ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE AVEC MÉMOIRE EN GÉNIE
CONCENTRATION RÉSEAUX DE TÉLÉCOMMUNICATION
M. Sc. A.

PAR
Moussa GONDA

RÉPARTITION EFFICACE DES SOUS-FLUX TCP DANS LES ÉQUIPEMENTS MULTIHÔTES DANS LES RÉSEAUX SANS FIL HÉTÉROGÈNES

MONTRÉAL, LE 27 AVRIL 2016

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. Zbigniew DZIONG, directeur de mémoire
Département de génie Électrique à l'École de technologie supérieure

M. Georges KADDOUM, président de du jury
Département de génie Électrique à l'École de technologie supérieure

M. Michel KADOCH, membre du jury
Département de génie Électrique à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 20 AVRIL 2016

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Il me plait de remercier toutes les personnes qui ont d'une manière ou d'une autre contribué à l'accomplissement du présent mémoire.

Mes sincères remerciements vont à l'endroit du Professeur DZIONG Zbigniew pour m'avoir accordé sa confiance pour la réalisation de ce mémoire, ainsi que ses conseils et son encadrement tout au long de ce mémoire.

Mes remerciements s'adressent également aux différents membres du jury pour leurs commentaires et remarques sur mon travail.

Je remercie également FIRMIN Mah pour l'aide qu'il m'a apportée du début à la fin de ce mémoire.

Toute ma reconnaissance à ma famille en particulier à mes parents sans qui je ne serais pas là où je suis actuellement et, ma femme née DJIBJI MOUSSA Nana Hadiza qui a pu supporter mon absence près d'elle durant mes études.

RÉPARTITION EFFICACE DES SOUS-FLUX TCP DANS LES ÉQUIPEMENTS MULTIHÔTES DANS LES RÉSEAUX SANS FIL HÉTÉROGÈNES

Moussa GONDA

RÉSUMÉ

Les équipements terminaux d'aujourd'hui disposent non seulement plusieurs interfaces radios mais aussi ont la capacité d'en utiliser parallèlement. Avec l'évolution de nombreuses technologies sans fil et des terminaux mobiles à interfaces réseau multiples, les usagers ont non seulement accès aux services n'importe où et n'importe quand, de n'importe quel réseau, mais aussi envisage l'utilisation simultanée des différents accès réseaux à l'aide des plusieurs interfaces. Le Multipath TCP qui est une extension de TCP a été développé pour permettre à des terminaux d'établir une communication capable d'exploiter les ressources de plusieurs chemins à travers des sous-sessions TCP, afin notamment d'optimiser l'usage des terminaux multi-interfaces. Cependant, le fonctionnement de ce protocole nécessite une méthode de répartition des flux des données qui reste à déterminer quand bien même il est déjà standardisé.

Le présent mémoire a pour objectif de déterminer une méthode de répartition des données côté émetteur afin de minimiser le problème inhérent à ce protocole qui est celui des paquets hors ordre au niveau de récepteur et qui affecte ses performances en termes de délai et débit. La méthode que nous proposons concerne le cas où deux sous-sessions TCP asymétriques en termes de débit et temps d'aller-retour sont utilisées. Cette méthode est basée sur le principe qu'un paquet généré par l'application ne doit être assigné qu'à la sous-session TCP via laquelle il sera reçu en ordre ou au cas échéant sera le plus tôt en ordre après réception.

Mots clés : multipath TCP, sous-sessions TCP, multi-interfaces, méthode de répartition

RÉPARTITION EFFICACE DES SOUS-FLUX TCP DANS LES ÉQUIPEMENTS MULTIHÔTES DANS LES RÉSEAUX SANS FIL HÉTÉROGÈNES

Moussa GONDA

ABSTRACT

Today's end hosts not only have multiple interfaces but also have capability of their simultaneous use. Under evolution of many wireless technologies and mobile terminals with multiple network interfaces, users not only have access to services anywhere, anytime and from any network, but also aim the simultaneous use of different access networks by using these multiple interfaces. Multipath TCP which is an extension to TCP, allowing the use of multiple paths between end hosts. It has been developed to allow terminals to be able to exploit the resources of multiple paths through several subflows with the objective to optimize the use of the end hosts capability. However, the operation of this protocol requires a special scheduling algorithm which is not yet determined by the standard.

This project aims to develop a scheduling algorithm to minimize the problem of out-of-order packets inherent to this protocol at receiver. The scheduling algorithm that we propose concern the use of two asymmetric subflows in terms of delay and throughput. Our scheduling algorithm is based on the principle that a packet generated by the application should only be assigned to the subflow on which it will be received in order or will be received with the minimum delay in out-of-order state.

Keywords: multipath TCP, subflow, multiple interfaces, scheduling algorithm

TABLE DES MATIERES

Page

INTRODUCTION 1

CHAPITRE 1 REVUE DE LITTERATURES 5
1.1 Introduction 5
1.2 TCP (Transport Control Protocol) 6
1.2.1 Fonctionnalites de Tcp 7
1.2.2 Format d'un segment TCP 7
1.3 Fonctionnement du MPTCP 9
1.3.1 Les objectifs fonctionnels du MPTCP 10
1.3.2 Architecture de MPTCP 11
1.3.3 Initialisation de la connexion MPTCP 12
1.3.4 JOINDRE UNE NOUVELLE SOUS-SESSION 13
1.3.5 Structures des sequences de donnees (data sequences mapping) 15
1.3.6 Acquittement des données 16
1.3.7 Fin de connexion 17
1.4 Présentation des multipath schedulers 17
1.4.1 Schedulers a buffers individuels 18
1.4.2 Schedulers a buffer partage 18
1.4.3 Schedulers hybrides 19
1.4.4 Les paramètres caractéristiques des schedulers 19
1.4.4.1 Scheduler indépendant des caractéristiques des sous-sessions 20
1.4.4.2 Scheduler utilisant les informations d'état de l'émetteur 20
1.4.4.3 Scheduler se basant sur la capacité des sous-sessions 20
1.4.4.4 Scheduler se basant à la fois sur la capacité et le délai 20
1.5 Recherches antérieures relatives aux multi-path schedulers 21

CHAPITRE 2 PROBLEMATIQUE ET OBJECTIVE 23
2.1 Introduction 23
2.2 La problématique 23
2.2.1 Head-of-line blocking 23
2.2.2 Scheduler 26
2.3 Objectif 26
2.4 Hypotheses 26

CHAPITRE 3 PRESENTATION ET FONCTIONNEMENT DU SCHEDULER PROPOSE 29
3.1 Introduction 29
3.2 Bases théoriques 29
3.2.1 Le temps d'aller et retour (RTT) d'un lien TCP 29
3.2.2 Fenêtre TCP 30
3.2.2.1 Fenêtred'émission 30

3.2.2.2 Fenêtre de réception 31
3.2.3 Capacité et débit d'uneconnexion TCP 32
3.2.4 Contrôle de congestion 34
3.3 Scheduler proposé 38
3.3.1 Principe de fonctionnement 38
3.3.1.1 Stratégie d'assignation des paquets aux sous-sessions 39
3.3.1.2 Estimation de la taille de la fg_{i+1} 41
3.3.1.3 Assignation des paquets aux sous-sessions 44
3.3.2 Principe d'émission-réception 45
3.4 Conclusion 48

CHAPITRE 4 SIMULATIONS : PRESENTATION ET ANALYSES DES RESULTATS 49
4.1 Introduction 49
4.2 Outil de simulation 49
4.3 Méthodologie 50
4.3.1 Description du système 50
4.3.2 Adaptation du système à l'environnement Matlab 51
4.3.3 Paramètres de simulation 60
4.4 Analyses des resultats de simulations 61
4.4.1 Le débit instantané 64
4.4.2 Débit moyen 65
4.4.3 Délai de réception 68
4.5 Conclusion 69

CONCLUSION 71

ANNEXE I LES ALGORITHMES D'ESTIMATIONS DU NOMBRE DES PAQUETS DE ZONES A ET B 73

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES 79

LISTE DES FIGURES

Page

Figure 1.1 Exemple type d'utilisation de la connexion MPTCP 6

Figure 1.2 Format de l'en-tête TCP 7

Figure 1.3 Simple scénario d'usage de MPTCP 10

Figure 1.4 Comparaison entre les architectures TCP et MPTCP..... 12

Figure 1.5 Echange des signaux d'initialisation d'une connexion MPTCP 13

Figure 1.6 Diagramme d'initialisation d'une connexion de MPTCP 15

Figure 1.7 Exemple de data sequence mapping..... 16

Figure 1.8 Configuration du scheduler à buffers individuels 18

Figure 1.9 Configuration du scheduler à buffer partagé 19

Figure 1.10 « Scheduler Design Parameters » 19

Figure 2.1 Ordre des paquets avant leur envoie..... 24

Figure 2.2 Exemple du problème de head-of-line blocking avec RTT2⁄RTT1=4 25

Figure 2.3 Exemple du problème de head-of-line blocking avec RTT2⁄RTT1= 5.5 25

Figure 3.1 Illustration graphique de la mesure du RTT_m..... 30

Figure 3.2 Exemple de la fenêtre d'émission..... 31

Figure 3.3 Types des données de la fenêtre de réception 32

Figure 3.4 Schéma de principe du scheduler proposé..... 38

Figure 3.5 Principe de la stratégie d'assignation des paquets de notre schedule..... 40

Figure 3.6 Scénario de la réémission méthode standard..... 46

Figure 3.7 Scénario de réémission de paquet perdu (méthode proposée)..... 47

Figure 4.1 Les deux types d'intervalles de temps..... 51

Figure 4.2 Différentes positions temporelle d'une cellule i d'un tableau 52

Figure 4.3 Exemple d'adaptation à l'environnement Matlab d'une session TCP 53

Figure 4.4 Tableau représentant le débit de transmission illustrée par le chronogramme de la figure 4.3 55

Figure 4.5 Exemple d'adaptation à l'environnement Matlab d'une connexion MPTCP . 56

Figure 4.6 Tableau représentant le débit de transmission illustrée par le chronogramme de la figure 4.5 60

Figure 4.7 Illustration de l'influence des ITAR sur le débit au niveau application pour chacune des sous-sessions (cas de Round Robin)..... 62

Figure 4.8 Illustration de l'influence des ITAR sur le débit au niveau application pour chacune des sous-sessions (cas de notre proposition)..... 63

Figure 4.9 Courbes représentatives des débits instantanés via SS1 et SS2 obtenues avec notre proposition et celle de la référence 64

Figure 4.10 Les courbes représentatives du débit moyen (2178372 octets reçus)..... 65

Figure 4.11 Les courbes représentatives du débit moyen (2 178 372 octets reçus)..... 67

Figure 4.12 Les courbes représentatives du débit moyen (6 310 176 octets reçus)..... 68

Figure 4.13 Délai en fonction de la taille des données reçues ……..... 69

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

4G	quatrième génération
NGN	Next Generation Network
TCP	Transport Control Protocol
MPTCP	Multipath TCP
IETF	Internet Engineering Task Force
SCTP	Stream Control Transmission Protocol
WiFi	Wireless Fidelity
HSPA	High Speed Packet Access
SYN	Synchronisation des numéros de séquences
ISN	Initial Sequences Number
ACK	Acknowledgment
URG	Pointeur des données urgentes
PSH	Push
RST	Reset
MP_CAPABLE	Multipath capable option
MP_JOIN	Multipath joint option
RFC	Request For Comments
HMAC	Hash-based Message Authentication Code
DSS	Data Sequences Signal
SS	Sous-session
RTT	Round Tripes Time
MSS	Maximum Size Segment

AIMD	Additive Increase, Multiplicative Decrease
HPLTC	High-Performance Language for Technical Computing
ITAR	Intervalle de Temps d'Attente de Réception
ITR	Intervalle de Temps de Réception
VT	Valeur Temporelle
RR	Round Robin

LISTE DES SYMBOLES ET UNITÉS DE MESURE

s	seconde
ms	ms
o	octet
Mo	Méga octets
Go	Giga octets

INTRODUCTION

L'accès aux réseaux sans fil est devenu universel dans les nouvelles générations de systèmes de communication telle que 4G (quatrième génération) ou NGN (Next Generation Network). Cette nouvelle génération n'est autre qu'un environnement sans fil hétérogène dans lequel les différents réseaux d'accès doivent coexister. Il incorpore une variété des technologies d'accès radio intégrées et gérées conjointement et assure aussi une couverture ubiquitaire aux utilisateurs mobiles.

Il apparaît surprenant que, bien que les exigences ont changé, l'architecture de base et les mécanismes pour le transport de données via l'internet sont restés assez semblables par rapport à son début. Au niveau des terminaux, tous les services fournis via l'internet se basent pour l'essentiel sur deux protocoles à savoir IPv4 et IPv6. Cependant, les objets connectés ainsi les services fournis par internet se diversifient et leur nombre augmente avec le temps. (Martin Beck, 2014).

Les équipements terminaux d'aujourd'hui disposent non seulement des multiples et variées interfaces radios, mais aussi ont la capacité d'en utiliser simultanément. Avec cette évolution de nombreuses technologies sans fil et des terminaux mobiles à interfaces réseau multiples, les usagers (qui deviennent chaque année encore plus nombreux) ont non seulement accès aux services n'importe où et n'importe quand, de n'importe quel réseau, mais aussi envisage l'utilisation simultanée des différents accès réseaux à l'aide des plusieurs interfaces. A titre d'exemple, la quasi-totalité des téléphones mobiles et autres équipements terminaux d'aujourd'hui disposent à la fois au moins une interface WiFi et interface cellulaire.

Cependant, avec le traditionnel Transport Control Protocol (TCP), on ne peut utiliser qu'un seul chemin pour le transfert des données entre deux équipements terminaux. De ce fait, le TCP supporte avec difficulté le trafic généré sur internet car il en résulte le problème des congestions et pertes des paquets au dépend de la qualité de service requise par les usagers (F. Melakessou *et al.*, 2007, page 568).

Les réseaux sans fil hétérogènes offrent un cadre pour l'amélioration des transferts en exploitant le paradigme de multi-hôte où les équipements terminaux, doivent découper les flux de données (du côté source) en sous-flux des données et réassembler ces sous-flux (du côté destination). Le choix de la répartition de ces sous-flux des données sur les différents chemins est primordial. A cause des différences de débit, délai et gigue entre les différents chemins, la performance du transfert multi-hôte peut être médiocre.

Le Multipath TCP (MPTCP), standardisé par Internet Engineering Task Force (IETF) vise l'utilisation concurrente des chemins disponibles entre deux équipements terminaux (A. Ford *et al*., 2013) de façon optimale. Il définit une sous-couche dans la couche transport qui est chargé d'élaborer et d'allouer des sous-sessions[1] à la couche réseau. Cependant le standard ne fournit pas une méthode de répartition des données sur ces sous-sessions.

En plus des contraintes physiques liées aux chemins empruntés, sur la base du service fourni, chaque application a des exigences spécifiques par rapport au protocole de transport tel que le MPTCP. Au moment où certaines applications requièrent des débits importants, d'autres ont plus des exigences de point de vue délai. Des services telle que la vidéo conférence, exigent simultanément des débits importants et le plus court délai possible.

D'autre part, l'ordre d'arrivée à la réception des sous-flux des données via les différents chemins disponibles entre les équipements terminaux dépend non seulement des caractéristiques de transfert de ces chemins mais aussi de la manière avec laquelle ces sous-flux des données sont répartis sur les sous-sessions à la source. Ce problème de l'arrivée hors ordre à la destination affecte les performances en termes de débit et délai du MPTCP bien qu'elles fassent parti des objectifs de ce dernier.

Donc, pour utiliser plusieurs chemins pour le transfert des données avec une connexion MPTCP, il est requis une méthode de répartition des données sur les différentes sous-sessions disponibles. A travers cette méthode de répartition, le MPTCP a à décider sur quelle sous-session il doit allouer chaque paquet. Dans notre document, nous nommons le module en charge de cette décision « scheduler ».

[1] Sous-session est la traduction française de subflow (M. BOUCADAIR et C. JACQUENET, 2015)

Le scheduler, à travers une stratégie, doit déterminer comment envoyer les données sur les différentes sous-sessions disponibles en tenant compte des changements hétérogènes et dynamiques selon les différents chemins empruntés par ces sous-sessions et les contraintes liées au contrôle de congestion correspondant. D'autre part, côté destination, le scheduler est chargé de délivrer les flux des données à la couche supérieure (application) dans le même ordre que ces flux des données étaient à l'émission.

Ainsi, dans ce mémoire nous proposons une stratégie d'assignation des flux des données sur les différentes sous-sessions côté source pour minimiser le nombre des paquets hors ordre à la destination de sorte à avoir des débits à l'arrivée relativement plus importants que si on utilisait un seul chemin pour le transfert. Pour valider la stratégie que nous proposons, nous avons effectué nos simulations en considérant deux chemins à caractéristiques temporelles et taux de transmissions asymétriques. Nous avons choisi l'aspect temporel des chemins car même dans un environnement sans pertes, un écart entre les rounds tripe time (RTT) engendre le problème des paquets hors ordre à la réception.

Dans le premier chapitre de notre mémoire, nous donnons un aperçu sur le TCP et le principe et le fonctionnement du MPTCP. Nous y présentons ensuite une revue de littératures relatives aux chedulers.

Le deuxième chapitre est la partie où nous présentons la problématique de notre recherche ainsi que les objectifs qui en sont liés.

Le troisième chapitre, est le lieu où nous avons fait un aperçu des bases théoriques de notre recherche. L'étude et principe de fonctionnement du scheduler que nous proposons y sont aussi présentés.

Le quatrième chapitre est consacré à la présentation et l'analyse des résultats de simulations obtenus. Nous y avons expliqué la méthodologie suivie pour mener nos simulations.

Une conclusion et des recommandations sont aussi faites.

CHAPITRE 1

REVUE DE LITTERATURES

1.1 Introduction

Malgré sa position quasi hégémonique pour le transport des données dans internet, le TCP standard impose par définition l'utilisation d'un unique chemin pour l'établissement et le maintien d'une connexion. Les usagers d'internet ne peuvent pas exploiter les évolutions technologiques les plus récentes en matière de terminaux à multi-interfaces, combinée à celle de pouvoir avantageusement utiliser les ressources de plusieurs chemins distincts susceptibles de supporter la communication.

De nouveaux protocoles de transport ont été conçus pour répondre à ces contraintes, comme SCTP « Stream Control Transmission Protocol » (Martin Becke, 2014). « Néanmoins, SCTP n'est pas massivement adopté, en raison de la prolifération des middleboxes (boîtiers intermédiaires dans une chaîne de communication), comme les NAT (Network Address Translator) et pare-feu, de nature à compromettre le bon fonctionnement des protocoles de transport, et notamment celui de SCTP ». (Mohamed BOUCADAIR et Christian JACQUENET, 2015).

> Compte tenu de la position dominante du protocole TCP, il a paru naturel d'en développer une extension pour permettre à des terminaux d'établir une communication capable d'exploiter les ressources de plusieurs chemins, afin notamment d'optimiser l'usage des terminaux multi-interfaces. Des mesures ont été effectuées pour démontrer qu'il est encore possible d'ajouter de nouvelles extensions TCP à large échelle. C'est ainsi qu'est né le protocole Multipath Transmission Control Protocol (MPTCP). (M. BOUCADAIR et C. JACQUENET, 2015).

Le MPTCP a précisément pour objectif d'optimiser la gestion d'une connexion TCP en permettant notamment l'exploitation de plusieurs chemins distincts qui pourront supporter autant de sous-sessions. La figure 1.1 est exemple type d'utilisation de la connexion MPTCP

où la station mobile accède au serveur à travers un point d'accès WiFi et HSPS simultanément.

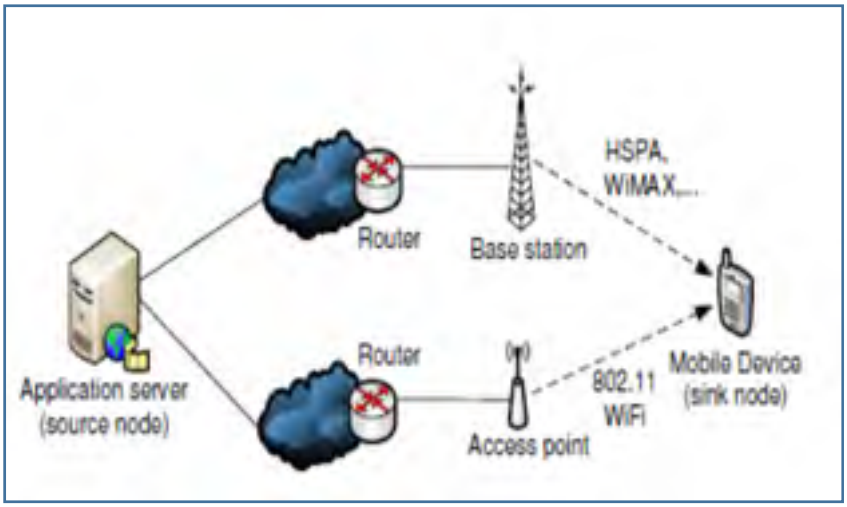

Figure 1.1 Exemple type d'utilisation de la connexion MPTCP
Tirée de Dizhi Zhou *et al.* (2013)

L'objectif clé de la conception de MPTCP est qu'il devrait être en mesure d'assurer une bonne performance sous diverses contraintes du réseau de chaque chemin. (Amanpreet Singh *et al*., 2012).

1.2 TCP (Transport Control Protocol)

Le protocole TCP comme tout protocole de la couche transport fournit une communication de type logique entre des processus d'application exploités sur des serveurs différents. Il assure un transfert de données bout à bout fiable en mettant en œuvre la détection et la correction d'erreurs, en gérant le contrôle de flux et négociant les conditions de transfert des données entre les deux extrémités de la connexion. Le segment (appelé aussi paquet) est l'entité géré par TCP. (Danièle Dromard et Dominique Seret, 2013).

1.2.1 Fonctionnalités de TCP

La détection et la correction d'erreurs que met en œuvre le TCP repose sur la numérotation des données ainsi que leur acquittement et aussi sur l'utilisation de la fenêtre de contrôle de congestion.

Il est à noter que la numérotation des données dans TCP se fait par octet et que les données transportées sont considérées comme des flux d'octets. La connexion TCP étant bidirectionnelle, les flux des données sont traités suivant le sens de transfert indépendamment par les extrémités concernées. Les données gérées par chacune des extrémités doivent être comprises dans les limites d'une fenêtre dont la taille est définie par un entier de 16 bits, donc la taille maximale d'une fenêtre TCP est de 65535 octets. (Danièle Dromard et Dominique Seret, 2013).

1.2.2 Format d'un segment TCP

TCP emploie un seul format de segment structuré comme représenté à la figure 1.2.

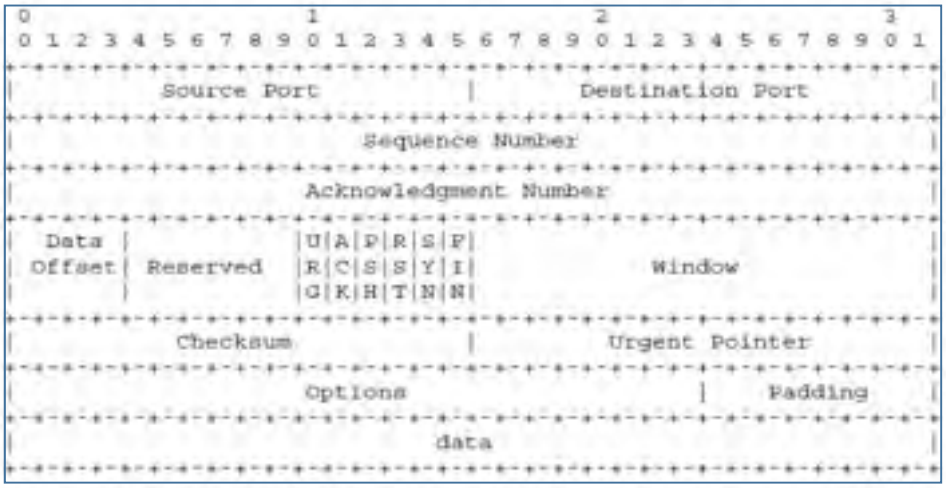

Figure 1.2 Format de l'en-tête TCP
Tirée de Jon Postel (1981)

- les champs Port source/destination déterminent les ports utilisés pour la connexion à chaque extrémité identifiant les deux applications ;
- numéro de séquence: le numéro du premier octet de données par rapport au début de la transmission excepté le segment de synchronisation (SYN). Si SYN est actif, le numéro de

séquence est le numéro de séquence initial (ISN) et le premier octet a pour numéro ISN+1 ;

- le numéro d'acquittement (acknowledgment ACK) est un champ qui contient le numéro de séquence du prochain octet que le récepteur s'attend à recevoir. Une fois la connexion établie, ce champ est toujours renseigné ;
- Data offset : La taille de l'en-tête TCP en nombre de mots de 32 bits. Il indique là ou commence les données. L'en-tête TCP, dans tous les cas à une taille correspondant à un nombre entier de mots de 32 bits ;
- le champ réservé est destiné à une utilisation future ;
- le champ bits de contrôle ou flags. Il est constitué des bits dont le rôle de chacun est défini dans le tableau 1.1 ;
- le nombre d'octets à partir de la position marquée dans l'accusé de réception que le récepteur est capable de recevoir ;
- le champ checksum : il est constitué en calculant le complément à 1 sur 16 bits de la somme des compléments à 1 des octets de l'en-tête et des données pris deux par deux (mots de 16 bits). Si le message entier contient un nombre impair d'octets, un 0 est ajouté à la fin du message pour terminer le calcul du Checksum. Cet octet supplémentaire n'est pas transmis. Lors du calcul du Checksum, les positions des bits attribués à celui-ci sont marquées à 0. Le Checksum couvre de plus un pseudo en-tête de 96 bits préfixée à l'en-tête TCP. Ce pseudo en-tête comporte les adresses Internet source et destinataires, le type de protocole et la longueur du message TCP. Ceci protège TCP contre les erreurs de routage. Cette information sera véhiculée par IP, et est donnée comme argument par l'interface TCP/Réseau lors des appels d'IP par TCP ;
- le champ de pointeur des données urgentes communique la position d'une donnée urgente en donnant son décalage par rapport au numéro de séquence. Le pointeur doit pointer sur l'octet suivant la donnée urgente. Ce champ n'est interprété que lorsque URG est marqué ;
- les champs options sont variables. Elles peuvent occuper un espace de taille variable à la fin de l'en-tête TCP. Ils formeront toujours un multiple de 8 bits. Toutes les options sont prises en compte par le Checksum.

Tableau 1.1 Signification des différents bits code de l'entête TCP

Bits de code de gauche à droite	Signification
Si URG = 1	Pointeur de données urgentes valide
Si ACK = 1	Champ d'accusé de réception valide
Si PSH = 1	Push requis
Si RST = 1	Réinitialisation de connexion
Si SYN = 1	Synchronisation des numéros de séquences
Si FIN = 1	Fin des données de l'émetteur

1.3 Fonctionnement du MPTCP

Le MPTCP est un ensemble d'extension au TCP standard pour fournir un service MPTCP permettant à la couche transport d'opérer à travers plusieurs chemins simultanément. (A. Ford *et al*., 2013, p. 4). Il exploite l'avantage d'utilisation des multiples interfaces actives en permettant la répartition des flux des données d'une session TCP sur plusieurs sous-sessions.

Comme le TCP standard, le MPTCP assure une communication bidirectionnelle entre deux équipements terminaux et ne requiert aucun changement des applications. Bien que les concepts de multi-hôte et multi-chemin[1] ne soient pas nouveaux, le MPTCP vise un déploiement à grande échelle à travers les objectifs de compatibilités vis-à-vis des applications et réseaux existants. (A. Ford *et al*., 2011). La figure 1.3 illustre typiquement le simple scénario d'usage d'une session MPTCP où deux équipements multi-hôtes et multi-adresses A et B se communicant entre eux fournissent deux sous-sessions disjointes. Il y a entre A et B, quatre sous-sessions possibles respectivement formés par les liens entre les paires d'adresses A1 et B1, A1 et B2, A2 et B1, et A2 et B2. Chaque paire des sous-sessions

[1] Ici multi-chemin est relatif à la couche transport à ne pas confondre le trajet multiple de la couche physique

(par exemple les sous-sessions entre A1 et B1 et entre A1 et B2) peuvent être associées à une connexion MPTCP.

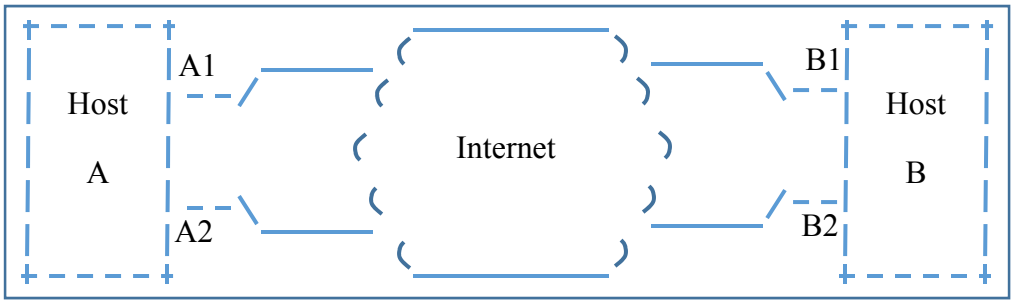

Figure 1.3 Simple scénario d'usage de MPTCP
Tirée de A. Ford *et al*. (2011, page 6)

Les contraintes clefs (A. Ford *et al*., 2013, p. 4) imposées à la norme MPTCP sont :

- la rétrocompatibilité avec le TCP standard courant pour accroitre la chance de son déploiement ;
- au moins un des équipements terminaux doit être multi-hôte.

1.3.1 Les objectifs fonctionnels du MPTCP

Pour inciter l'utilisation du multi-chemin, le MPTCP doit être en mesure d'atteindre les objectifs fonctionnels suivants :

- améliorer le débit : le MPTCP, en supportant l'utilisation concurrente de multiples chemins, doit présenter le minimum des performances incitatives à son déploiement en assurant une connexion à travers multiples chemins avec un débit au pire de cas meilleurs que celui du meilleurs chemin (A. Ford *et al*., 2011) ;
- améliorer la résilience : le MPTCP doit aussi supporter l'utilisation des multiples chemins de façon interchangeable à de fin de résilience en permutant les segments à transmettre ou à retransmettre sur n'importe quel chemin disponible (A. Ford *et al*., 2011). Il ne doit pas au pire des cas être moins résilient que le TCP standard.

1.3.2 Architecture de MPTCP

Le MPTCP est un protocole de la couche transport et se veut transparent aussi bien pour la couche application que pour la couche réseau. La couche application voit une connexion MPTCP comme une connexion TCP standard. Cependant, pour la couche réseau, chacune des sous-sessions MPTCP apparaît comme une session TCP standard dont les segments présentent des nouveaux types d'options TCP. Le MPTCP gère la création, la suppression ainsi que l'utilisation de ces sous-sessions pour le transfert des données (A. Ford *et al.*, 2013).

La figure 1.4 montre la différence entre l'architecture de la couche transport pour TCP standard et son architecture pour le MPTCP. Situé en dessous de la couche application, le MPTCP doit gérer les sous-sessions en dessous par les mécanismes suivants :

- le mécanisme de la gestion des chemins, à travers lequel le MPTCP détecte et utilise les multiples chemins disponibles entre deux équipements terminaux ;
- le mécanisme de coordination des paquets à travers lequel le MPTCP subdivise en segments les flux d'octets venant de l'application. Il assigne ces segments sur les différentes sous-sessions disponibles après avoir répertorié chaque segment par son numéro de séquence correspondant à sa position dans le flux des données venant de l'application. C'est grâce à ce numéro que l'opération de réordonnancement est possible à la réception. Au niveau de récepteur, le MPTCP réassemble et réordonne si nécessaire les paquets venant des différentes sous-sessions TCP pour former le flux des données qu'il délivre à la couche application ;
- le mécanisme d'interface aux sous-sessions TCP à travers lequel la compatibilité avec la couche réseau est assurée. Au niveau de l'émetteur, chaque sous-session TCP ajoute son propre numéro de séquence aux segments qu'il reçoit du mécanisme de répartition avant de les acheminer vers la couche réseau. Au niveau du récepteur, chaque sous-session TCP renvoie les paquets reçus au mécanisme de coordination des différentes sous-sessions ;
- le mécanisme de contrôle de congestion à travers lequel le MPTCP coordonne le contrôle de congestion à travers chacune des sous-sessions.

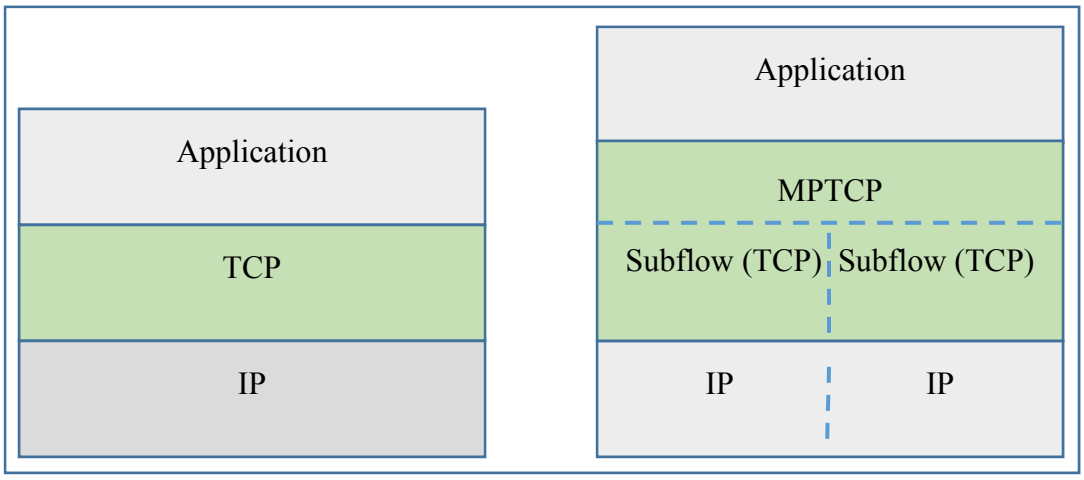

Figure 1.4 Comparaison entre les architectures TCP et MPTCP
Tirée de A. Ford *et al.* (2013, page 6)

1.3.3 Initialisation de la connexion MPTCP

L'initialisation de la connexion MPTCP commence via une seule session avec échange des segments SYN, SYN/ACK et ACK. L'option Multipath Capable (MP_CAPABLE) est insérée dans le champ option de chacun de trois segments TCP ci-dessus (A. Ford *et al.*, 2013) où les équipements A et B (hots A et host B) sont respectivement émetteur et récepteur comme présenté à la figure 1.5.

L'option MP_CAPABLE signale que son émetteur peut fonction en multi-chemin et est disposé à fonctionner sur ce type de connexion. Elle permet aux équipements A et B d'échanger entre eux les informations clefs nécessaires pour l'authentification des sous-sessions à joindre à la connexion. L'équipement B reçoit l'écho des informations clefs relatives à l'équipement A via le segment SYN. L'équipement A reçoit l'écho des informations clefs relatives à l'équipement B et via le segment SYN/ACK. Les informations clefs permettent l'identification de la connexion (à travers les interfaces, les adresses IP et ports additionnels disponibles) mais aussi la sécurisation de la connexion.

L'option MP_CAPABLE est utilisée par seulement la première sous-session (sous-session maîtresse) pour l'identification de la connexion. Pour joindre la connexion existante, les autres sous-sessions (sous-sessions esclaves) utilisent une autre option dite MP_JOIN.

A la fin de ces trois phases appelées « three-way handshake », la sous-session maîtresse est établie.

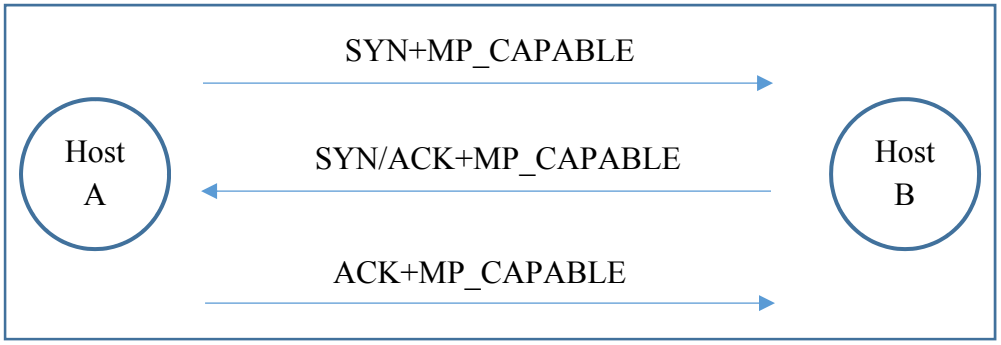

Figure 1.5 Echange des signaux d'initialisation d'une connexion MPTCP

1.3.4 Joindre une nouvelle sous-session

Une fois que chacun de deux équipements terminaux a connaissance de ses propres adresses et celles de l'autre à travers l'option MP_CAPABLE, une nouvelle sous-session peut être initialisée entre une paire d'adresses inutilisée. Cependant il faut noter que la nouvelle sous-session peut être initialisée entre une paire des ports inutilisés (A. Ford *et al.*, 2013).

Un de deux équipements en connexion peut initialiser une nouvelle sous-session mais il est attendu que ça soit l'équipement à l'origine de la connexion (A. Ford *et al.*, 2013).

La nouvelle sous-session, comme la session TCP standard avec les SYN/ACK échanges. Il identifie la connexion à joindre à travers la « Connection (MP_JOIN) TCP option » en utilisant les informations clefs échangées à travers MP_CAPABLE option (A. Ford *et al.*, 2013).

L'option MP_JOIN est présente pendant le three-way handshake c'est-à-dire dans chacun des paquets SYN, SYN/ACK et ACK. Cependant dans chaque cas, cette option a un nouveau format. Le détail dans la RFC 6824 (A. Ford *et al.*, 2013).

Les informations échangées à travers MP_JOIN option sont (A. Ford *et al.*, 2013):

- un hachage cryptographique des informations clefs du récepteur appelé « token » qui permet l'indentification de la connexion MPTCP en question. Il est présent dans le paquet SYN ;
- un nombre aléatoire relatif à l'équipement émetteur (R-A) et un autre relatif au récepteur (R-B). Ces deux nombres aléatoires sont respectivement présents dans les paquets SYN et SYN/ACK ;
- adresse ID qui n'a de signification que dans une simple connexion où il identifie l'adresse source si jamais l'entête IP serait changé par un middlebox. Il est présent dans le paquet SYN ;
- « Hash-based Message Authentification Code (HMAC) » généré avec l'algorithme de hachage SHA-1. Il est présent dans les paquets SYN/ACK et ACK. Il permet une authentification sécurisée entre les équipements participant la même connexion MPTCP.

Si on considère deux équipements A et B les multi-hôtes liés par une connexion MPTCP, la figure 1.6 suivant représente le diagramme d'authentification de cette connexion MPTCP où :

- Key-A représente les informations clefs A et Key-B représente celles de B ;
- Token-A est le token de A, Token-B celui de B ;
- R-A est le nombre aléatoire relatif à A et R-B celui relatif à B ;
- HMAC-A relatif à A et HMAC-B relatif à B.

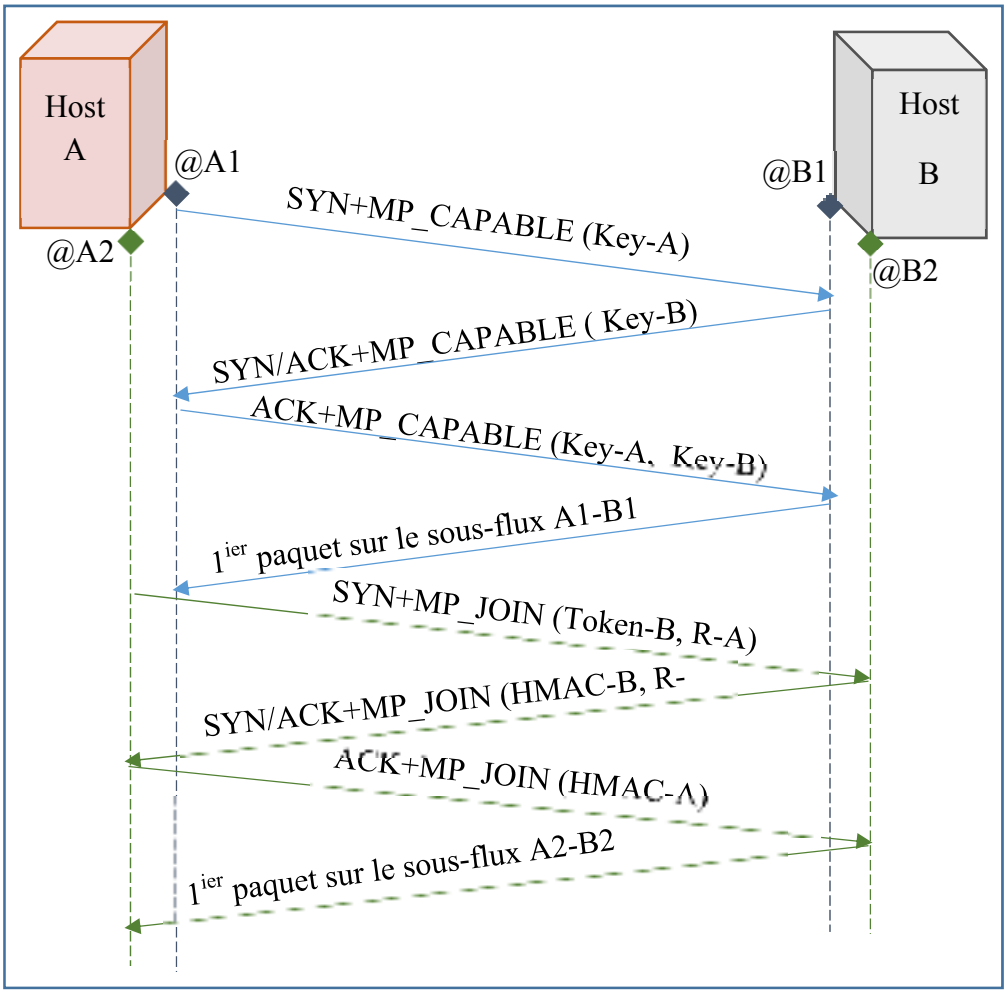

Figure 1.6 Diagramme d'initialisation d'une connexion de MPTCP

1.3.5 Structures des séquences de données (data sequences mapping)

Le flux des données, en tant qu'ensemble est géré à travers la composante de data séquences mapping de l'option DSS (Data Sequences Signal). Cette option définit la manière avec laquelle les séquences des sous-sessions sont liées aux séquences des données permettant ainsi à la réception d'assurer une livraison des données dans leur ordre vers la couche d'application (A. Ford *et al.*, 2013).

Le data sequence mapping spécifie la manière avec laquelle l'espace des séquences des sous-sessions est liée à l'espace des séquences des données en terme de numéros des séquences de

départ pour les sous-sessions et le niveau données ainsi que pour la taille des données qui en est associée (A. Ford *et al.*, 2013). Le data sequence mapping est illustré à la figure 1.7.

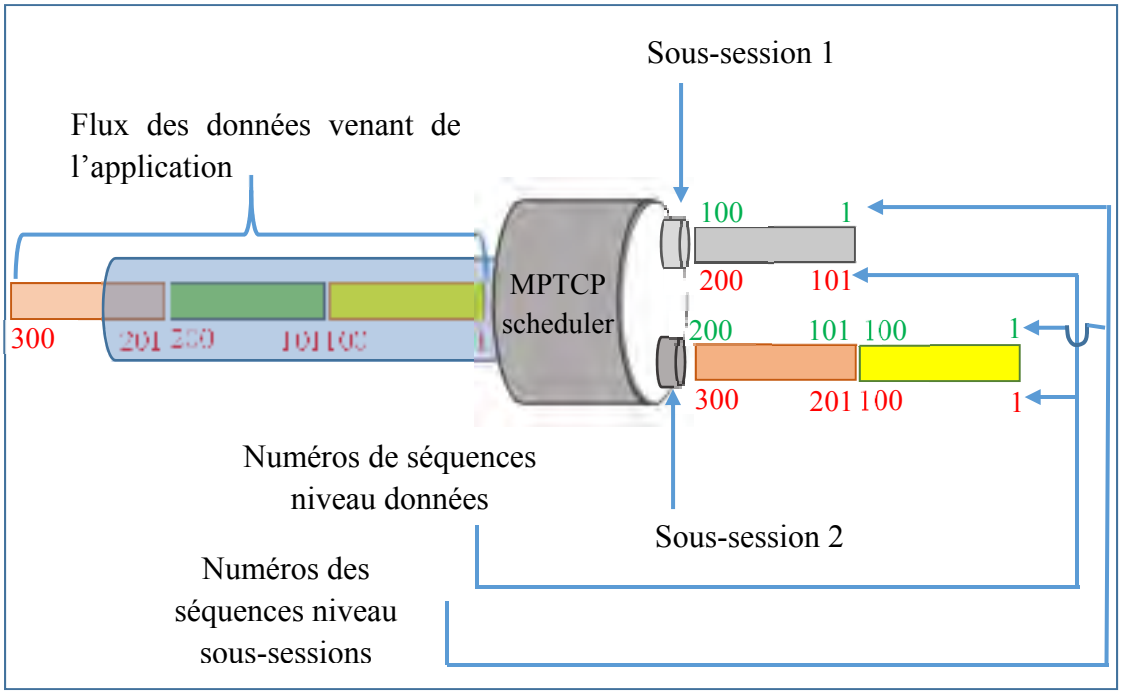

Figure 1.7 Exemple de data sequence mapping

Le numéro de séquence d'une donnée est une valeur absolue c'est-à-dire détermine la position réelle de cette donnée dans le flux des données alors que le numéro de séquence au niveau de la sous-session a une valeur relative donc ne déterminant pas la vraie position d'une donnée. Exemple les données comprises entre l'octet 101 et l'octet 200 se trouve entre l'octet 1 et l'octet 100 au niveau de la SS1 (Voir figure 1.7).

1.3.6 Acquittement des données

Pour assurer une résilience bout en bout, le MPTCP fournit un acquittement au niveau connexion (Data ACK). Data ACK agit comme un acquittement cumulatif pour la connexion dans son ensemble. Par contre l'acquittement au niveau de la sous-session ne prouve que la réception ce niveau donc ne prouve pas qu'il n'y a pas de gap dans le flux des données au niveau connexion.

Data ACK prouve que les données et tous les signaux MPTCP requis sont reçus et acceptés par le terminal en correspondance. L'utilité clef du signal Data ACK est que c'est lui qui indique l'extrémité gauche de la fenêtre de réception annoncée, c'est-à-dire le dernier octet de la dite fenêtre. Cette fenêtre concerne toutes les sous-sessions en ce sens que toutes les données provenant de ces sous-sessions doivent y être reçues.

Un des atouts de la séparation des acquittements de niveau connexion et de niveau des sous-sessions est que chaque sous-session peut libérer l'espace mémoire de son buffer occupée par le segment acquitté avant le Data ACK. Cependant, si toutes les sous-sessions partagent le même buffer, aucune donnée ne doit être écrasée de buffer avant l'écho du Data ACK.

1.3.7 Fin de connexion

Pour le TCP standard, « FIN » signal au récepteur que l'émetteur n'a plus des données à envoyer. Pour permettre un fonctionnement indépendant de chacune des sous-sessions et garder son apparence au TCP tout au long de son chemin, FIN en MPTCP n'affecte que la sous-session sur laquelle il est envoyé (A. Ford *et al.*, 2013).

Pour le MPTCP, la fermeture d'une application se fait avec « DATA_FIN » qui est une indication signalant que l'émetteur n'a plus des données à envoyer. DATA_FIN a la même signification qu'a FIN pour le TCP standard et est signalé par le flag « F » de l'option DSS. DATA_FIN peut être aussi utilisée pour vérifier que toutes les données ont été reçues avec succès (A. Ford *et al.*, 2013).

1.4 Présentation des multipath schedulers

Les multipath schedulers se décomposent en trois catégories :

- schedulers à buffers individuels ;
- schedulers à buffer partagé ;
- schedulers hybrides.

1.4.1 Schedulers à buffers individuels

Cette catégorie de schedulers utilise une stratégie selon laquelle les paquets des données sont alloués à une sous-session aussitôt qu'ils sont générés par l'application sur la base des paramètres individuels de cette sous-session tels que délai et capacité (Amanpreet Singh *et al.*, 2012). Cela exige que chaque sous-session ait son propre buffer (voir figure 1.8).

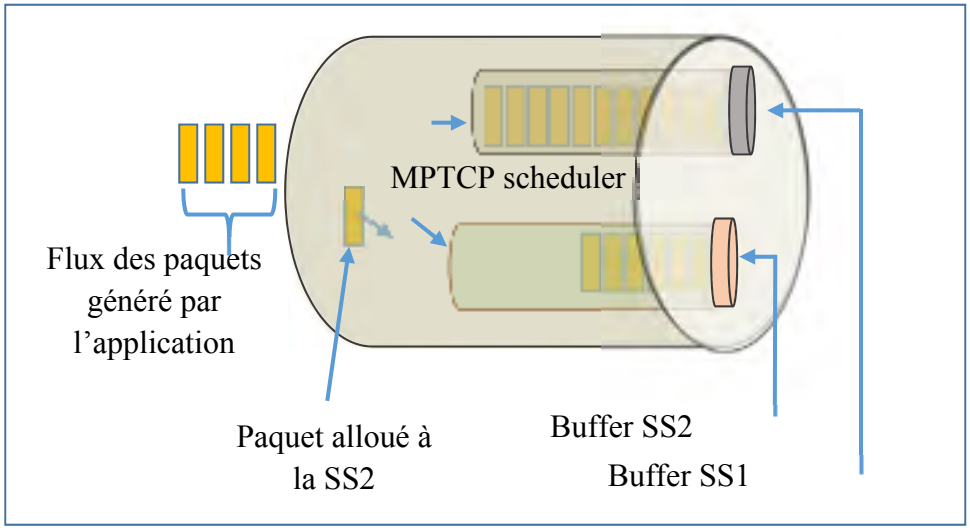

Figure 1.8 Configuration du scheduler à buffers individuels

1.4.2 Schedulers à buffer partagé

Cette catégorie de schedulers est illustrée à la figure 1.9 où fcg_1 est la fenêtre de contrôle de congestion de la SS1 et fcg_2 celle de la SS2.

La stratégie de cette catégorie de schedulers consiste à stocker les paquets des données générés par l'application dans un seul buffer. Les paquets ne sont alloués qu'à la sous-session prête à les transmettre c'est-à-dire ayant une fenêtre ouverte (Amanpreet Singh *et al.*, 2012). Le buffer est donc partagé par toutes les sous-sessions.

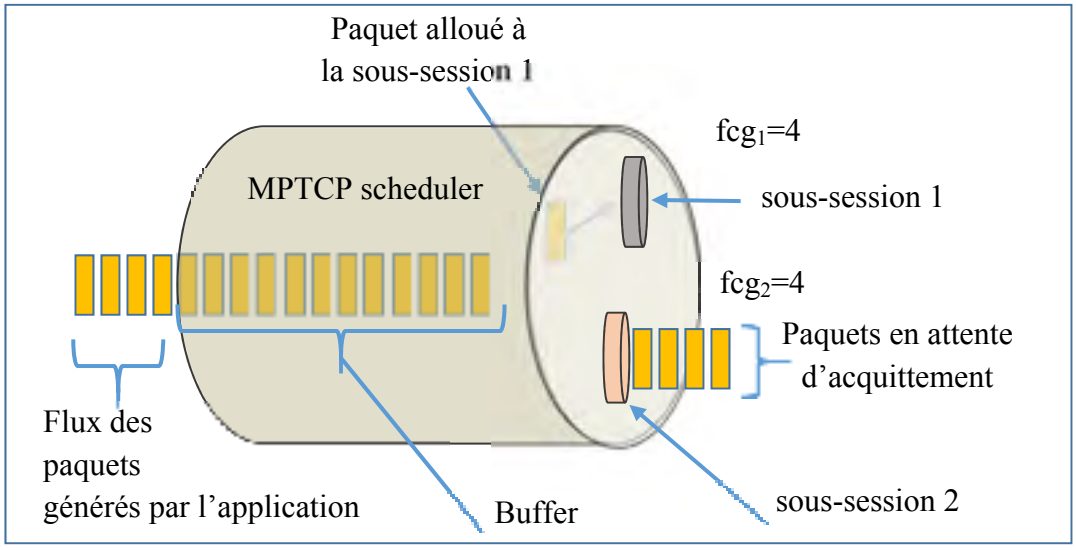

Figure 1.9 Configuration du scheduler à buffer partagé

1.4.3 Schedulers hybrides

Cette catégorie de schedulers combine les deux stratégies précédentes en maintenant des courtes listes d'attente au niveau de buffer de chaque sous-session en plus de buffer qu'elles partagent.

1.4.4 Les paramètres caractéristiques des schedulers

L'assignation des données à une sous-session, peut prendre en compte plusieurs les variables (voir figure 1.10), par exemple capacité de la sous-session ou délai sur la sous-sessions et d'autres paramètres d'entrée pertinents (Amanpreet Singh *et al*., 2012).

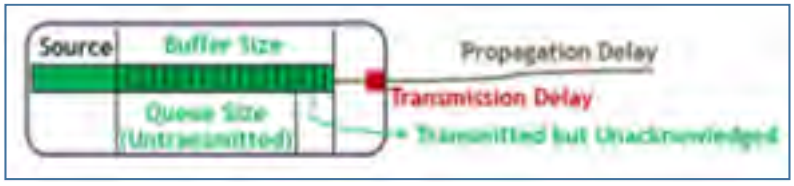

Figure 1.10 « Scheduler Design Parameters »
Tirée de Amanpreet Singh *et al*. (2012)

Il existe plusieurs stratégies d'assignation des données aux sous-sessions qui s'appuient sur la configuration du scheduler à buffers individuels.

1.4.4.1 Scheduler indépendant des caractéristiques des sous-sessions

C'est le « Round Robin scheduler » (Amanpreet Singh *et al*., 2012) dont la stratégie consiste à allouer séquentiellement les paquets aux sous-sessions.

1.4.4.2 Scheduler utilisant les informations d'état de l'émetteur

La stratégie de ce type de scheduler se base sur la détermination de la sous-session qui est disponible au niveau de l'émetteur suivant la taille de la liste d'attente et la taille du buffer (Amanpreet Singh *et al*., 2012). Donc l'allocation des paquets est faite suivant deux types de priorité.

- on peut se fixer comme priorité la taille de la liste d'attente au niveau chaque sous-session. Dans ce cas, les paquets sont à chaque fois alloués à la sous-session ayant la plus courte liste d'attente ;
- l'autre priorité est la taille du buffer associé à chaque sous-session. Ici on assigne en priorité les paquets à la sous-session qui a le plus petit nombre des paquets bufférisés.

1.4.4.3 Scheduler se basant sur la capacité des sous-sessions

La stratégie de ce type de scheduler consiste à allouer les paquets à une sous-session sur la base de la capacité de celle-ci. Ici l'assignation des paquets aux différentes sous-sessions se fait proportionnellement aux capacités individuelles de chacune des sous-sessions ou suivant la priorité selon laquelle les paquets sont alloués à la sous-session qui les transmettra le plus tôt.

1.4.4.4 Scheduler se basant à la fois sur la capacité et le délai

La stratégie ici est de tenir en compte d'aussi bien de la capacité que du délai de la sous-session. Les paquets sont alloués à la sous-session capable de les livrer le plus tôt.

1.5 Recherches antérieures relatives aux multi-path schedulers

Bien qu'elle soit d'une grande importance, l'IETF laisse la problématique de multipath scheduler ouverte aux chercheurs, donc rien de spécifique n'est défini dans « internet drafts » (Martin Beck, 2014).

Si deux chemins ont une large différence en termes de bandes passantes et délais, sans un scheduling algorithme approprié, le MPTCP sur ces chemins sera moins performant que le TCP standard (Amanpreet Singh *et al*., 2012). Or, la large différence en termes de bandes passantes et délais est une réalité pour les réseaux sans fil hétérogènes.

« Linux MPTCP implementation » (Costin Raiciu *et al*.,2012) est une référence de l'implémentation du MPTCP et dans sa version 0.88, adopte un mécanisme appelé « opportunistic retransmission and penalization » où on essaie de compenser les limites de la fenêtre de réception dues aux écarts entre les RTT des différents chemins empruntés en retransmettant les paquets non acquittés à travers une autre sous-session et réduisant de moitié la fenêtre de congestion de la sous-session lente (Simone Ferlin-Oliveira *et al*., 2014). Cette démarche paraît logique du point de vue partage de charge. Mais dans les réseaux sans fil hétérogènes avec un écart important entre les RTT, cette stratégie peut paraître caduque car on peut avoir plusieurs accusés de réception sur un chemin de RTT plus petit sans en avoir un sur le chemin de RTT plus grand. Si à chaque accusé de réception on réduit de moitié la fenêtre de congestion de la sous-session lente (à RTT plus grand), cette fenêtre peut être remise à zéro à partir d'un certain nombre d'accusés de réception de la sous-session la plus rapide (à RTT plus petit) et par conséquent les performances en termes de débit ne seront pas au rendez-vous.

Une autre stratégie est proposée dans (ALLen L. Ramaboli *et al.*, 2013) qui consiste à allouer chaque paquet à la sous-session à travers laquelle il sera reçu le plus tôt. L'objectif visé par cette stratégie est de réduire le nombre des paquets hors ordre à la réception en espérant améliorer le débit. Comme pour la stratégie de (Costin Raiciu *et al*.,2012), ici aussi les performances en terme de débit ne seront pas au rendez-vous. Dans cette approche, il y a

problème de non répartition de charge surtout si l'écart entre les RTT est élevé. En cas de perte sur la sous-session la plus rapide, l'objectif visé risque de ne pas être atteint.

« Lowest-RTT-First (LowRTT) » est une autre stratégie qui donne la priorité à la sous-session ayant le plus petit RTT tant que sa fenêtre de congestion est ouverte (Christoph Paasch *et al.*, 2014). Cette stratégie n'est pas différente de weighted round robin (WRR). Dans les réseaux sans fil hétérogène, cette stratégie ne sera pas efficace dans le cas où l'écart entre les RTT est grand car il en résultera un nombre important de paquets hors ordre donc apparition du head-of-line blocking.

« Out-of-order Transmission for In-order Arrival Scheduling » est une stratégie proposée dans (Fan Yang, Qi Wang et Paul D. Amer, 2014). Ici il s'agit d'assigner le paquet à la sous-session via laquelle il sera reçu le plus tôt, l'instant d'assignation du paquet étant le repère. Cette stratégie garantit une livraison en ordre des paquets avant que les sous-sessions atteignent leurs capacitté maximale mais au dépend de débit et conduira à une sous-utilisation de la sous-session ayant la plus grand RTT.

La quasi-totalité des propositions existantes dans la littérature présente des performances en dessous des attentes surtout dans les réseaux sans fil hétérogènes où les chemins ont des caractéristiques asymétriques que sont :

- le temps d'aller et retour (RTT) ;
- la bande passante ;
- le taux des pertes.

CHAPITRE 2

PROBLEMATIQUE ET OBJECTIVE

2.1 Introduction

Pour que les lecteurs de notre document aient une vision plus claire des problèmes autour desquels se concentre l'effort de notre recherche, il nous paraît important de consacrer ce chapitre à la problématique de notre recherche et les objectifs qui y sont liés.

2.2 La problématique

Dans les réseaux sans fil hétérogènes, la performance de la connexion MPTCP surtout en termes de débit et délai n'est pas satisfaisante. Cette sous-performance peut être expliquée par le délai induit par le phénomène connu sous le nom de « head-of-line blocking » (Christoph Paasch *et al.*, 2014). L'effet d'une stratégie d'assignation des paquets inadapté au niveau de l'émetteur n'en est pas aussi étranger (Amanpreet Singh *et al*., 2012).

2.2.1 Head-of-line blocking

Il y a réordonnancement lorsque le phénomène de head-of-line blocking se produit, c'est-à-dire que les paquets qui arrivent à la destination hors ordre requis doivent attendre l'arrivée des paquets auxquels ils succèdent à la source. Le TCP interprète mal le réordonnancement des paquets en le considérant comme résultat de la perte due à la congestion et réagit en réduisant la fenêtre de congestion et requérant une retransmission rapide non nécessaire. Par conséquence, le débit peut dramatiquement baisser. Le réordonnancement peut également induire des délais supplémentaires qui peuvent dégrader les performances des applications qui en sont sensibles (Allen L. Ramaboli *et al.*, 2013).

Si la connexion MPTCP utilise des chemins hétérogènes, le délai induit par le réordonnancement devient important si l'écart entre les RTT est important. Pour illustrer ce fait, considérons le scheduler à buffer partagé occupé par des paquets prêts à être envoyés représenté à la figure 2.1.

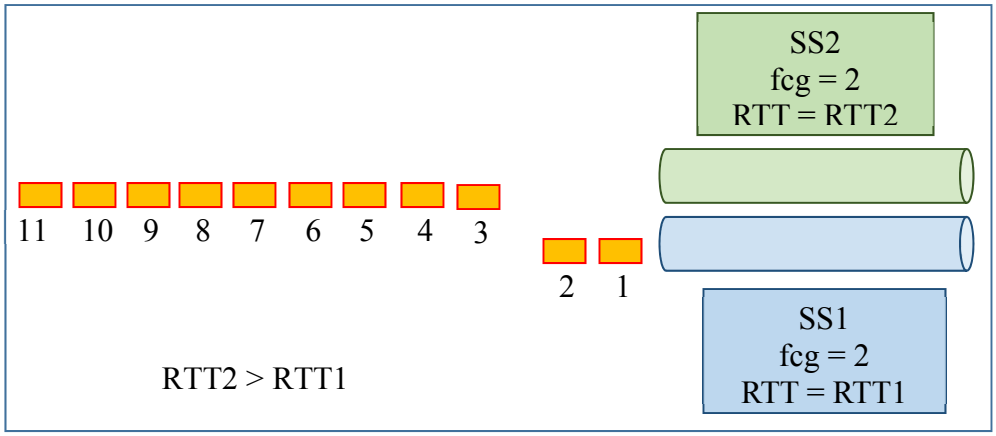

Figure 2.1 Ordre des paquets avant leur envoie

En considérant que le RTT de la SS1 est plus court que celui de la SS2, à l'instant t_0 (voir figure 2.2), selon la stratégie d'assignation des paquets implémentée dans linux, on envoie les paquets 1 et 2 via la SS1. A l'instant t_1, la fenêtre de congestion de la SS1 (fcg1) est occupée par les paquets 1 et 2 non encore acquittés, donc les paquets 3 et 4 sont envoyés via la SS2.

Si le RTT de la SS1 est $RTT1 = 20$ ms et celui de la SS2 est $RTT2 = 80$ ms, à la fin de la réception de tous les paquets, on a trois paquets hors ordre comme le montre la figure 2.2. Ainsi, si Δ est le temps nécessaire à l'ordonnancement d'un paquet, le réordonnancement de deux paquets induit un délai de $3 * \Delta$.

Si le RTT de la SS1 est toujours $RTT1 = 20$ ms et celui de la SS2 devient $RTT2 = 110$ ms, à la fin de la réception de tous les paquets, on a sept paquets hors ordre comme le montre la figure 2.3. Le réordonnancement de deux paquets induit ainsi un délai de $7 * \Delta$.

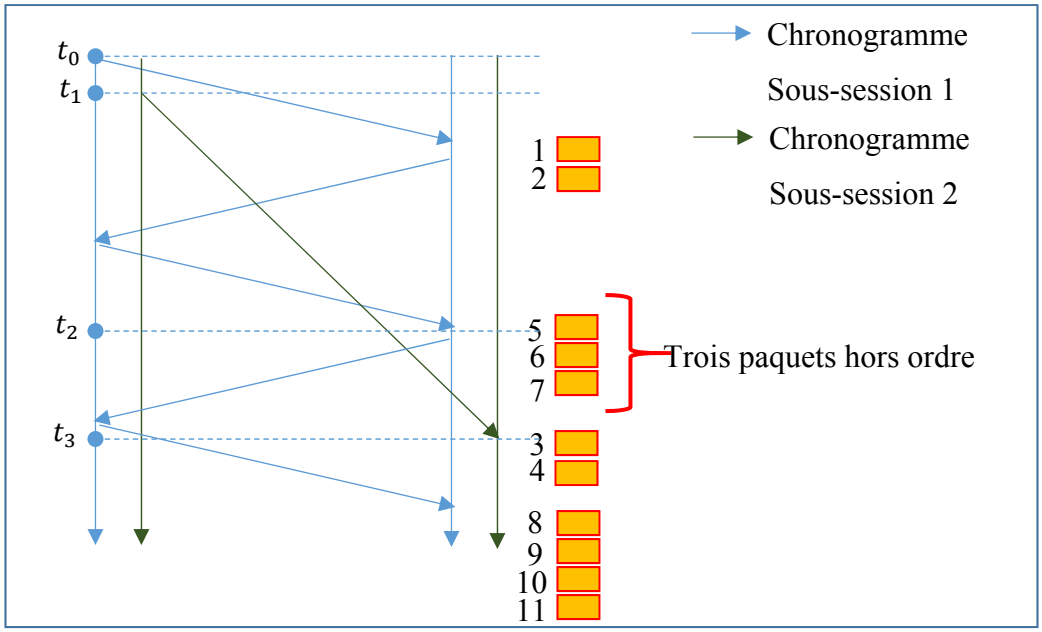

Figure 2.2 Exemple du problème de head-of-line blocking avec RTT2/RTT1=4

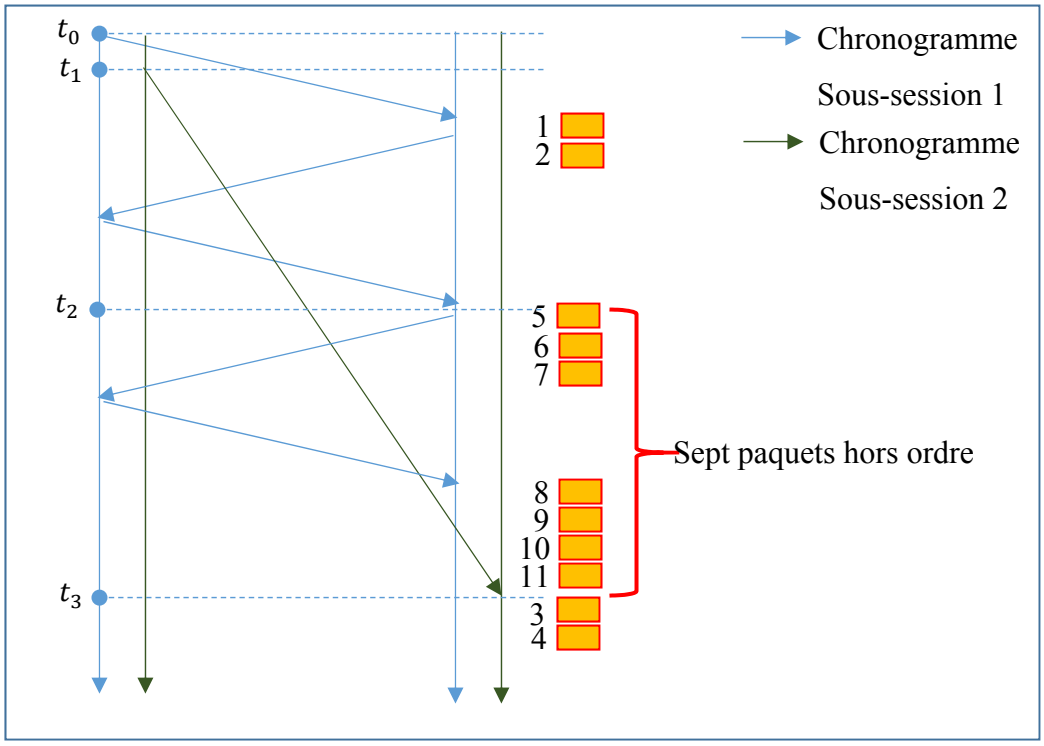

Figure 2.3 Exemple du problème de head-of-line blocking avec RTT2/RTT1= 5.5

On voit à partir des figures 2.2 et 2.3 que le problème de head-of-line blocking s'accentue si l'écart entre les RTT s'élargit. Il va s'accentuer d'avantage si en plus de l'écart entre les RTT, il y a les pertes des paquets.

La solution au problème de head-of-line blocking repose sur l'arrivée en ordre des paquets à la réception.

2.2.2 Scheduler

L'inadaptation de scheduler pour une transmission multipath peut gravement affecter les performances attendues. Par exemple le scheduler implémenté dans linux est inadapté pour le scenario de la figure 2.1. Comme on le voit sur les figures 2.2 et 2.3, il génère le problème des paquets hors ordre et ce problème s'aggrave si l'écart entre les RTT s'agrandit.

2.3 Objectif

L'objectif principal de ce mémoire est de déterminer une stratégie d'assignation des paquets en mesure d'améliorer les performances actuelles du MPTCP en termes de débit et délai.

Même sans pertes des paquets, le goulot d'étranglement des performances du MPTCP en termes de débit et délai est le problème de head-of-line blocking. L'objectif spécifique qui résulte de l'objectif principal est donc de minimiser le nombre des paquets qui arrivent hors ordre à la réception dû à la transmission multipath sans porter atteinte à la répartition de charge que prône le MPTCP.

2.4 Hypothèses

Il faut noter que notre travail de mémoire se focalise sur le cas d'une connexion MPTCP à deux sous-sessions à caractéristiques hétérogènes et asymétriques.

Pour l'atteinte de l'objectif de notre mémoire, nous proposons :

- que la stratégie d'assignation des données venant de la couche d'application aux sous-sessions soit appliquée à l'intérieur de la fenêtre de réception globale ;

- cette stratégie doit garantir que toutes les données envoyées à travers les sous-sessions pour une même fenêtre globale arrivent en ordre ou avec un temps d'attente le plus court possible au niveau du récepteur ;
- que le paquet perdu à travers la sous-session la plus lente soit retransmis à travers la plus rapide mais le paquet perdu à travers la sous-session la plus rapide doit y être retransmis.

Dans une connexion MPTCP, chaque sous-session se présente comme une session TCP à part entière. Donc en cas de la perte d'un paquet au niveau d'une sous-session, il en résulte des paquets hors ordre venant de la dite sous-session au niveau récepteur du fait du mécanisme de la retransmission d'un paquet perdu.

Or, l'ordre des paquets empruntant les différentes sous-sessions de la même connexion MPTCP au niveau application est lié et par conséquent, la perte d'un paquet au niveau d'une sous-session met hors ordre des paquets provenant des autres sous-sessions accentuant ainsi le problème de head-of-line blocking. Par effet domino, ce problème va se poser avec acuité si la perte a lieu sur la sous-session la plus lente.

A titre d'exemple, en se référant à la figure 2.3, si le paquet 3 se perd, il doit être retransmis. La retransmission du paquet 3 ne devrait avoir lieu qu'après deux acquittements négatifs de ce dernier. Donc, en plus des paquets déjà hors ordre, tous les paquets à émettre via SS2 avant la retransmission du paquet 3, tous ceux qui seront émis par SS1 et reçus avant la réception du paquet 3 seront hors ordre.

Ainsi, pour minimiser toujours le nombre des paquets arrivant hors ordre à la réception, nous proposons un mécanisme à travers lequel, la retransmission d'un paquet perdu sur une sous-session lente via la sous-session la plus rapide. Par ce mécanisme on va éviter un nombre non négligeable des paquets hors ordre comme il est illustré au chapitre 3.

La démonstration de l'atteinte de l'objectif de notre mémoire est faite à travers des scénarios simulés dans Matlab.

CHAPITRE 3

PRESENTATION ET FONCTIONNEMENT DU SCHEDULER PROPOSE

3.1 Introduction

Bien qu'un des objectifs autour du MPTCP soit d'améliorer le débit surtout au niveau application, il existe encore un problème non résolue qui est celui de l'arrivée hors ordre à la destination. Il en résulte des performances en termes de débit et délai de livraison en dessous de l'attente. Pour minimiser ces effets indésirables, l'une des solutions possibles est de déterminer une stratégie d'allocation efficace de paquets aux sous-sessions côté source à travers le module scheduler.

3.2 Bases théoriques

3.2.1 Le temps d'aller et retour (RTT) d'un lien TCP

Vue du côté source, le RTT est le temps écoulé entre l'instant de l'envoie d'un segment et l'instant de la réception de son accusé de réception (voir figure 3.1). Pour calculer le RTT (qui est actualisé à chaque transmission), on recueille les informations comme l'instant auquel un segment TCP est émis (t_e) et celui auquel l'accusé de réception correspondant lui parvient (t_r). Le module TCP retient une estimation pondérée de RTT qui s'appuie à la fois sur le mesuré (RTT_m) et précédemment estimé (RTT_e) (Danièle Dromard et Dominique Seret, 2013).

Le RTT résultant de l'estimation actualisée tient compte d'un facteur de pondération α compris entre 0 et 1, donné par :

$$RTT = \alpha * RTT_e + (1 - \alpha) * RTT_m \tag{3.1}$$

$$RTT_m = t_r - t_e \tag{3.2}$$

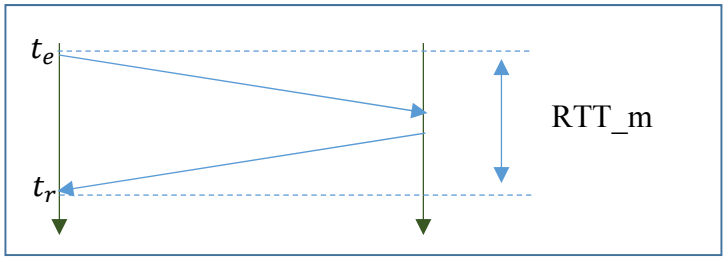

Figure 3.1 Illustration graphique de la mesure du RTT_m

L'option d'estampille horaire (timestamps) permet au TCP de mesurer avec précision la valeur de RTT dans le réseau. Le TCP source indique sur 32 bits l'instant t_e d'émission dans le champ réservé à cet effet qui y joue le rôle de référence temporelle. Le TCP destinataire retourne cette option en écho en y indiquant son instant d'émission. Le RTT est déterminé par la différence entre t_e et l'instant de la réception de l'accusé de réception du segment émis t_r. Le rôle de l'estampille horaire de la destination est de lever les ambiguïtés en cas d'acquittement retardé ou autre (Claude Servin, 2013).

3.2.2 Fenêtre TCP

Sans le concept de fenêtrage, côté source, le module TCP n'émet un segment que si le segment précédemment émis est acquitté, ce qui veut dire que le débit de transmission est limité à un segment par RTT. Le concept de fenêtrage TCP permet d'éviter ce souci. Il existe deux types de fenêtrage : fenêtrage d'émission et fenêtrage de réception.

3.2.2.1 Fenêtre d'émission

Afin d'accélérer les transmissions, on émet plusieurs paquets sans attendre de recevoir un acquittement jusqu'à un nombre des paquets limite : c'est la fenêtre d'émission (voir figure 3.2). Les paquets étant numérotés, il n'est pas nécessaire que tous les acquittements parviennent à l'émetteur : l'acquittement d'un paquet acquitte en même temps les paquets précédents.

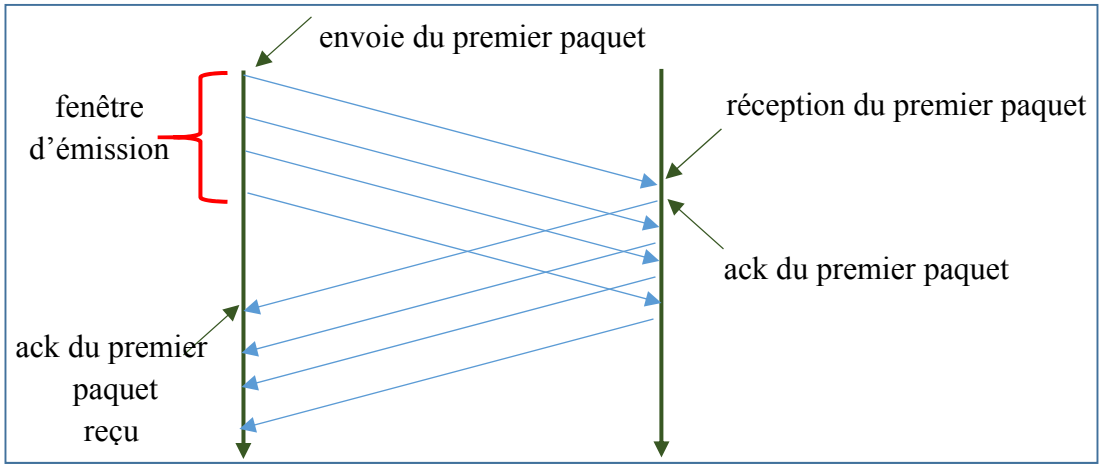

Figure 3.2 Exemple de la fenêtre d'émission

Si les paquets ont emprunté des trajets différents, ils peuvent parvenir à la destination en désordre. Ils sont remis dans l'ordre et stockés provisoirement, mais ils ne sont acquittés que lorsque le récepteur ait reçu tous les paquets précédents.

Si par exemple les paquets 1, 2, 3 et 4 sont émis dans une fenêtre d'émission, si à la destination, on reçoit les paquets 1, 3 et 4, le paquet 1 est acquitté mais les paquets 3 et 4 ne le seront qu'après la réception du paquet 2. En ce moment le paquet 4 est acquitté, ce qui est synonyme de l'acquittement des paquets 2 et 3.

3.2.2.2 Fenêtre de réception

Pour limiter la quantité de données pouvant être envoyées simultanément et fournir un contrôle du flux côté destination, le destinataire impose une fenêtre de réception. La fenêtre de réception détermine le nombre d'octets ou paquets que le destinataire autorise la source à envoyer. La source ne peut envoyer que les octets compris à l'intérieur de la dite fenêtre. La fenêtre de réception suit le flux d'octets sortants de l'émetteur et le flux d'octets entrants du récepteur.

Comme il peut y avoir des données de la fenêtre de réception qui ne sont pas encore récupérées par l'application et des données qui sont reçues mais pas reconnues, la fenêtre de réception TCP a une structure, comme illustré par la figure 3.3.

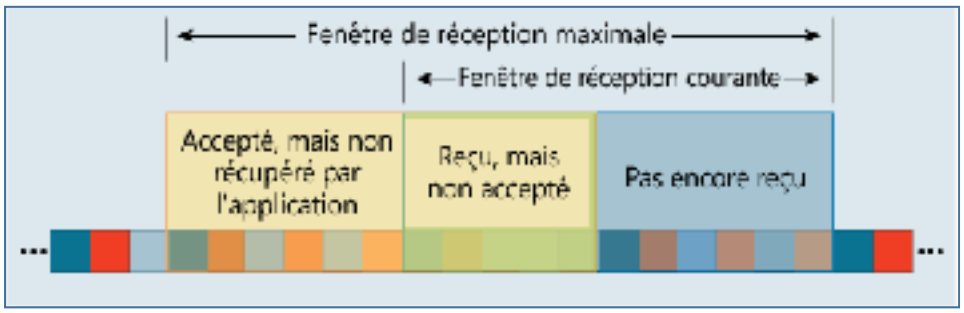

Figure 3.3 Types des données de la fenêtre de réception
Tirée de Joseph Davies (2007)

La fenêtre de réception maximale a une taille fixe qui est celle du buffer à la destnation. La fenêtre de réception courante a une taille variable et correspond à la quantité restante de données que le destinataire autorise la source à envoyer. La taille de la fenêtre de réception courante est la valeur du champ fenêtre indiquée dans les segments ACK renvoyés à la source et elle correspond à la différence entre la taille de la fenêtre de réception maximale et la quantité des données reçues et reconnues mais non récupérées par l'application.

3.2.3 Capacité et débit d'une connexion TCP

Pour optimiser le débit TCP (en supposant que le chemin de transmission est dépourvu d'erreurs), la source doit envoyer suffisamment de paquets pour remplir le canal logique qui la lie au destinataire. Connaissant le débit (D) et le RTT du lien, la capacité (C) de la connexion TCP peut être calculée par la formule suivante (Joseph Davies, 2007) :

$$C\ (en\ bits) = D(en\ bits/s) * RTT(en\ s) \tag{3.3}$$

Le débit (D) d'une connexion TCP peut ne pas être identique à l'émission et à la réception car les octets émis ne sont pas à priori tous reçus. A l'émission, il est donné par la relation suivante :

$$D(octets/s) = \frac{fenêtre\ d'émission(en\ octets)}{RTT\ (en\ s)} \tag{3.4}$$

A la réception, le débit est donné par la relation suivante :

$$D(octets/s) = \frac{nombre\ d'octets\ reçus\ pendant\ RTT}{RTT\ (en\ s)} \tag{3.5}$$

Le débit moyen d'une connexion TCP est donné par la relation (3.6) où MSS (Maximum Size Segment) est la taille maximale d'un segment et W le nombre de segments reçu pendant RTT (Claude Servin, 2013).

$$\overline{D}(octets/s) = \frac{3}{4}\frac{W * MSS}{RTT} \tag{3.6}$$

La source rentre dans la phase de fast recovery à chaque perte de segment. La durée de cette phase en RTT correspondant à l'évolution d'un segment par RTT soit W/2. Dans ces conditions, le nombre N des segments émis pendant ladite phase est donnée par la relation (3.7).

$$N = \frac{3}{4}W * \frac{W}{2} = \frac{3W^2}{8} \tag{3.7}$$

Quelle qu'en soit la cause, d'erreur ou congestion, pour les N segments émis, il n'y a qu'un segment en erreur, la probabilité (p) d'erreur est donc de 1/N. La relation (3.9) donne ainsi W en fonction de p.

$$p = \frac{1}{N} \Rightarrow N = \frac{1}{p} \tag{3.8}$$

$$N = \frac{3W^2}{8} \Rightarrow \frac{1}{p} = \frac{3W^2}{8} \Rightarrow W = 2\sqrt{\frac{2}{3p}} \tag{3.9}$$

De la relation (3.9), on transforme (3.6) en (3.10).

$$\overline{D}(octets/s) = \frac{3}{2}\frac{MSS}{RTT}\sqrt{\frac{2}{3p}} \tag{3.10}$$

Le nombre d'octets reçus pendant RTT est au maximum égal à la taille maximale de la fenêtre de réception. La taille du champ fenêtre dans l'en-tête TCP étant de 16 bits, la taille de fenêtre de réception maximale est 2^{16} octets soit 65 536 octets, donc le débit maximal est :

$$Dmax(octets/s) = \frac{65536\ octets}{RTT\ (en\ s)} \tag{3.11}$$

A partir de l'équation (3.6), le débit est limité (même avec sa valeur maximale), ce qui peut être préjudiciable pour une liaison haut débit. Par exemple, pour une liaison dont le RTT est de 25 ms, le débit maximal que la connexion TCP ne peut pas dépasser est 2.5 Mo/s alors que si la liaison a une capacité par exemple 2 Mo, le débit maximal atteignable est de 80 Mo/s très loin du débit maximal imposé par la connexion TCP.

Mais grâce à l'option d'échelle de fenêtre (D. Borman *et al.*, 2014) qui doit être invoqué lors de l'établissement de la connexion (segment SYN), on peut augment la taille de la fenêtre à l'aide de la relation suivante :

$$fenêtre\ de\ réception\ maximale = 65536 * 2\hat{}n \tag{3.12}$$

L'exposant « n » est au maximum égale à 14 pour une fenêtre de réception maximale de 1 Go (D. Borman *et al.*, 2014).

3.2.4 Contrôle de congestion

Les machines qui émettent et reçoivent les segments de données TCP ne le font pas toutes au même rythme (unités centrales, bande passante). A la réception, le flux des données envoyé par la source doit être à l'intérieur de la fenêtre de réception : c'est le contrôle de flux qui empêche à la source de submerger la destination. Cependant, cela ne résout pas le problème de congestion.

La congestion est gérée par la fenêtre de congestion. Le nombre d'octets que l'on peut transmettre sans attendre les acquits correspondants est le minimum de la taille des deux fenêtres (fenêtre de réception et fenêtre de congestion). Le contrôle de congestion a pour but d'éviter les pertes inutiles de segments sans pour autant nuire à leur débit de transmission. Le

contrôle de congestion de TCP est effectué de bout en bout de la source à la destination et n'est pas fonction de la congestion du réseau. Ceci indique que lorsque le réseau n'est pas congestionné il pourrait y avoir une congestion au niveau TCP.

Une source TCP perçoit le phénomène de congestion par des pertes de segments, à travers soit :

- expiration du temps d'attente de l'accusé de réception (ACK) ;
- réception de trois ACK identiques.

Lors d'un phénomène de perte de segment, TCP a recours à la décroissance multiplicative : la valeur de la fenêtre de congestion est diminuée de moitié. Lorsque la congestion est résorbée, TCP procède à une augmentation progressive de sa fenêtre de congestion.

L'algorithme de contrôle de congestion de TCP est appelé algorithme AIMD (Additive Increase, Multiplicative Decrease).

Il existe 4 algorithmes de contrôle de congestion interdépendants tels qu'ils sont détaillés dans (M. ALLman *et al.*, 2009) :

- démarrage lent (Slow Start) ;
- évitement de la congestion (Congestion avoidance) ;
- retransmission rapide (Fast retransmission) ;
- récupération rapide (Fast recovery).

Pour une session TCP standard, le contrôle de congestion est utilisé pour gérer le rythme avec lequel les données sont envoyées au niveau de l'émetteur. Si chaque sous-session MPTCP utilise le contrôle de congestion comme une session TCP standard, il y aura un partage de flux inéquitable lorsque les chemins empruntés par leurs différentes sous-sessions rencontrent un même goulot d'étranglement. Cela ne va pas dans le sens d'atteinte des objectifs attendus du contrôle de congestion pour une session MPTCP qui sont (Raiciu, *et al.*, 2011):

- amélioration de débit : le MPTCP doit au moins le débit qu'assure le TCP standard sur le meilleur chemin qui lui est disponible ;
- ne pas faire de mal en ce sens qu'une session multi-chemin ne devrait prendre plus de capacité de toutes les ressources partagées par ses différents chemins que la capacité prise par des simples sessions indépendantes utilisant ces chemins ;
- partage de ressource qui veut dire qu'une session multi-chemin doit à la limite de possibilité pouvoir décongestionner le chemin le plus congestionné en envoyant plus des données sur les chemins les moins congestionnés.

Pour le MPTCP, c'est l'algorithme de contrôle de congestion proposé dans (Raiciu, *et al.,* 2011) qui est suggéré. Cet algorithme est appelé « Coupled Congestion Control algorithm». Pour « slow star », « fast retransmission » et « fast recovery », il fonctionne identiquement comme pour TCP standard. Le coupled Congestion Control algorithm amène le changement pour la phase « congestion avoidance » (Raiciu, *et al.,* 2011).

Si on considère fcg_i la fenêtre de congestion de la SSi, RTT_i et MSS_i respectivement le RTT et la taille maximale de segment de la SSi, en maintenant la fenêtre de congestion en octets, on a les équation (3.13) et (3.14 (Raiciu, *et al.,* 2011).

A chaque accusé de réception reçu sur la sous-session i :

$$fcg_i = fcg_i + \min\left(\frac{\alpha * MSSi * MSS_i}{fcg_{tot}}, \frac{MSSi * MSS_i}{fcg_i}\right) \qquad (3.13)$$

A chaque signal de congestion sur la SSi :

$$fcg_i = fcg_i - \frac{fcg_i}{2} \qquad (3.14)$$

fcg_{tot} est la fenêtre globale c'est-à-dire la fenêtre partagée par toutes les sous-sessions et elle est donnée par l'équation (3.15) où N est le nombre des sous-sessions, fcg_{i_j} est la fenêtre de la SSi au moment j de la réception d'un accusé de réception.

$$fcg_{tot} = \sum_{i=1}^{N} fcg_{i\_j} \tag{3.15}$$

α est un facteur d'agressivité et est donné par l'équation (3.16).

$$\alpha = fcg_{tot} \frac{\max\limits_{1 \le i \le N} \frac{fcg_{i\_j}}{RTT_i^2}}{\left(\sum_{i=1}^{N} \frac{fcg_{i\_j}}{RTT_i}\right)^2} \tag{3.16}$$

Quand l'algorithme de contrôle de congestion maintien la fenêtre de congestion fcg en paquets, les expressions précédentes changent. Pour un acquittement positif, le coupled congestion control algorithm est une simple extension à celui du TCP standard. Pour TCP standard, il existe une fenêtre additionnelle fcg_{cnt} dont le rôle est de traquer le nombre des paquets acquittés depuis la dernière incémentation de fcg. On ne va incrémenter fcg que lorsque la relation suivant est vérifiée (Raiciu, *et al.,* 2011) :

$$\text{fcg}_{\text{cnt}} > \text{fcg} \tag{3.17}$$

Pour le coupled congestion control algorithm, $\text{fcg}_{\text{cnt\_i}}$ joue le même rôle comme expliqué ci-haut pour la SSi et sa fenêtre de congestion fcg_{i} s'incrémente de 1 quand la relation suivante est vraie :

$$fcg_{cnt\_i} > \max\left(\frac{\alpha_{sc} * fcg_{tot}}{\alpha}, fcg_i\right) \tag{3.18}$$

Le paramètre α_{sc} (alpha_scale) est utilisé pour assurer une précision dans le calcul de α. Il doit être de l'ordre de 2^n avec n un petit entier (typiquement $\alpha_{sc} = 512$) (Raiciu, *et al.,* 2011).

L'expression (3.11) est équivalent à l'expression (3.19) suivante :

$$\alpha = \alpha_{sc} * fcg_{tot} * \frac{fcg_{max}}{\left(\sum_{i=1}^{N} \frac{RTT_{max} * fcg_i}{RTT_i}\right)^2} \tag{3.19}$$

Ainsi, la relation (3.18) devient :

$$fcg_{cnt\_i} > \max\left(\frac{\left(\sum_{i=1}^{N}\frac{RTT_{max} * fcg_i}{RTT_i}\right)^2}{fcg_{max}}, fcg_i\right) \qquad (3.20)$$

3.3 Scheduler proposé

Le scheduler joue un rôle important tant à l'émission à travers la stratégie d'assignation des données aux différentes sous-sessions participantes à une connexion MPTCP qu'à la réception à travers le réassemblage (et l'ordonnancement si nécessaire) des données provenant des différentes sous-sessions participantes à une connexion MPTCP.

3.3.1 Principe de fonctionnement

Le principe de scheduler que nous proposons repose sur le schéma de principe d'un scheduler hybride qui est un scheduler disposant d'un buffer principale partagé par toutes les sous-sessions et des buffers individuels un pour chaque sous-session (voir figure 3.4).

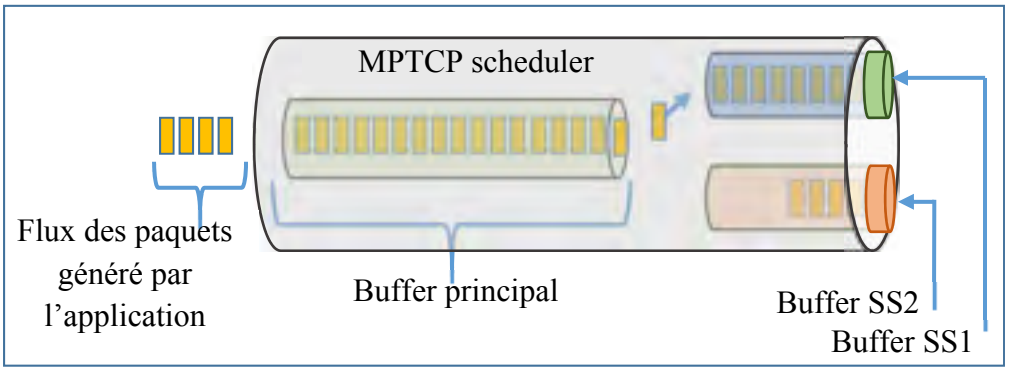

Figure 3.4 Schéma de principe du scheduler proposé

L'ensemble des données se trouvant dans le scheduler proposé se répartissent comme suit :

- les données délivrées par l'application qui sont bufférisées dans le buffer principal avant qu'elles soient assignées à une sous-session précise suivant la stratégie;

- les données déjà assignées à la SS1 qui sont toutes bufférisées dans le buffer de celle-ci. C'est l'ensemble des données assignées à SS1 pour la fenêtre globale en cour (fg_i) que sont les paquets émis en attente d'acquittement et les paquets en instance d'émission et les données assignées à SS1 pour la fenêtre globale suivante (fg_{i+1}) ;
- les données déjà assignées à la SS2 qui sont toutes bufférisées dans le buffer de celle-ci. C'est l'ensemble des données assignées à SS2 pour la fg_i que sont les paquets émis en attente d'acquittement et les paquets en instance d'émission et les données assignées à SS2 pour la fg_{i+1}.

3.3.1.1 Stratégie d'assignation des paquets aux sous-sessions

La stratégie de notre scheduler s'applique sur des données dont la taille est celle de la fenêtre globale. La stratégie de notre scheduler est telle que les paquets sont assignés aux sous-sessions de sorte que tous les paquets à l'intérieur d'une fenêtre globale, soient arrivés en ordre ou passent le minimum de temps hors ordre à la réception (voir figure 3.5). Sur cette figure, on distingue :

- la zone A qui est l'intervalle de temps d'émission des paquets via SS1 qui sont susceptibles d'être reçus avant le premier paquet émis via SS2 pour la même fenêtre globale, c'est-à-dire pendant $\Delta t1_{i+1}$ pour la fg_{i+1} ;
- la zone B qui est l'intervalle de temps d'émission des paquets via SS1 susceptibles d'être reçus après le premier paquet et avant le dernier paquet émis via SS2 pour la même fenêtre globale, c'est-à-dire pendant $\Delta t2_{i+1}$ pour la fg_{i+1}.

L'assignation des paquets par le scheduler ne se fait pas au rythme des ACK. Par exemple, pour une fg_{i+1} on estime sa taille après l'instant de la réception du dernier acquittement via SS2 pour le compte de la fg_{i-1} (t_{rack_i-1}) et on assigne le nombre des paquets estimés aux sous-sessions pendant la fenêtre temporelle de transmission (ftt) de fg_i (ftt_i) et avant l'instant d'envoi du dernier paquet via SS1 susceptible d'arriver à la réception avant le dernier paquet de la fg_i (t_{e_i}). Donc l'estimation et l'assignation des paquets d'une fenêtre globale se font pendant l'émission des paquets de la fenêtre globale à laquelle elle succède.

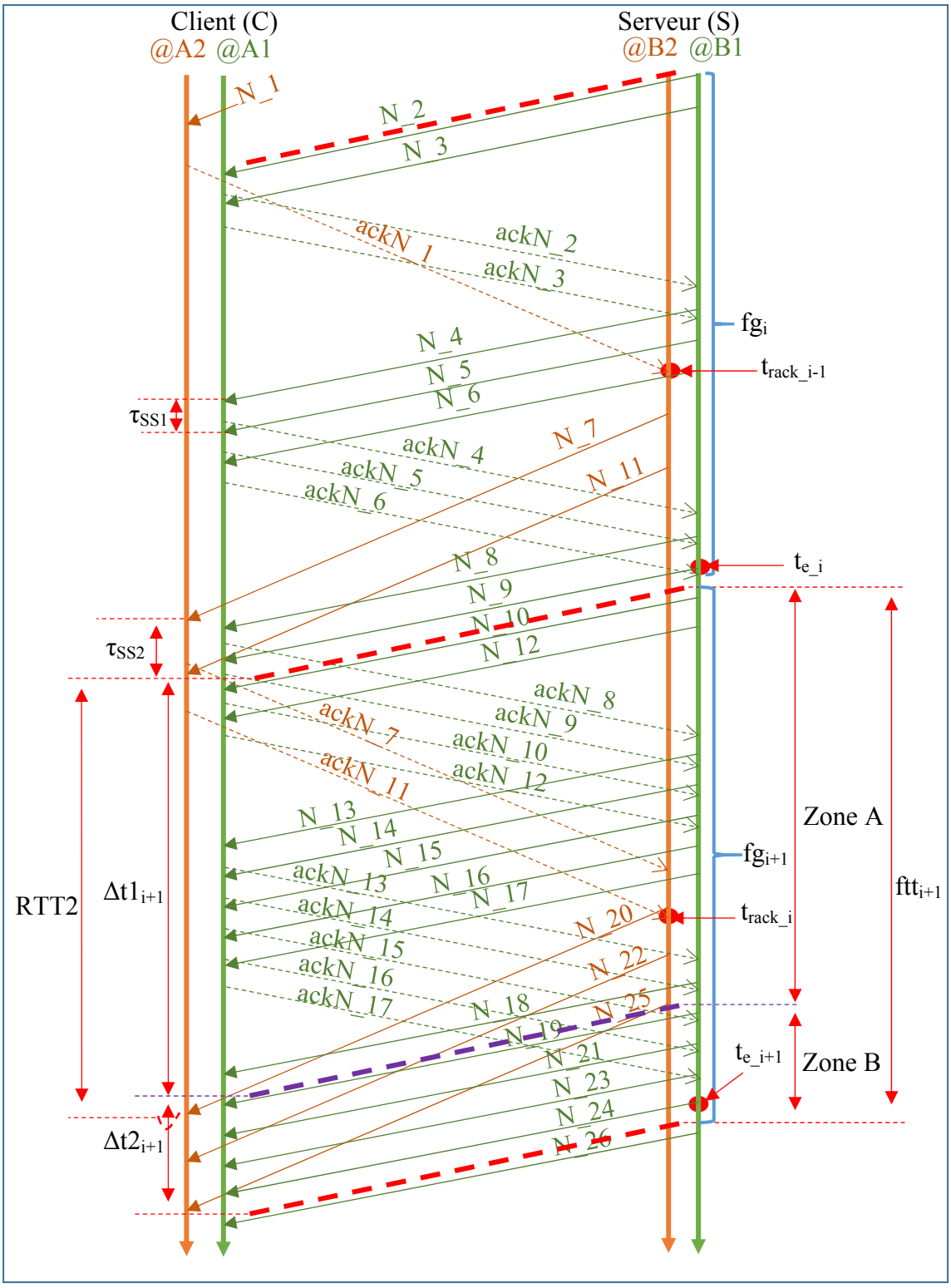

Figure 3.5 Principe de la stratégie d'assignation des paquets de notre schedule

Pour estimer la taille de fg_{i+1}, on commence par estimer les informations temporelles $\Delta t1_{i+1}$ et $\Delta t2_{i+1}$ qui dépendent de l'évolution de la fenêtre de congestion de SS2 (fcg_{SS2}). La valeur courante de la fcg_{SS2} (fcg_{SS2_i}) est connue à l'instant t_{rack_i-1}. Une fois les estimations de $\Delta t1_{i+1}$ et $\Delta t2_{i+1}$ obtenues, on peut estimer le nombre des paquets à assigner à SS1 (Np_SS1_{i+1}) qui dépend du nombre de paquets à émettre via SS1 dans la zone A (Np_zA_{i+1}) et dabs la zone B (Np_B_{i+1}).

On détermine $\Delta t1_{i+1}$ par la relation (3.21) et $\Delta t2_{i+1}$ par la relation (3.22) où fcg_{SS2_i} et fcg_{SS2_i+1} sont respectivement la taille courante de de fcg_{SS2} à l'instant t_{r_i-1} et la taille suivante de fcg_{SS2} suivant l'algorithme de contrôle de congestion et la phase dans laquelle se fait la transmission.

$$\Delta t1_{i+1} = RTT2 - (fcg_{SS2_i} - 1) * \tau_{SS2} \tag{3.21}$$

$$\Delta t2_{i+1} = (fcg_{SS2\_i+1} - 1) * \tau_{SS2} \tag{3.22}$$

3.3.1.2 Estimation de la taille de la fg_{i+1}

L'élargissement de la taille de fcg_{SS1} n'est limité que par l'algorithme de contrôle de congestion. La fenêtre de congestion via SS1 (fcg_{SS1}) n'est pas synchronisée à la ftt d'une fenêtre globale, donc la transmission d'une fcg_{SS1_j} peut se dérouler à cheval de ftt_i et de ftt_{i+1}. Comme la stratégie d'assignation s'applique à l'intérieur de la fenêtre globale, si fcg_{SS1_j} se trouve à cheval de ftt_i et de ftt_{i+1}, les séquences de ses paquets de à l'intérieur de la ftt_{i+1} sont supérieures à celles de tous les paquets se trouvant dans ftt_i. Tous les paquets d'une fg_{i+1} doivent être transmis pendant ftt_{i+1} qui est donnée par la relation (3.23).

$$ftt_{i+1} = \Delta t1_{i+1} + fcg_{SS2\_i+1} * \tau_{SS2} \tag{3.23}$$

La taille de la fg_{i+1} estimée est la somme des estimations de Np_SS2_{i+1} et de Np_SS1_{i+1}

L'estimation du nombre des paquets (Np_SS2_{i+1}) à assigner SS2 après l'instant t_{rack_i-1} revient à déterminer fcg_{SS2_i+1} (équation 3.24) suivant l'algorithme de contrôle de congestion car sa valeur courante fcg_{SS2_i} connue.

$$Np\_SS2_{i+1} = fcg_{SS2\_i+1} = fcg_{SS2\_i} + \Delta fcg_{SS2\_i+1} \tag{3.24}$$

La variation subie par fcg_{SS2} (Δfcg_{SS2_i}) pour passer de fcg_{SS2_i-1} à fcg_{SS2_i} est connue à l'instant t_{rack_i-1}. La variable Δfcg_{SS2_i+1} qui représente la variation que va subir fcg_{SS2} pour passer de sa valeur courante fcg_{SS2_i} à la suivante fcg_{SS2_i+1} n'est pas connue car à ce moment, aucun paquet de fcg_{SS2_i} n'est acquitté. La variation Δfcg_{SS2_i+1} est estimée sur hypothèse d'une évolution positive de (phase de démarrage lent ou phase d'évitement de congestion de la transmission) suivant l'algorithme de contrôle de congestion.

L'algorithme de contrôle de congestion utilisé pour les estimations est le Coupled Congestion Control algorithm décrit à la section 3.2.4.

Pour la phase de démarrage lent on a :

$$\Delta fcg_{SS2\_i+1} = \begin{cases} fcg_{SS2\_i} & si \quad fcg_{cnt\_SS2} = fcg_{SS2\_i} \\ 0 & si\ non \end{cases} \tag{3.25}$$

La variable fcg_{cnt_SS2} désigne le nombre des paquets acquittés via SS2 depuis la dernière incrémentation de fcg_{SS2}.

Pour la phase d'évitement de congestion on a :

$$\Delta fcg_{SS2\_i+1} = \begin{cases} 1 & si \quad fcg_{cnt\_SS2} > M1 \\ 0 & si\ non \end{cases} \tag{3.26}$$

En exploitant la relation (3.20), on détermine M1 par la relation (3.27). Les indices i et j ici respectivement la $i^{ième}$ et $j^{ième}$ fenêtres de congestion de la SS2 et de la SS1 depuis l'initialisation de la connexion. La différence d'indice est due au fait que fcg_{SS1} peut évoluer plusieurs fois durant une seule évolution de fcg_{SS2} selon l'écart entre RTT1 et RTT2.

$$M1 = max\left(\frac{\frac{RTT2 * fcg_{SS1\_j}}{RTT1} + fcg_{SS2\_i}}{\max(fcg_{SS2\_i}, fcg_{SS1\_j})}, fcg_{SS2\_i}\right) \tag{3.27}$$

Pour SS1, le nombre des paquets à lui assigner (Np_SS1_{i+1}) dépend non seulement de l'évolution de de fcg_{SS2_i} à fcg_{SS2_i+1} mais aussi de celle de fcg_{SS1} depuis sa valeur à l'instant t_{rack_i-1} jusqu'à sa valeur à l'instant d'émission du dernier packet via SS1 pendant ftt_i (t_{e_i+1}).

Comme pour Np_SS2_{i+1}, pour estimer Np_SS1_{i+1}, la variation Δfcg_{SS1_j+1} que subit fcg_{SS1} pour passer de fcg_{SS1_j} à fcg_{SS1_j+1} est estimée sur hypothèse d'une évolution positive (phase de démarrage lent ou phase d'évitement de congestion de la transmission).

Pour la phase de démarrage lent on a :

$$\Delta fcg_{SS1\_j+1} = \begin{cases} fcg_{SS1\_j} & si \quad fcg_{cnt\_SS1} = fcg_{SS2\_j} \\ 0 \; si \; non \end{cases} \tag{3.28}$$

La variable fcg_{cnt_SS1} désigne le nombre des paquets acquittés via SS1 depuis la dernière incrémentation de fcg_{SS1}.

Pour la phase d'évitement de congestion on a :

$$\Delta fcg_{SS1\_i+1} = \begin{cases} 1 & si \quad fcg_{cnt\_SS1} > M2 \\ 0 \; si \; non \end{cases} \tag{3.29}$$

On détermine $M2$ par la relation (3.30).

$$M2 = max\left(\frac{RTT2 * fcg_{SS1\_j}}{RTT1} + fcg_{SS2\_i}, fcg_{SS1\_i} \right) \tag{3.30}$$

Pour SS1, à l'instant t_{rack_i-1}, l'émetteur peut connaître les variables suivantes :

- de la $j^{ième}$ fcg_{SS1} courante (fcg_{SS1_j}) ;
- de la valeur courant de fcg_{cnt_SS1} ;
- du nombre des paquets non encore émis à l'intérieure de la fcg_{SS1_j} (Np_ne_j) ;
- du dernier instant d'émission via SS1 avant t_{rack_i-1} (t_{ec}) ;
- du nombre des paquets à l'intérieure de fg_i assignés à SS1 (Np_SS1_i);
- du nombre des paquets à l'intérieure de fg_i assignés à SS1 déjà acquittés (Np_ack_i).

Connaissant ces variables, on estime Np_SS_{i+1} par l'algorithme 3.1. Les estimations de Np_zA_{i+1} et Np_zB_{i+1} sont faites sur la base de l'évolution positive de fcg_{SS1} de sa valeur courante à l'instant t_{rack_i-1} à l'instant t_{e_i+1}. Pour les estimations de Np_zA_{i+1} et Np_zB_{i+1} voir respectivement Annexe I, Algorithme-A I-1 et Annexe I, Algorithme-A I-2.

Après l'estimation de Np_SS1_{i+1}, on déduit la taille de la fenêtre fg_{i+1} par :

$$fcg_{i+1} = Np\_SS1_{i+1} + Np\_SS2_{i+1} \quad (3.31)$$

Algorithme 3.1 Estimation de Np_SS1_{i+1}

Estimation de Np_SS1_{i+1}

Variables d'entrée : fcg_{SS1_j}, fcg_{SS1cnt}, t_{ec}, τ_{SS1}, RTT1, Np_SS1_i Np_ne_j, Np_ack_i, Np_unack_i, $\Delta t1_i$, $\Delta t2_i$, $\Delta t1_{i+1}$, $\Delta t2_{i+1}$.
Variable de sortie : NP_SS1_{i+1}.

1. Si le dernier acquittement via SS2 pour la fenêtre globale précédente reçu
2. Si $\Delta t1_{i+1} > \tau_{SS1}$
3. Bool = 1 ;
4. On appelle l'algorithme d'estimation de Np_zA_{i+1} ;
5. Si non
6. $Np\_zA_{i+1} = 0$;
7. Bool = 0 ;
8. Fin si ligne 2
9. On appelle l'algorithme d'estimation de Np_zB_{i+1} ;
10. $Np\_SS1_{i+1} = Np\_zA_{i+1} + Np\_zB_{i+1}$;
11. Si non
12. Ne rien faire;
13. Fin si ligne 1

3.3.1.3 Assignation des paquets aux sous-sessions

Une fois l'estimation de la taille pour la fg_{i+1} finie, le scheduler commence à assigner les paquets aux sous-sessions comme le décrit l'algorithme 3.2. Dans cet algorithme, on a :

- la variable Np_as comme compteur des paquets assignés ;
- n_1 comme compteur des paquets émis via SS1 dans la zone B ;
- n2 le compteur des paquets émis via SS2.

On définit aussi une horloge c_1 pour suivre l'évolution du temps de transmission via SS1 de la zone B et une autre horloge c_2 pour suivre l'évolution du temps de transmission via SS2.

Algorithme 3.2 Assignation des paquets à émettre à l'intérieure de fg_{i+1}

Assignation des paquets à émettre à l'intérieure de fg_{i+1}

Variables d'entrée : fg_{i+1}, Np_SS2_{i+1}, Np_zA_{i+1}, Np_zB_{i+1}, τ_{SS1}, τ_{SS2}.
Variable de sortie : pas de variable de sortie.

1. Initialisation : Np_as = 0 ; n1 = 0 ; n2 = 0 ; c1 = $\tau_{SS1}/2$; c2 = τ_{SS2} ;
2. Tant que Np_as < Np_zA_{i+1}
3. Assigner un paquet à SS1 ;
4. Incrémenter de Np_as de 1;
5. Fin tant que ligne 2
6. Assigner un paquet à SS2 ;
7. Assigner un paquet à SS1 ;
8. Incrémenter de Np_as de 1 ;
9. Incrémenter n2 de 1;
10. Incrémenter n1 de1 ;
11. c2 =c2 + τ_{SS2} ;
12. Tant que n2 < Np_SS2_{i+1}
13. Tant que c1 < c2
14. Si n1 < Np_zB_{i+1}
15. Assigner un paquet à SS1 ;
16. Incrémenter Np_as de 1 ;
17. Incrémenter n1 de 1 ;
18. c1 =c1 + τ_{SS1} ;
19. Si non
20. n1 = Np_SS2_{i+1} ;
21. c1 = c2 + n1* τ_{SS1} ; pour arrêter d'allouer des paquets à SS1 ;
22. Fin si ligne 14
23. Fin tant que ligne 13
24. Assigner un paquet à SS2 ;
25. Incrémenter de Np_as de 1 ;
26. Incrémenter n2 de 1;
27. c2 =c2 + τ_{SS2} ;
28. Fin tant que ligne 12

3.3.2 Principe d'émission-réception

A la différence de la méthode standard illustrée à la figure 3.6, la seule particularité pour notre proposition est que nous préconisons en cas de perte de segment via SS2, qu'un ack

spécial soit envoyé via SS1 (pointillés jaunes figure 3.7) par le récepteur pour en informer l'émetteur.

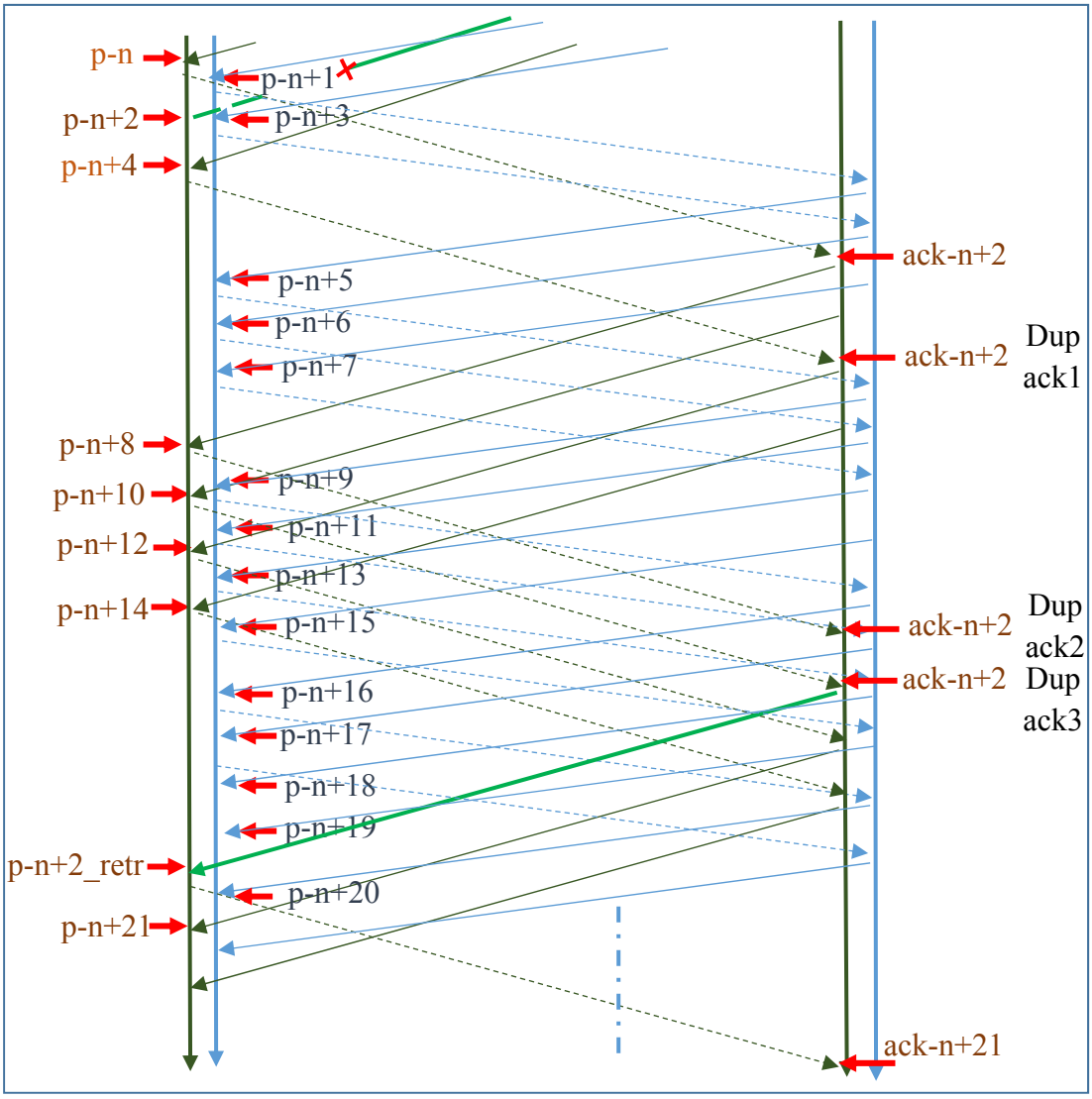

Figure 3.6 Scénario de la réémission méthode standard

On voit bien qu'avec la méthode standard, la réémission de p-n+2 ne doit intervenir qu'après le troisième acquittement dupliqué du paquet p-n+2 (Dup ack3) via SS2. De ce fait, les paquets de séquences comprises entre p-n+3 et p-n+19 doivent être bufférisés avant la réémission de p-n+2 (rtr_p-n+2 ligne verte figure 3.6) soit 17 paquets.

Pour notre proposition, on peut insérer dans le segment de l'ack spécial, la séquence niveau données du paquet perdu en utilisant l'option DSS du TCP. Une fois cet ack spécial reçu par l'émetteur, grâce à l'option DSS, le paquet en question est réafecté à la SS1 où il devient le paquet en instance d'envoi. Au niveau SS2, la séquence niveau session du paquet perdu est réafectée au paquet en instance d'envoi (paquet p-n+8 figure 3.7), de telle sorte que l'émission de ce dernier soit vue au niveau SS2 comme étant la réponse à la requête de retransmission. La fcg_{SS2} est réduite de moitié.

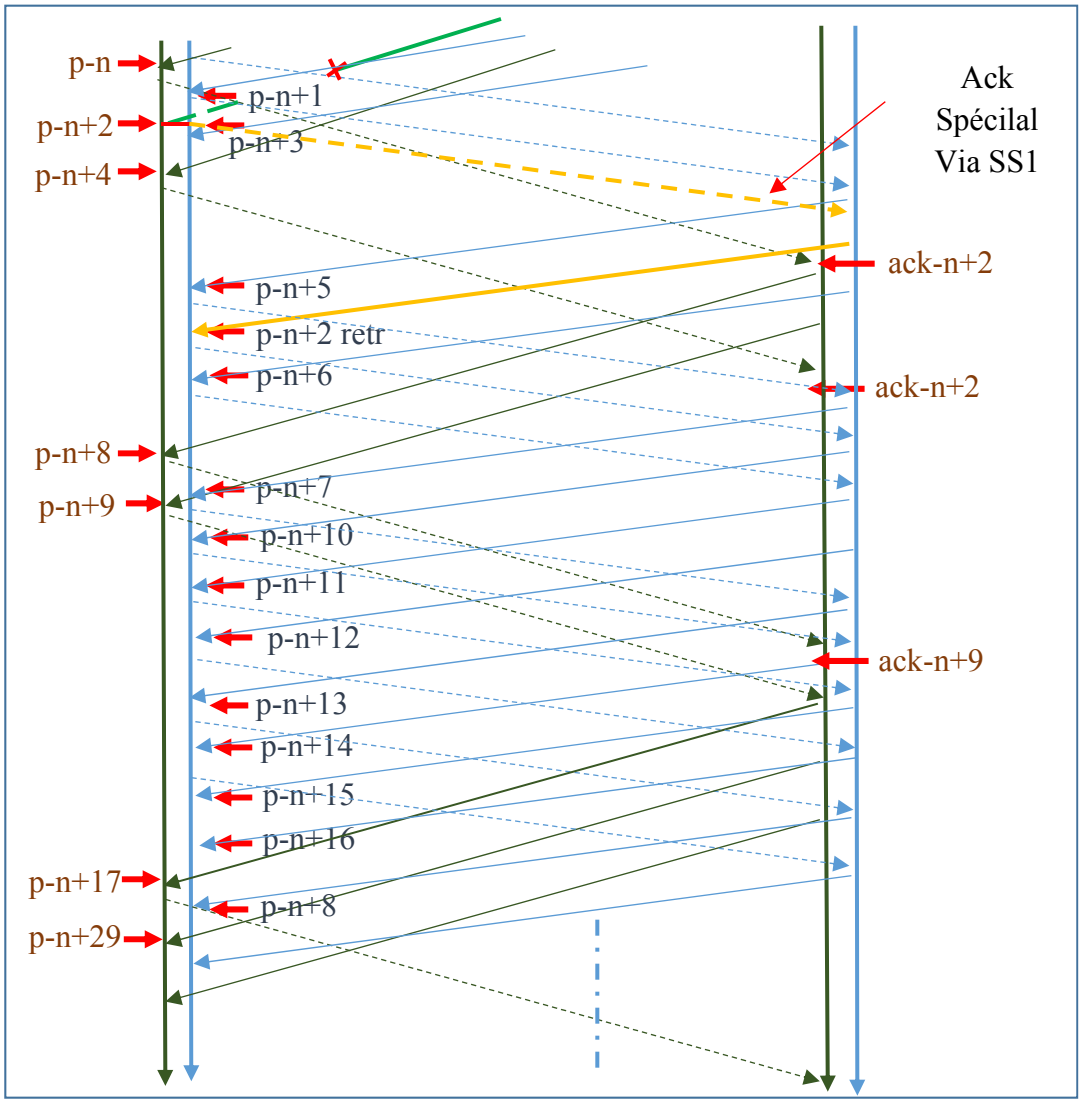

Figure 3.7 Scénario de réémission de paquet perdu (méthode proposée)

Avec la méthode que nous proposons, seuls les paquets de séquences comprises entre p-n+3 et p-n+5 doivent être bufférisés avant la réémission de p-n+2 (rtr_p-n+2 ligne jaune figure 3.7). Le paquet p-n+7 étant déjà assigné à la SS1, la retransmission de p-n+2 via SS1 décale son l'instant d'émission entrainant ainsi l'arrivé hors ordre de p-n+8 via SS2.

Il faut noter que les numéros des séquences des paquets au niveau des figures 3.6 et 3.7 représentent les séquences niveau données. Comme les paquets doivent être reçus en ordres, on remarque que leur progression est suivant l'ordre de réception. Ce sont les couleurs qui font distinguer la sous-session empruntée par les paquets. Nous voulons tout juste illustrer l'avantage de l'acquittement spécial par rapport la méthode standard. Ce mécanisme, permet de minimiser le nombre des paquets hors ordre (voir les figures 3.6 et 3.7) dû à la perte d'un segment

3.4 Conclusion

Ce chapitre est le lieu de l'étude théorique de notre mémoire. Dans ce chapitre nous avons fait l'étude du fonctionnement de scheduler que nous proposons. Nous y avons expliqué à travers les algorithmes d'estimation et d'assignation des paquets comment notre scheduler fonctionne. Nous avons aussi expliqué le mécanisme par lequel la retransmission d'un paquet perdu via une sous session lente se fait via la sous-session la plus rapide.

Notre proposition va contribuer à minimiser les nombre des paquets reçus hors ordre avec comme résultat, l'amélioration des performances du MPTCP. Les résultats de simulation de notre proposition sont étudiés au chapitre 4.

CHAPITRE 4

SIMULATIONS : PRESENTATION ET ANALYSES DES RESULTATS

4.1 Introduction

L'objectif de notre travail de recherche est de déterminer un scheduler avec une stratégie d'assignation des paquets aux sous-sessions utilisées dans une connexion MPTCP de sorte que les performances de la dite connexion en termes de débit soient améliorées. C'est ainsi que nous avons proposé un scheduler dont le principe de fonctionnement et la stratégie qu'il met en œuvre sont expliqués à la section 3.3.

Dans ce chapitre, nous présentons les résultats de simulation de notre travail ainsi que leurs analyses afin de vérifier leur cohérence et la validité des performances théoriques du scheduler proposé. C'est le weighted round robin (WRR) scheduler implémenté dans linux pour une connexion MPTCP qui constitue notre base référentielle pour l'analyse des résultats de notre travail.

4.2 Outil de simulation

Matlab qui veut dire « matrix laboratory » est un HPLTC (High-Performance Language for Technical Computing) sous licence de MathWorks, Inc. Il intègre les calculs, visualisations et programmations. Matlab est un système interactif dont le vecteur est l'élément de base ce qui permet de résoudre beaucoup des problèmes des calculs techniques en particulier ceux qui peuvent être modélisés sous formes vectorielles et matricielles.

MATLAB a évolué sur une période de plusieurs années avec la participation de nombreux utilisateurs. Il est au niveau des universités, un outil pédagogique standard pour des cours d'initiation et de perfectionnement en mathématiques, en ingénierie et en science. MATLAB dispose d'une gamme de solutions spécifiques aux applications appelées « toolboxes ». Les toolboxes sont des fonctions Matlab (M-Files) qui étendent l'environnement Matlab pour résoudre des catégories particulières de problèmes.

Certaines problématiques de la transmission des données dans le réseau de télécommunication peuvent être modélisées sous forme vectorielle. Le problème de notre recherche en est un et peut ainsi bien être modélisé et simulé à l'aide de Matlab.

4.3 Méthodologie

4.3.1 Description du système

Le système que nous avons simulé est défini par une connexion MPTCP entre deux équipements terminaux (fonctionnant en client et serveur) à travers deux sous-sessions TCP SS1 et SS2 dont les taux de transmission et les RTT sont asymétriques. On suppose une connexion sans pertes. La taille des segments est constante et est identique pour toutes les deux sessions, elle est à la taille maximale du segment (MSS).

La SS1 est caractérisée par les paramètres suivants:

- un RTT (RTT1) supposé constant ;
- une fenêtre de congestion fcg1 évoluant suivant l'algorithme de contrôle congestion;
- un taux de transmission B1 (en octets/ms) considéré identique à la réception.

La SS2 est caractérisée par les paramètres suivants:

- un RTT (RTT2) supposé constant ;
- une fenêtre de congestion fcg2 évoluant suivant l'algorithme de contrôle congestion;
- un taux de transmission B2 (en octets/ms) considéré identique à la réception.

L'algorithme de contrôle congestion est le Coupled Congestion Control algorithm (voir section 3.2.4).

Il faut noter que B1 est supérieur à B2 et RTT2 est supérieur à RTT1. La connexion est initiée par SS1.

4.3.2 Adaptation du système à l'environnement Matlab

Les paramètres sur lesquels doivent se concentrer notre analyse sont le débit et le délai de livraison qui dépendent du RTT et du taux de la transmission B de la sous-session. Avec Matlab, le débit peut être représenté par un tableau dont la taille doit évoluer en fonction du temps. Le contenu d'une cellule i de ce tableau est un nombre entier d'octets qui dépend de la valeur temporelle (VT[1]) de la dite cellule, du taux de réception de la sous-session et de sa position temporelle. La valeur temporelle du tableau qui est la somme des VT de toutes ses cellules le délai de livraison des données reçues.

La position temporelle d'une cellule d'un tableau se détermine par rapport à deux types d'intervalles de temps (voir figure 4.1).

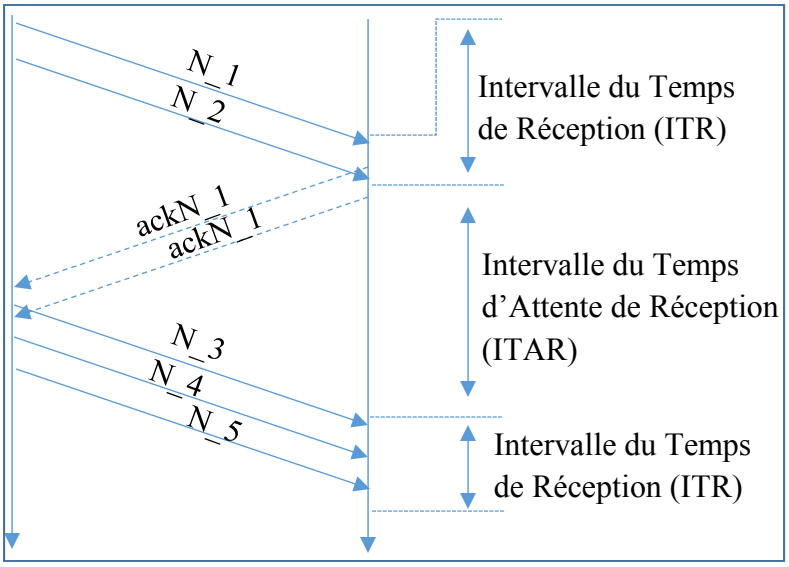

Figure 4.1 Les deux types d'intervalles de temps

Une cellule de tableau peut se trouver dans l'une des positions temporelles décrites à la figure 4.2 suivante.

[1] Une VT, est un intervalle de temps représentatif d'une cellule d'un tableau

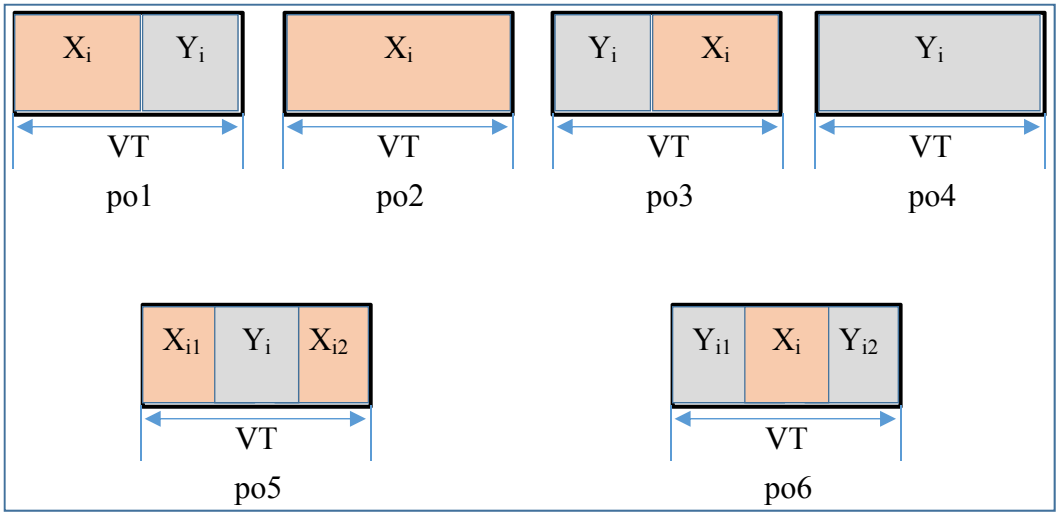

Figure 4.2 Différentes positions temporelle d'une cellule i d'un tableau

- position po1 : la VT de la cellule est divisée en deux parties. La première partie coïncide avec l'ITR de X_i octets. La deuxième partie coïncide avec un ITAR durant lequel Y_i octets pourraient être reçus ;
- position po2 : toute la VT de la cellule coïncide avec l'ITR de X_i octets ;
- position po3 : la VT de la cellule est divisée en deux parties. La première partie coïncide avec un ITAR durant lequel Y_i octets pourraient être reçus. La deuxième coïncide avec l'ITR de X_i octets ;
- position po4 : toute la VT de la cellule coïncide avec un ITAR pendant lequel Y_i octets pourraient être reçus;
- position po5 : la VT de la cellule est divisée en trois parties. La première partie coïncide avec l'ITR de X_{i1} octets. La deuxième partie coïncide avec un ITAR pendant lequel Y_i octets pourraient être reçus. La dernière partie coïncide avec l'ITR de X_{i2} octets ;
- position po6 : la VT de la cellule est divisée en trois parties. La première partie coïncide avec un ITAR pendant lequel Y_i octets pourraient être reçus. La deuxième coïncide avec l'ITR de X_i octets. La dernière partie coïncide avec un ITAR pendant lequel Y_{i2} octets pourraient être reçus.

Le contenu en octets de la cellule i de tableau est X_i pour les positions po1, po2, po3 et po6. Il est $X_{i1}+X_{i2}$ pour la situation po5 et 0 octet pour la situation po4.

Exemple : une session TCP de RTT fixe et égal à 15 ms est établie comme l'indique la figure 4.3. Le MSS de cette connexion est 2048 octets et son taux de réception B de 1000 octets/ms. La réception a lieu entre les instants t_0 et t_1. La VT choisie d'une cellule du tableau représentant le débit en fonction de temps à la réception est 5 ms.

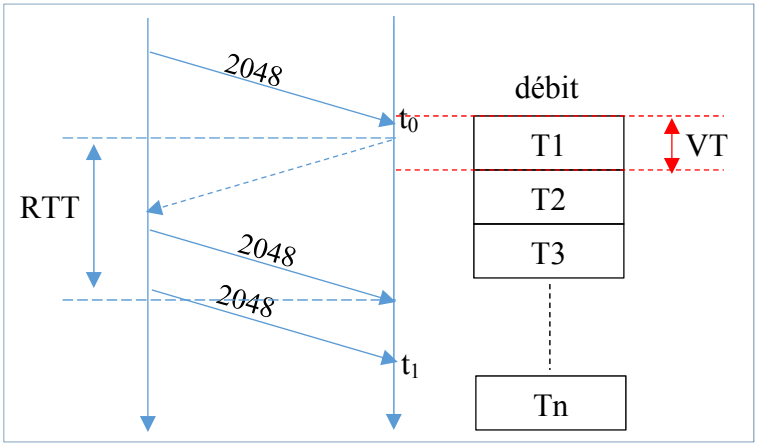

Figure 4.3 Exemple d'adaptation à l'environnement Matlab d'une session TCP

A l'instant t_0, la fcg est égale à 1, donc le premier ITR est :

$$ITR_1 = fcg * MSS/B = 1 * MSS/B = 2{,}048\ ms \quad (4.1)$$

Le premier ITAR ($ITAR_1$) est égal à RTT soit 15 ms. L'ITR_1 étant inférieure à VT et l'$ITAR_1$ étant au-delà de VT du T_1, donc T_1 se trouve à la position po1. L'ITR de X_1 (ITR_X_1) est ainsi égal à ITR_1. On a donc

$$ITR\_X_1 = ITR_1 \Rightarrow X_1 = ITR\_X_1 * B = 2048\ octets \quad (4.2)$$

On en déduit l'ITAR de Y_1 ($ITAR\_Y_1$) par :

$$ITAR\_Y_1 = VT - ITR\_X_1 = 2{,}952\ ms \quad (4.3)$$

Au-delà de la VT de T_1, l'$ITAR_1$ restant devient :

$$ITAR_1 = ITAR_1 - ITAR\_Y_1 = 12{,}048\ ms \tag{4.4}$$

L'$ITAR_1$ restant est supérieur à 2*VT, T_2 et T_3 se trouvent ainsi à la position po4, et l'ITAR de Y_2+Y_3 ($ITAR\_Y_2\_Y_3$) est égal à 2*VT et l'$ITAR_1$ restant devient :

$$ITAR_1 = ITAR_1 - ITAR\_Y_2\_Y_3 = 2{,}048\ ms \tag{4.5}$$

Ainsi on a :

$$X_2 = X_3 = 0 \tag{4.6}$$

Au-delà du T_3, l'$ITAR_1$ restant étant inférieur à VT, T_4 se trouve à la position po3 et $ITAR\_Y_4$ est égal à l'$ITAR_1$ restant. L'ITR_X_4 est :

$$ITR\_X_4 = VT - ITAR\_Y_4 = 2{,}952\ ms \Rightarrow X_4 = ITR\_X_4 * B = 2952\ octets \tag{4.7}$$

Le deuxième ITR (ITR_2) est :

$$ITR_2 = fcg * MSS/B = 2 * MSS/B = 4{,}096\ ms \tag{4.8}$$

Au-delà du T_4, ITR_2 restant devient :

$$ITR_2 = ITR_2 - ITR\_X_4 = 1144\ octets \tag{4.9}$$

ITR_2 restant étant inférieur à VT et que c'est la fin de réception, T_5 se trouve à la position po1. L'ITR de X_5 (ITR_X_5) est ainsi égal à ITR_2 restant. On a donc :

$$ITR\_X_5 = ITR_2 \Rightarrow X_5 = 1144\ octets \tag{4.10}$$

On obtient ainsi le tableau comme l'indique la figure 4.4. Il en résulte le débit moyen suivant :

$$débit\ moyen = \frac{\sum_{i=1}^{5} T_i}{5 * VT} = 245{,}76\ octets/5ms \tag{4.11}$$

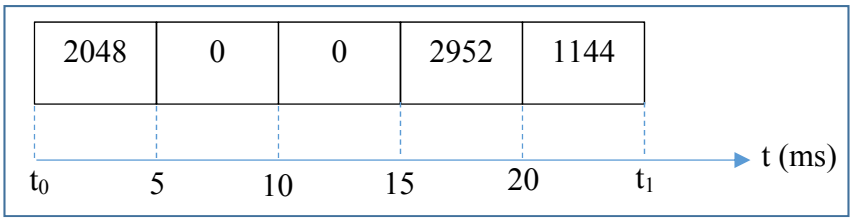

Figure 4.4 Tableau représentant le débit de transmission illustrée par le chronogramme de la figure 4.3

Pour une connexion MPTCP, on a plusieurs sous-sessions fonctionnant en parallèle et il doit y avoir autant des tableaux pour la réception. Cependant, la mesure de débit au niveau des sous-sessions ne donne aucune information sur le débit d'octets qui passent à la couche application. La raison est qu'un paquet reçu hors ordre via une sous-session doit attendre tous les paquets (venant souvent via une autre sous-session) qui doivent le précéder avant de passer à la couche application. Donc, c'est le débit moyen d'octets qui passent à la couche application qui permet de mesurer la performance de la stratégie d'assignation des paquets aux sous-sessions à l'émission car il dépend aussi du temps d'attente de de réordonnancement.

Exemple : Une connexion MPTCP est établie à l'aide de deux sous-sessions TCP comme l'indique la figure 4.5. L'assignation de paquet se fait selon le principe de WRR. La SS1 représentée en bleu a pour RTT 15 ms et un taux de réception B1 de 1000 octets/ms. La SS2 représentée en marron a pour RTT 25 ms et un taux de réception de 250 octets/ms. Toutes les deux sous-sessions utilisent un MSS de 2048 octets. L'intervalle de temps de réception à considérer commence à t_0 et finit à t_1 et à l'instant t_0, il reste 24 ms dans RTT de SS2. Les deux tableaux ont la même VT par cellule.

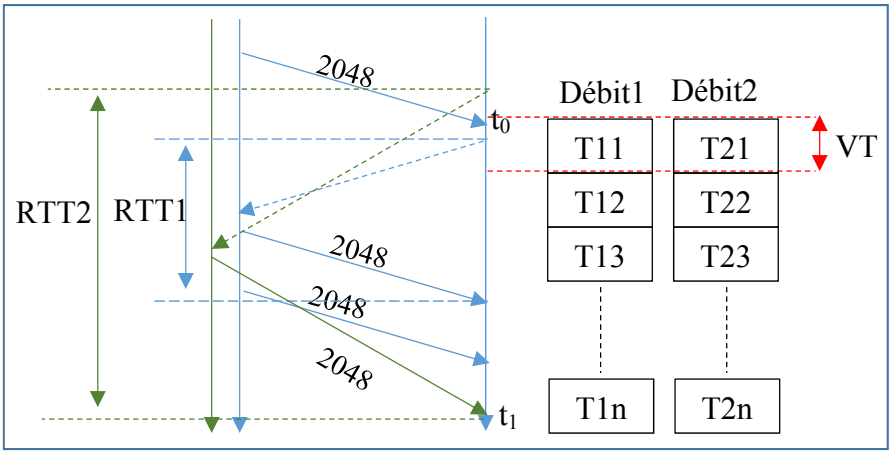

Figure 4.5 Exemple d'adaptation à l'environnement Matlab d'une connexion MPTCP

Pour cet exemple, pour indiquer les positions temporelles des cellules de chacun de deux tableaux, pour SS1 et SS2, X_i devient respectivement X_{1i} et X_{2i} et Y_i devient respectivement Y_{1i} et Y_{2i}. La VT de T_{1i} et en même temps celle de T_{2i}.

A l'instant t_0, la fcg1 est égale à 1, donc le premier ITR de SS1 (ITR_{11}) est :

$$ITR_{11} = fcg1 * MSS/B1 = 1 * MSS/B1 = 2{,}048\ ms \qquad (4.12)$$

Le début de réception coïncide avec le premier ITAR de SS2 ($ITAR_{21}$). Il est égal à 24 ms.

Le premier ITAR de SS1 ($ITAR_{11}$) est égal à RTT soit 15 ms. L'ITR_{11} étant inférieur à VT et l'$ITAR_{11}$ étant au-delà de VT du T_{11}, donc T_{11} se trouve à la position po1. L'ITR de X_{11} (ITR_X_{11}) est ainsi égal à ITR_{11}. On a donc

$$ITR\_X_{11} = ITR_{11} \Rightarrow X_{11} = ITR\_X_{11} * B1 = 2048\ octets \qquad (4.13)$$

T_{21} se trouve quant à lui à la position po4 par conséquent on a :

$$X_{21} = 0\ octet\ et\ ITAR\_Y_{21} = 5\ ms \qquad (4.14)$$

On déduit l'ITAR de Y_{11} ($ITAR\_Y_{11}$) par :

$$ITAR\_Y_{11} = VT - ITR\_X_{11} = 2{,}952\ ms \qquad (4.15)$$

Au-delà de la VT de T_{11}, l'$ITAR_{11}$ et l'$ITAR_{21}$ restants deviennent :

$$ITAR_{11} = ITAR_{11} - ITAR\_Y_{11} = 12{,}048\ ms \quad (4.16)$$

$$ITAR_{21} = ITAR_{21} - ITAR\_Y_{21} = 19\ ms \quad (4.17)$$

L'$ITAR_{11}$ et l'$ITAR_{21}$ restants sont supérieurs à 2*VT, T_{12} et T_{13} du tableau1 et T_{22} et T_{23} du tableau2 se trouvent ainsi à la position po4, et l'ITAR de Y_{12}+Y_{13} (ITAR_Y_{12}_Y_{13}) et l'ITAR de Y_{22}+Y_{23} (ITAR_Y_{22}_Y_{23}) sont égaux à 2*VT et l'$ITAR_{11}$ et l'ITAR21 restants deviennent :

$$ITAR_{11} = ITAR_{11} - ITAR\_Y_{12}\_Y_{13} = 2{,}048\ ms \quad (4.18)$$

$$ITAR_{21} = ITAR_{21} - ITAR\_Y_{22}\_Y_{23} = 9\ ms \quad (4.19)$$

On a ainsi :

$$X_{12} = X_{13} = X_{22} = X_{23} = 0 \quad (4.20)$$

Au-delà du T_{13}, l'$ITAR_{11}$ restant est inférieur à VT de T_{14}. Le deuxième ITR de SS1 (ITR_{12}) débute donc dans la VT de T_{14}. Si le premier segment de l'ITR_{12} est reçu en ordre, le segment suivant ne l'est pas. Il n'est reçu à la couche application qu'après la réception du segment de SS2 qui le précède à l'émission. Cette attente induit le deuxième ITAR de SS1 ($ITAR_{12}$). Si ITR_X_{14} est relatif à la réception du premier segment d'ITR_{12}, on a :

$$X_{14} = MSS = 2048\ octets \Rightarrow ITR\_X_{14} = MSS/B1 = 2{,}048\ ms \quad (4.21)$$

$$ITR\_X_{14} + ITAR_{11} = 4{,}096\ ms \quad (4.22)$$

La somme de l'$ITAR_{11}$ restant et l'ITR_X_{14} est inférieure à VT, donc T_{14} se trouve à la position po6. Ainsi on a :

$$ITAR\_Y_{141} = ITAR_{11} = 2{,}048\ ms \quad (4.23)$$

L'$ITAR_{12}$ est donnée par :

$$ITAR_{12} = ITAR_{21} - (ITAR\_Y_{141} + ITR\_X_{14}) + ITR_{22} \quad (4.24)$$

ITR_{22} est relatif à la réception du segment via SS2 à la fin de l'$ITAR_{21}$. On a donc :

$$ITR_{22} = MSS/B2 = 8{,}192\ ms \tag{4.25}$$

$$ITAR_{12} = 13{,}096\ ms \tag{4.26}$$

L'$ITAR_{12}$ part au-delà de la VT de T_{14}, donc, on a :

$$ITAR\_Y_{142} = VT - (ITAR\_Y_{141} + ITR\_X_{14}) = 0{,}904\ ms \tag{4.27}$$

L'$ITAR_{12}$ et l'$ITAR_{21}$ restant après T_{14} sont :

$$ITAR_{12} = ITAR_{12} - ITAR\_Y_{142} = 12{,}192\ ms \tag{4.28}$$

$$ITAR_{21} = ITAR_{21} - VT = 4\ ms\ et\ X_{24} = 0 \tag{4.29}$$

L'$ITAR_{21}$ restant est inférieure à la VT de T_{25} et on a :

$$ITAR\_Y_{25} = ITAR_{21} = 4\ ms \tag{4.30}$$

L'ITR_{22} commence dans la VT de T_{25} va au-delà de celle-ci. T_{25} se trouve ainsi à la position po3 et on a :

$$ITR\_X_{25} = VT - ITAR\_Y_{25} = 1\ ms \tag{4.31}$$

$$\Rightarrow X_{25} = ITR\_X_{25} * B2 = 250\ octets \tag{4.32}$$

L'$ITAR_{12}$ restant après T_{14} est supérieur à la VT de T_{15}. L'$ITAR_{12}$ restant après T_{15} est :

$$ITAR_{12} = ITAR_{12} - VT = 7{,}192\ ms\ et\ X_{15} = 0\ octets \tag{4.33}$$

L'ITR_{22} restant après T_{25} est :

$$ITR_{22} = ITR_{22} - ITR\_X_{25} = 7{,}192\ ms \tag{4.34}$$

L'ITR_{22} et l'$ITAR_{12}$ restants au-delà de T_{15} sont supérieurs à la VT de T_{16}, donc ils deviennent après T_{16} :

$$ITR_{22} = ITR_{22} - VT = 2{,}192\ ms\ et\ X_{26} = VT * B2 = 1250\ octets \tag{4.35}$$

$$ITAR_{12} = ITAR_{12} - VT = 2{,}192\ ms\ et\ X_{16} = 0\ octet \tag{4.36}$$

Au-delà de T_{26}, l'ITR_{22} restant est inférieur à la VT de T_{27}, et c'est la fin de la réception via SS2, donc T_{27} se trouve à la position po1. Ainsi on a :

$$ITR\_X_{27} = ITR_{22} \Rightarrow X_{27} = ITR\_X_{27} * B2 = 548\ octets \tag{4.37}$$

Au-delà de T_{16}, l'$ITAR_{12}$ restant est inférieur à la VT de T_{17} mais le deuxième reçu pendant l'ITR_{12} retrouve son ordre.

$$ITAR\_Y_{17} = ITAR_{12} \Rightarrow VT - ITAR\_Y_{17} = 2{,}808\ ms \tag{4.38}$$

$$X_{17} = MSS \Rightarrow ITR\_X_{17} = MSS/B1 = 2{,}048\ octets \tag{4.39}$$

La différence entre la VT de T_{17} et l'$ITAR\_Y_{17}$ est supérieure à ITR_X_{17}, mais comme c'est la fin de réception via SS1, T_{17} se trouve à po3.

On obtient ainsi les tableaux comme l'indique la figure 4.6. Il en résulte les débits moyens pour la SS1 (4.40) et pour la SS2 (4.41).

Pour SS1 on a :

$$débit\ moyen = \frac{\sum_{i=1}^{7} T_{1i}}{7 * VT} = 175{,}54\ octets/5ms \tag{4.40}$$

Pour SS2 on a :

$$débit\ moyen = \frac{\sum_{i=1}^{7} T_{2i}}{7 * VT} = 58{,}51\ octets/5ms \tag{4.41}$$

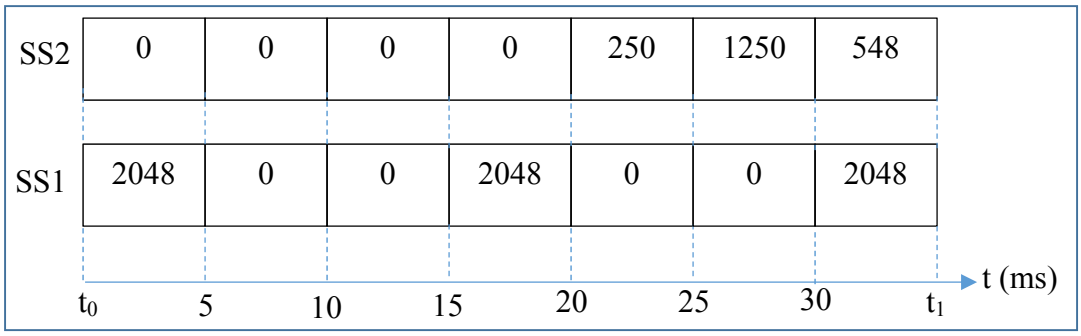

Figure 4.6 Tableau représentant le débit de transmission illustrée par le chronogramme de la figure 4.5

4.3.3 Paramètres de simulation

Les paramètres de simulation sont :

- le RTT1 qui est le RTT de la SS1 ;
- le RTT2 qui est le RTT de la SS2 ;
- le taux de réception B1 (1000 octets/ms) de la SS1 ;
- le taux de réception B2 (250 octets/ms) de la SS2 ;
- la fcg1 qui est la fenêtre de congestion de la SS1 évoluant avec le temps suivant l'algorithme de contrôle de congestion ;
- la fcg2 qui est la fenêtre de congestion de la SS2 évoluant avec le temps suivant l'algorithme de contrôle de congestion ;
- la taille du MSS (1024 octets) identique pour les deux sous-sessions ;
- la VT (5 ms) identique pour les tableaux représentant les débits au niveau de chacune des sous-sessions.

Remarques : Les valeurs numériques de RTT1 et RTT2 sont fixées suivant le scénario simulé.

4.4 Analyses des résultats de simulations

Le paramètre sur lequel nous nous basons pour analyser et comparer les résultats de différents scénarios est le débit moyen en fonction du temps. Avec ce paramètre, nous pouvons apprécier les améliorations apportées par notre travail.

Le débit moyen au niveau application via une sous-session i (Dmoy_i) à l'instant t_0 est donné par :

$$Dmoy_i = \frac{Nombre\ total\ d'octets\ reçus\ au\ niveau\ application}{t_0} \qquad (4.42)$$

Le nombre total d'octets reçus au niveau application est la somme de tous les octets reçus en ordre ou réordonnées et ensuite reçus via la sous-session de début de la réception jusqu'à l'instant t_0. Le nombre total d'octets reçus pour une sous-session à l'instant t_0 dépend des ITR et des ITAR rencontrés pendant la réception (revoir la figure 4.1).

Le débit moyen pour une connexion MPTCP, est la somme des débits moyen au niveau application de toutes les sous-sessions.

Il est important de noter que le débit moyen pour une connexion MPTCP au niveau des sous-sessions ne donne aucune information sur la stratégie d'assignation à l'émission pour la simple raison qu'il ne tient pas en compte des octets hors non encore reçus à l'instant de la mesure.

Les rythmes de réception n'étant pas synchronisés au niveau des sous-sessions, les ITR et ITAR à ce niveau ne peuvent dépendre que de l'algorithme de contrôle de congestion. Mais au niveau application, les ITR et les ITAR dépendent non seulement de l'algorithme de contrôle de congestion mais aussi de l'ordre d'arrivée via les deux sous-sessions comme on le voit à la figure 4.7 où :

- l'ITR_{ij_s} est le $j^{ième}$ ITR au niveau de la sous-session i ;
- l'ITR_{ij_a} est le $j^{ième}$ ITR au niveau application relatif à la sous-session i ;
- l'$ITAR_{ij\_s}$ est le $j^{ième}$ ITAR au niveau de la sous-session i ;
- l'$ITAR_{ij\_a}$ est le $j^{ième}$ ITAR au niveau application relatif à la sous-session i.

La position temporelle d'un ITR_{ij_a} pendant la réception est influencée par l'ordre des données qui y sont reçues et de ce fait, la largeur d'un $ITAR_{ij\_a}$ pendant la réception est influencée par l'ordre des prochaines données à recevoir au niveau de la sous-session i.

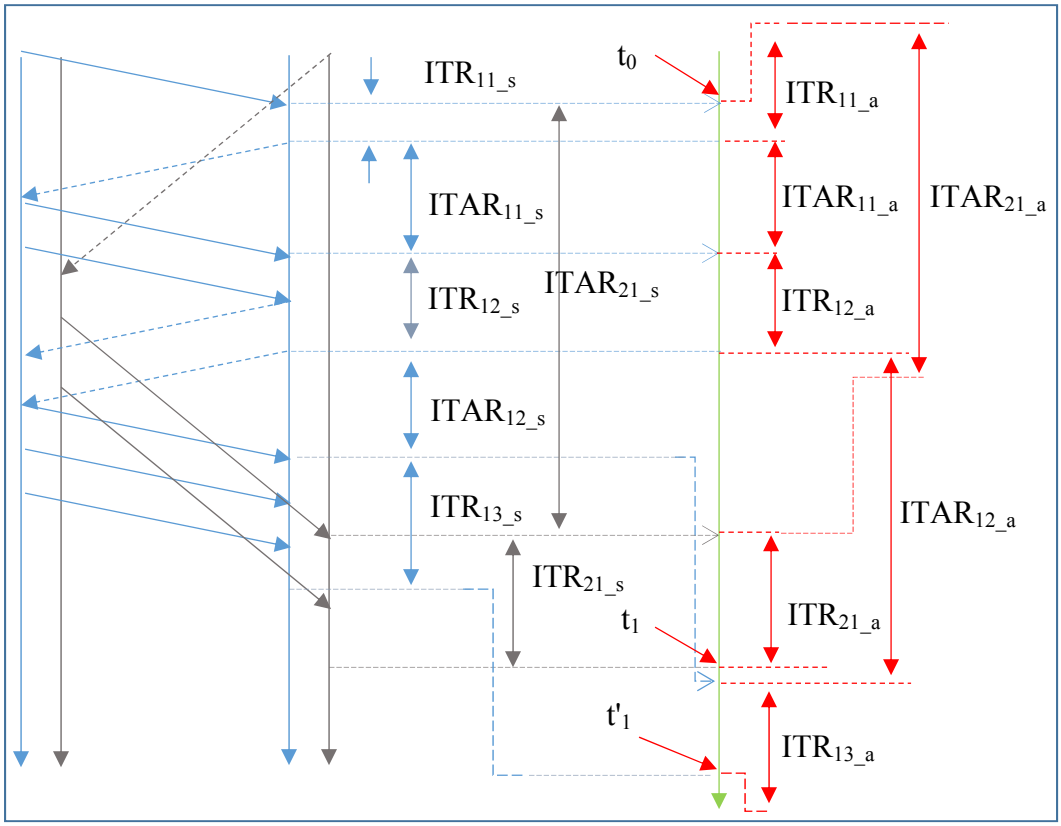

Figure 4.7 Illustration de l'influence des ITAR sur le débit au niveau application pour chacune des sous-sessions (cas WRR)

Si la réception a lieu entre t_0 et t_1 au niveau sous-session, à cause des paquets reçus hors ordre, la réception a lieu entre t_0 et t'_1 mais pour un même nombre des octets reçus. De ce fait, à partir du débit moyen, nous sommes en mesure de juger de la performance de notre scheduler non seulement en termes de débit mais aussi en termes du délai de livraison si nous définissons celui-ci comme étant le temps mis pour recevoir tous les paquets envoyés par l'émetteur avec une connexion ininterrompue. Il y a aussi des ITAR pour notre proposition, tels qu'ils sont illustrés par la figure 4.8.

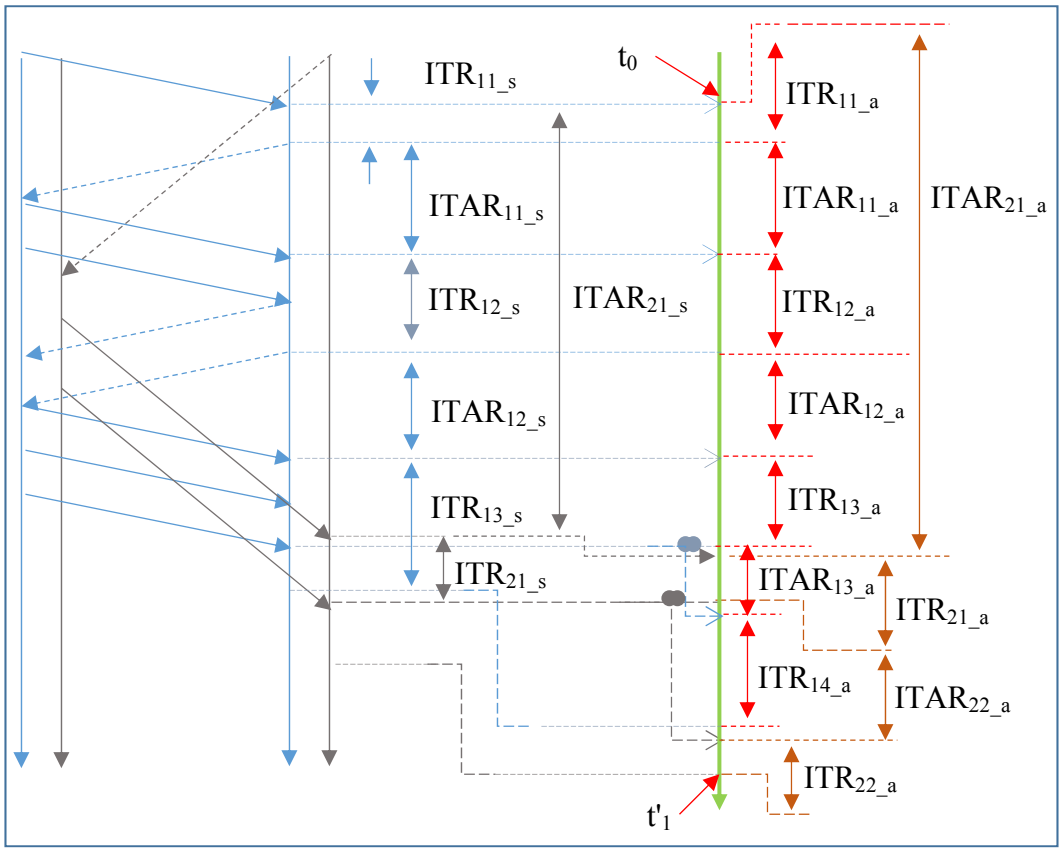

Figure 4.8 Illustration de l'influence des ITAR sur le débit au niveau application pour chacune des sous-sessions (cas de notre proposition)

Que ça soit pour notre proposition ou celle de notre référence, les $ITAR_{ij\_s}$ se rétrécissent au rythme d'élargissement des ITR_{ij_s} déterminé par la phase dans laquelle se fait la réception.

Pour notre proposition, les ITAR ont une largeur qui ne peut atteindre MSS/B2 lorsque le nombre maximal des segments pendant RTT2 (relation 4.43) est atteint car seuls les ITAR de type $ITAR_{13\_a}$ et $ITAR_{22\_a}$ y existent.

$$Nseg_{max} = \frac{RTT2}{MSS * B2} \tag{4.43}$$

4.4.1 Le débit instantané

Il est important de noter que les courbes représentatives du débit pour notre proposition ainsi que pour la référence, sont en fonction du temps de réception. Cela est obtenu en conditionnant la fin de réception avec la même quantité des données pour les deux cas.

A travers la figure 4.9, on observe que le débit instantané est plus important au niveau de la SS1 qu'au niveau de SS2 que ça soit notre proposition ou WRR. Pour notre proposition, la valeur minimale instantanée du débit via SS1 est légèrement inférieure à 1000 octets/5ms pendant presque tout le temps de la réception alors qu'elle est souvent égale à 0 octet/5ms pour WRR. La même observation avec une moindre mesure peut être faite entre la valeur minimale instantanée du débit via SS2 pour notre proposition par rapport à celle débit via SS2 pour la référence.

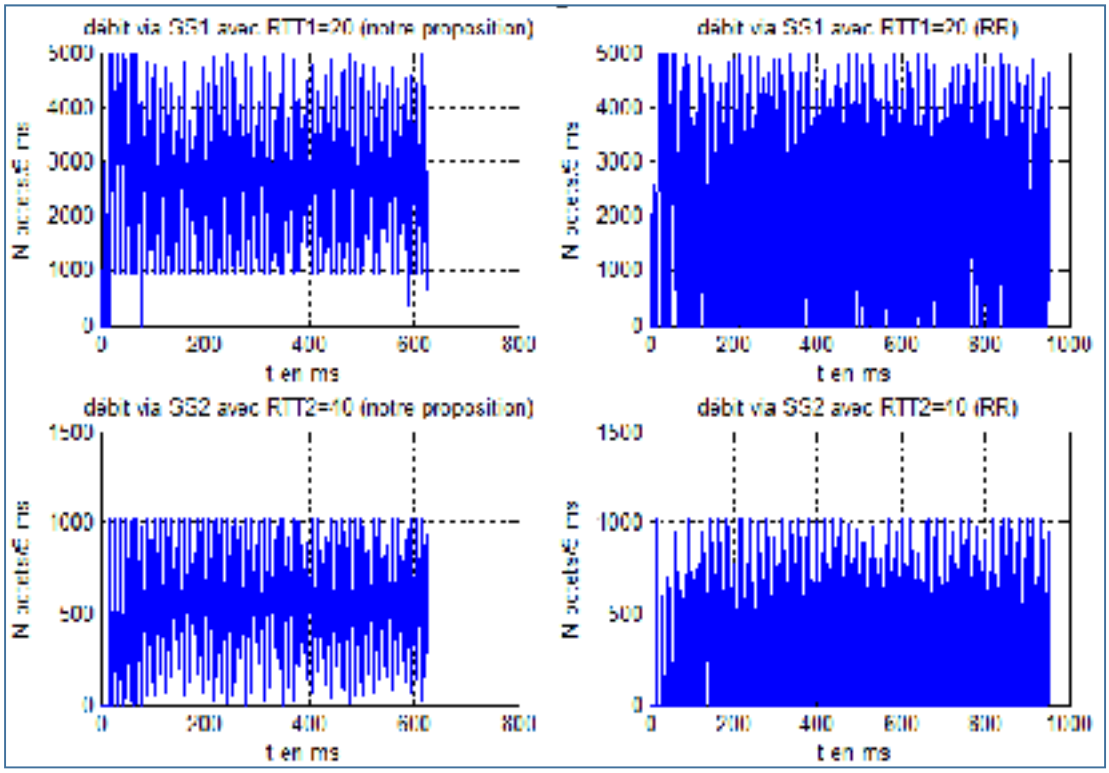

Figure 4.9 Courbes représentatives des débits instantanés via SS1 et SS2 obtenues avec notre proposition et WRR

Pour pouvoir juger la performance en termes de délai, l'observation la plus visible est que le temps de réception est largement plus court pour notre proposition que celui obtenu avec WRR.

A travers les observations, on peut dire que notre proposition apporte une amélioration notable en termes de débit mais surtout en termes de délai. Cette amélioration peut être expliquée par la réduction des ITAR au niveau application grâce la minimisation du nombre des paquets hors ordre.

4.4.2 Débit moyen

Avec le débit moyen, les observations précédentes deviennent plus évidentes (voir figure 4.10) surtout le degré d'amélioration en termes de débit.

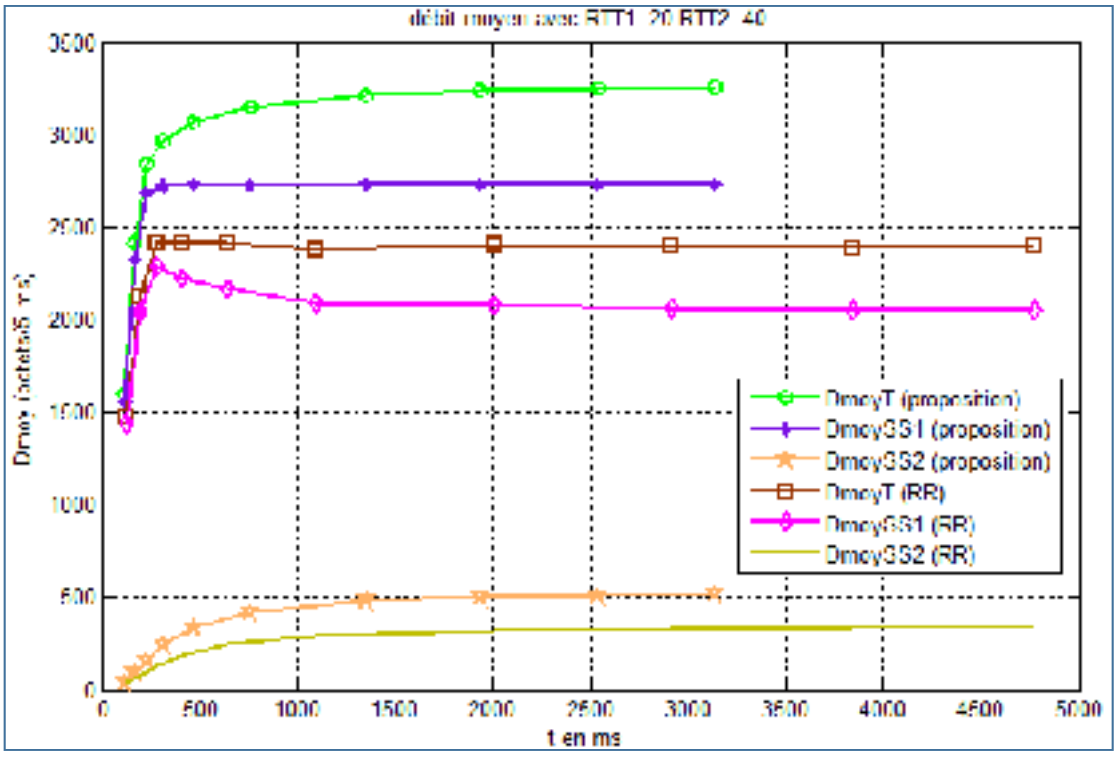

Figure 4.10 Les courbes représentatives du débit moyen (2178372 octets reçus)

Une observation à ne pas omettre est cette croissance linéaire au début de la réception pour toutes les courbes. Son explication réside sur l'élargissement des ITR liée à l'algorithme de

congestion qui est dans la phase de démarrage lent dans cet intervalle de temps. Dans cette phase, il y a plus de chance de recevoir des paquets en ordres via toutes les sous-sessions.

Cependant, après le démarrage lent, il est nettement visible que les courbes représentatives du débit moyen total (DmoyT) et celle de la courbe de débit moyen via SS1 (DmoySS1) pour notre proposition ainsi que pour le Round Robin (RR) s'aplatissent avec une légère croissance de DmoyT de notre proposition. Cet aplatissement est lié à la phase d'évitement de congestion et la réception d'un nombre de plus en plus important des paquets hors ordre surtout pour le RR où c'est plutôt une légère décroissance.

Les courbes de débit moyen via SS2 (DmoySS2) poursuivent très lentement leur croissance, donc la relative décroissance de DmoyT pour le RR est liée à celle de DmoySS1 étant donné que DmoyT est la somme de ce débit et DmoySS2.

Les résultats des figure 4.9 et figure 4.10 sont obtenus avec les mêmes paramètres de simulations. Les plus importants pour le MPTCP sont les RTT1 et RTT2. Quelle influence ces paramètres peuvent-ils avoir sur le débit et le délai pour notre proposition ? Pour répondre à cette question, nous avons simulé le débit moyen avec une nouvelle valeur de RTT2 (donc un nouvel écart entre RTT1 et RTT2), les valeurs des autres paramètres de simulation restés inchangés et nous avons les résultats tels qu'ils sont indiqués par la figure 4.11.

Les courbes de la figure 4.11 nous montrent une nette amélioration de débit au début de la réception aussi bien pour notre proposition que pour le RR par rapport aux courbes de la figure 4.10.

Cependant, il y a une décroissance plus notable de DmoyT qui est liée certainement à celle de DmoySS1 après le point d'inflexion. Si pour DmoySS1 pour RR la décroissance résulte de l'augmentation des paquets hors ordre pendant la phase d'évitement de congestion, à quoi est due celle de DmoySS1 pour notre proposition étant donné que tous les paquets sont censés être reçus en ordre ?

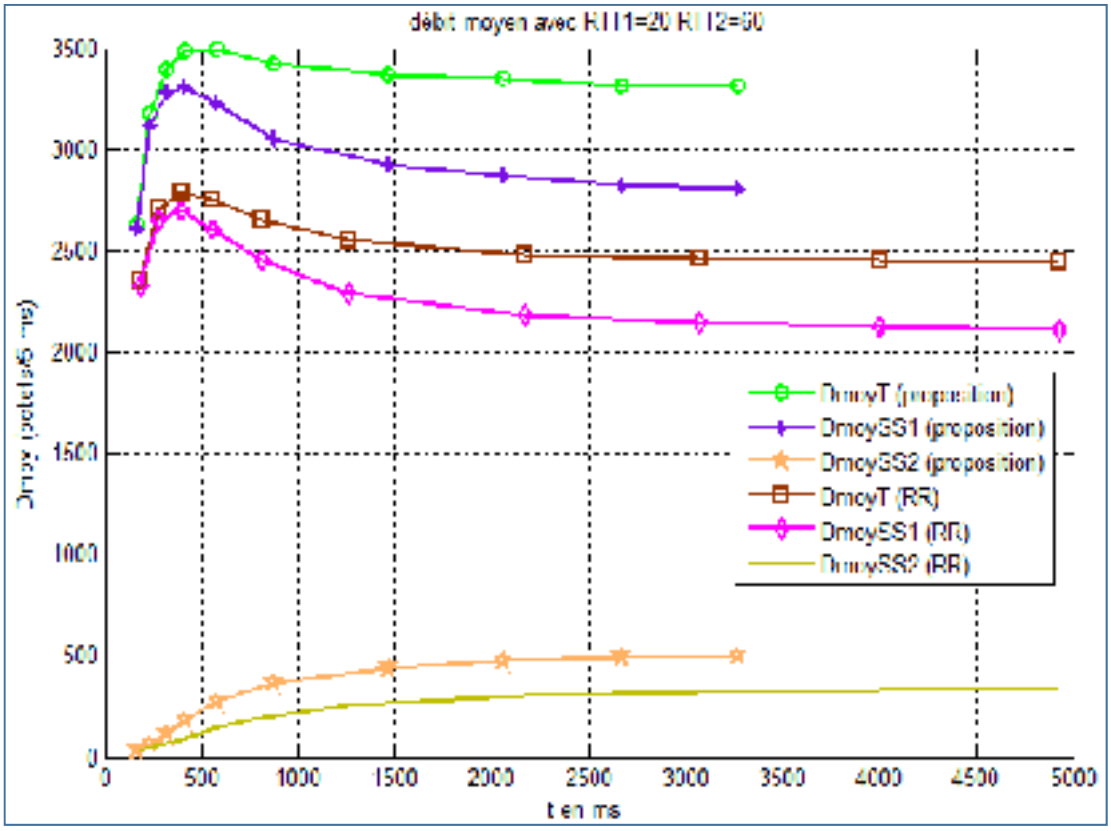

Figure 4.11 Les courbes représentatives du débit moyen (2 178 372 octets reçus)

Du fait de l'augmentation de l'écart entre RTT1 et RTT2, le nombre maximal de segment pendant RTT1 peut être atteint alors les ITAR de type $ITAR_{21\_a}$ (voir figure 4.8) sont encore non négligeables. Ceux-ci vont exister encore plus longtemps si le seuil d'entrer dans la phase d'évitement de congestion est inférieur au nombre maximale des segments pendant RTT2 parce qu'ils se rétrécissent au rythme d'augmentation de fcg2. Donc, c'est le rétrécissement des ITAR de type $ITAR_{21\_a}$ qui justifie la décroissance de DmoySS1 car il en résulte des ITAR de type $ITAR_{13\_a}$.

Si RTT2 augmente et taux de réception via SS2 reste inchangé, il va exister des ITAR même dans la phase d'évitement de congestion et c'est cela la justification de la décroissance de DmoySS1 pour notre proposition.

Pour voir si la décroissance de DmoySS1 va se poursuivre si le temps de réception est plus large, nous avons simulé cette fois avec les données plus importantes et nous avons obtenu la figure 4.12.

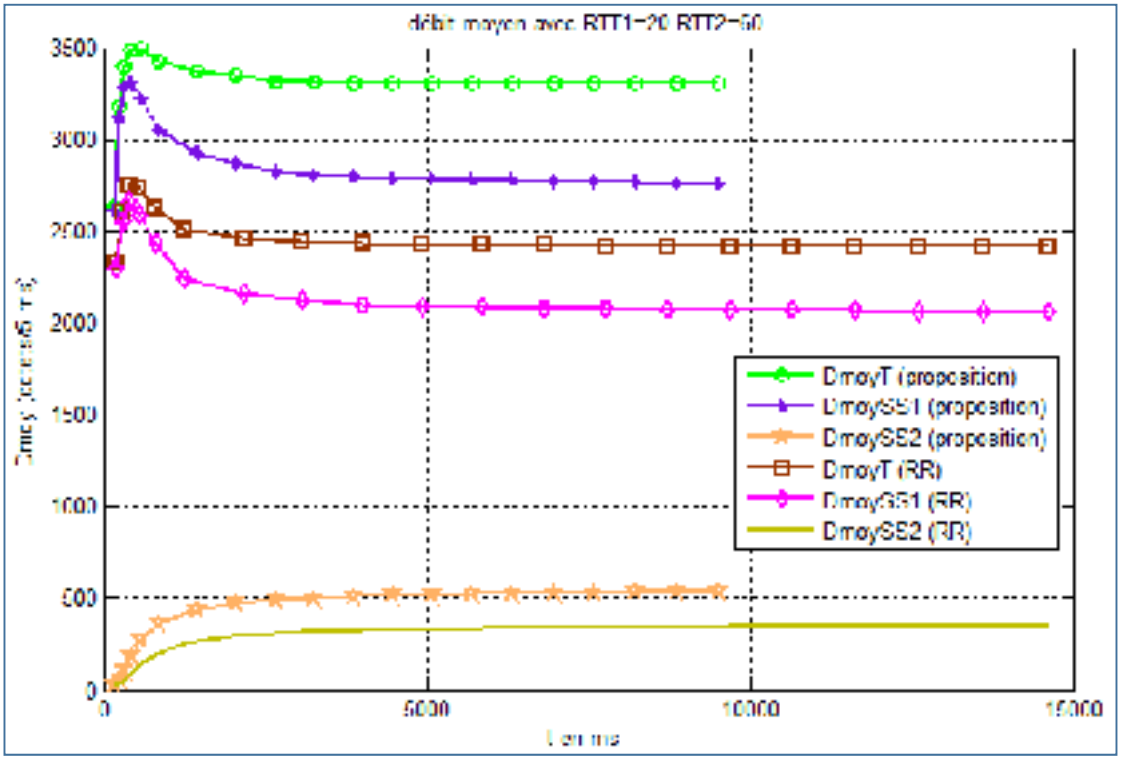

Figure 4.12 Les courbes représentatives du débit moyen (6 310 176 octets reçus)

4.4.2 Délai de réception

Pour observer plus claire la performance de notre proposition en terme de délai, nous relever le délai de réception en fonction de la taille de des données reçue. Pour cela, nous avons gardé fixe les valeurs de RTT1 et RTT2. Il en résulte les courbes présentées à la figure 4.13.

On y observe une augmentation linéaire du délai de réception en fonction de la taille des données. Mais l'observation la plus intéressante est celle de l'écart qui s'accentue avec la taille des données entre le délai de réception pour notre proposition et le délai de réception pour RR. Cela s'explique par le fait que le problème des paquets hors s'accentue avec le temps alors que plus la taille des données est importante, plus le temps de réception s'allonge.

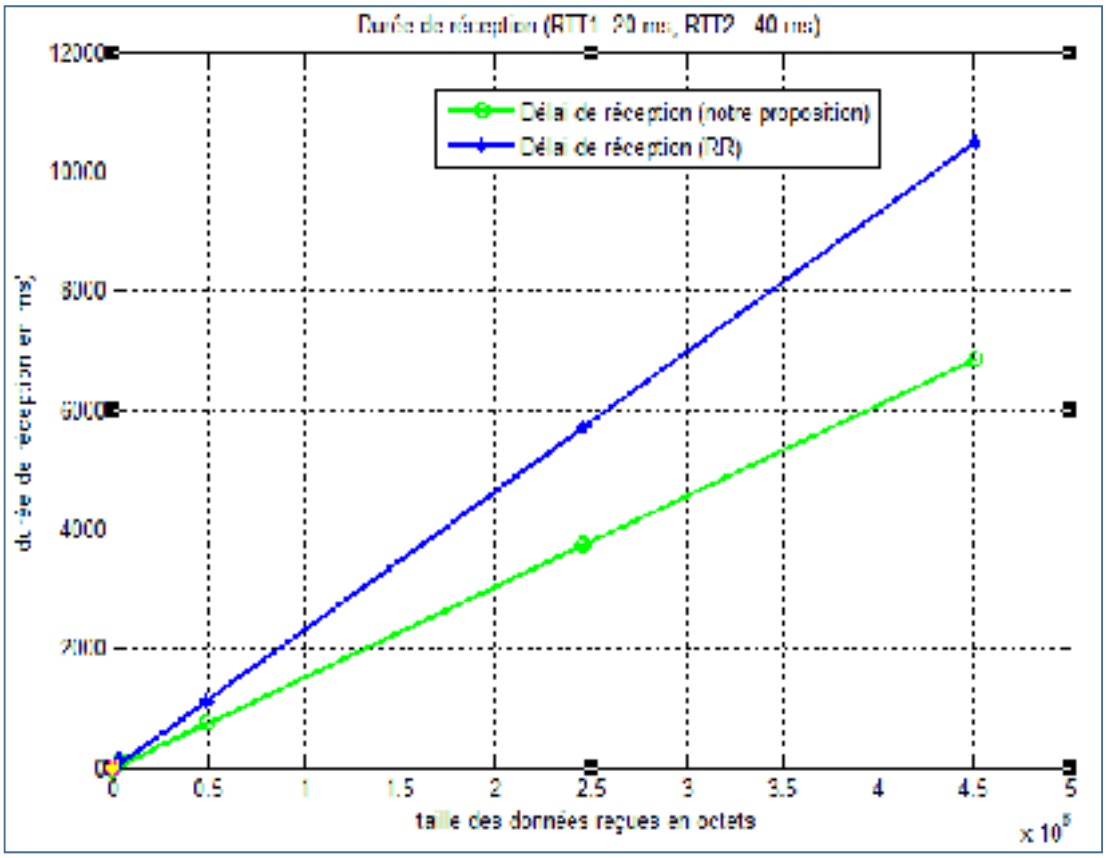

Figure 4.13 Délai de réception en fonction de la taille des données reçues

4.5 Conclusion

Les résultats de simulation suivant les différents scénarios nous montrent que les performances de la connexion MPTCP en termes de débit et délai de livraison sont nettement améliorées par à celles du scheduler implémenté dans linux.

CONCLUSION

Les travaux de recherche que nous avons menés dans le cadre de notre mémoire, nous ont été d'une grande utilité pour comprendre le MPTCP. A travers le premier chapitre, l'étude du fonctionnement du MPTCP nous a permis de mieux comprendre les apports technologiques qu'il peut fournir pour optimiser l'usage des terminaux multi-interfaces mais aussi un meilleur aperçu de ses limites. Parmi ces limites, il y a le problème des paquets hors ordre à la réception autour duquel nous avons développé la problématique de notre recherche dans le deuxième chapitre. Cette problématique s'articule autour de la minimisation du nombre des paquets hors ordre à la réception avec comme objectif l'amélioration des performances du MPTCP en termes de délai et de débit.

Tel qu'il est étudié et expliqué dans le troisième chapitre et analysé à l'aide des résultats de simulation au quatrième chapitre, le scheduler que nous proposons pour améliorer les performances du MPTCP surtout en termes du débit et délai de livraison est un heureux aboutissement pour notre mémoire.

Notre scheduler assure les meilleures performances en termes de débit et de délai de livraison comparé aux résultats du scheduler implémenté dans linux.

Nos simulations sont faites pour une transmission sans pertes, en considérant une connexion MPTCP utilisant deux sous-sessions seulement et dans un environnement moins réaliste. Un autre point que nous n'avons abordé dans l'analyse de notre proposition est le cas d'une connexion MPTCP avec plus de deux sous-sessions. Sur ce point, nous pensons qu'il est possible d'adapter notre scheduler à une connexion MPTCP utilisant plus de deux sous-sessions avec la même stratégie. Cela nous amène à faire les recommandations suivantes pour le futur :

- adapter la stratégie d'assignation des paquets au cas où il y a au-delà de deux sous-sessions. Cela passe par le classement de toutes les sous-sessions en fonction de leur RTT et la fenêtre temporelle de transmission des paquets d'une fenêtre globale doit être

déterminée par le RTT de la sous-session la plus lente et l'évolution de la fenêtre de congestion de celle-ci ;

- toujours pour le cas de plusieurs sous-sessions, la retransmission des paquets perdus via une sous-session doit se faire via la sous-session la plus rapide ;
- effectuer une simulation dans un environnement réseau plus proche de la réalité.

ANNEXE I

LES ALGORITHMES D'ESTIMATIONS DU NOMBRE DES PAQUETS DE ZONES A ET B

Les algorithmes décrits dans cet annexe permettent l'estimation du nombre des paquets des zones A (Np_zA_{i+1}) et B (Np_zB_{i+1}) et à assigner à SS1 pour la fenêtre globale i+1.

Algorithme-A I-1 Estimation de Np_zA_{i+1}

Estimation de Np_zA_{i+1}

Variables d'entrée : fcg_{SS1_j}, fcg_{SS1cnt}, t_{ec}, τ_{SS1}, RTT1, Np_SS1_i Np_ne_j, Np_ack_i, $\Delta t1_i$, $\Delta t2_i$, $\Delta t1_{i+1}$, $\Delta t2_{i+1}$.

Variables de sortie : Np_zA_{i+1}, Np_ne_j, t_{ec}.

1. $Np\_unack_i = 0$; // Nombre des paquets de fg_i non transmis à l'instant t_{ei}
2. Si $fcg_{SS1cnt} = 0$
3. ck=t_{ec} +RTT1-(fcg_{SS1_j}-1)* τ_{SF1} ;
4. On calcul fcg_{SS1_j};
5. $Np\_ne_j = fcg_{SS1\_j}$;
6. Si non
7. ck = t_{ec} ;
8. Fin si ligne 1 ;
9. Tant ck < $\Delta t1_{i+1} + \Delta t2_{i+1}$
10. Tant que Np_ne_j != 0
11. Si ck < $\Delta t1_i + \Delta t2_i$
12. ck = ck + $\tau_{SF1;}$
13. Np_nej = Np_nej -1;
14. Np_acki = Np_acki +1;
15. Si non
16. ck = 0 ;
17. Si $Np\_ack_i < Np\_SS1_i$
18. $Np\_unack_i = Np\_SS1_i – Np\_ack_i$;
19. Si non
20. Np_unacki = 0;
21. Fin si ligne 17;
22. Quitter la boucle tant que ligne 10;

Algorithme-A I-1 Estimation de Np_zA_{i+1} (suite 1)

Estimation de Np_zA_{i+1}

Variables d'entrée : fcg_{SS1_j}, fcg_{SS1cnt}, t_{ec}, τ_{SS1}, RTT1, Np_SS1_i Np_ne_j, Np_ack_i, $\Delta t1_i$, $\Delta t2_i$, $\Delta t1_{i+1}$, $\Delta t2_{i+1}$.

Variables de sortie : Np_zA_{i+1}, Np_ne_j, t_{ec}.

```
23.              Fin si ligne 11 ;
24.          Fin tant que ligne 10 ;
25.          Si Np_nej != 0
26.               Si Np_unacki <= Np_nej
27.                    Si Np_unacki*τSF1 < Δt1i+1
28.                         Np_zAi+1 = Np_unacki;
29.                         ck=ck + Np_unacki*τSF1;
30.                         Np_nej = Np_nej – Np_unacki;
31.                         Np_unack = 0;
32.                    Si non
33.                         Np_zAi+1 = Δt1i+1 / τSF1;
34.                         Np_unacki=Np_unacki –Np_zAi+1;
35.                         Np_nej = Np_nej – Np_zAi+1;
36.                         ck=ck+ Δt1i+1;
37.                    Fin si ligne 27
38.               Si non
39.                    Si Np_nej*τSF1 < Δt1i+1
40.                         Np_zAi+1 = Np_nej;
41.                         Np_unacki=Np_unacki –Np_nej;
42.                         ck=ck + Np_nej*τSF1 + RTT1- (fcgSS1_j – 1)* τSF1;
43.                         Calcul la nouvelle valeur de fcgSS1 (fcgSS1_j);
44.                         Np_nej = fcgSS1_j;
45.                    Si non
46.                         Np_zAi+1 = Δt1i+1 / τSF1;
47.                         Np_unacki=Np_unacki – Np_zAi+1;
48.                         Np_nej = Np_nej – Np_zAi+1;
49.                         ck=ck+ Δt1i+1;
50.                    Fin si ligne 38;
51.               Fin si ligne 26;
```

Algorithme-A I-1 Estimation de Np_zA_{i+1} (suite 2)

Estimation de Np_zA_{i+1}

Variables d'entrée : fcg_{SS1_j}, fcg_{SS1cnt}, t_{ec}, τ_{SS1}, RTT1, Np_SS1_i Np_ne_j, Np_ack_i, $\Delta t1_i$, $\Delta t2_i$, $\Delta t1_{i+1}$, $\Delta t2_{i+1}$.

Variables de sortie : Np_zA_{i+1}, Np_ne_j, t_{ec}.

```
52.          Si non
53.                ck=ck + RTT1- (fcg_SS1_j – 1)* τ_SF1;
54.                Calcul la nouvelle valeur de fcg_SS1 (fcg_SS1_j);
55.                Np_nej = fcg_SS1_j;
56.          Fin si ligne 25 ;
57.      Fin tant que ligne 9 ;
58.      t_ec = ck ;
```

Algorithme-A I-2 Estimation de Np_zB_{i+1}

Estimation de Np_zB_{i+1}

Variables d'entrée : fcg_{SS1_j}, fcg_{SS1cnt}, t_{ec}, τ_{SS1}, RTT1, Np_SS1_i Np_ne_j, Np_ack_i, $\Delta t1_i$, $\Delta t2_i$, $\Delta t1_{i+1}$, $\Delta t2_{i+1}$, Bool.

Variables de sortie : Np_zB_{i+1}.

```
1.  Np_unack_i = 0 ; // Nombre des paquets de fg_i non transmis à l'instant t_ei
2.  Si fcg_SS1cnt = 0
3.        ck=t_ec +RTT1-(fcg_SS1_j-1)* τ_SF1 ;
4.        On calcul fcg_SS1_j;
5.         Np_ne_j = fcg_SS1_j;
6.  Si non
7.      ck = tec ;
8.  Fin si ligne 1 ;
9.  Tant ck < Δt1_i+1 + Δt2_i+1
10.           Tant que Np_ne_j != 0&&Bool = 0
11.                 Si ck < Δt1_i + Δt2_i
12.                       ck = ck + τ_SF1;
13.                        Np_nej = Np_nej -1;
14.                        Np_acki = Np_acki +1;
15.                  Si non
```

Algorithme-A I-2 Estimation de Np_zB_{i+1} (suite1)

Estimation de Np_zB_{i+1}

Variables d'entrée : fcg_{SS1_j}, fcg_{SS1cnt}, t_{ec}, τ_{SS1}, RTT1, Np_SS1_i Np_ne_j, Np_ack_i, $\Delta t1_i$, $\Delta t2_i$, $\Delta t1_{i+1}$, $\Delta t2_{i+1}$.

Variables de sortie : Np_zB_{i+1}.

```
16.                 ck = 0 ;
17.                 Si Np_acki < Np_SS1i
18.                       Np_unacki = Np_SS1i – Np_acki;
19.                 Si non
20.                       Np_unacki = 0;
21.                 Fin si ligne 17;
22.                 Quitter la boucle tant que ligne 10;
23.           Fin si ligne 11 ;
24.     Fin tant que ligne 10 ;
25.     Si Np_nej != 0
26.           Si Np_unacki <= Np_nej
27.                 Si Np_unacki*τSF1 < Δt1i+1
28.                       Np_zBi+1 = Np_unacki;
29.                       ck=ck + Np_unacki*τSF1;
30.                       Np_nej = Np_nej – Np_unacki;
31.                       Np_unack = 0;
32.                 Si non
33.                       Np_zBi+1 = Δt1i+1 / τSF1;
34.                       Np_unacki=Np_unacki –Np_zBi+1;
35.                       Np_nej = Np_nej – Np_zBi+1;
36.                       ck=ck+ Δt1i+1;
37.                 Fin si ligne 27
38.           Si non
39.                 Si Np_nej* τSF1 < Δt1i+1
40.                       Np_zBi+1 = Np_nej;
41.                       Np_unacki=Np_unacki –Np_nej;
42.                       ck=ck + Np_nej*τSF1 + RTT1- (fcgSS1_j – 1)* τSF1;
43.                       Calcul la nouvelle valeur de fcgSS1 (fcgSS1_j);
44.                       Np_nej = fcgSS1_j;
45.                   Si non
46.                       Np_ zBi+1 = Δt1i+1 / τSF1;
```

Algorithme-A I-2 Estimation de Np_zB_{i+1} (suite 2)

Estimation de Np_zB_{i+1}

Variables d'entrée : fcg_{SS1_j}, fcg_{SS1cnt}, t_{ec}, τ_{SS1}, RTT1, Np_SS1_i Np_ne_j, Np_ack_i, $\Delta t1_i$, $\Delta t2_i$, $\Delta t1_{i+1}$, $\Delta t2_{i+1}$, Bool.

Variables de sortie : Np_zB_{i+1}.

47. $Np\_unacki=Np\_unacki - Np\_zB_{i+1}$;
48. $Np\_ne_j = Np\_ne_j - Np\_zB_{i+1}$;
49. $ck=ck+ \Delta t1_{i+1}$;
50. Fin si ligne 38;
51. Fin si ligne 26;
52. Si non
53. $ck=ck + RTT1- (fcg_{SS1\_j} - 1)* \tau_{SF1}$;
54. Calcul la nouvelle valeur de fcg_{SS1} (fcg_{SS1_j});
55. $Np\_nej = fcg_{SS1\_j}$;
56. Fin si ligne 25 ;
57. Fin tant que ligne 9 ;

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

A. Ford, C. Raiciu, M. Handley, et O. Bonaventure. 2013. « TCP Extensions for Multipath Operation with Multiple Addresses ». Internet Requests for Comments, IETF, RFC 6824, Jan. 2013.

A. Ford, C. Raiciu, M. Handley, S. Barre et J. Iyengar. 2011. « Architectural Guidelines for Multipath TCP Development ». Internet Requests for Comments, RFC 6182, IETF, Mar. 2011.

Allen L. Ramaboli, Olabisi E. Falowo, and Anthony H. Chan. 2013. « Using Multiple Links Simultaneously to Increase Capacity for Multi-homed Terminals in Heterogeneous Wireless Networks ». In 2013 IEEE 27th International Conference: Advanced Information Networking and Applications (AINA). (Barcelona, Mar. 25-28 2013), p. 788-793. IEEE.

Amanpreet Singh, Carmelita Goerg, Andreas Timm-Giel et Michael Scharf. 2012. « Performance comparison of Scheduling Algorithms for Multipath Transfer ». In Globecom 2012: Next Generation Networking and Internet Symposium. (Anaheim, Dec. 3-7 2012), p. 2653-2658. IEEE.

C. Raiciu, M. Handly et D. Wischik. 2011. «Coupled Congestion Control for Multipath Transport Protocols ». Request for Comments, RFC 6356, IETF, 2011.

Costin Raiciu, Christoph Paasch, Sebastien Barre, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure et Mark Handley. 2012. « How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP » In NSDI'12 Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation**. (**Berkeley, Av. 25 2012), p. 1-14. USENIX Association Berkeley.

Christoph Paasch, Simone Ferlin, Ozgu Alay et Olivier Bonaventure. 2014. « Experimental Evaluation of Multipath TCP Schedulers ». In International Conference on Emerging Networking Experiments And Technologies : CSWS '14 Proceedings of the 2014 ACM SIGCOMM workshop on Capacity sharing workshop. (New York, Aut. 18 2014), p. 27-32. ACM New York.

Claude Servin. 2013. Réseaux et Télécoms : Cours avec 129 exercices corrigés 3è édition. DUNOD, p. 980.

Danièle, Dromard et Dominique Seret. 2013. Architecture des réseaux informatiques : Synthèse de cours et exercices corrigés. Montreuil : Pearson, 279 p.

D. Borman, B. Braden, V. Jacobson et R. Scheffenegger. 2014. « TCP Extensions for High Performance ». Request for Comments, RFC 7323, IETF, sep. 2014.

Dizhi Zhou, Wei Song et Minghui Shi. 2013. « Goodput Improvement for Multipath TCP by Congestion Window Adaptation in Multi-Radio Devices ». In The 10th Annual IEEE CCNC: Wireless Networking Track. (Las Vegas, Jan. 11-14 2013), p. 508 – 514. IEEE.

Fan Yang, Qi Wang et Paul D. Amer. 2014. « Out-of-order Transmission for In-order Arrival Scheduling for Multipath TCP ». In 2014 28th International Conference on Advanced Information Networking and Applications Workshops (WAINA). (Victoria, BC Ma. 13-16 2014), p. 749-752. IEEE.

Foued Melakessou , Ulrich Sorger et Zdzislaw Suchanecki. 2007. « MPTCP: Concept of a Flow Control Protocol Based on Multiple Paths for the Next Generation Internet ». In International Symposium on Communications and Information Technologies (ISCIT, 2007). (Sydney, 17-19 Oct. 2007), p. 568-573. IEEE.

J Postel. 1981. « Transmission Control Protocol DARPA Internet Program Protocol Specification » Internet Requests for Comments, RFC 793, DARPA, Sep. 1981.

Joseph Davies. 2007. « The Cable Guy : Réglage automatique de la fenêtre de réception TCP ». In le site de TechNet Magazine. En ligne <https://technet.microsoft.com/fr-fr/magazine/2007.01.cableguy.aspx>. Consulté le 25 décembre 2015.

M. Allman, V. Paxson et E. Blanton. 2009. « TCP Congestion Control ». Request for Comments, RFC 5681, IETF, sep. 2009.

Martin, Becke. 2014. « Revisiting the IETF Multipath Extensions on Transport Layer ». DISSERTATION to obtain the academic grade doctor rerum naturalium (dr. rer. nat.) in Computer Science, Institute for Computer Science and Business Information Systems, University of Duisburg-Essen, 207 p.

Mohamed BOUCADAIR et Christian JACQUENET. 2015. « Techniques de l'ingénieur : L'expertise technique et scientifique de référence » In le site de techniques de l'ingénieur : Communiquer sur des chemins multiples Atouts du protocole MPTCP**.** En ligne. **<**http://www.techniques-ingenieur.fr>. Consulté le 28/11/2015.

Simone Ferlin-Oliveira, Thomas Dreibholz et Özgü Alay. 2014. « Tackling the Challenge of Bufferbloat in Multi-Path Transport over Heterogeneous Wireless Networks ». In 2014 IEEE 22nd International Symposium of Quality of Service (IWQoS). (Hong Kong, Ma. 26-27 2014), p.123-128. IEEE.

Yung-Chih Chen et Don Towsley. 2014. « On Bufferbloat and Delay Analysis of MultiPath TCP in Wireless Networks ». In 2014 IFIP Networking Conference. (Trondheim, Jun. 2-4 2014), p. 1-9. IEEE.