditions”, ICT2007, Malaysia, May 2007.
2. Mériem Afif, Philippe Martins, Sami Tabbane, Philippe Godlewski, “SCTP Extension for EGPRS/WLAN Handover Data”, P2MNET PERF06, Tampa Florida, November 2006.
3. Mériem Afif, Philippe Martins, Sami Tabbane, Philippe Godlewski, , “Radio aware SCTP extension for handover data in EGPRS”, PIMRC 2006, Helsinki Finland, September 2006.
4. Mériem Afif, Philippe Martins, Sami Tabbane, Philippe Godlewski, “A SCTP – Layer 2 Cross Layer Mechanism for Data Handover in Wireless Networks (application to EGPRS)”, ICT 2006, Portugal May 2006.
5. Mériem Afif, Philippe Martins, Sami Tabbane, “ Performance Evaluation of SCTP over E-GPRS Air Interface For HTTP Traffic”, ICECS 2005, Gammarth Tunisia, December 2005.

# Bibliographie

[3GPP48.018] : 3GPP TS 48.018, «General Packet Radio Service (GPRS), Base Station System (BSS) - Serving GPRS Support Node (SGSN), BSS GPRS Protocol (BSSGP)», V6.13.0 Release 6, May 2006.

[3GPP24.008] : 3GPP TS 24.008, «Digital cellular telecommunications system (Phase 2+), Universal Mobile Telecommunications System (UMTS), Mobile radio interface Layer 3 specification, Core network protocols, Stage 3», version 4.16.0 Release 4, December 2005.
[3GPP23.060] : 3GPP TS 23.060, «Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS) ; General Packet Radio Service (GPRS), Service description : Stage 2», version 6.11.0, Release 6, December 2005.
[3GPP44.060] : 3GPP TS 44.060, «Digital cellular telecommunications system (Phase 2+), General Packet Radio Service (GPRS), Mobile Station (MS) - Base Station System (BSS) interface, Radio Link Control / Medium Access Control (RLC/MAC), protocol», version 6.15.0 Release 6, November 2005.
[3GPP101.349] : 3GPP TS 04.60, «Digital cellular telecommunications system (Phase 2+), General Packet Radio Service (GPRS), Mobile Station (MS) - Base Station System (BSS) interface, Radio Link Control/ Medium Access Control (RLC/MAC) protocol», version 8.27.0 Release 1999, September 2005.
[3GPP0460] : 3GPP TS 04.60, «Digital cellular telecommunications system (Phase 2+);General Packet Radio Service (GPRS); Mobile Station (MS) - Base Station System (BSS) interface; Radio Link Control/Medium Access Control (RLC/MAC) protocol», version 8.10.0 (30-07-2001), Release99.

[AAH03] : C Aahlund, A Zaslavsky, «Multihoming with Mobile IP», IEEE Proceedings of the 6th International Conference on High Speed Networks and Multimedia, July 2003.
[AAH03a] : C Aahlund, R Braennstroem, A Zaslavsky, «A Multihoming approach to Mobile IP», Proceedings of the first Swedish National Computer Networking Workshop (SNCNW), September 2003.
[AND04] : André-Luc Beylot, Riadh Dhaou and Mahamadou Issoufou Tiado, «Time, Space and Level Cyclic Dependance of Highly Complex and Dynamic Networks (Application to Dynamic Cross-layer of Mobile Systems)», Contribution to D.JRA.5.5.1a and D.JRA.5.5.1b, May 2004.
[ARM03] : Armando L. Caro Jr. «SCTP patch of ns-2 simulator», NS-2 SCTP module release 3.3, Protocol Engineering Laboratory (PEL), 2003.
[ARM02] : Armando L.Caro Jr., Janardhan R.Iyengar, Paul D.Amer, Gerard J.Heinz, Randall A.Stewart, «Modeling SCTP Latency with Multihoming and Failovers», U.S.Army Research Laboratory, Communications and Networks Consortium, 2002.
[AYD03] : I Aydin, C Shen, «Cellular SCTP: A Transport-Layer Approach to Internet Mobility», IEEE proceedings the 12th International Conference on Computer Communications and Networks, ICCCN 2003, October 2003.

[BEM04] : P. Beming, M. Cramby, N. Johansson, J. Lundsjo, G. Malmgren, J. Sachs, «Beyond 3G Radio Access Network Reference Architecture», 59th IEEE Vehicular Technology Conference VTC 2004-Spring, vol.4, pp.2047-2051, May 2004.
[BRU97] : Bruce A. Mah, "An Empirical Model of HTTP Network Traffic," *infocom*, p. 592, INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution, 1997.

[CHA04] : M Chang, M Lee, S Koh, «A Transport Layer Mobility Support Mechanism», Networking Technologies for Broadband and Mobile Networks International Conference ICOIN 2004, February 2004.
[CIS03] : Cisco Systems. «*S*CTP: A detailed overview of the protocol and a examination of the socket API», Cours Cisco, Mars 2003.

[DAV04] : David Soldani, Nilmini Lokuge and Antti Kuurne, «Service Performance Monitoring for EGPRS Networks Based on treatment classes», 12th IEEE International Workshop on Quality of Service (IWQOS), June 2004.
[DIA03] : T. Diab, P. Martins, P. Godlewski et N. Puech. «The Eifel TCP extension over GPRS RLC : Effects of long radio blackouts», IEEE Vehicular Technology Conference, VTC Fall 2003, October 2003.
[DZM05] : Dzmitry Kliazovich and Fabrizio Granelli, «Cross-Layer Congestion Control in Ad-Hoc Wireless Networks», Ad Hoc Networks Journal, 2005.

[EMM03] : Emmanuel Seurre, Patrick Savelli, Pierre-Jean Pietri, «EDGE for mobile internet», Boston, MA : Artech House, 2003 Artech house mobile communications series.
[ERI02] : Erik Lundsten, «Improving 3G performance for the mobile Internet», Master of Science Thesis, Wireless@KTH Seminar in cooperation with Telia Research, November 2002.

[FHP01] : Ulrich Fiedler, Polly Huang, and Bernhard Plattner, «Http Traffic generator», NS-Script, December 2001, www.tik.ee.ethz.ch/~fiedler/provisioning.html.
[FRA03] : François Buntschu, Rudolf Scheurer and Antoine Delley, «SCTP (*Stream Control Transmission Protocol*) Une alternative à TCP et UDP ?», EPFL, FI- 9-18 Novembre 2003.

[GER04] : Gerard J.Heinz II and Paul D.Amer, «Priorities in Stream Transmission Control Protocol (SCTP) Multistreaming», sci2004-heinz-SCTP-Prioritized-Multistreaming, March 2004.
[GER03] : Gerard J.Heinz II, «Priorities in Stream Transmission Control Protocol (SCTP) Multistreaming», Protocol Engineering Laboratory, University of Delaware, Thesis report, Spring 2003.
[GUN04] : N.Gundu, «Mobility vs Multihoming», Seminar on Internetworking, Telecommunications Software and Multimedia Laboratory, Helsinki University of Technology, TML-C15, Espoo, spring 2004.

[HAK06] : Hakan Axelsson, Peter Björken, Peter de Brain, Stefan Eriksson and Häkan Persson, «GSM/EDGE continued evolution», Ericsson Review, no.1, 2006.

[INW03] : Inwhee Joe and Latha Kant, «SCTP with an improved cookie mechanism for wireless networks through modeling and simulation», 58th IEEE VTC Fall, vol.4, pp.2559-2563, October

2003.
[IVA02] : Ivan Arias Rodriguez, «Stream Control Transmission Protocol The design of a new reliable transport protocol for IP networks», Helsinki University of Technology, Electrical and Communications Engineering Department Networking Laboratory, Thesis report, Espoo 12 Februray 2002.
[IYE04] : J.Iyengar, K.Shah, P.Amer, R.Stewart, «Concurrent Multipath Transfer Using SCTP Multihoming», SPECTS 2004.
[JAV 02] : Javier Romero and Juan Melero, «GSM, GPRS and EDGE performance : evolution towards 3G/UMTS», John Wiley & sons, 2002.
[JUN04] : JW Jung, YK Kim, HK Kahng, «SCTP Mobility Highly Coupled with Mobile IP», Telecommunications and Networking - ICT 2004 : 11th International Conference on Telecommunications, August 2004.

[KER03] : Kerstin B.Johnsson and Donald C.Cox, «An Adaptive Cross-layer Scheduler for Improved QoS Support of Mixed Data Traffic on Wireless Data Systems», 58th IEEE Vehicular Technology Conference, VTC-Fall 2003, vol.3, pp.1618- 1622 , October 2003.
[KEV03] : Kevin Fall, Kannan Varadhan, «The *ns* Manual (formerly *ns* Notes and Documentation», VINT Project, December 2003.
[KHI2] : "Test du Khi-deux", home page. Http://perso.orange.fr/alain.pichereau/chi2test.html.

[LEH03] : W. Lehr and Lee W.McKnight, «Wireless Internet access : 3G vs. Wifi?», Telecommunications Policy, Elsevier Science, pp.351-370, 2003.
[LIL06] : Liljana Gavrilovska, «Cross-Layering Approches in Wireless Ad Hoc Networks», Wireless Personal Communications, vol.37, pp.271-290, 2006.
[LUD00] : R. Ludwig, R. H. Katz, «The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions», ACM Computer Communications Review Journal, vol. 30, no. 1, January 2000.

[LUO03] : J Luo, E Mohyeldin, N Motte, M Dillinger, «Performance investigations of ARMH in a Reconfigurable Environment», SCOUT Workshop, Paris 2003.

[MAG04] : P Magnusson, J Lundsjo, J Sachs, P Wallentin, «Radio Resource Management Distribution in a Beyond 3G Multi-Radio Access Architecture», Global Telecommunications Conference, GLOBECOM'04, vol.6, pp.3472- 3477, November/December 2004.
[MAR04] : Marco Conti, Gaia Maselli, Giovanni Turi and Sylvia Giordano, «Cross-Layering in Mobile Ad Hoc Network Design», IEEE Computer Society, pp.48-51, February 2004.
[MIH02] : A Mihailovic, T Suihko, M West, «Aspects of Multi-Homing in IP Access Networks», IST Mobile Communications Summit, 2002.

[PAH00] : Pahlavan, K et al, «Handoff in hybrid Mobile Data Networks», IEEE Personal Communications, vol.7, no.2, pp.34-47, 2000.
[PAN99] : Q. Pang et. Al, «Service Scheduling for general packet radio service classes», IEEE Wireless Communications and Networking Conference (WCNC), vol.3, pp. 1229-1233, September 1999.
[PED04] : A Peddemors, H Zandbelt, M Bargh, «A Mechanism for Host Mobility Management supporting Application Awareness», Proceedings of the 2nd international conference on Mobile Systems, applications, and services (MobiSys'04), pp. 231 – 244, June 2004.

[RAI04] : V.T. Raisinghani and S.Iyer, «Cross-Layer Design Optimizations in Wireless Protocol Stacks», Computer Communications 27 (2004), Elsevier Publishing, 2004.
[RAJ02] : R Rajamani, S Kumar, N Gupta, «SCTP versus TCP : Comparing the Performance of Transport Protocols for Web Traffic», report, University of Wisconsin-Madison, May 2002.
[RAN03] : R. Stewart, L. Ong, I. Arias-Rodriguez, K. Poon, P. Conrad, A. Caro, M. Tuexen. «Stream Control Transmission Protocol (SCTP) Implementers Guide», draft-ietftsvwg-sctpimpguide-08.txt, Internet-Draft, February 2003 (Work in Progress).
[RAN03a] : R. Stewart, M. Ramalho, Q. Xie, M. Tuexen, P. Conrad. «SCTP Partial Reliability Extension», draft-stewart-tsvwg-prsctp-04.txt, Internet Draft, May 2003 (Work inProgress).
[RAN01] : Randall Stewart, Chris Metz, «SCTP New Transport Protocol for TCP/IP», IEEE Internet Computing, vol.5, no.6, pp.64-69, November-December 2001.
[RFC3522] : R.Ludwig and M.Meyer, «The Eifel Detection Algorithm for TCP», RFC3522, Internet Engineering Task Force, April 2003.
[RFC3481] : H. Inamura, G. Montenegro, R. Ludwig, A. Gurtov, F. Khafizov, «TCP over Second (2.5G) and Third (3G) Generation Wireless Networks», RFC3481, Internet Engineering Task Force, February 2003.
[RFC3309] : J. Stone, R. Stewart, D. Otis, «Stream Control Transmission Protocol (SCTP) Checksum Change», RFC3309,Internet Engineering Task Force, September 2002.
[RFC3344] : C. Perkins, « IP Mobility Support for Ipv4», RFC3344, Internet Engineering Task Force, August 2002.
[RFC3286] : L. Ong, J. Yoakum, «An Introduction to the Stream Control Transmission Protocol (SCTP)»,RFC3286,Internet Engineering Task Force, May 2002.
[RFC3150] : S.Dawkins, G.Montenegro, M .Kojo, V.Magret, «End-to-end Performance Implications of Slow Links», RFC3150, Internet Engineering Task Force, July 2001.
[RFC2960] : R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, V. Paxson, «Stream Control Transmission Protocol», RFC2960, Internet Engineering Task Force, October 2000.
[RFC2581] : M. Allman, V. Paxson, W. Stevens, «TCP Congestion Control»,RFC2581, Internet Engineering Task Force, April 1999.
[RFC2522] : P. Karn and W. Simpson, «Session-Key Management Protocol», RFC2522, Internet Engineering Task Force, March 1999.
[RFC1954] : P. Newman, W. L. Edwards, R. Hinden, E. Hoffman, F. Ching Liaw, T. Lyon, G. Minshall, «Transmission of Flow Labelled IPv4 on ATM Data Links Ipsilon Version 1.0», RFC1954, Internet Engineering Task Force, May 1996.
[RFC1191] : J. Mogul, S. Deering, «Path MTU Discovery», RFC1191, Internet Engineering Task Force, November 1990.
[RIE04] : M. Riegel, M. Tuexen, «Mobile SCTP», draft-riegel-tuexen-mobile-sctp-04.txt, Internet-Draft , October 2004 (Work in Progress).
[RQX02] : R. Stewart, Qiaobing Xie, «Stream Control Transmission Protocol (SCTP) : a reference guide», Addison wesley, London 2002.

[SAC04] : J. Sachs, H. Wiemann, P. Magnusson, P. Wallentin, J.Lundsjo, «A Generic Link Layer in a Beyond 3G Multi-Radio Access Architecture», International Conference on Communications, Circuits and Systems, ICCCAS 2004, vol.1, pp. 447- 451, June 2004.
[SAC04a] : J Sachs, L Munoz, R Aguero, J Choque, G.Koudouridis, R.Karimi, L.Jorguseski, J.Gebert, F.Meago, F.Berggren, «Future Wireless Communication Based on Multi-Radio Access»,Wireless World Research Forum (WWRF) 2004.

[SAC03] : J Sachs, E Res, EED GmbH, G Herzogenrath, «A Generic Link Layer for Future Generation Wireless Networking», IEEE International Conference on Communications (ICC'03), vol.2, pp 834-838. May 2003.
[SAM02] : Sami Tabbane, «Ingénierie des réseaux cellulaires», Paris Hermès Science, 2002 Réseaux et télécommunications.
[SAN03] : Sanjay Shakkottai, Theodore S.Rappaport and Peter C.Karlsson, «Cross-Layer Design for Wireless Networks», IEEE Communications Magazine, October 2003.
[SEO04] : Seok Joo Koh, Moon Jeong Chang and Meejeong Lee, «mSCTP for Soft Handover in Transport Layer», IEEE Communication Letters, vol.8, no. 3, March 2004.
[SHA05] : Shaojian Fu, Mohammed Atiquzzaman and William Ivancic, «Evaluation of SCTP for Space Networks», IEEE Wireless Communications, vol.12, no 5, October 2005, pp. 54-62.
[SHA04] : Shaojian Fu and Mohammed Atiquzzaman, «SCTP: state of the art in research, products, and technical challenges», IEEE Communications Magazine, vol.42, no. 4, April 2004, pp.64-76.
[STE05] : R. Stewart, M. Ramalho, Q. Xie, M. Tuexen, P. Conrad, «Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration», draft-ietf-tsvwg-addip-sctp-11.doc, Internet-Draft, February 2005 (Work in Progress).
[STU03] : P. Stuckmann, «The GSM Evolution, Mobile packet data services», John Wiley&Sons, LTD 2003.
[SUK03] : Sukwoo Kang, Matthew Fields, «Experimental Study of the SCTP compared to TCP», Computer Communications and networking Fall 2003.

[THI04] : Thierry E.Klein, Kin K.Leung and Haitao Zheng, «Enhanced Scheduling Algorithms for Improved TCP Performance in Wireless IP Networks», IEEE Globecom'04, vol.5, pp.2744- 2748, November/December 2004.

[XIN05] : XinWang and Koushik Kar, «Cross-Layer Rate Control for End-to-End Proportional Fairness in Wireless Networks with Random Access», Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc'05), May 2005.
[XIN02] : W Xing, H Karl, A Wolisz, H Mueller, «M-SCTP: Design and prototypical implementation of an end-to-end mobility concept», Proceedings of 5th Intl. Workshop The Internet Challenge: Technology and Applications, October 2002.

[YLI03] : J Ylitalo, T Jokikyyny, T Kauppinen, AJ Tuominen, J Laine, «Dynamic Network Interface Selection in Multihomed Mobile Hosts», IEEE, Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03), January 2003.
[YUE06] : Yue Wang, «A Tutorial of 802.11 Implementation in ns-2», MobiTec Lab, CUHK, January 2006, http://www.winlab.rutgers.edu/~zhibinwu/pdf/tr_ns802_11.pdf.